FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Adam Blažek

Efficient Video Retrieval using Complex Sketches and Exploration based on Semantic Descriptors

Department of Software Engineering

Supervisor of the master thesis:	RNDr. Jakub Lokoč, Ph.D.
Study programme:	Computer Science
Study branch:	Theoretical Computer Science

Prague 2016

Title: Efficient Video Retrieval using Complex Sketches and Exploration based on Semantic Descriptors

Author: Adam Blažek

Department: Department of Software Engineering

Supervisor: RNDr. Jakub Lokoč, Ph.D., Department of Software Engineering

Abstract: This thesis focuses on novel video retrieval scenarios. More particularly, we aim at the Known-item Search scenario wherein users search for a short video segment known either visually or by a textual description. The scenario assumes that there is no ideal query example available. Our former known-item search tool relying on color feature signatures is extended with major enhancements. Namely, we introduce a multi-modal sketching tool, the exploration of video content with semantic descriptors derived from deep convolutional networks, new browsing/visualization methods and two orthogonal approaches for textual search. The proposed approaches are embodied in our video retrieval tool Enhanced Sketch-based Video Browser (ESBVB). To evaluate ESBVB performance, we participated in international competitions comparing our tool with the state-of-the-art approaches. Repeatedly, our tool outperformed the other methods. Furthermore, we show in our user study that even novice users are able to effectively employ ESBVB capabilities to search and browse known video clips.

Keywords: video retrieval color signatures query-by-sketch decaf descriptors

Hereby, I dedicate this work to my family which encouraged and supported me during my studies.

I am deeply grateful to my supervisor RNDr. Jakub Lokoč, Ph.D. for his expertise, time and guidance not only on professional level.
I am thankful to Doc. RNDr. Tomáš Skopal, Ph.D. who enabled me to present this work at a number of international conferences.

Finally, I would like to express my gratitude to my colleagues who helped me to realize my ideas; namely, David Kuboň and Filip Matzner.

This research has been supported by Czech Science Foundation (GAČR) projects P202/11/0968 and P202/12/P297, SVV projects SVV-2014-260100 and SVV-2015-260222, and Charles University in Prague Grant Agency GAUK project no. 1134316.

Contents

1. Preface

Nowadays world is without any doubt greatly influenced by modern information technologies. Digital revolution is rapidly transforming virtually every branch of industry and countless aspects of our every day life. Devices capable of recording and storing data are omnipresent and interconnected. Nonetheless, the Internet is nothing like cold digital plains occupied by machines, it is in fact the exact opposite. Billions of peoples are communicating through social networks on daily basis with texts, emails, images and videos.

On YouTube alone, more than 300 hours of video content is being uploaded every minute. With personal devices capable of capturing images and videos we indeed face an explosion of multimedia. Multimedia databases are vigorously growing in the terms of size, complexity and diversity. Hence, we see the arising need of effective solutions for browsing and retrieving multimedia collections.

Similarly to vision being the dominant source of information from our environment, video represents one of the richest, most complex and also challenging type of multimedia. Within five minutes, one writes an email while another records a several gigabytes of video. Collections comprising hundreds of hours of video content are becoming commonly available for educational, professional and personal use.

With diverse applications, novel video retrieval scenarios are emerging. Furthermore, the video data are generally not annotated; thus, it is not possible to employ regular textual search techniques. In this thesis, we tackle the topic of video retrieval and browsing. We propose, implement and evaluate a tool for interactive browsing of large video collections. Main attention is paid to the scenario where users search for a known video segment having no suitable example object.

We base our tool on simple yet effective description of a color distribution in video key-frames. Users express their search intent in the form of color-position sketches. This basic approach is further combined with edge-based descriptors, semantic similarity search, text-based retrieval and effective browsing. In comparison to other state-of-the-art tools, our approach appears vital and actually outperforms a number of them. Our system has become a decent base-line for the problem of known-item search in video.

All the up-to-date solutions to the problem including ours put users to the central role. The performance is therefore influenced by the level of expertise and many other factors. Meanwhile, artificial intelligence is able to answer questions asked in natural language within fractions of seconds. Effortlessly, we obtain informations such as where is the nearest coffee shop. We do expect similar solution to be available for video retrieval in the foreseeable future.

2. Introduction

With the advance of various recording devices, multimedia creation is no longer a domain reserved for professionals. Devices being used on daily basis such as smartphones or tablets are able to capture high-quality images and videos in a convenient way. Hence, large video collections are becoming common, yet their sizes, contents and applications are indeed diverse. To portray the variability of video collections, let us describe few examples:

Medical videos for surgery after inspection and training of physicians are often recorded. During the after inspection, physicians control critical phases of the surgery and select screenshots for documentation (in some countries required by legislation). During their education, surgeons have to observe a number of operations before they can assist at one. Educational database of operations captured on video might accelerate this process. Students might look up for the particular type of operation or retrieve cases when a specific complication happened.

Industrial research videos help to better understand and analyze complex physical processes in various products. For example, high-speed cameras can be used to record processes in an engine during the combustion. The results of such analysis can help to design new types of engines with lower fuel consumption and producing less harmful emissions.

Personal video collections are probably the most growing category of videos. Memorable moments from people vacation, weddings etc. are being commonly captured and stored by billions of ordinary users. Nonetheless, finding one particular shot within tens or hundreds of hours of video content collected over years can be quite challenging even for a single personal archive.

Professional media productions create high quality videos, often equipped with subtitles and soundtracks. Thousands of short clips are organized and labeled for easy navigation and access. The provided annotation however, might be ambiguous, misleading or even missing.

Once the quantity of video data reaches some level, the *findability* of specific scenes becomes one of the main issues of video management systems. The videos can be searched by means of available attributes (e.g., date of creation, GPS location, length). However, these attributes do no help to find semantic information recorded in the videos. Some engines [22, 40] implement keyword search, where users specify their search intents using natural language. The keyword search stands or falls on the automatic annotation of the videos [41, 79, 85]. While humans are able to effortlessly and precisely process visual stimuli and recognize objects, persons or ongoing events, the same does not apply to machines. Although computer vision has achieved major breakthroughs [60] in recent years, many problems are far from being satisfactory solved. For example, when it comes to automatic video event detection state-of-the-art methods [82] achieve no more than 45% mean average precision for specific benchmarks. Since it is still a difficult task to automatically annotate video collections, the search engines employ also content-based similarity search using query by example. In this research area, deep learning [38] has started a revolution boosting the performance of similarity search significantly. However, this scenario can be considered as long as users have an appropriate query example object. Therefore, some research directions focus on the *Known-item Search* (KIS) tasks where users are searching for a particular video clip known either visually or by textual description (i.e., no example video is available).

To solve KIS tasks, automatic annotation and content-based similarity search are employed in combination with exploration scenarios, where users conveniently browse and explore the database contents or lists of results. For example, users start with KIS query specification, browse the results and pick one of the results for similarity search. Iterations of results browsing, query refinement and similarity search form interactive and often effective user-centric retrieval process.

Although various advanced KIS retrieval tools exist, the *usability gap* can prevent from using them. It is common that once the keyword search fails, ordinary users often skip to classical players with intuitive playback controls. Clearly not optimal, nor scalable, but very intuitive solution. Hence, as users become a significant element of the retrieval process, the interfaces have to be intuitive, responsive and convenient. In other words, KIS tools have to provide comprehensible query initialization, data visualization and exploration interfaces.

In order to initialize KIS search, the state-of-the-art KIS tools often rely on automatic concept detection methods. Even with lower than desirable precisions, object recognition and localization or concept detection techniques might provide valuable additional information for video retrieval[1]. With effective browsing features, a system for video retrieval [53, 63] based on automated concept detectors may provide reasonable results. Alternatively, video content might be described with robust low-level features such as color distribution [50] or optical flow [11]. Differently to high-level concept detection, the features can be directly extracted from the data. Nonetheless, these low-level features are often hard to be specified by users or they are not descriptive enough.

During the last decade, several orthogonal approaches for KIS have emerged [76]. Unlike classical benchmarks for automatic retrieval, the performance assessment of tools for KIS video retrieval is complicated problem on its own. As users are involved in the process, the results are influenced by many factors including the level of user expertise. For this reason, comparative competitions such as Video Browser Showdown (VBS) [73] are being organized. In VBS case, each participating team develops its own tool for video retrieval with which solves plurality of KIS tasks. In particular, both visual and textual KIS tasks are solved by experts and novices during live competition. In this way, the state-of-the-art methods are directly compared. Another relevant workshop is TRECVID [80] where algorithms for concept detection, object localization and event recognition in video are compared. In 2017, VBS will cooperate with TRECVID and use the same IACC.3 video data set comprising 600 hours of video.

[1]Note that in some specific cases, machine learning algorithms even surpassed human performance [26].

2.1 Our contribution

In this thesis, we propose, implement and evaluate a video retrieval tool that provides an aid for KIS in video collections. We aim at the situation when no additional information for video content is available (i.e., no subtitles, no soundtrack). In the early versions of the tool, users could express their search intent in a form of color-position sketch drawings, which led to the name Sketch-based Video Browser (SBVB). The actual version introduces multi-modal sketches and supports also keyword search to initialize textual known item search tasks. Interactive exploration of the database content is further supported with a number of features including semantic similarity search, compact visualizations and interactive navigation summaries.

Note that the first version of the tool was introduced and defended as my Bachelor Thesis [9] in 2014. Since then, the work has continued and we enhanced SBVB with a lot of new features, modalities and browsing options. In other words, SBVB evolved to Enhanced SBVB (ESBVB) which is presented in this thesis. Most of the contributions were published at international conferences. Hereby, we include the list of our published papers:

1. **Signature-Based Video Browser** [46]
   Demo paper describing the overall functionality of SBVB.

2. **On Effective Known Item Video Search Using Feature Signatures** [48]
   Demo paper describing our retrieval model.

3. **Video Retrieval with Feature Signature Sketches** [7]
   Full paper on the overall concept and index optimizations.

4. **Enhanced Signature-Based Video Browser** [8]
   Extended demo paper describing the enhancements done in 2014.

5. **Multi-sketch Semantic Video Browser** [39]
   Extended demo paper describing the enhancements done in 2015.

6. **Known-Item Search in Video Databases with Textual Queries**
   Short paper describing our textual query interface and model. Paper is accepted to the conference on Similarity Search and Applications 2016 in Tokyo. Its contents are covered with Sections 6.3 and 9.2.

7. **Interactive Video Search Tools: a Detailed Analysis of the Video Browser Showdown 2015** [14]
   Journal paper thoroughly analyzing the results from VBS 2015.

Demo papers 1, 4 and 5 were accepted to VBS 2014, 2015 and 2016 respectively where our tool competed with the other state-of-the-art methods. In short, we won VBS 2014 and 2015 and ended up 3rd in VBS 2016. Detailed results and analysis are presented in Chapter 8.

2.2 Thesis Structure

The thesis is organized as follows. We start with introduction to similarity search and image/video retrieval as preliminaries to our work (Chapter 3). Before we describe our system, we portray the state-of-the-art of video retrieval in Chapter 4. The description of the tool is divided in two chapters. Firstly, in Chapter 5 we present the original SBVB version with many (not yet published) additional details about the retrieval model. Secondly, in Chapter 6 we go through all the enhancements done in the last two years.

The architecture of our ESBVB implementation (available in DVD) is discussed in Chapter 7. In the following Chapter 8, we present and discuss the results from VBS 2014 – 16. Next, we discuss the results of two user studies in Chapter 9 and finally, we conclude the work and outline our future work in Chapter 10.

An inseparable part of the thesis is a DVD containing the ESBVB implementation, the user tutorial and other materials. Additionally, we maintain a project web page available at `http://siret.cz/project/sbvb`.

The proposed models and algorithms were tested on a dataset of 200 hours of video content kindly provided by BBC. The dataset comprises various BBC programmes including TV shows, documents and broadcasts. If not stated otherwise, the depicted images originate from this dataset. All rights regarding these materials are reserved to BBC.

3. Preliminaries

In this chapter, we build the apparatus needed to describe our tool and algorithms. In particular, we start with defining concepts related to metric spaces, similarity search and general object retrieval. We follow with domain specific concepts such as digital image processing and image descriptors.

3.1 Metric and Vector Spaces

A *Metric space* $\mathbb{M}$ is a pair $\mathbb{M} = (\mathbb{D}, \delta)$, where $\mathbb{D}$ denotes a descriptor universe and δ is a distance function $\mathbb{D} \times \mathbb{D} \to \mathbb{R}$ ($\mathbb{R}$ is the set of real numbers) satisfying the following conditions $\forall x, y, z \in \mathbb{D}$:

$$\delta(x,y) \geq 0 \qquad \text{(non-negativity)}$$

$$\delta(x,y) = 0 \quad \text{iff} \quad x = y \qquad \text{(identity)}$$

$$\delta(x,y) = \delta(y,x) \qquad \text{(symmetry)}$$

$$\delta(x,y) + \delta(y,z) \geq \delta(x,z) \qquad \text{(triangular inequality)}$$

We understand the elements of $\mathbb{D}$ as object descriptors (or simply objects) and the function δ as a distance or dissimilarity measure between them. The crucial property of metric spaces is the triangular inequality as it induces a number of interesting properties.

The well known example of a metric space is defined over the d-dimensional vector space $\mathbb{R}^d$ with the following formula:

$$\delta(x,y) = L_2(x,y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2} \tag{3.1}$$

The distance function (3.1) is also called the Euclidean distance. With $d = 2$ or $d = 3$ we get the regular space with δ measuring the actual distance between two points. The distance function (3.1) might be generalized to

$$L_p(x,y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{\frac{1}{p}} \tag{3.2}$$

for an arbitrary $p \geq 1$. The effects of different values of p are demonstrated in Figure 3.1, wherein we display unit ball regions [1] centered in $(0, 0)$.

[1]For point x a unit ball is the set of all y having $\delta(x,y) \leq 1$.

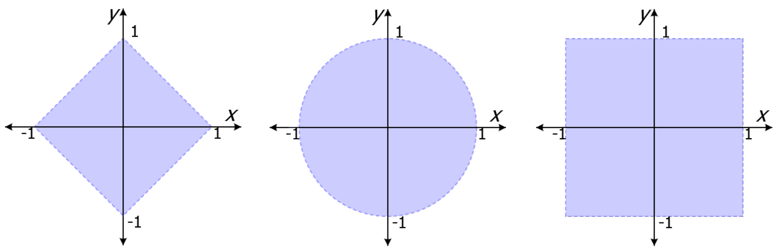


Figure 3.1 Unit balls centered in $(0, 0)$ under different L_p metric functions. From left to right: $p = 1$ (Manhattan distance), $p = 2$ (Euclidean distance) and $p = \infty$ (Chebyshev distance).

3.1.1 Similarity Search

Often, $\mathbb{S} \subset \mathbb{D}$ represents a database of objects subjected to various queries. As we have available a dissimilarity measure δ we are able to identify the most similar (or least dissimilar) objects within $\mathbb{S}$ for a particular query $q \in \mathbb{D}$. The query could be expressed in natural language as "find k closest objects for q". Formally, we consider two popular types of queries:

k-nn Given a query object $q \in \mathbb{D}$ and $k \in \mathbb{N}$; return first k objects from the ascending ordering of all objects $x \in \mathbb{S}$ with respect to $\delta(q, x)$ (i.e., k-nearest neighbors from $\mathbb{S}$ to q)

range Given query object $q \in \mathbb{D}$ and a range $r \in \mathbb{R}_0^+$; return $X = \{x \,|x \in \mathbb{S} \wedge \delta(q, x) \leq r\}$ (i.e., all the objects within the specified range)

Both queries might be processed with naive algorithm which calculates all the distances between database objects and query object q. Unfortunately, the distance function δ could be computationally expensive and/or the number of objects in $\mathbb{S}$ might be very high.

3.1.2 Indexing Metric Spaces

In order to avoid the computation of the distance function during processing of similarity search queries, various indexing techniques were proposed [87]. Roughly, the idea is to pre-compute some auxiliary data structures so that we can filter the database objects that should not be retrieved. The filtering conditions often utilize the triangular non-equality in some form. Sometimes, it is acceptable to allow filtering which also filters objects that should be retrieved. In such case, we are talking about approximate similarity search.

To demonstrate how filtering in a metric space might look alike assume the following example. In metric space $(\mathbb{R}^2, L_2)$ with objects $O = \{o_1, o_2 \ldots\}$ we pre-compute the distances to the selected pivot p (that is $L_2(p, o_1)$, $L_2(p, o_2) \ldots$). Given a **range** query (q, r) we do the following:

1. Compute $L_2(q, p)$
2. Filter $O' = \{o \,|\, o \in O,\, |L_2(q, p) - L_2(p, o)| \leq r\}$
3. Return $O'' = \{o|o \in O',\, L_2(q, o) \leq r\}$

The situation is depicted in Figure 3.2. Note that we do not compute any distances in the second step! The distance is computed only to the pivot and objects that pass the filtering. This filtering is beneficial for expensive distance functions. Mathematically, the condition in the second step is justified with

$L_2(q,p) - L_2(p,o) > r$ (filtered object)

$L_2(q,o) + L_2(p,o) >= L_2(q,p)$ (triangular inequality)

$L_2(q,o) >= L_2(q,p) - L_2(p,o)$

Now, by combining 1st and 3rd row we obtain:

$L_2(q,o) > r$ (out of range)

I.e., a filtered object o can not be in the range r from q.

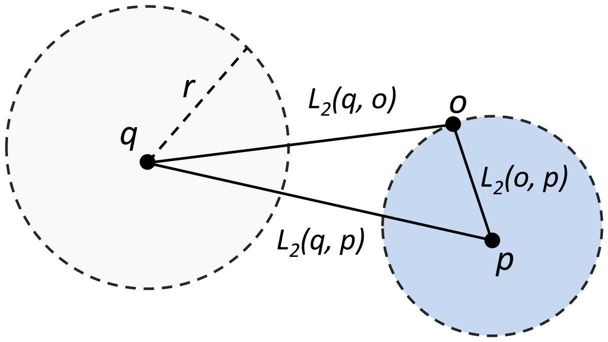


Figure 3.2 Range query (q,r) configuration in which we can filter the object o thanks to the pivot constraint $L_2(q,p) - L_2(p,o) > r$.

Our pivot and the actual query range effectively identify parts of the database that can be completely skipped. With additional pivots and constraints we are able to further partition the database and tighten the searched area even more. The metric space approach provides also ball and generalized hyperplane partitioning methods enabling filtering of groups of objects [87]. Although we do not propose novel constraints and filtering techniques, we utilize state-of-the-art metric indexing methods in our system.

The metric space approach is an efficient choice especially for high-dimensional data that cannot be indexed by traditional spatial access methods [71]. However, for uniformly distributed low-dimensional vector data, classical grid indexes could be a more efficient choice [7].

3.2 Image Retrieval and Similarity Search

Despite we aim at video retrieval problems, most of the concepts are exactly the same in the image domain. As images are less complex than video content, we start with defining basic image related concepts and we move to the video domain later on.

An image is often defined as a real function of two spatial variables.

$$f(x,y) = z \quad ; \quad f : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}^+ \tag{3.3}$$

That is, for a given point (x, y) we understand $f(x, y)$ as the intensity at that point. Although working with images as functions may clarify and ease a number of complex problems, we are going to prefer the discrete definition. For our needs a gray-scale image I is a matrix of size $M \times N$ of integer values in one byte range $0 - 255$. An image element I_{ij} is named pixel. A color image can be defined as a stack of 3 gray-scale images, each image or channel representing either red, blue or green color component (Fig. 3.3).

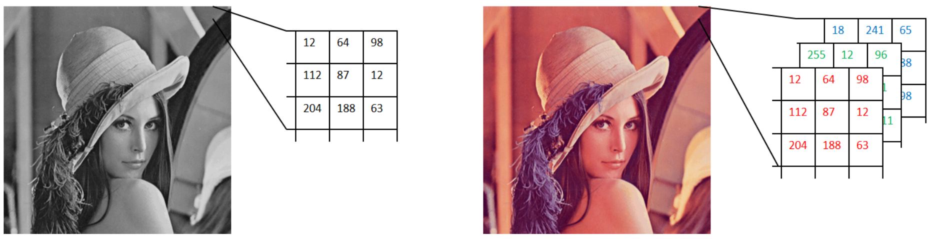


Figure 3.3 Discrete understanding of digital images. A gray-scale image (left) and a color image (right).

3.2.1 Image Retrieval

Nowadays, databases of millions of images are quite common; thus, algorithms, indexes and techniques for large scale image retrieval are in the spotlight. Out of all the problems being solved, we define the following scenarios:

qbe Query by Example: given an example image, we are searching for either other instances of the same object or visually similar images.

sbir Sketch-based Image Retrieval: searching for particular object/image, users do not have an example available; thus, the query is specified as a user-drawn sketch.

In fact, those image retrieval scenarios can be solved by the metric spaces approach and similarity queries. Furthermore, we know that with proper indexing techniques, similarity search queries might be processed efficiently.

Now, the question is what is the proper descriptor space for images and of course what distance function shall be utilized. Often these two challenges are solved together, i.e. we design an image descriptor together with a distance function.

3.2.2 Image Descriptors

The selection of an appropriate image descriptor is often a complicated problem. There is no always-best option and with different tasks and datasets the optimal descriptor would also differ. We do not provide an exhaustive list of image descriptors; instead, we demonstrate the overall idea of descriptor design on the simple example of a color histogram descriptor which captures the distribution of colors in images.

Speaking about descriptors capturing color distribution in images, we already have one – the images themselves. Unfortunately, for image retrieval problems raw images are rather useless for a number of reasons. At first, we would have to deal with the very large number of dimensions (the number of pixels times the number of channels). Furthermore, even an image shifted by a few pixels would be quite different (pixel-wise) from the original; thus, we would have to design more sophisticated perceptually sensitive similarity measures that are more expensive.

Basic color histograms compactly represent color frequencies in images. Basically, we sum up the numbers of pixels with specific colors. Alternatively, we do this summation channel-wise as is depicted in Figure 3.4. As we lost the spatial distribution of colors, color histogram is so called global descriptor capturing overall image characteristics.

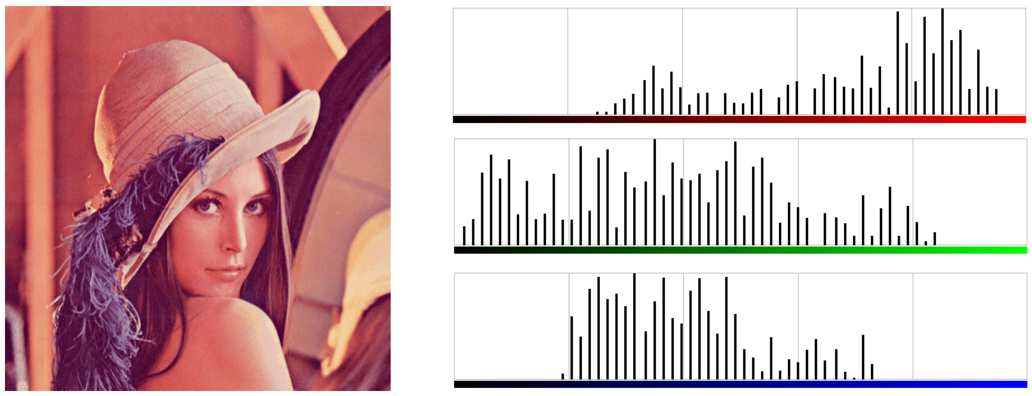


Figure 3.4 Per-channel color histogram of Lena color image.

Now, imagine an image with randomly shuffled pixels. Its color histogram would be the same as the color histogram of the original image although the images are not similar at all (Fig. 3.5). This should be no surprise as we know that we simply completely lost the spatial information.

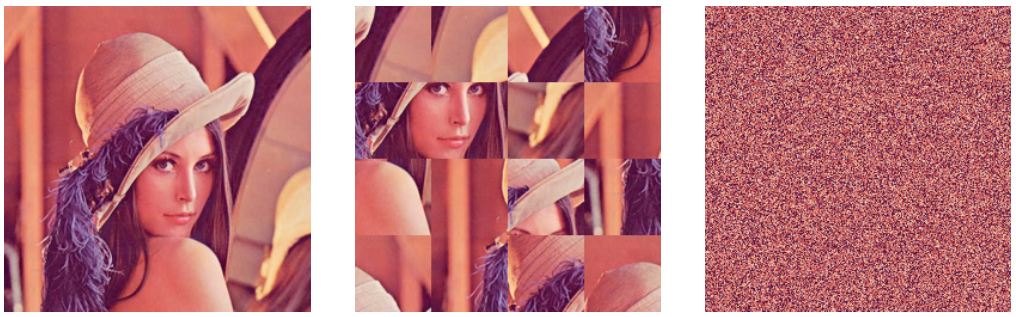


Figure 3.5 Image of Lena (left), the same image with randomly shuffled tiles (center) and with randomly shuffled pixels (right). All the images would have exactly the same color histogram.

We might enhance the global color histogram descriptor so that it roughly captures the color distribution with simple trick. We divide the image in $K \times L$ equally sized parts and calculate the color histogram for each part separately (Fig. 3.6).

In this way, images in Figure 3.5 would yield different descriptors. Side effect is that the local histograms descriptor is precisely $K \times L$ times larger than one global histogram.

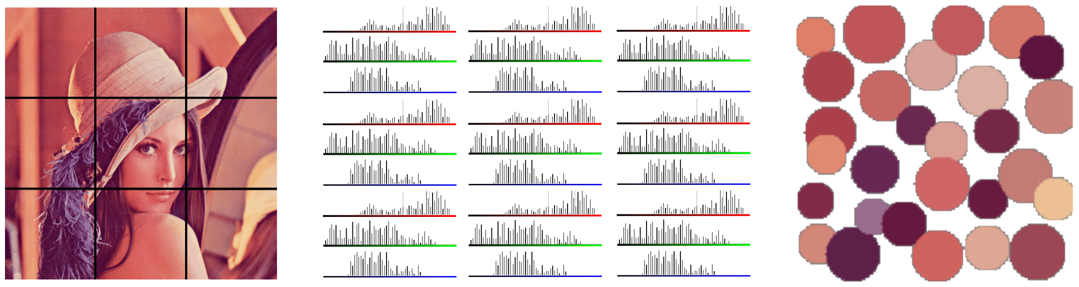


Figure 3.6 An image (left), its local image histogram descriptor (center) and feature signature (right).

With 1-byte per color channel representation, total of $256^3 = 16777216$ different colors may be specified. A color histogram with this dimensionality would not be any more compact than the image itself. For this reason, we often quantize the colors in each channel. Finally, a difference between two histograms might be measured with their correlation.

Color histogram is quite simple yet in some scenarios surprisingly powerful image descriptor. Nonetheless, it is easy to come up with scenarios when it fails completely. For example, although images of dalmatian and golden retriever are semantically close (both are dogs), their color histograms would be completely different.

In our system, we utilized the feature signature color descriptor (see Fig. 3.6 – right) which represents the image as a set of distinct color regions. Differently form local color histograms, feature signatures do not follow a rigid grid structure and thus adapts better to image contents. We provide more elaborate description of feature signatures in Section 5.1.

It is clear that the selection of a suitable image descriptor is the crucial step in designing an image retrieval system. We might utilize descriptors capturing different image properties such as its edges, complexity etc. or even combine multiple descriptors together. Through this work, we describe several additional image descriptors and discuss the reasons for their selection.

3.2.3 Image Convolution

Seemingly aside from image retrieval and image descriptors is the image convolution operation. Mathematically, the convolution is defined as a functional operator, i.e., a mapping between two function spaces where in our case the functions are images. If we stay with the discrete images it is a sliding window operator which maps an image to another image. The window is called the convolution filter or kernel and is in fact a small matrix (e.g.: 3×3 or 5×5). With an input image I and a kernel K of dimensions $W \times H$, the output image O is defined as:

$$O(i,j) = \sum_{k=-\lfloor \frac{W}{2} \rfloor}^{\lfloor \frac{W}{2} \rfloor} \sum_{l=-\lfloor \frac{H}{2} \rfloor}^{\lfloor \frac{H}{2} \rfloor} K\left(k + \left\lfloor \frac{W}{2} \right\rfloor, l + \left\lfloor \frac{H}{2} \right\rfloor\right) I\left(i+k, j+l\right) \tag{3.4}$$

I.e., the kernel is centered at (i, j), point-wise multiplied with the underlying portion of image I and summed to produce the pixel value of the filtered image $O(i, j)$ (Fig. 3.7).

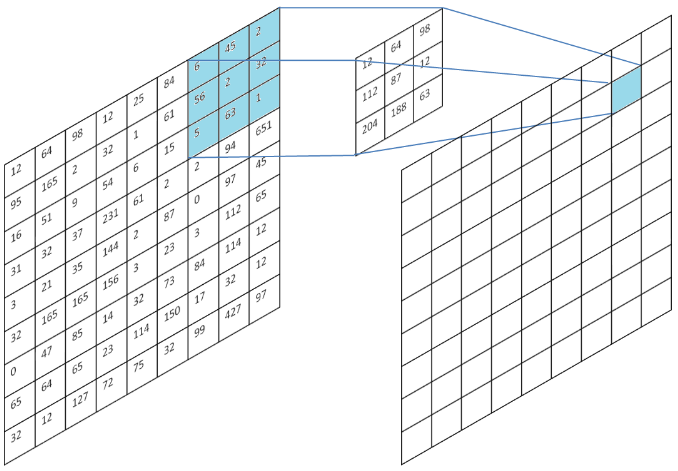


Figure 3.7 Discrete 2D convolution schema. The convolution filter or kernel (middle) is slided over the original image (left). At each position, point-wise multiplication and summation produce one pixel value in the resulting filtered image (right).

The output image is dependent on the values of the utilized convolution kernel and, of course, on the input image. We might design filters for amplifying image edges, blurring images etc.

Convolution operations are fundamental building blocks of deep convolutional neural networks (DCNN) described in the following section. Furthermore, we utilize convolution for detecting edges in images (Section 6.1).

3.3 Deep Learning as Frontier in Image Retrieval

For a long time, hand crafted features based on detection of key points in images were dominant in image retrieval, image matching, image registration etc. Huge effort was paid for crafting, refining and nurturing these features. A typical example of such hand-crafted feature is the Scale-Invariant Feature Transform image descriptor (SIFT) [49]. The SIFT descriptor pursues the idea that certain key points are characteristic for the objects captured in images. Ideal key points should be invariant to various image transformations such as rotation, scaling etc. SIFT features actually hold these properties and thus, became very popular for various image related problems. An example of detected SIFT key-points might be seen in Figure 3.8.

Visual object recognition and image classification has been one of the major challenges in computer vision for decades. Since its introduction, SIFT and other similar features played a major role in virtually every solution proposed for these problems. Given a set of images divided to certain classes, we start with detection of key points. These key points are consequently quantized to centers of clusters (so called Visual Words) obtained by clustering of all the extracted key points.

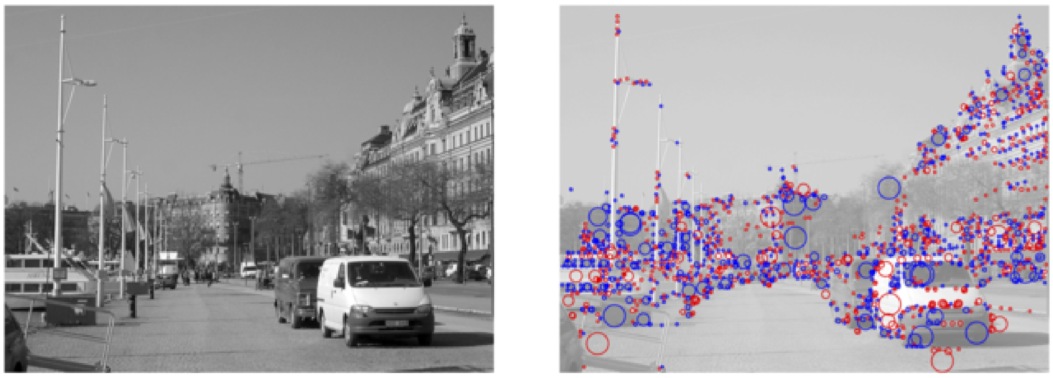


Figure 3.8 A gray-scale image (left) and its SIFT key points (right). Images courtesy of [45]

Each image is represented by the histogram of these Visual Words, which is known as the Bag of Visual Words (BoVW) model [2].

Machine learning algorithms, such as Support Vector Machines (SVM) [27], can be employed to classify the resulting feature vectors. A common processing pipe-line designed in 2012 or earlier would comprise these steps:

1. Key-points detection
2. Key-points clustering (visual vocabulary building)
3. Calculating BoVW representation
4. Training SVM classifier

While each step of the pipe-line can be optimized with variety of improvements and tricks, everything depends on the properties of the key points being utilized. It is unclear if there is any better way of selecting and representing the key points. The performance of visual recognition systems is being evaluated annually at ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [68]. ILSVRC comprises 1000 object categories, each having around 1000 example images. In years 2010 and 2011 the state-of-the-art systems following roughly the sketched pipe-line did not achieve better than 25% top-5 error rate [3] which is quite far from the desirable levels.

Then, deep convolutional neural networks (DCNN) emerged on the scene in 2012. The renaissance of neural networks was initialized by daringly large model utilizing convolutional layers, thorough sub-sampling with max pooling and clever prevention of over-fitting with so called drop-out. Famous AlexNet [38] was designed for ILSVCR 2012 and achieved less than 16.5% error rate. The community quickly adopted the new course and pushed the performance of DCNNs on image classification tasks even further (see Table 3.1 for details).

The power and beauty of DCNNs lies in their architecture mimicking the mammalian visual cortex. Crucial components of DCNNs for visual recognition

[2] An alternative to BoVW model are the Fisher [72] or VLAD [34] Vectors encodings. Both are global image descriptors obtained by an aggregating of the local image features (e.g. SIFTs).

[3] Top-5 error rate refers to the percentage of images for which the real category was not within the top-5 categories suggested by a particular visual recognition system.

Table 3.1 ILSVCR image classification results from 2010–2014.

Year	**Method**	**Top-5 error**
2010	Key points + SVM	29.8%
2011	Key points + SVM	25.6%
2012	DCNN	16.4%
2013	DCNN	11.7%
2014	DCNN	7.4%

are convolution layers containing small convolution filters with learned parameters. Once learned, convolution filters detect local patterns present in images. As we delve deeper, detected patterns become more complex until finally, we are able to separate the image categories with regular neural network with couple of fully connected layers. Although the architecture is not complicated (see Figure 3.9), one must make use of many tricks to actually create a DCNN model which generalizes well. Bottom line is that features are learned instead of crafted by hand and the whole system is learned in the end-to-end manner. Furthermore, some of the learned models are publicly available which allows us to stay within a user-level and utilize them as black-box components. The details how we utilize DCNNs are provided in Sections 6.2 and 6.3.

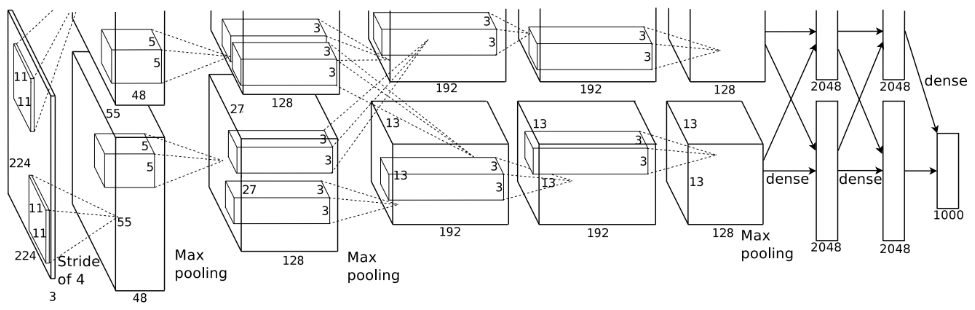


Figure 3.9 AlexNet architecture comprising convolutional layers (local pattern detectors), max-pooling layers (down-sampling) and fully connected layers (classifier). Courtesy of [38].

4. Known-item Search Tools

Video retrieval is a broad research topic touching other areas such as machine learning, information retrieval, database systems and cognitive science. It is out of the scope of this work to provide exhaustive introduction to all these areas. Instead, we portray the context of our work with a short description of video retrieval systems designed to solve the same tasks as we try to solve using our Sketch-based Video Browser.

In particular, a short overview of the top performing teams during the last five years of the Video Browser Showdown are presented. Although many interesting approaches and tools competed at the Video Browser Showdown, only the top-3 teams are summarized for each year in this chapter. If a team appeared with their tool multiple times in the set of top-3 scoring teams, the evolution of the tool is chronologically summarized.

The overall results of VBS 2012–2016 are captured in Table 4.1 wherein our tool is denoted as SIRET. (E)SBVB is thoroughly described in Chapters 5 and Chapters 6. In depth analysis of the VBS results is provided in Chapter 8.

Year	2012	2013	2014	2015	2016
1st place	KLU1	NII-UIT	SIRET	SIRET	HTW
2nd place	NUS	DCU	NII-UIT	IMOTION	KLU-UU
3rd place	KLU2	KLU1	KLU1	UU	SIRET
Teams	8	6	7	9	9
DB size	1h*	1h*	25h	100h	250h

Table 4.1 Video browser showdown winners and competition settings. * the name of a one hour video was specified before each search task.

4.1 KLU team

The KLU team from the Institute of Information Technology, Klagenfurt University has successfuly presented several top-3 scoring tools at VBS. The first tool (KLU1 in Table 4.1) is the AAU video browser [16, 18] that won VBS in 2012 and then took the third place for two times in 2013 and 2014 (out of competition). The AAU video browser relies on (exhaustive) human computing by providing parallel or hierarchical browsing methods. The tool relies mainly on the ability of the users to track several video streams in parallel. Surprisingly, the tool performed well also on the collection comprising 25 hours of video. The second version of the AAU video browser tool uses also a content-based analysis of the video to provide augmented navigation bars depicting repeating segments [18]. The second presented tool (KLU2 in Table 4.1) focuses on video browsing with a 3D thumbnail ring arranged by color similarity [75] and participated only in 2012. The second tool relies also on the sequential inspection of the video. Instead of time preserving visualization, the tool arranges the keyframes by their dominant hue color in the HSV color space. The third tool, developed together with UU team (KLU-UU in Table 4.1), utilizes also content-based filtering considering automatic concept detection and color feature signatures. The tool focuses also on the collaborative video search (involving more users) combining video retrieval using a desktop application with human-based visual inspection of a tablet application [30].

4.2 NII-UIT team

The NII-UIT tool was designed by the international team from National Institute of Informatics in Tokyo and the University of Information Technology and the University of Science in HCM City. The tool participated three times at the VBS, where in years 2013 and 2014 the tool took the first and the second place [42, 55]. The first version of the tool relies on coarse-to-fine presentation, where each one hour video clip is divided into short compact segments and then grouped using a hierarchial clustering method. The tool employs also filtering methods reducing the number segments for inspection. Category, layout and color distribution based filters are considered. The second version of the tool adds also sequential patterns that can be used to find pairs of consecutive scenes with predefined patterns. In order to identify concepts, support vector machines in connection with classical features (BoVW and HOG) are trained.

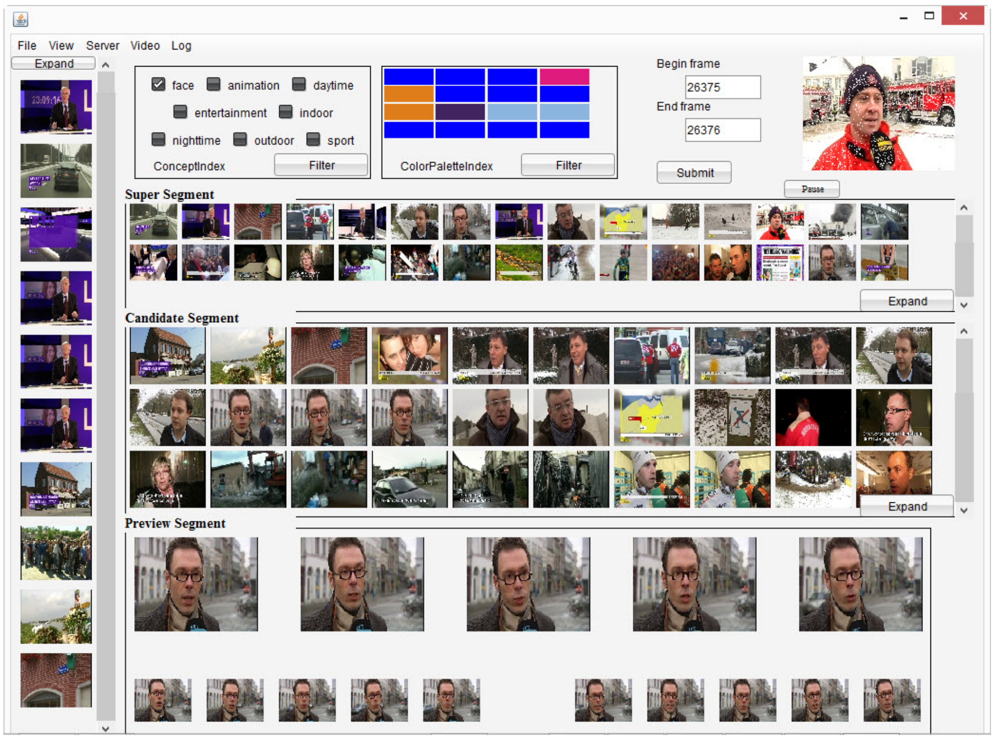


Figure 4.1 The tool winning VBS 2013 developed by the NII-UIT team. Image courtesy of [42].

4.3 HTW team

The Visual Computing Group from HTW Berlin, University of Applied Sciences, has presented a novel browsing approach [5] based on a hierarchical graph and visually sorted image maps. In order to create the graph (in the preprocessing phase), the authors consider visual features and semantic features learned from a convolutional neural network. In order to preserve complex image relations during efficient navigation, a novel method projecting images from the graph organization to 2D plane is utilized. To initialize the search, time ordered sketches, categories or frequent scenes can be employed. The results of the search (by sketch or similarity) are visually sorted and the user can use each displayed image to jump either to the graph or to the video clip. If the image appears at multiple levels of the graph, the highest level is selected.

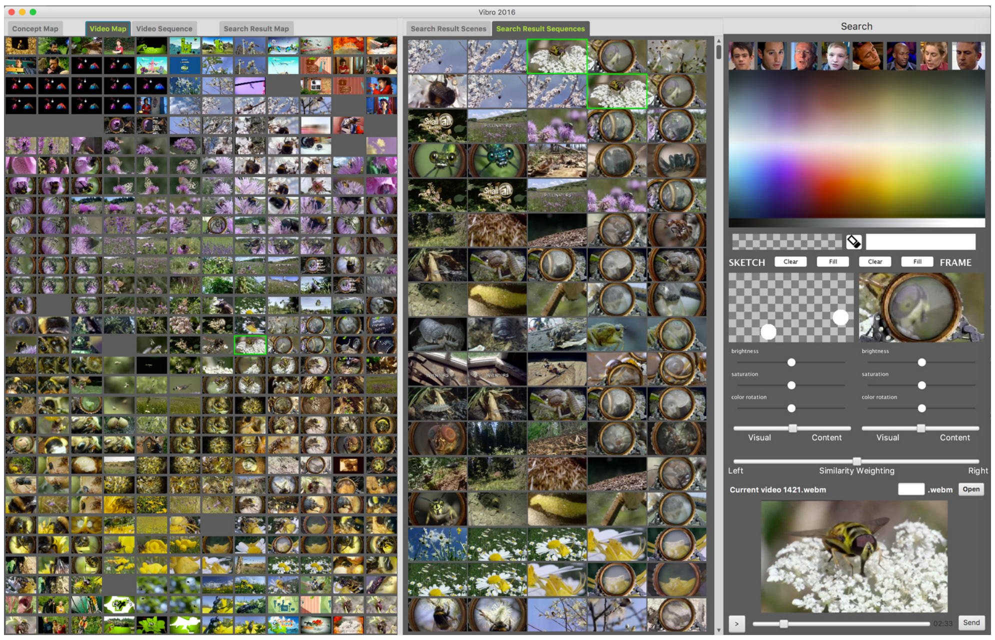


Figure 4.2 The tool winning VBS 2016 developed by the HTW team. Image courtesy of [5].

4.4 IMOTION team

The IMOTION system is a joint effort of the Numediart Institute for Creative Technologies at the University of Mons (Belgium), the Intelligent User Interfaces Lab at Koç University (Turkey), and the Databases and Information Systems Research Group at the University of Basel (Switzerland) [63]. A multitude of low-level (global, regional and motion based) and high-level (deep learning based) features is considered in the system. The system supports rough color or edge sketches, motion queries or query by example. The three modes are complemented by a refinement of query results by means of a relevance feedback.

4.5 UU team

The team from the Department of Information & Computing Sciences, Utrecht University, The Netherlands has presented a story-board based interface for mobile video browsing [31]. The tool considers limiting factors in the interface design (e.g., small screen, touch and tilting actions) and relies only on human computing and interaction, excluding video analysis and machine-based query processing. Hence, for 100 hours of video in VBS 2015 and uniform sampling of keyframes every second, 360000 keyframes has to be inspected. Considering research works about optimal images sizes for storyboards on mobiles, 625 images on one screen were considered for the employed device. These settings result in 550 screens that have to be visited manually. The tool uses a compact space filling curve for temporal arrangement of thumbnails and a simplified storyboard navigation interaction restricted to up/down motions.

4.6 DCU team

The team from the Dublin City University focused on visual concept classification, visual similarity search and clustering, and face browsing and search [77]. The visual concept classification helps with filtering of non-relevan scenes, where the concept detectors used popular BoVW representations and support vector machines. For similarity search in the set of keyframes, detected sparse SIFT descriptors were aggregated to VLAD vectors. The PCA was used to reduce the dimensionality of VLAD vectors. The agglomerative clustering on the VLAD descriptors extracted for similarity search was performed to support coarse browsing of groups of similar keyframes. Since the videos at VBS often contain faces, the tool also provides a face view that shows all the faces found in the video and supports navigation operations in the space of faces.

4.7 NUS team

The team from the School of Computing, National University of Singapore has prepared a tool integrating visual content, text and concept based search approaches [86]. For the visual and conceptual inputs, the user can further specify a temporal order to enhance the search performance. More specifically, multiple color based features can be entered to select matching scenes, or a sequence of concept bundles can be used to describe the searched scene. The text based approach enables users to select a category enabling relevance ranking of the results. The ranking method relies on the tag files of the top 100 videos from YouTube and textual words extracted by ASR and OCR. For each query modality, also a related sample can be selected to display the k nearest samples from the database.

5. Sketch-based Video Browser

Our tool for known item video retrieval has been developed since 2013 and was initially presented as Signature-based Video Browser (SBVB) in 2014 [47], later it was renamed to Sketch-based Video Browser. In this chapter, we recapitulate the key ideas embodied in SBVB which is going to serve us as a reference point for the next Chapter 6. Besides the already published contents [9, 47] additional details are provided. In particular, the not yet published details are:

- the feature signatures extraction algorithm and its hyper-parameters,
- the discussion on distance function options for the video retrieval model,
- and partial results caching in video retrieval model

The SBVB tool is build around a simple yet effective feature signatures descriptor [66] which flexibly captures a color distribution in the key-frames and at the same time allows to specify color-position sketches. The database contents are ranked according to the similarity with the defined sketch and the top matching key-frames are presented together with their preceding and following key-frames. Especially visual KIS tasks can be solved using such interaction loop, where in each iteration users modify the color sketch and refine results. In this chapter, we thoroughly describe all the significant parts of the original SBVB tool – feature signatures, the retrieval model (Section 5.1) and the user interface (Section 5.2).

5.1 Feature Signatures

The wide variety of image and video descriptors and similarity measures was introduced during the last decades [33, 43, 61, 67]. Each descriptor was designed to optimize different criteria and to capture different image or video properties. To address the visual KIS problem, we would like to utilize a descriptor that

Eff is robust, compact, scalable and provide enough filtering power

Comp can be extracted and indexed with low computational and memory demands

User can be easily specified and understood by users

While the first two properties **Eff** and **Comp** are being held by many descriptors it is the last **User** property that often imposes the actual challenge for KIS retrieval. To provide an effective solution for the visual KIS scenario, it is vital when users are able to express their search intent in a simple and convenient way.

One of the candidates for the image descriptor which satisfies all the desired properties are color-position feature signatures (FS) [66]. With FSs, an image is represented by a set of its distinct color regions where the number of regions and their sizes may vary. Rigorously, an image i is described with FS_i defined as

$$FS_i = \{r_{ij}\}_j \tag{5.1a}$$
$$r_{ij} = (x, y, L, a, b, r) \tag{5.1b}$$

In other words, a feature signature (Eq. 5.1a) is a set of positioned color regions r_{ij}, whereas each color region (Eq. 5.1b) is defined by the position of its centre (x, y), its color in the Lab color space [1] (L, a, b) and its size or alternatively its diameter r. Example of visualized feature signatures are depicted in Figure 5.1.

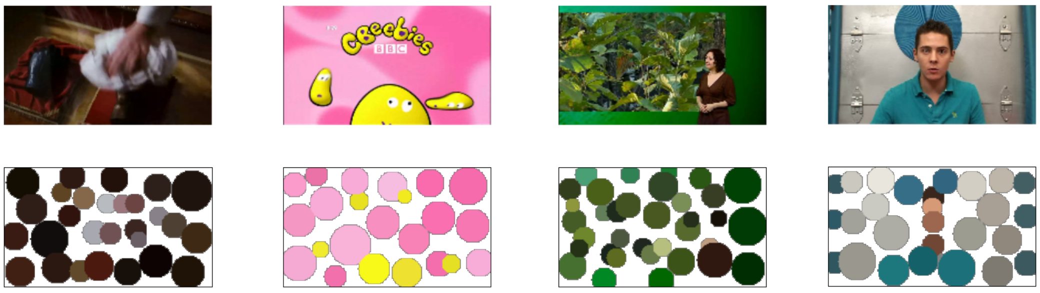


Figure 5.1 Example key-frames and the visualizations of their feature signatures.

Now, given a video clip to be searched, we assume that users are able to memorize the overall composition of one or more scenes. Perhaps with a simple sketching tool, the memorized scene composition might be materialized and used as an actual query to the system. We designed a simple signature sketching tool (depicted in Fig. 5.2) where only colored circles (user-defined centroids) are placed instead of complicated drawings. In this way, we describe the dataset images/key-frames in the same way/format as users are expressing their search intents.

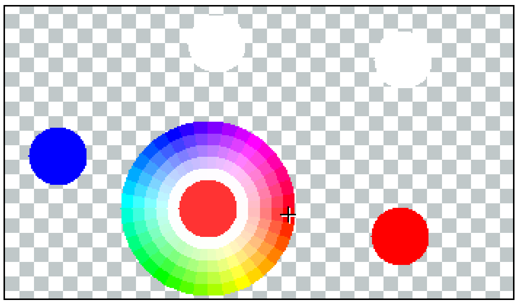


Figure 5.2 FS sketching canvas. An arbitrary combination of colored circles might be defined in order to express the memorized distinct color regions in the searched scene. Colors might be picked with our custom color-picker displayed ad-hoc for each centroid.

In the rest of this section, we provide details on FS extraction algorithm and FS sketch-based image retrieval.

5.1.1 Extraction Algorithm

Our feature signatures extraction algorithm is an adaptive variant of the k-means clustering [24]. The original K-means clustering is a well known algorithm for

[1] Lab is a color space designed so that the Euclidean distance corresponds to human-perceived color difference uniformly over the whole space.

finding k clusters of objects from a vector space R^n of arbitrary dimension n. Translated to our world of images, we are finding distinct color regions (clusters) in a color-position space of image pixels.

First, we describe the vanilla k-means clustering algorithm. Second, we introduce our variant of k-means clustering tailored to the problem of finding distinct color regions in images. As our algorithm have a number of hyper parameters, we also walk through the process of the parameters fine-tuning.

K-means clustering [2]

Given a set of points X and an initial set of k means $m_1^{(1)} \dots m_k^{(1)}$ (e.g. random points from X) the two following steps are repeated until a terminating condition is reached.

Assigning points to means: Assign the points to the nearest mean.

$$S_i^{(t)} = \{x \in X \mid L_2(x, m_i^{(t)}) \leq L_2(x, m_j^{(t)}) \, \forall j = 1 \dots k\} \tag{5.2}$$

Updating means: Update the means according to the previous assignment.

$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x \in S_i^{(t)}} x \tag{5.3}$$

The algorithm converges to a local optimum once $S_i^{(t)} = S_i^{(t+1)}$ for every i; however, it is not guaranteed that the global optimum will be found in this process. It is also common to terminate the algorithm earlier, for example, after a limited number of iterations. A simple example of k-means clustering in 2D space is depicted in Figure 5.3.

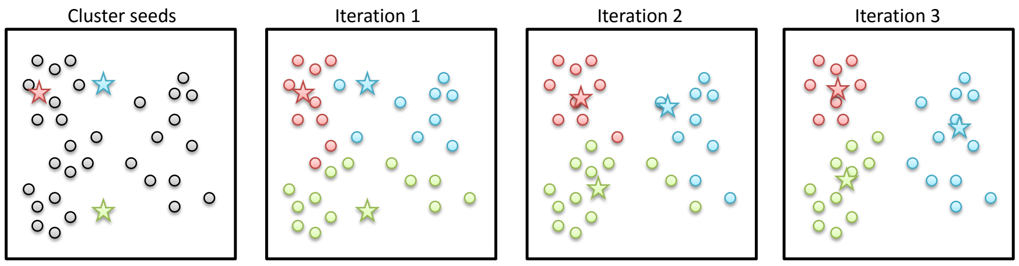


Figure 5.3 K-means clustering in 2D space with $k = 3$. Current cluster centers are depicted with stars.

Adapting K-means for Feature Signatures Extraction

In principle, k-means might be utilized directly for finding color regions in images. In this case, objects to be clustered are the image pixels, i.e., 5-dimensional color-position points (x, y, L, a, b). As the resolution of images might be quite high, we might cluster only a randomly selected subset of image pixels.

[2]Description of k-means clustering is adapted from [9].

We understand the set of centers of the detected clusters as our color-position image descriptor – feature signature. The feature signature might be visualized as colored circles where the circle diameters are defined with the number of pixels assigned to the respective clusters (see again Figure 5.1). We might understand the feature signature as a rough approximation of the original image.

Nonetheless, the vanilla k-means algorithm does not fit our needs. In particular, we would like to achieve the following properties:

Var Different images should have different number of clusters (based on their complexity)

Uni Smaller yet distinct color regions shall be preserved

Dim Different dimensions may have different importance (e.g., color vs. position)

While the **Dim** property might be achieved by scaling the dimensions prior running the algorithm, the vanilla version of the k-means algorithm does not ensure neither **Var** nor **Uni** properties. Therefore, we are introducing our variant of k-means algorithm adapted to feature signatures extraction. The main idea is to start with densely and uniformly distributed small clusters and with continuous merging of similar clusters and deletion of small clusters obtain a flexible feature signature representation. This idea might be embodied as follows:

1. Scale images down to P pixels
2. Scale x, y coordinates with a scaling factor S_p and a, b coordinates with a scaling factor S_c
3. Initialize cluster centers to the uniform spatial grid with span of $G \times G$ pixels
4. Update clusters color coordinates to the local average of pixel colors
5. Repeat N times:
    (a) Assign pixels to the nearest cluster center
    (b) Recalculate cluster centers
    (c) Merge clusters C_1 and C_2 if
        $L_2(C_1, C_2) < M_1$ and
        $L_2(\text{color}(C_1), \text{color}(C_2)) < M_2$
    (**Var**, **Uni**)
    (d) Delete clusters with less than DF^{iter} pixels (**Var**)
    (e) Recalculate cluster centers

The behavior of our algorithm is well illustrated in an example depicted in Figure 5.4. We can see that with more iterations we obtain a coarser yet more compact representation of the original image. Disadvantage is that the algorithm contains a number of hyper-parameters (summarized in Table 5.1).

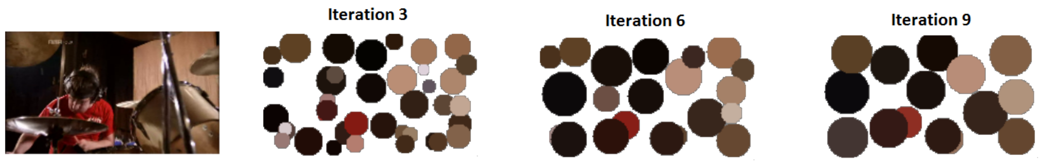


Figure 5.4 An image (left) and visualizations of extracted FSs after 3,6 and 9 iterations of our FS extraction algorithm.

Table 5.1 Hyper-parameters of feature signatures extraction algorithm.

Notion	Name	Affects
P	Resolution	comp. cost, precision, number of clusters
S_p, S_c	Dimension scaling	color-position balance
G	Cluster density	comp. cost, precision, number of clusters
N	DNN	comp. cost, FS coarseness and compactness
M_1	Merge threshold	num. of clusters, FS coarseness and comp.
M_2	Merge color threshold	distinct color preservation
D	Delete threshold	FS coarseness and compactness
F	D increment	speed of compacting

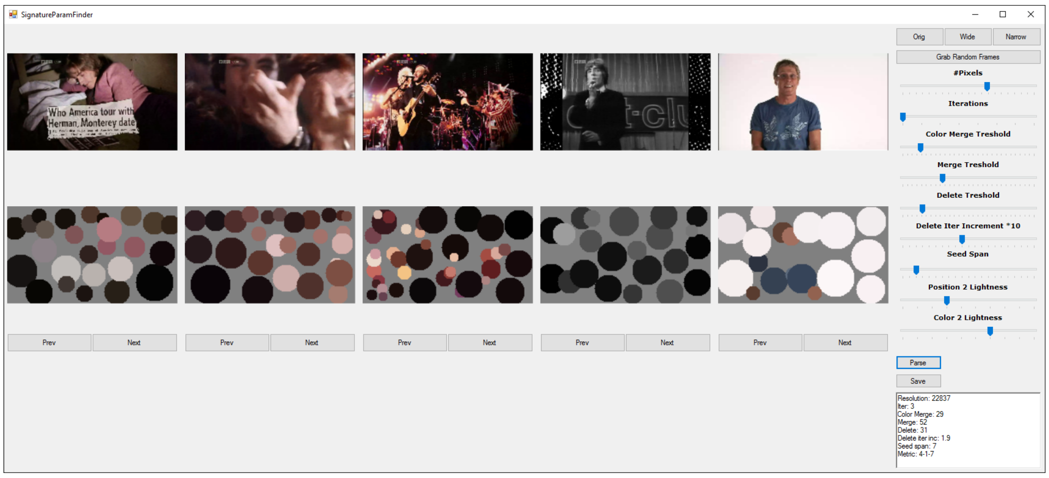
Our extraction algorithm inherits most of the properties of the vanilla k-means. Similarly to the k-means, most of the computations might be done in parallel. Nonetheless, there is one subtle yet important difference. Since we do not initialize cluster centers randomly, our extraction algorithm is deterministic and the chance of missing a color region by chance is eliminated. Furthermore, the extraction algorithm is more stable; thus, two consecutive key-frames will more likely have similar signatures.

Fine-tuning of Feature Signatures Extraction

Evaluating the performance of clustering algorithms is quite complicated. In many cases, it is subjective what is actually a cluster and what is not. Even if we are able to design a well defined measure of a clustering quality, evaluating all the possible combinations of hyper-parameters might not be computationally feasible. For these reasons, we designed a human-aided tool for finding appropriate parameters of our FS extraction algorithm. In the simple interface depicted in Figure 5.5, users are enabled to set up all the hyper-parameters and immediately observe the algorithm outputs.

Appropriate hyper-parameters might be estimated with the following workflow:

1. Load several random key-frames.
2. Adjust parameters until satisfactory results are obtained
3. Check setup robustness by examining results with adjacent key-frames (similar key-frames should be represented with similar FS)
4. Repeat steps 1,2 and 3 until a stable setup is found



```
Resolution: 22837
Iter: 3
Color Merge: 29
Merge: 52
Delete: 31
Delete iter inc: 1.9
Seed span: 7
Metric: 4-1-7
```

Figure 5.5 A screenshot of simple interface for human-aided optimization of hyper-parameters of our FS extraction algorithm.

Although the number of parameters is considerably high, in practice a reasonable setup can be reached in only couple of iterations of the proposed work-flow. The estimated setup might be saved and immediately utilized in our tool. In this way, it is possible to customize the extraction algorithm for a particular dataset if needed. Although, our default setup appeared to perform quite well on all the data we encountered, it is possible that with a specific type of data (e.g. black&white movies or cartoons) all the subsequent algorithms would perform better with a different setup.

5.1.2 Feature Signatures Video Retrieval Model

To index video files, we select roughly one key-frame for every second, resulting in the set of key-frames $\mathbb{F} = \{F_1.F_2 \dots F_N\}$. For all the key-frames $F_i \in \mathbb{F}$, feature signatures $FS_i = \{r_{ij}\}$ are extracted where r_{ij} denotes the j-th centroid of the i-th feature signature. As mentioned earlier, users are enabled to define several sketch centroids which we would like to match to the extracted feature signatures. Since users may memorize only the most distinct color regions from the searched scene, we expect only a few query centroids to be specified; hence, the model uses local instead of global matching.

For a user defined query $FS_u = \{r_{uv}\}_{v=1}^{m}$ and a centroid distance measure δ, we start with calculating distances to all the key-frames for each of the sketch centroids separately. For the sketch centroid r_{uv} the distance to the i-th key-frame is defined as:

$$dist_{uvi} = \min_{\forall r_{ij} \in FS_i} \delta(r_{ij}, r_{uv}) \tag{5.4}$$

I.e., $dist_{uvi}$ is the distance to the closest of the key-frame centroids. For a sketch centroid r_{uv} we denote the set of distances to all the key-frames as D_{uv}. Formally,

$$D_{uv} = \{dist_{uvi} \,|\, i = 1 \dots N\} \tag{5.5}$$

Now, the distances for the sketch centroid r_{uv} are scaled to [0,1] interval according to the minimal and maximal value in D_{uv}.

$$rank_{uvi} = \frac{(dist_{uvi} - \min D_{uv})}{(\max D_{uv} - \min D_{uv})} \tag{5.6}$$

Thanks to the scaling, all the centroid rankings are comparable and we might obtain the overall ranking by simply averaging them. In particular, the rank of the key-frame i is

$$rank_{ui} = \underset{\forall r_{uv} \in FS_u}{\text{avg}} \; rank_{uvi} \tag{5.7}$$

To complete our ranking model, we need to define the centroid distance measure δ. Given a user-defined centroid q and a database centroid o, $\delta(o, q)$ shall define their distance or dissimilarity. As both spatial and color spaces are suitable for L_p metrics, our first choice for δ was the regular Euclidean distance:

$$\textbf{A} \;\; \delta_a(o,q) = L_2(o,q) = \sqrt{\sum_{d \in \{x,y,L,a,b\}} (d_o - d_q)^2}$$

Nonetheless, δ_a actually ignores one part of the extracted feature signatures information – the centroid size/radius. As we fixed the sketch circles sizes [3], including r in δ_a would only favor centroids of a particular size. For these reasons, we introduce two additional distance measures:

$$\textbf{B} \;\; \delta_b(o,q) = \sqrt{\sum_{d \in \{L,a,b\}} (d_o - d_q)^2 + \max\left(0, \sqrt{\sum_{d \in \{x,y\}} (d_o - d_q)^2} - r_o\right)^2}$$

$$\textbf{C} \;\; \delta_c(o,q) = \sqrt{\sum_{d \in \{L,a,b\}} (d_o - d_q)^2 + \max\left(0, \sum_{d \in \{x,y\}} (d_o - d_q)^2 \quad - r_o^2\right)}$$

The idea is to take into account the database centroid radius r_o. The fundamental observation is that larger database centroids are effectively closer to the query centroid q. Hence, we subtract the radius r_o from the spatial part of the Euclidean distance. To elucidate the idea, consider the proposed distance functions without the color coordinates:

$$\delta'_a(o,q) = \sqrt{\sum_{d \in \{x,y\}} (d_o - d_q)^2}$$

$$\delta'_b(o,q) = \max\left(0, \sqrt{\sum_{d \in \{x,y\}} (d_o - d_q)^2} - r_o\right)$$

$$\delta'_c(o,q) = \max\left(0, \sqrt{\sum_{d \in \{x,y\}} (d_o - d_q)^2 \quad - r_o^2}\right)$$

Now, we are able to visualize at least δ'_a and δ'_b distance functions in Figure 5.6. We see that while δ'_a measures the distance between centroid centers, δ'_b cuts this distance at the database centroid boundary. As a result, larger centroids are more likely to be matched. In practice, we find that the distance measure δ_b provides results closer to user expectations; thus, we utilize it in our SBVB implementation.

To include the temporal relations derived from the video, users are enabled to draw two consecutive sketches. In such case, we rank both sketches separately and find the best matches with a sliding window [4].

[3] Specifying sketch circle sizes was rather confusing for users. In practice, users were placing multiple circles of the same color to capture large color areas.

[4] Details of the two sketch ranking are available in [9] or [7]

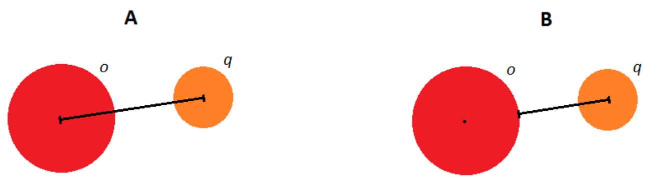


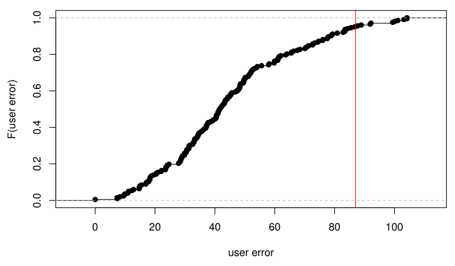
Figure 5.6 Simplified centroid distance functions δ'_a (A) and δ'_b (B).

5.1.3 Towards Higher Efficiency of FS Retrieval Model

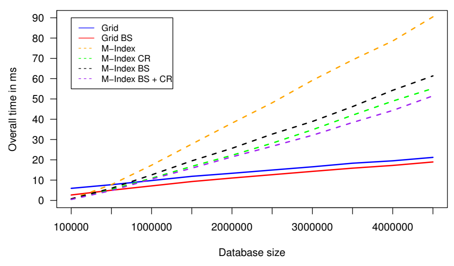
Computing the ranking from the definition would be quite costly as equation (5.4) is basically saying that all the pairwise distances between the database centroids and query centroids must be computed. In practice, computing the full ranking requires several seconds on our dataset which makes the work with the tool inconvenient.

Looking closely to users behavior, we find that the error in specifying the sketches is limited [7, 9]. More precisely, within hundreds of query centroids the distance to the searched key-frame (i.e., user error) is less than 86 in 95% of the cases (the absolute value 86 corresponds to the actual settings of our tool). The cumulative distribution function of the user error is apparently quite similar to the normal distribution (Fig. 5.7a). We conclude that the database might be subjected to range queries. Effectively, using range queries and indexing we can retrieve less than 10% of database objects without negative impact on the overall performance [9].

To select a proper index for our 5-dimensional feature space, we measured the overall time needed to process range queries with several index variants [9]. Based on the results (Fig. 5.7b), we identified a simple spatial grid with additional bounding-sphere filtering constraint to be the best choice.



(a) Cumulative distribution function of user error.



(b) Grid and M-index [58] performance under range queries.

Figure 5.7 Results of experiments suggesting that feature signatures feature space might be subjected to range queries. [7, 9].

To push the efficiency even further, we exploit the observation that the query is being build incrementally, i.e., one centroid is added, modified or deleted at a time. Next, $rank_{uvi}$, that is the ranking for key-frame i and query centroid r_{uv}, is independent on the remaining query centroids. Putting both observations together, we might define update rules for each of the add, modify and delete centroid operations (see Table 5.2).

Now, we can describe the refined algorithm for our FS video retrieval model. We cache all the partial centroid rankings $rank_{uwi}$. When a centroid r_{uv} is

Table 5.2 FS sketch ranking update rules.

Action	New query	New ranking $rank_{u'i}$
Centroid added	$FS_{u'} = FS_u \cup \{r_{uv}\}$	$\frac{\lvert FS_u\rvert rank_{ui}+rank_{uvi}}{\lvert FS'_u\rvert}$
Centroid modified	$FS_{u'} = FS_u \cup \{r_{uv}\} \setminus \{r_{uw}\}$	$\frac{\lvert FS_u\rvert rank_{ui}+rank_{uvi}-rank_{uwi}}{\lvert FS'_u\rvert}$
Centroid deleted	$FS_{u'} = FS_u \setminus \{r_{uw}\}$	$\frac{\lvert FS_u\rvert rank_{ui}-rank_{uwi}}{\lvert FS'_u\rvert}$

added/modified, we calculate the respective centroid rankings $rank_{uvi}$ and update the overall ranking according to the appropriate formula from Table 5.2. The only step requiring computing δ is calculating $rank_{uvi}$. As a result, the amortized computational cost for FS sketch ranking is independent on the number of centroids in the query.

5.2 Results Presentation

The results of FS sketch ranking are either matched key-frames or pairs of key-frames. In both cases, we present the matched key-frames in rows together with the preceding and following key-frames to ease the identification of the correct result (Fig. 5.8). Typically, around ten result rows can fit one screen while additional results might be listed with mouse scrolling. Each result row might be dragged to the left or right in order to explore a wider neighborhood of the matched key-frame.

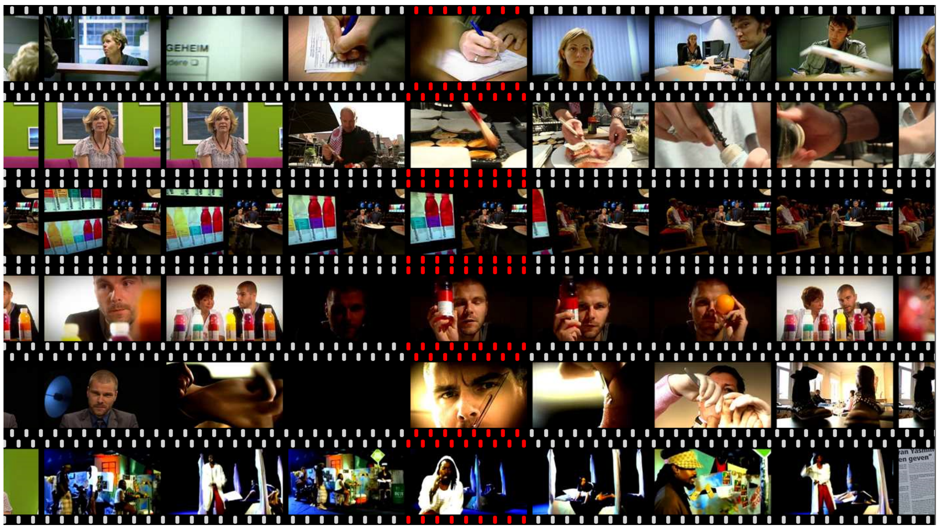


Figure 5.8 Results presentation in SBVB 2014. Matched key-frames are marked with red decorations.

5.2.1 Interactivity

Our results presentation interface is simple yet also responsive, convenient and lucid. Furthermore, we support its interactivity with several interesting features.

First noteworthy one is the option to pick up an arbitrary centroid from the presented key-frames and use it in the query sketch. We expect this to be useful in two use cases – to pick up a suitable color or to pick up several centroids from one particular key-frame and search for the key-frames similar to it.

Another feature supporting the interactivity of our tool is a browser-like history of user queries. Each modification of the query is tracked and users might navigate themselves through the history of queries with widely used undo and redo actions. With proper caching of results and images, we are able to offer instant responses which makes work with the SBVB tool more effective.

5.3 Summary

In this chapter, we presented the first release of our tool for video retrieval – Sketch-based Video Browser. The tool specializes on a convenient interface for visual KIS scenarios. Users may express their search intent in a form of simple color-position sketches. The retrieval model is based on feature signatures, a flexible image descriptor capturing distinct color regions in video key-frames. Furthermore, we presented several optimizations for the retrieval model including partial results caching and the selection of appropriate distance function. The top matching key-frames are presented together with the following and preceding key-frames (Fig. 5.8) from the video (so called the *temporal context*). With the optimized FS retrieval model, the UI might be responsive and convenient to use.

6. Enhanced Sketch-based Video Browser

Video retrieval with feature signatures sketches was proven to be effective in many scenarios, nonetheless, its filtering power is limited. Facing data collections of hundreds or thousands of hours of video, the color-position sketching would impose high demands on user precision that cannot be reached. We found users struggling with specifying the colors and spend most of the time with time demanding browsing of the top results.

To increase the filtering/ranking power, we enhanced SBVB with additional modalities. Users may select the most appropriate one or define composite queries. Each modality leverages different retrieval models and indexing techniques. The whole system scales up to hundreds of hours of video content. The Enhanced SBVB tool (ESBV) comprises:

- Edge sketching and matching with edge histograms
- Similarity search with features extracted from deep convolutional networks
- Text-based search

The enhancements did not target only the filtering/ranking part of the application. Various features supporting interactivity and easing the results browsing were introduced as well. To name a few, results are displayed in a compact manner, users are allowed to balance the number of results on the screen with the size of displayed neighborhood etc.

This chapter is organized as follows. In Sections 6.1, 6.2 and 6.3, the additional search modalities are presented. The enhanced multi-modal video retrieval model is introduced in Sections 6.4 and 6.5 and the filtering options of ESBVB are then summarized in Section 6.6. The improvements of the results browsing part of ESBVB are described in Section 6.7 and finally, the overall user interface is briefly discussed in Section 6.8.

6.1 Edge-based Features

Occurrence and characteristics of edges, corners or similar edge-based features are often utilized in image-related problems. Edges, i.e., step changes in the intensity of adjacent pixels can be effectively detected with simple convolution filters such as Sobel [83]. Filtering images with these filters may be understood as calculating their derivatives (Fig. 6.1), or smoothed derivatives in Sobel's case. With an additional processing, edge features may provide image segmentation, information about certain shapes present in the image and in the center of our interest – features suitable for similarity search and image retrieval.

At the same time, humans are highly sensible to edges. Even those with the lack of drawing skills are able to capture the most distinct edges such as horizon in landscape images or similar straight lines. With simple free sketch tools, we can obtain fair approximates of the searched image/scene in the edge domain (see Figure 6.2).

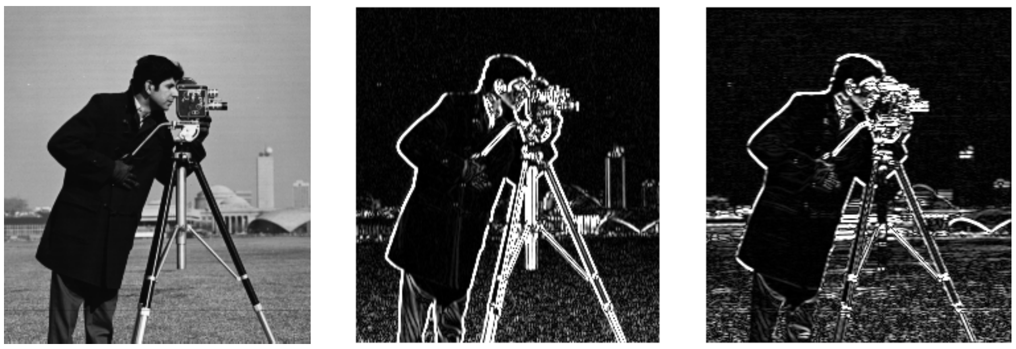


Figure 6.1 An image and its derivation in x (horizontal) and y (vertical) axis.

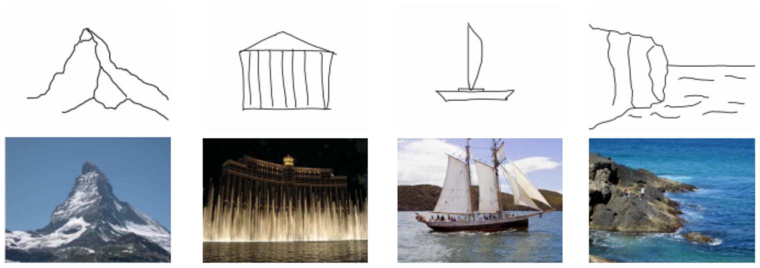


Figure 6.2 Example images and the corresponding free hand sketches. Images courtesy of [20].

Now, the challenging part is to design an appropriate similarity measure between the sketches and detected edges in images. Attractive yet unfeasible way would be to measure the similarity directly in the pixel domain [6]. Alternatively, we can perform local matching [23] which searches for similar local patterns. Both the similarity measure and edge features representation matters [28].

All these approaches, however, are being tested on small to medium image databases of well captured objects divided into several categories. Our situation is quite different. Our dabase comprises millions of very heterogeneous key-frame images while many of them capture complex scenes with many objects present or blurry transitions and various graphics. We expect the edge sketching retrieval scenario to be applicable only in certain cases where edges are the dominant feature in the scene (see Fig. 6.3).

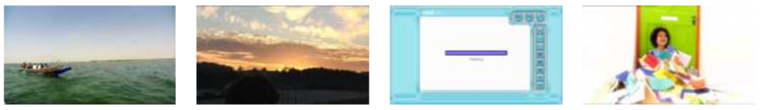


Figure 6.3 Example key-frames with distinct edge-features that can be easily sketched.

For these reasons, we need to utilize edge-based features that are robust and capture the overall composition of images rather than subtle details. Edge histograms [1] appeared to satisfy these demands. The main idea is to sum the detected edges in all the desirable directions (horizontal, vertical etc.) over predefined set of regions [2]. In particular, we followed [62] where the summation

[1]Also known as histogram of oriented gradients (HOG) features [15].

[2]Similarly to local color histograms.

is done on multiple levels which allows to capture global as well as local edge information.

In practice, our edge histogram features are 120 dimensional integer valued vectors. More precisely, the images are divided in several regions in which we sum horizontal, vertical and diagonal gradients. For each region and direction, a bin in the histogram is reserved. Furthermore, gradients in adjacent regions are summed and stored in additional bins and finally the global sum of gradients is added to the histogram. I.e., gradients on local, semi-local and global levels are captured. The similarity between two edge histograms can be defined as regular L_1 metric. Regarding user sketches, edges and edge histogram features are computed in the same manner as for the key-frame images.

In practise we find that the utilized edge histograms are effective especially when few dominant edge lines are sketched, yet not suitable for detailed drawings. This is compliant with previously defined goals. Furthermore, edge sketching can be combined with current color-position sketching as is described in Sub-section 6.4.1. Some example edge sketches together with the respective best matches are depicted in Figure 6.4.

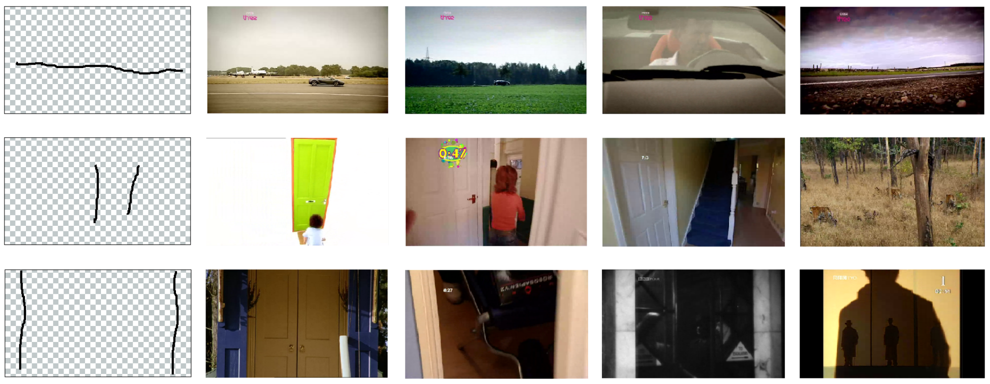


Figure 6.4 Example edge sketches (left) and the respective key-frame matches.

6.1.1 Similarity Search with Edge Histograms

Although the dimensionality of edge histograms is mediocre and the utilized metric function is not very expensive, evaluating distances between the defined sketch and all the dataset images is not feasible. Empirically, for a dataset of million images it took several seconds to compute all the distances on a commodity hardware.

Is the computation of all the distances necessary? In an example query depicted in Fig. 6.5 we can see that most of the results are rather dissimilar to the edge sketch. Furthermore, we display only top k results as users are able to examine only limited number of results. Therefore, we might either approximate or skip computing distances to the images beyond k-th position.

In similar situations, the approximate k nearest neighbors (k-NN) search is the method of choice. The idea is to cheaply compute the set of k most promising objects only to which the actual distances are calculated. For our purposes, we selected 1-level pivot indexing. Firstly, N pivots are randomly selected from the dataset. Secondly, all the database objects are associated with the nearest pivot. After that, a k-NN search is done in the following steps:

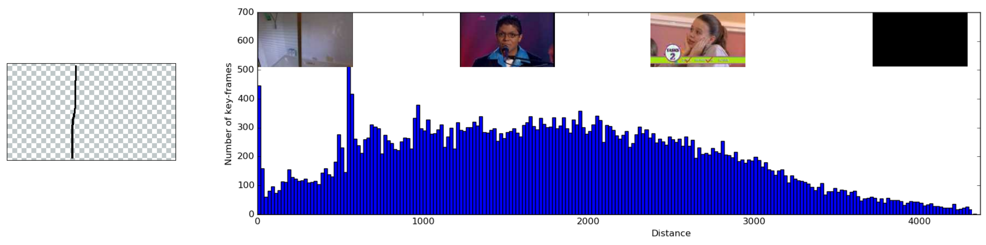


Figure 6.5 Query edge sketch (left) and histogram of distances of a subset of our dataset. Images are aligned according to x axis, i.e. The leftmost image is the best match while the rightmost is one of the worst.

1. Calculate distances from the query object to all the pivots
2. Sort the pivots by the distances in ascending order
3. Return database objects associated with the pivots until at least $c > k$ candidate objects are retrieved
4. Calculate distances to the candidate objects and return actual k-NN

Where parameters c and k are selected empirically. With large c, we would achieve better precision paid with higher computational cost. As users may examine only limited number of results, we may argue that reasonable value for k is around 1000.

Alternatively, we can view the algorithm as the Voronoi partitioning of the space and examining the Voronoi cells according to the distances between the cell centers (pivots) and the query object.

With this approximate search schema, we efficiently avoid computing the majority of the distances. The number of pivots and k can be estimated empirically for particular applications. Note that we scale the computed distances to [0,1] interval for later use in our system.

6.2 Similarity Search with DeCAF

Aside from the fact that DCNNs can provide accurate model for image classification, they have a number of exciting properties. The most interesting for us are the Deep Convolutional Activation Features (DeCAF) [19]. DeCAF, as the name indicates, is the activation of the fully connected layers during feed-forward pass through the network. Experiments suggest that DeCAF might be used as a substitute for SIFT or SURF descriptors in general, but more importantly, a simple L_2 metric already represents an acceptable similarity measure. Apparently, the non-linear transformation in higher DCNN layers produces good features often immune to changes in angle, rotation, illumination and other transformations.

The crucial characteristic of DeCAFs is that they generalize well to datasets other than the training one. In particular, DeCAFs outputted from network trained for ILSVRC provide suitable retrieval performance also on keyframes from our dataset. Similarly to [59], we incorporate DeCAF image retrieval as our search by example model.

Note that DeCAFs happen to be quite high dimensional (e.g. 4096) which imposes high computational and memory demands. It was shown that DeCAF features might be binarized without significant loss in precision. Furthermore, an arbitrary algorithm for dimensionality reduction such as PCA might be employed [1]. As a proof of concept, in our system we work with DeCAFs in their raw form, while the dimension reduction/binarization techniques are the subject of our future work.

Similarly to the edge histogram case, it is not tractable to compute all the distances between query image and all the database images. As the dimensionality is higher, we need to retrieve even smaller portion or rather a fixed number of objects to keep the computation fast enough for real-time responses. In this case, we picked up the state-of-the-art method for indexing high dimensional metric spaces – M-Index [57]. M-Index provides effective and efficient processing of approximate k-NN queries and can be tailored to our system.

Query-by-example DeCAF similarity search might be available side by side with all the other searching options. The details how we combine the searching modalities will be provided in section 6.4.2. We present some example queries and the corresponding best matches in Figure 6.6.

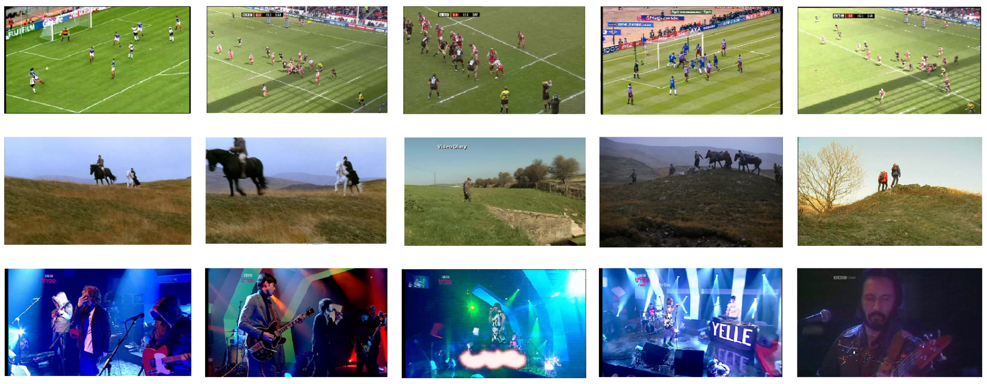


Figure 6.6 Example of query-by-example searches with DeCAFs. Queries (leftmost image) and the corresponding best matches.

6.3 Querying with Natural Language

As previously stated, we assume the searched scene to be known either visually or by textual description. The later, naturally, posses a greater challenge to our system as users may struggle with specifying visual clues via sketching canvas. The textual description might be misleading or simply lack distinct visual clues (colors, edges). Take for example a simple scene described with sentence *Man wearing a red t-shirt is filmed sitting inside his car.* What shade of red is it? Is the scene taken from outside or inside of the car? What color is the car? How would you sketch it? As a matter of fact, we observed that users dealing with textual KIS often prefer simple browsing of the database content rather than the filtering part of our system. Furthermore, it was shown that simple sequential storyboard search [32] might perform well on textual tasks (for 100 hours of video). Next, once a visually similar result is obtained, query by example search (e.g. with DeCAF features) often lead to success.

Given a textual description of the searched scene, it is difficult to initialize the search in a content-based way. If the textual description does not contain any clues for color or edge/shape, but only a concept label, the keyword search becomes the preferred filtering choice. Even if the search item is known visually, it might be easier, especially for novice users, to describe it textually. For these reasons we focus also on a textual known-item search task initialization.

We compare two orthogonal approaches to find an initial query object using keyword search. The first approach employs an external image retrieval system designed for effective keyword search (e.g., Google Images). In such system, users can usually materialize their ideas and use some of the returned image to initialize similarity search. The problem of this approach is that the first idea of the user does not have to correspond (in the terms of similarity) with the contents of the dataset. For this reason, the second approach is dataset-oriented, where automatic annotations are created for the searched video. Unlike other state-of-the-art approaches, our approach is not restricted to speech transcripts only [84] nor do they require manual pre-annotation [44]. Given the annotations, user-defined keyword query is matched directly to the contents of the dataset. Such an approach has been used in several known-item search tools [52]. We enhance this approach with text processing and exploration of semantic relations between words. Since the user is part of the search process, only a comparative user study can decide which of the two approaches is more effective given specific retrieval tasks.

6.3.1 External Image Search Engine

The first approach we investigate is to use the text query in an external keyword-based image search engine to get sample images and use these to initialize the search. Such an engine can be, for example, well-established Google Images search. However, this approach has two significant limitations. First, a sufficiently fast Internet connection is necessary and the engine has to be available. Furthermore, as the selected image has to be preprocessed for the video retrieval system, all the feature extraction techniques (or services) have to be available for instant usage. Second, the selected image does not have to correspond to the content of the searched scene, while the materialized image may be distracting for the search.

Our system can be extended easily just by providing an additional text search field in the application. When user issues a textual query, the application downloads top k images from the external image retrieval server. The images are displayed to the user and the user selects a candidate image for search. DeCAF descriptors are then extracted and used to initialize similarity search browsing. Expected use case is depicted in Figure 6.7.

6.3.2 ImageNet Labels

In the second approach investigated in this work, we consider a search method based on labels automatically assigned to key-frames using arbitrary ImageNet [69] classification model; in our case, Deep Neural Network [79]. While the model provides highly specific labels from the ImageNet set of 1000 labels, users tend to form more elaborate yet generic queries or even whole sentences. For example, we

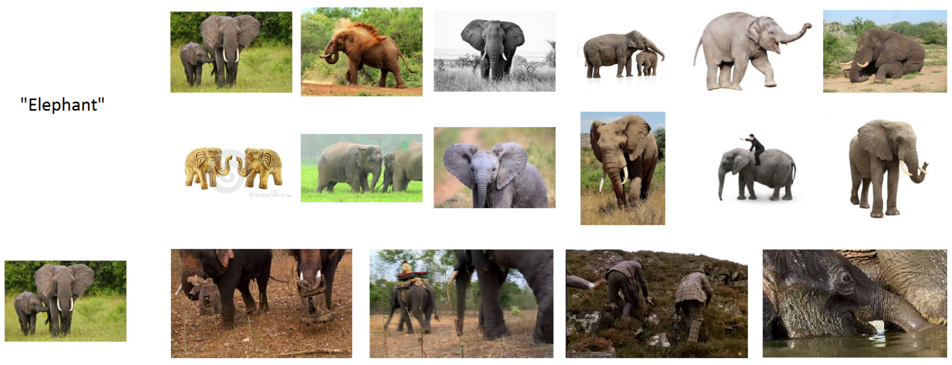


Figure 6.7 Example search scenario utilizing external search engine to initialize search. We start with textual query "Elephant" issued to Google Images Search Engine. Retrieved images (first and second rows) are displayed to the user. Consequently, one of the image is selected for DeCAF similarity search (last row).

would like to match a key-frame labeled as 'golden retriever' with a query 'A dog is playing with a ball'. To interconnect the worlds of ImageNet labels and user queries, we introduce two different processing pipelines: key-frames labeling and query processing and matching.

Key-frames labeling.

During the video preprocessing, the top five labels for each key-frame are extracted together with their probabilities. Every ImageNet label is in fact a synset (a group of elements, which are semantically equivalent) from the WordNet [21] lexical database. Of the five extracted labels, a tree of their hypernyms is built, such as the one in Fig. 6.8. The nodes represent synsets and edges represent WordNet hypernym–hyponym relations between them. a probability of a newly inferred synset is a sum of probabilities of its children. Using this tree, we can deduce probability values even for labels not available in the ImageNet labels set. Both extracted and inferred labels and their probabilities are stored in an inverted index for efficient retrieval.

Query processing and matching.

The processing of a user query follows a simple procedure. First, stop-words (such as 'a', 'been' or 'your') are removed for higher efficiency while the remaining words are transformed into their basic forms. Since only nouns (objects) are present in ImageNet, all other parts of speech are also removed. Consequently, the noun's meanings are explored using WordNet and a set of synsets is assigned to each of the nouns, which usually contains several different meanings of the word (e.g. a noun 'horse' might be an animal as well as a gymnastic equipment).

Now, we iteratively generalize each query synset (exchange it for its hypernym) until it is present in the database. For example, a query synset 'horse' (not present) might be exchanged after few iterations for synset 'animal' (present). This way, each query synset would yield some results — in the worst case, we would search for the most general synset – 'entity' which is present in every key-frame. In

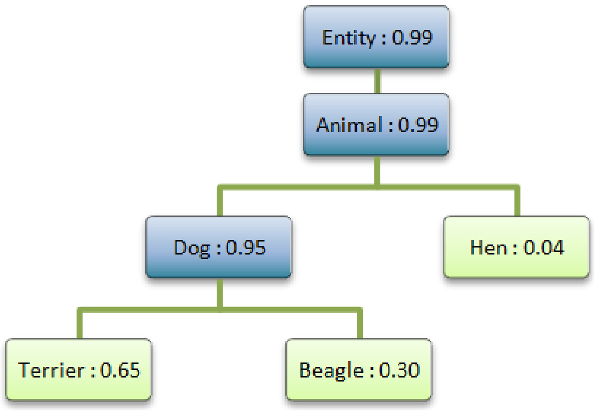


Figure 6.8 A synset tree of one particular key-frame. Each node represents one synset together with the probability of its presence in the key-frame. The tree includes both the synsets extracted by the model (green) and the inferred ones (blue).

other words, we are going to search for the most specific synsets we can find in the dataset.

For each query synset, we rank all the key-frames with the probabilities of occurrence of the synset in the key-frames. The rankings of query synset s are weighted by $1 - avg_p(s)$ where $avg_p(s)$ stands for average probability of synset s over all the dataset key-frames. In other words, we prefer more specific synsets (rarely present) over generic synsets, which have a weaker filtering power. This scheme follows the same idea as the term frequency weighing technique [70] utilized in general textual search.

Example queries together with the respective top matching results are depicted in Fig. 6.9. We might get impression that caffe

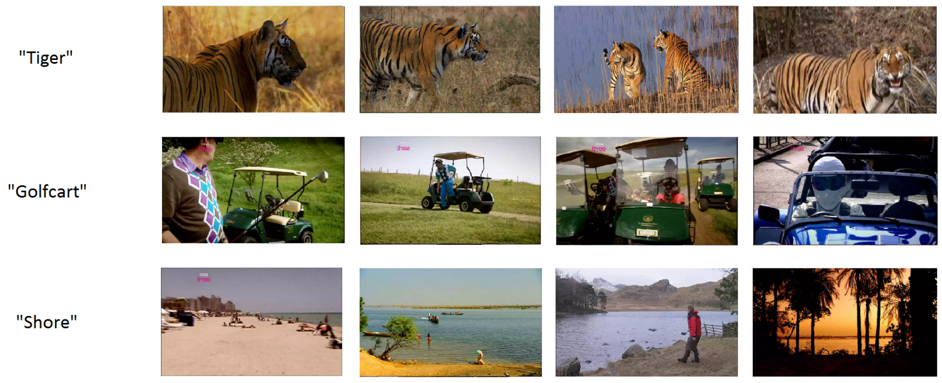


Figure 6.9 Example of textual queries processed with ImageNet labels search.

User feedback loop.

Since all the word meanings are extracted from the query, it is vital for the user to be able to overlook and control the actual searched labels. We provide a checklist of them and only a subset might be selected. The effects of this procedure affect the results immediately. Experiments reveal that this provides a highly used and convenient way to further specify the search, now based on information actually contained in the dataset.

6.4 Combining Modalities

In the previous sections, we incorporated several additional modalities (edge sketches, DeCAF similarity search etc.) yet we did not clarify how they are combined. We introduce a simple yet effective model for handling queries composited from multiple modalities. First, we describe our multi-modal sketching canvas comprising both the original feature signatures and novel edges drawing. Second, we provide a weighing scheme for overall ranking when a composite query is issued.

6.4.1 Multi-modal Sketches

Although feature signature and edge sketches may seem quite different, nothing actually prevents us from drawing them both in one canvas. We reserve the left mouse button for placing and manipulating centroids while pressing the right mouse button and dragging the mouse over canvas would draw the appropriate edge curve. Both the centroids and edge curves might be moved, deleted or copied as the user wishes. An example story comprising several common sketch manipulation is depicted in 6.10.

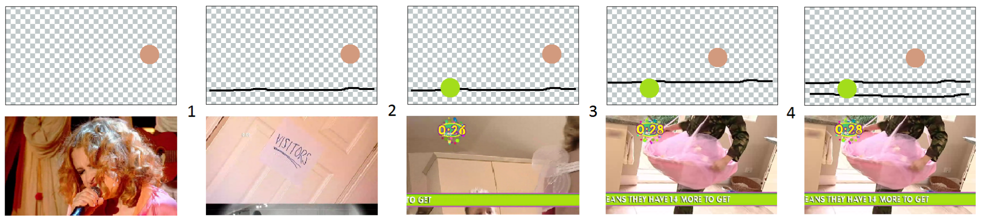


Figure 6.10 Multi-modal sketch time-lapse capturing some of the common sketch modifications. The best matching key-frames at every step are displayed under the respective sketches. The time-lapse starts with a sketch containing one centroid and 4 additional steps follow.

1. A horizontal edge line is drawn
2. A green centroid is picked up from displayed results
3. Both the pink centroid and edge line are moved
4. An extra edge line is drawn

6.4.2 Composite Query Ranking

Although each of the modalities may be sufficient to effectively filter the database content, the combination of modalities is expected to be even more powerful. Thus, we allow user to define composite queries comprising of multiple modalities, e.g., combined feature signature and edge sketches or feature signature sketches accompanied with textual descriptions. Users may select both the modalities and their importance ranging from 1 to 5. The modality with higher importance is going to have higher impact on the results ordering.

Our overall ranking scheme for composite queries is basically a weighted average over the partial rankings. We define the ranking scheme as follows. To rank database key-frames F_i with a query comprising modalities M, each modality $m \in M$ with importance $I_m \in \{1, 2, .., 5\}$. We define the modality weight as

$$w_m = \frac{I_m}{\sum_{m \in M} I_m} \tag{6.1}$$

For each modality m and each key-frame i we expect the ranking $r_{im} \in [0, 1]$, 0 being the worst match while 1 is the best. Then, the overall ranking for the key-frame i is defined as

$$r_i = \sum_{m \in M} w_m r_{im} \tag{6.2}$$

Note that only a subset of key-frames is actually ranked by each modality. For key-frames not ranked by a modality m we define $r_{im} = 0$, i.e. The worst match. Consequently, the key-frames shall be sorted in ascending order according to their overall rankings r_i. This schema can be recognized as a late fusion as opposed to an early fusion, where the modalities are combined in the feature space and only one ranking is calculated. It was shown that late fusion models perform slightly better [81].

As defining such composite query might be difficult, we aid novice users and automatically set modality importance to zero for all but the most recently modified modality. In this way, only one modality is used at a time. Once users get used to the behaviour of our system, they may decide to turn this feature off and explore the filtering power of composite queries.

Several examples of composite queries are depicted in Figure 6.11.

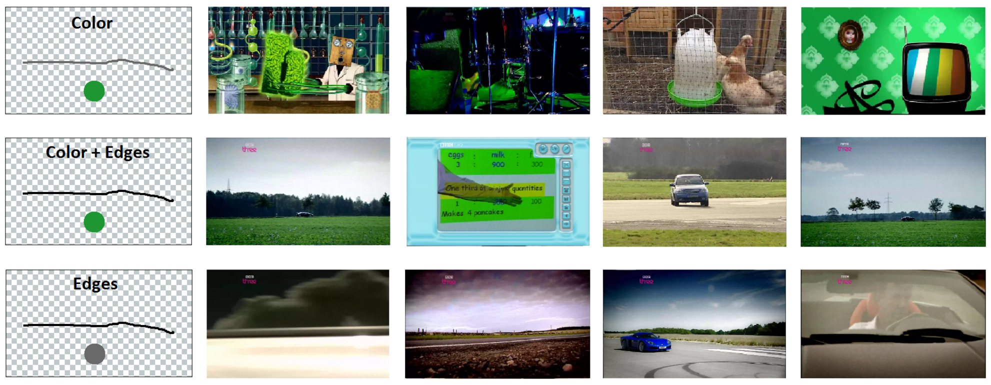


Figure 6.11 A multi-modal sketch drawn when attempting to retrieve a landscape scene capturing grass and sky. While both the feature signatures (top) and edge histograms (bottom) models fail to retrieve the scene, their combination (middle) achieves the desired result.

6.5 Sketch Sequences

From the very beginning, SBVB contained an option to specify two consecutive sketches. In this case, the search engine would search for video segments matching to the drawn sketches in the given order at most 20 seconds apart. Although this option significantly raises the filtering power, the time constraint of 20 seconds seemed not restrictive enough. More precisely, users typically sketched two subsequent scenes while the system found two matches up to 20 seconds apart. For this reason, we introduced an interval slider allowing users to specify the time interval in which the matched key-frames must appear.

Note that nothing forces us to stay with matching of just two consecutive sketches. We would be able to specify and match a sequence of sketches of arbitrary length. Nonetheless, we did not implement such option for two reasons. Firstly, specifying several sketches would be complicated and time consuming. Secondly, we find memorizing more than two scenes quite challenging.

6.6 Summary of Searching Features

While the original tool SBVB offered only feature signature sketches for searching the video content, ESBVB includes several other modalities. Namely, ESBVB now supports searching with:

- Feature Signatures Sketches
- Edge Sketches
- Example Image (DeCAF similarity search)
- Text Queries

Users may define a composite query with any of the mentioned modalities. To rank the key-frames, the modalities are combined proportionally to the user defined modality importances. Appropriate indexing techniques are employed to retain convenient user experience even when dealing with hundreds of hours of video content. As a result, even the slightest adjustments of the query might be immediately reflected in the results. This indeed interactive experience supports ESBVB effectiveness and allows users to intuitively embrace the behavior of the search engine.

The newly incorporated features and searching options increase the retrieval power of our system, though, the original functionality is preserved. Novice users may stick with single-modal queries while experienced users may select an appropriate combination of modalities for each task.

6.7 Enhancing Results Browsing

One can discriminate two main components of our system – content searching and results browsing. We already know that the searching component received number of enhancements and the same actually applies to the browsing component. The most important enhancements might be summarized as follows:

- Scene detection and compacting
- Interactive Navigation Summary [74]
- Coarse to fine results presentation
- Additional video and scene filtering

All of the enhancements are motivated by the user feedback from early versions of SBVB and we include the motivation in the discussion. None of the enhancements is crucial yet they provide very effective solutions in certain situations.

6.7.1 Scene detection and compacting

The automated scene detection in a video is a well defined problem with a variety of algorithms available [36, 37]. Most of them share the same idea of detecting the magnitude of change in consecutive frames and placing a scene cut if this measure exceeds a defined threshold. We follow this idea in our system where the magnitude of change is calculated as the distance of the DeCAF features of the respective frames. The threshold was estimated empirically to fit our dataset (significantly different dataset may require re-estimation).

The detected scenes or scene cuts find several applications in almost every video retrieval tool. Commonly, each scene is being represented by only one of its frames which drastically reduces the number of key-frames to be indexed. Furthermore, only the selected representative key-frame might be displayed within the results. Despite the attractivity of this dataset reduction, we do not follow this procedure for several reasons. Firstly, it might be complicated to select the most suitable representative for each scene. Secondly, a mistake in scene detection might make some of the video segments virtually non-retrievable. Thus, we retain the original uniform sampling of key-frames which might seem redundant in some cases, nonetheless, it ensures the highly desired robustness for our system.

Nonetheless, we utilize the detected scene in two different ways.

Scene-based results filtering. Dense key-frames sampling induces a high redundancy of results, therefore, we are filtering the results prior to displaying them. The original algorithm for removing redundant results [9] resulted in a set where *no results were too close to each other*. With detected scene cuts we might easily extend this algorithm so that we obtain set of results where *no results were too close to each other or within the same scene*. In this way, we remove results that are already quite far apart yet still basically points to the same scene and displaying both would confuse users.

Scene compaction. Furthermore, we utilize the detected scenes for a compact results presentation. Originally, each matched key-frame (result) was displayed with the following and preceding key-frames. Now, key-frames are compacted into scenes, each having only one key-frame fully visible, while the others are cropped and arranged into pseudo key-frames (see Figure 6.12). As users hover over the results the fully visible key-frame moves which allows to examine the scene in full detail. Note that the temporal information, i.e. The length of the scene, is also preserved. This technique saves a significant amount of space on the row. As a consequence, we might display more temporal context for each of the matched key-frames.

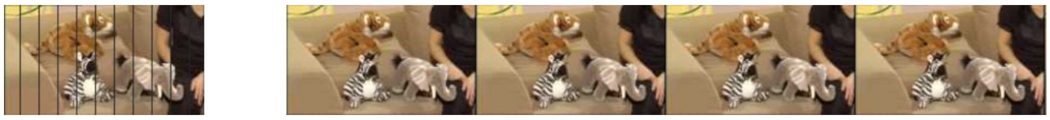


Figure 6.12 A scene displayed as a sequence of key-frames (right) and the same scene displayed compactly (left).

6.7.2 Coarse to Fine Results Presentation

The way of presenting the results was rather simplistic so far. Each of the matched key-frames was presented together with its context on one row. This limits the number of results per page to units based on the available screen resolution. We find this simple presentation not convenient, especially when users intent to explore the dataset rather than identify the correct result among well matching content. We address this issue with option to specify the number of result columns displayed. Users may balance between the size of the results temporal context and the number of results per page. The original scheme is one extrema while results without contextual information yet high density is the other (captured in Figure 6.13).

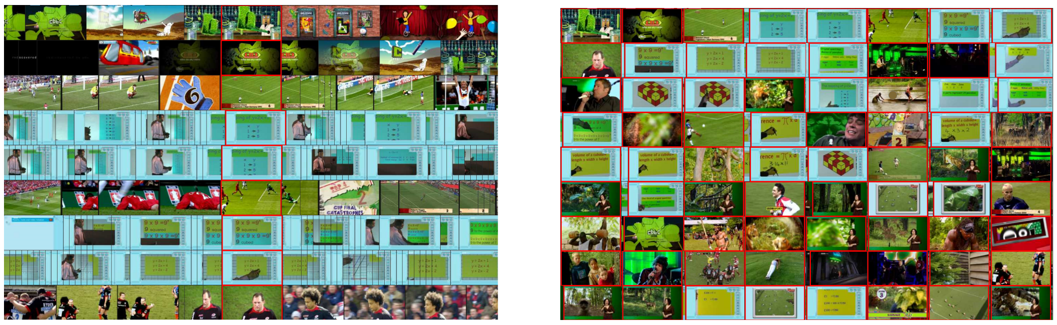


Figure 6.13 Two different setups for results displaying – large results context (left) vs. low results context (right).

We find both views quite useful. The typical scenario is to begin without the temporal context and once promising results are found, the view is set up so that the contextual information allows to discriminate the correct result.

6.7.3 Interactive Navigation Summary

Through our experience with SBVB we occasionally encountered a situation where we found the video containing the searched scene yet not the scene itself. In these situations, moving the result row to actually browse the video is indeed cumbersome. When browsing a single video, the random access with a regular seeker is often preferred. In order to deal with these situations, we included similar functionality to our system.

One row, actually the bottom one, is enriched with Interactive Navigation Summary which works similarly to the well-known seeker bar. Additionally, the content of the video being browsed is summarized and visualized. Authors of Interactive Navigation Summary proposed several features that might be summarized, nonetheless, we stick with the most straightforward one – dominant colors of the key-frames.

The top-5 dominant colors, calculated during the pre-processing of each keyframe, are displayed at appropriate positions in the seeker bar, as can be seen in Figure 6.14. The Interactive Navigation Summary provides top-level information on the video content and enables focusing on the promising parts only. On top of that, we visualize the positions of the matched key-frames with triangular pointers at the respective positions to guide users to the areas with clusters of matches potentially containing the searched scene.

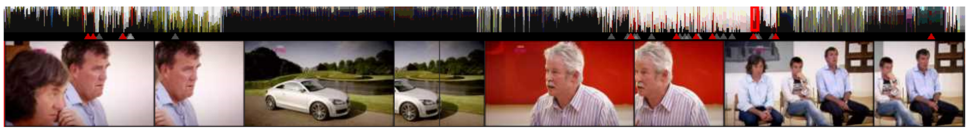


Figure 6.14 Interactive navigation summary (top) serves as regular seeker bar with additional information. In this case, we display 5 most dominant colors for each of the key-frames.

6.7.4 Additional Video and Scene Filtering

As the database grows, searching becomes more difficult and even minor improvements and lesser filtering options might make a difference between a successful and unsuccessful search. We are introducing variety of features, each solving a particular situation and slightly reducing the amount of content needed to be examined. None of the filtering features is actually essential and we expect them to be used by rather experienced users in certain specific scenarios. It is assumed that the database consist of multiple video files.

Video Exclusion. When searching for a certain scene, users may immediately identify certain results as irrelevant (e.g. cartoon scenes). We would like to exclude not only the scene but actually the whole video from the search. ESBVB includes this option.

Scene Exclusion. Similarly, considerable difficulties are induced when a certain repetitive scene constantly appears among the top results (e.g. TV news studio). Excluding such scene from results would significantly elucidate the browsing of the results. Now, we allow users to do so. The list of excluded scenes is visibly maintained and any result that is close to any of the excluded scenes (measured by DeCAF similarity) is not displayed.

Video Focus. In some cases, a scene different from the searched one yet clearly from the same video may be encountered first. In this case, we would like to continue with the search only within this video and we know which one is it. Users are allowed to focus the further search to only one video, i.e., exclude all the videos but the selected one.

Scene Marking. a scene visually similar to the searched one happens to be retrieved during the search process. Unfortunately, it is particularly hard and time consuming to discriminate these similar scenes. Any scene might be marked so that when retrieved again with a different query it can be skipped right away. We display the marked scenes in gray-scale.

6.7.5 Thumbnails Caching

With datasets of hundreds of hours of video content both filtering and displaying results become a challenge. We are utilizing a variety of indexes, approximations and heuristics to keep the ranking fast enough to retain real-time responses. All for nothing, however, if the displaying of the key-frame thumbnails takes a long time. Imagine a dataset of 250 hours of video. Dense key-frames sampling produces nearly 1 million key-frames for which we need to have thumbnails available. That is more than 5 Gigabytes of approx. 170 times 90 pixels large images encoded in JPG and roughly 8 times more (40 Gigabytes) when stored as bitmaps. Clearly unfeasible to be maintained in the main memory of a common notebook or PC.

To keep the user experience as convenient as possible we implemented 3-level thumbnails caching. The levels are the following:

1. Blocks of B consecutive images encoded as JPGs stored on the hard-drive
2. Blocks of B consecutive images encoded as JPGs stored in the main memory
3. Blocks of D consecutive bitmaps stored in the main memory

where B is divisible by D [3]. Each time an image shall be shown, a request dispatched by the topmost cache layer is made. If the respective block is not available it is requested from the underlying layer and so on. To better utilize resources, all the processing is done in parallel.

The question is when and how the cache shall be cleared. We achieved satisfactory results with the following schema (Least Recently Used):

- Keep track of the most recently accessed blocks
- Set up the maximal number of blocks to be kept
- If the maximal number of blocks is reached, clear (release) the least recently used ones

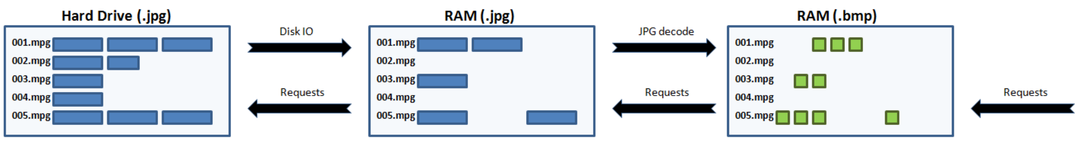


Figure 6.15 3-level image caching scheme implemented in ESBVB.

This schema (depicted in Fig. 6.15) might be implemented with a bidirectional list and a hashmap. The hashmap serves for accessing the blocks while accessed

[3]In our implementation, we set B=D.

blocks are moved to the front of the list. The blocks to be released can be found at the end of the list. Each level of the cache may act independently regarding its maximal number of blocks etc.

How does such system behaves? With properly set up cache parameters we obtain responsive GUI and keep the memory footprint low. We may enhance the system with additional heuristics such as when D-block is requested, we request also its preceding and following blocks (H1) or request blocks needed for the following results page in advance (H2). All together, we might observe the following desirable properties:

- The images are loaded and displayed independently, thus, the cached ones are displayed immediately and the rest once they become available.
- Single adjustment of a query would yield similar results, some of them with cached images.
- Exploring a result context is smooth as the bitmap images are pre-cached thanks to H1.
- Results page listing is responsive thanks to H2
- Resubmitting a recently submitted query would yield results with cached images.

6.8 User Interface

Even the best algorithms, models and filtering techniques can not be effective without a convenient and responsive user interface (UI). It is because users still play the major role in the whole process and we expect this statement to be true for the following years if not decades. We believe that UI requires at least the same amount of attention as does the back-end system.

It is therefore surprising that our UI did not receive significant changes despite more than three years of development. Out of many development versions, we present those with which we participated at VBS through years 2014–2016 (Figs. 6.16, 6.17 and 6.18). We might see that the overall schema remained almost the same. The UI consists of two parts – the querying interface and results area, the later occupying the majority of the screen. We try to keep all the UI features as lucid as possible. The querying interface contains two sketching canvases (since 2014), an interval slider for specifying the time interval between the sketches (since 2015), sliders for specifying the modality importance (since 2015) and a textual query interface (since 2016).

Regarding the results area, the major improvements are the extra detail row (bottom) equipped with the Interactive Navigation Summary and the compact scene presentation (Section 6.7.1) both introduced in the 2015 version. Additionally to the 2014 version, each result row might be explored without limitations, i.e., the whole video might be examined through one result.

The up to date version is captured in Figure 6.19 wherein we label all the UI components. Furthermore, we include a user guide for the ESBVB tool in a form of videos in the attached DVD.

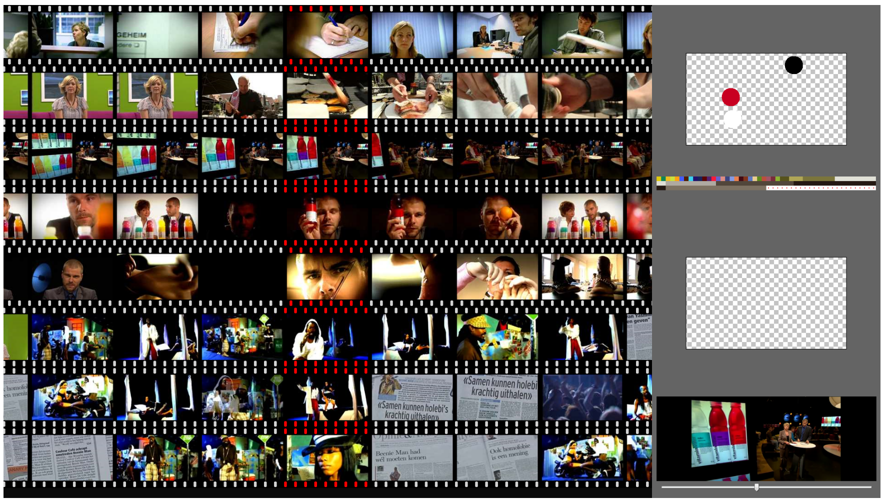


Figure 6.16 Screenshot of SBVB @ VBS 2014.

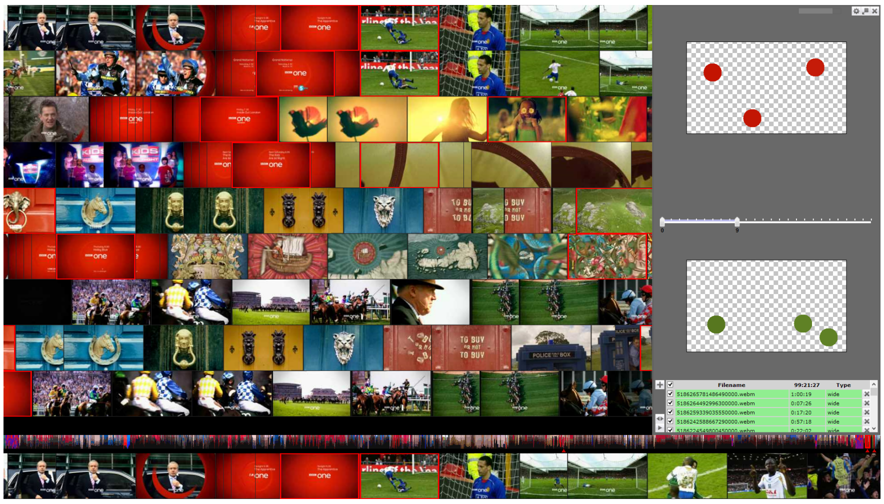


Figure 6.17 Screenshot of ESBVB @ VBS 2015.

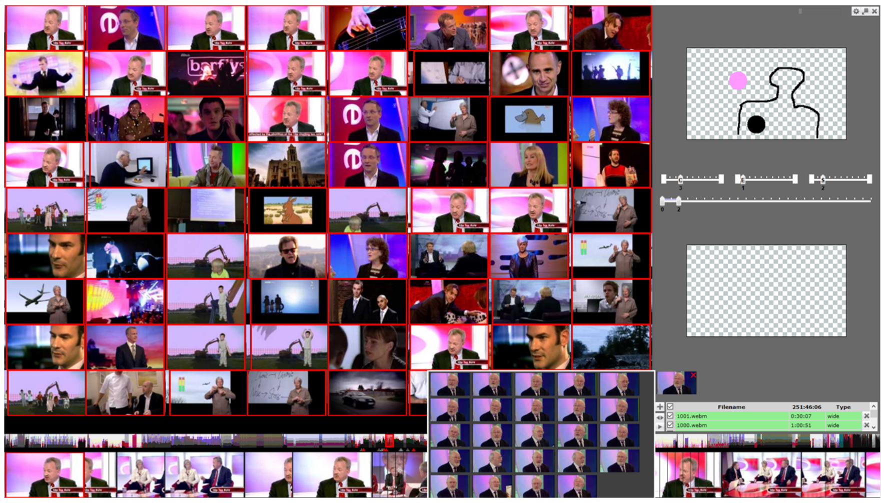
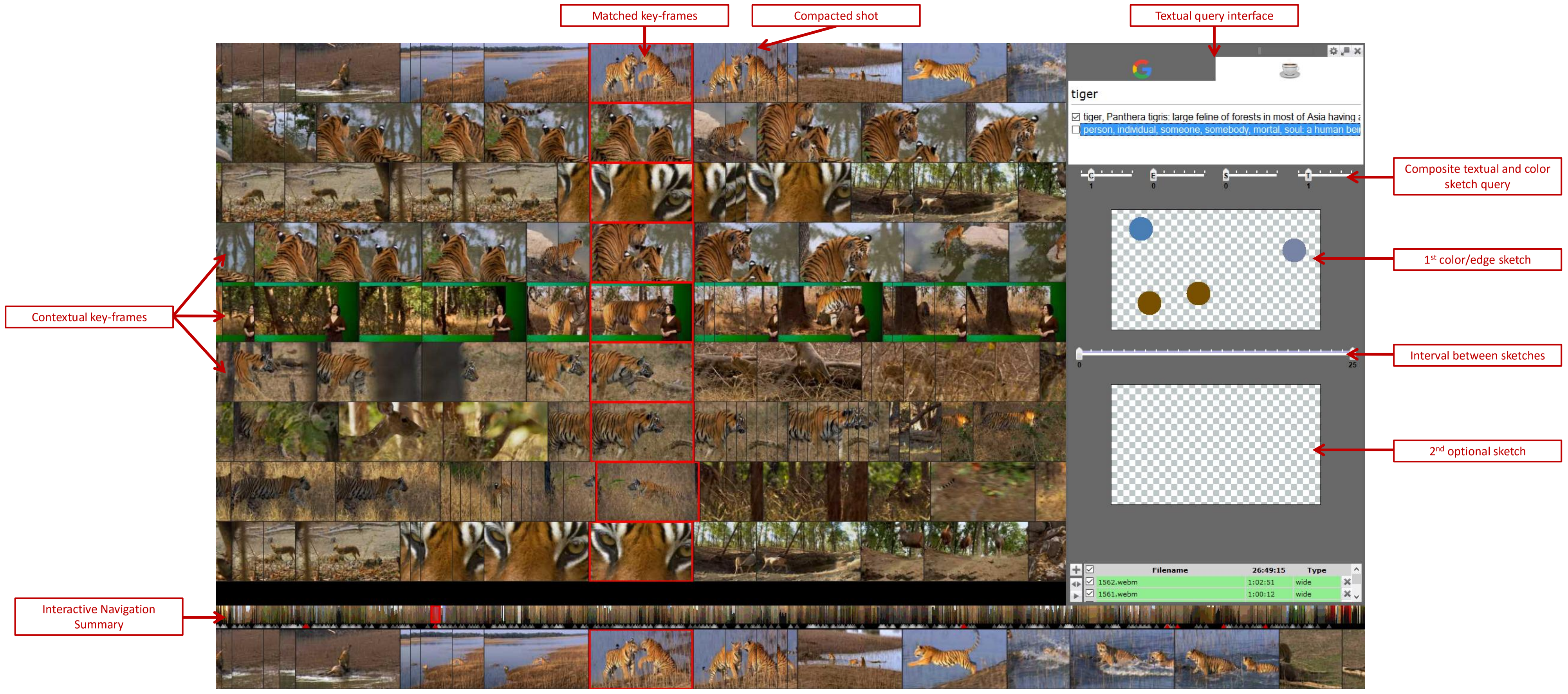


Figure 6.18 Screenshot of ESBVB @ VBS 2016.



tiger, Panthera tigris: large feline of forests in most of Asia having a

person, individual, someone, somebody, mortal, soul: a human bei

Filename	26:49:15	Type
1562.webm	1:02:51	wide
1561.webm	1:00:12	wide

Figure 6.19 Overall screenshot of up to date ESBVB with additional labels pointing out noteworthy UI features.

7. Implementation Architecture

In this chapter, we describe the overall architecture of our implementation of ESBVB. We are going to provide rather a top-level overview; nonetheless, we also occasionally delve deep to discuss several noteworthy details.

The implementation itself is available in the attached DVD. It is implemented in C# programming language and thus requires the .NET runtime environment. The application utilizes multiple 3rd party libraries (available under 2-clause or 3-clause BSD license) in a form of dynamically linked libraries compiled for 64bit Windows operating system. The most important libraries are Open Computer Vision library [10] and Caffee deep learning framework [35]. Furthermore, we include the application sources distributed as MS Visual Studio 2015 project available at the attached DVD.

Through this chapter, we reference the actual C# classes defined in the source code. The class names are highlighted with the `verbatim` font.

7.1 Components

The overall architecture is portrayed in Figure 7.1 wherein 6 top-level components are discriminated.

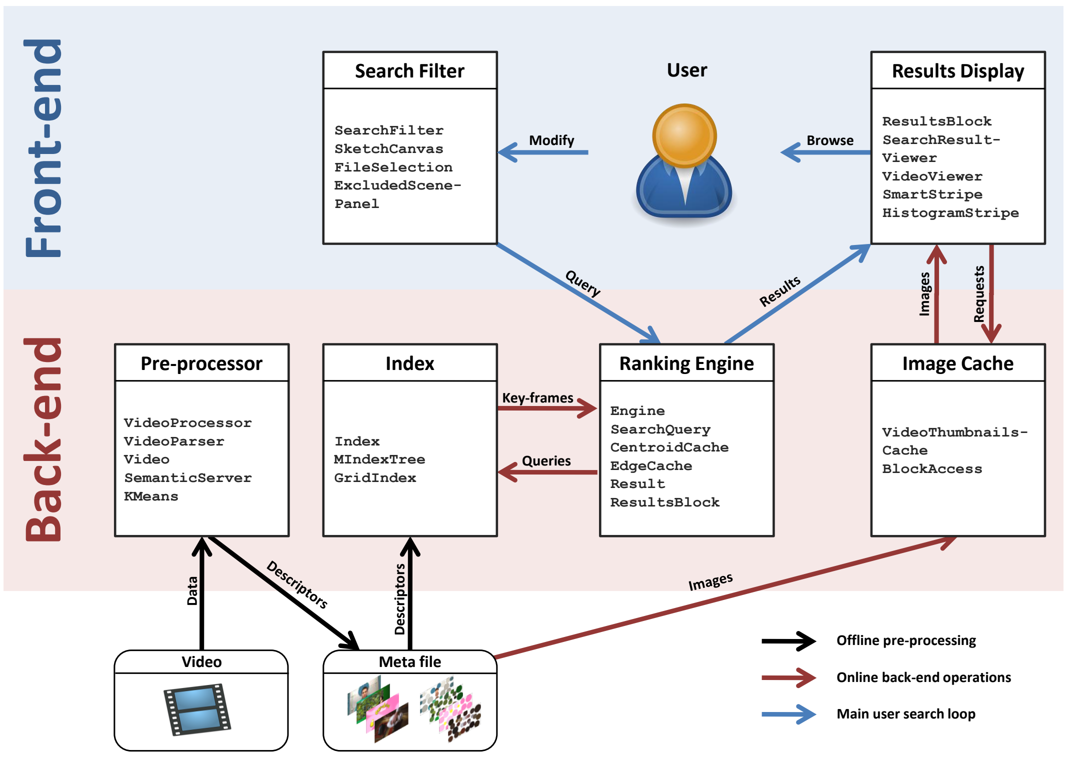


Figure 7.1 Overal (E)SBVB architecture. Application components are depicted in boxes containing the list of important class names. The communication between components is depicted with arrows.

Pre-processor analyzes video key-frames and stores descriptors and key-frame thumbnails to meta file.

Index builds auxiliary data structures for efficient processing of descriptor range and k-nn queries.

Search Filter allows users to specify their search intent.

Ranking Engine ranks the database content given a query from the Search Filter.

Results Display allows users to browse the results provided by the Ranking Engine.

Image Cache accelerates access to video key-frame thumbnails. (Section 6.7.5)

Search Filter and Results Display components contain UI features; hence, we identify them as parts of the application front-end. The remaining components form the back-end of the application.

User interactions might be described as the loop between query specification and results browsing (captured with blue arrows in Figure 7.1). To facilitate the search loop Ranking Engine ranks the database key-frames every time a query is issued. As the ranking is calculated on-line, high demands on the efficiency are imposed. Hence, the query is processed by the Index component. Similarly, loading of images is accelerated by the Image Cache component.

A video file has to be pre-processed prior to the actual search. This is done in the Pre-processor component where all the needed descriptors and key-frame thumbnails are extracted and stored in a meta-file. Afterwards, the descriptors might be rapidly loaded and indexed to enable the content-based retrieval of the respective video file.

In the following sections, we describe the application components in more detail.

7.2 Back-end

When a video is opened, we identify whether its meta data files containing key-frame thumbnails and the respective descriptors are available. If not, the video must be pre-processed, otherwise the descriptors are loaded to a continuously maintained in-memory Index. Hereby, the video becomes accessible using our video retrieval model. When a user issues a query, the Ranking Engine ranks the video key-frames that are retrieved from the Index (e.g. with FS sketch query, the engine calculates the ranking according to Section 5.1.2). Ranked key-frames are consequently passed to the Results Display.

We might identify three main roles of the back-end – video pre-processing, indexing and query ranking, each of which is discussed in the following sub-sections dedicated to the respective components.

7.2.1 Video Pre-processor

In this step, ESBVB extracts all the image/video descriptors and stores them in meta files located next to the video files. This is perhaps the most computationally demanding operation performed by ESBVB; nonetheless it is done only once. At this moment, the Pre-processor extracts the following items for each key-frame:

1. Thumbnail image
2. FS descriptor (Section 5.1)
3. Edge histogram (Section 6.1)
4. Top-5 dominant colors (Section 6.7.3)
5. DeCAF descriptor (Section 6.2)
6. ImageNet labels (Section 6.3.2)

The extraction of DeCAF descriptors and ImageNet labels require running a DCNN model and is the most computationally expensive item of our list. A simple server-like program written in C++ is available for this purpose. It communicates through the standard input and output, accepting filenames on its input and responding with either DeCAF, ImageNet labels or both. ESBVB runs one or more instances of this program and manages the communication in the `SemanticServer` class.

The extraction itself is done within the `VideoParser` class which requires an instance of `VideoProvider` providing the video key-frames and an instance of `VideoParsingListener` which stores the extracted descriptors. In our case, it is an instance of the `Video` class and the extracted meta data are stored in files located next to the video file. The extraction process is implemented as the 1 Producer – N Consumers design pattern, where the producer produces video key-frames and consumers are extracting the descriptors in concurrent threads. The video pre-processing schema is depicted in Figure 7.2.

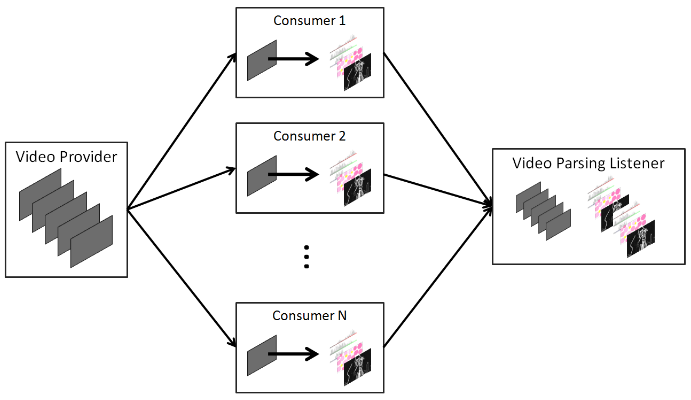


Figure 7.2 Schema of video pre-processing.

7.2.2 Video Index

To enable efficient similarity search, the descriptors have to be properly indexed. The Index component, in particular the `Index` class, provides an interface for k-nn and range queries described in Chapters 5 and 6. An instance of the `Video` class might be added or removed from the `Index` at any time. Internally, it holds multiple instances of image descriptor indexes, each serving a portion of videos. When a query is issued, it queries the underlying partial indexes and returns the aggregated results (Schema for FS indexes is depicted in Fig. 7.3). The descriptors and indexes implemented in ESBVB at this moment are summarized in Table 7.1.

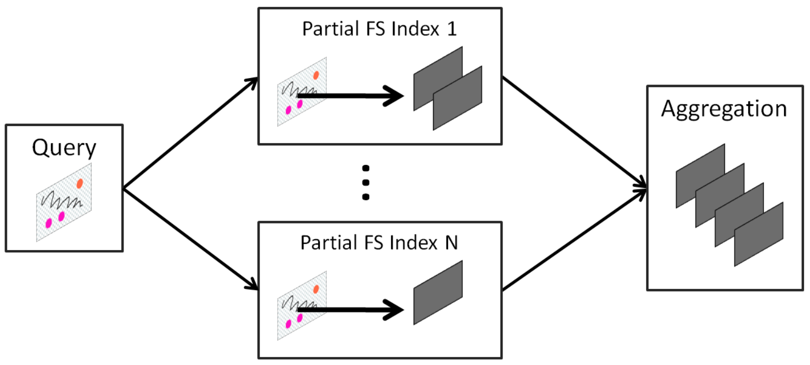


Figure 7.3 Schema of querying the FS index.

Table 7.1 A summary of utilized image descriptors and respective indexes, type of queries and implementation classes.

Descriptor	**Index**	**Query**	**Class**
Feature Signatures	Grid	**range**	`GridIndex`
Edge Histogram	1-level pivoting	**k-nn**	`List`
DeCAF	M-Index	**k-nn**	`MIndexTree`
ImageNet labels	Inverted index	**full**	`Dictionary`

7.2.3 Query Ranking Engine

The query ranking follows a schema similar to the indexing. The labor is divided between several threads, each ranks a portion of videos. The partial rankings are afterwards aggregated, scaled, modalities are combined and the results are merged and sorted. The whole process, captured in Figure 7.4, is done in the Engine component. The final rankings are passed to the front-end part of the application.

All the computations are quite straightforward except the results merging. The goal is to select only one result (the best one) from each cluster of results. For this reason, we maintain two data structures – an ordered binary tree of results rankings and lists of results in the original temporal order. Now, we iteratively pop the top result from the tree, examine its neighborhood through the lists and remove neighbor results from both structures until the tree is empty.

7.3 Front-end

The front-end part of ESBVB provides the interface for multi-modal query specification and displays the results obtained from the back-end. As all the components

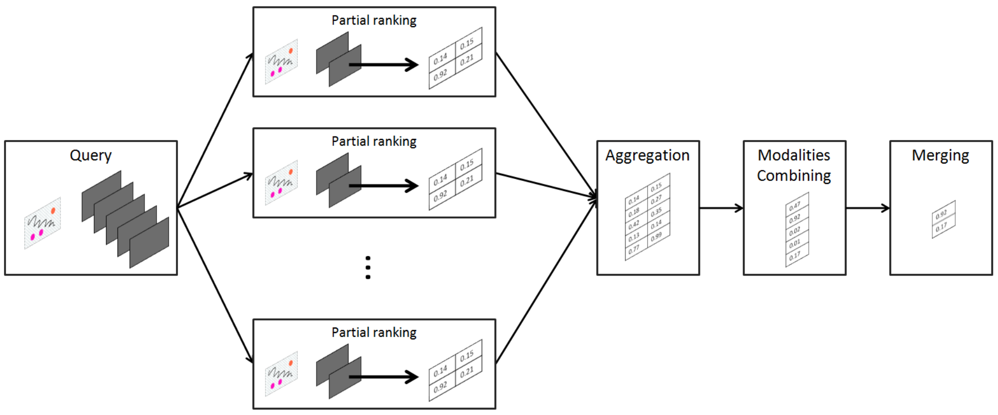


Figure 7.4 Video ranking scheme of a multi-modal query.

are actually visible to users, we start with locating them in a screen-shot of ESBVB. Afterwards, we discuss the functionality of each component separately.

The overall scheme of the front-end components hierarchy is captured in Figure 7.5 where the Results Display and Search Filter components correspond to those in Figure 7.1. All the top-level components gather the user input and may produce additional events which are captured and handled by their parent components. The components are stand-alone, i.e., they might be re-used at a different locations or even in another application.

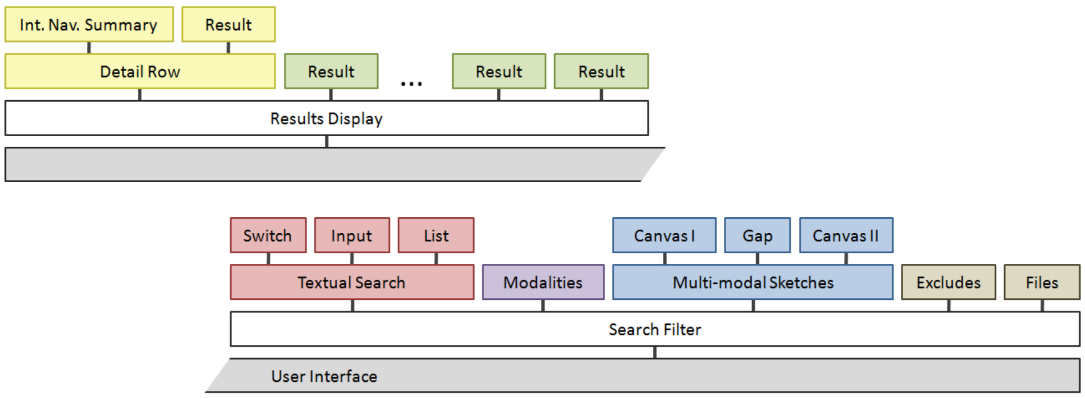


Figure 7.5 Scheme of the front-end components of ESBVB. The scheme is cut in half and depicted in two rows.

The majority of the UI area is occupied with the results presenting component (**Results Display**). The remaining area is dedicated to the **Search Filter** and its components with which users are defining the query or filtering conditions (class `SearchFilter`). The components are highlighted in Figure 7.6 with colors corresponding to Figure 7.5. The functionality of all the components are described in the following sections.

7.3.1 Search Filter

In the **Search Filter**, users define their search intent, i.e., form a query. The **Search Filter** contains several components including two sketching canvases

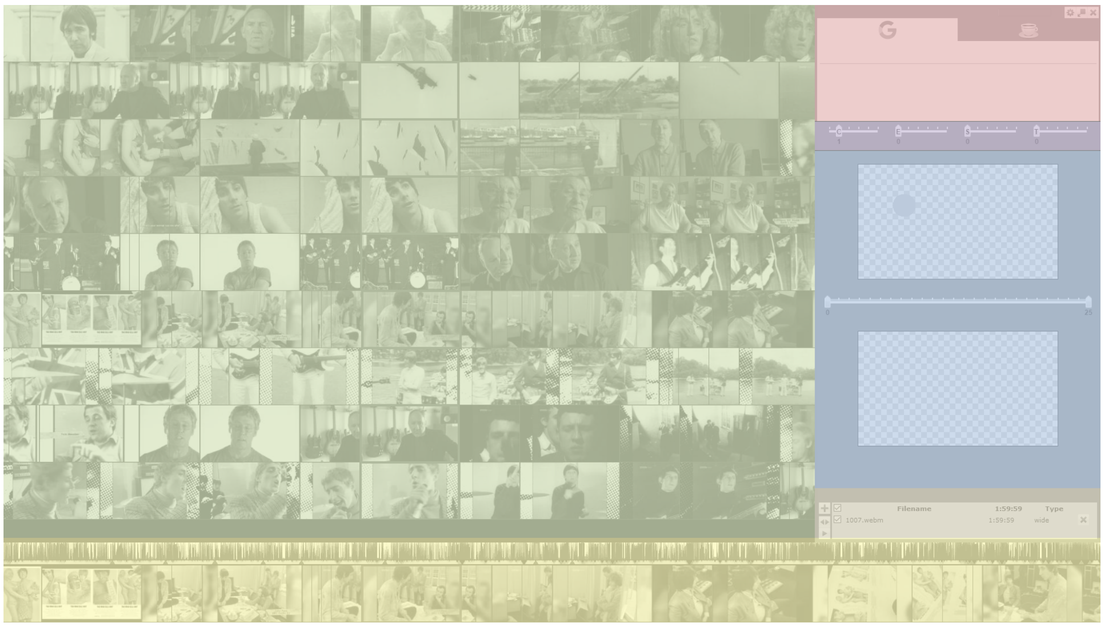


Figure 7.6 ESBVB front-end components. Colors are compliant with Figure 7.5.

which are characteristic for ESBVB. The components and their functionalities are enumerated in the following list:

With the **Textual Search** component a query in natural language (english) might be issued. Users may select either the Google or ImageNet labels search scenario (Section 6.3). An input field for the query itself is available together with the list of actually searched terms.

Modalities are combined as users define in this component. Four sliders (FS, edges, DeCAF and text) define modality importances (Section 6.4.2).

Multi-modal sketches are the key-stones of ESBVB. We offer two multi-modal drawing canvases allowing to sketch the searched scene with color-position centroids and edge lines. Furthermore, a two sketch query might be issued and the time interval between the sketched scene can be defined as well.

The **Excludes** component holds and visualizes all the scenes excluded from the search (Section 6.7.4). Additionally, users may examine which results are not displayed just to be sure that the searched scene is not among them.

The **Files** component manages the video files and allows a video-level filtering. Next to the list of currently opened videos, there are buttons for adding videos and playing a random segment.

7.3.2 Results Display

All the back-end computations are carried out in a background thread keeping the user interface responsive. Every modification of the query launches the new ranking and terminates the old one. Each time a ranking computation is finished, the front-end displays the results.

Result is a component displaying one result, i.e., a matched key-frame or a pair of key-frames. By default, each **Result** occupies one row and displays key-frames following and preceding the matched one. Any **Result** might be dragged to the left or right in order to examine wider neighborhood of the matched key-frame. This functionality is turned off if users set up to display multiple results per row.

Detail Row component is located at the very bottom of the screen. It composes of one regular result row and an interactive navigation summary (Section 6.7.3) functioning as well known seeker bars.

8. Performance Evaluation

Evaluating the performance of any KIS video retrieval tool is a challenging task, as users are the significant part of the search process. The performance is influenced by many factors and there is no rigorous measurement or a single valued quality indicator. It is therefore extremely important that workshops such as VBS or TRECVID are challenging teams worldwide to develop solutions for various video retrieval related tasks. These events undergo continuous development just as do the participating tools. Datasets are growing, complexity increasing and all this pushes the state-of-the-art further.

We report our results with respect to other state-of-the-art tools for video retrieval from VBS 2014, 2015 an 2016. The conditions, tasks and the datasets are different for each year; thus, we report the results separately. Nonetheless, the goals, task definitions and the scoring system remained the same through years.

Each participating team develops a tool for KIS in video. The video data are distributed several months before the event, leaving enough time for arbitrary pre-processing. The event is organized as a competition where each team tries to realize KIS of two types:

Visual KIS A short clip 10 to 20 seconds long is played once. The goal is to identify the video from which the clip originates as well as the exact position of the clip in the respective video. Video recording is prohibited.

Textual KIS Only a textual description of a clip in the collection is given. The goal is the same as in the Visual KIS.

The tasks are presented on site and teams receive points for each successfully solved task. The time for solving the tasks is limited to 3 to 6 minutes per task. To determine whether a team solved the task correctly, a submission to the evaluation server must be send and false submissions are penalized. Point rewards are defined with the following formula:

$$\frac{100 - 50\frac{t}{T_{max}}}{\max\left(1, s-1\right)} \tag{8.1}$$

Where t is the time of the correct submission, T_{max} is the time limit for this task and s is the number of submissions. In other words, the goal is to locate the clip as quickly as possible without making any mistakes.

During the session, the current scores are screened and the session itself is accessible for public which makes VBS indeed interactive, entertaining and exciting event. In fact, a part of the session includes novice users from the audience.

8.1 VBS 2014

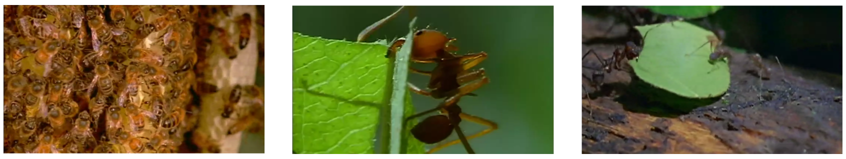
We entered the field of video retrieval via participation at VBS 2014 with our tool SBVB. The dataset comprised 26 hours of various video content provided by BBC including TV shows, TV News broadcast and documents in both wide-screen and 4:3 format. The session was divided in 6 task categories summarized in Table 8.1. In 2014, the following 7 teams participated: DCU [78], KLU [12],

Table 8.1 VBS 2014 task categories.

Category	**Scope**	**User**	**KIS**	**Tasks**	**Limit**
Single Visual	single video	expert	Visual	10	3 min.
Single Textual	single video	expert	Textual	10	3 min.
Archive Visual	video archive	expert	Visual	7	6 min.
Archive Textual	video archive	expert	Textual	7	6 min.
Single Visual II	single video	expert	Visual	10	3 min.
Single Visual Novice	single video	novice	Visual	10	3 min.

JRS [2], CERTH [51], SIRET (**us**) [46], NII-UIT [55] and AAU (out of the competition) [17].

For single video tasks, the particular video file was announced just before the task started. 3 out of 7 teams including us were able to participate in an optional ad-hoc task category where additional dataset was distributed only 1 hour in advance. In total, 54 KIS tasks were carried allowing to score up to 5400 points. An example of textual task is depicted in Figure 8.1.



"First a close-up of a beehive with many bees,
then close-up shots of ants cutting and
carrying large green leaves."

Figure 8.1 One of the textual KIS tasks from VBS 2014. Participants were given the textual description (bottom), three selected key-frames of the actual clip (top) were not revealed until the end of the task.

8.1.1 VBS 2014 Results

The results are captured in Figure 8.2 where we might see that we won by a significant margin (472 points to be precise), NII-UIT team ended up 2nd and JRS 3rd. The portion of points obtained by the teams for each category are summarized in Figure 8.3.

Out of 27 expert visual KIS tasks, we correctly submitted 26 within the time limit. In particular, we achieved a remarkable result in the first single visual category where we were able to obtain 999 out of 1000 points– in the majority of cases we found the target segment even before its playback ended. Regarding the expert textual tasks, we correctly submitted 16 out of 17 tasks and finally, a novice user was able to solve 7 out of 10 visual tasks in the single video task category with SBVB.

It comes as no surprise that the archive categories pose a greater challenge to all the teams. On the other hand, we would expect the Textual KISs to be significantly harder than Visual KISs; nonetheless, the results on the VBS dataset do not support this hypothesis, given the competition settings. The amount of

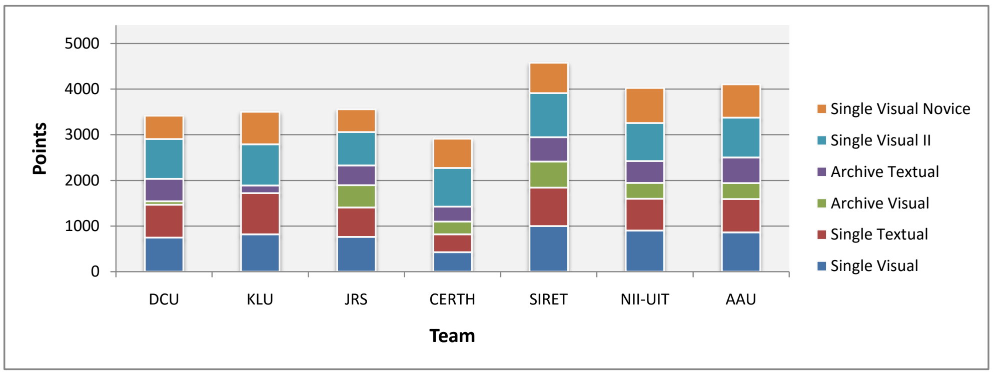


Figure 8.2 Total points acquired by the teams participating on VBS 2014.

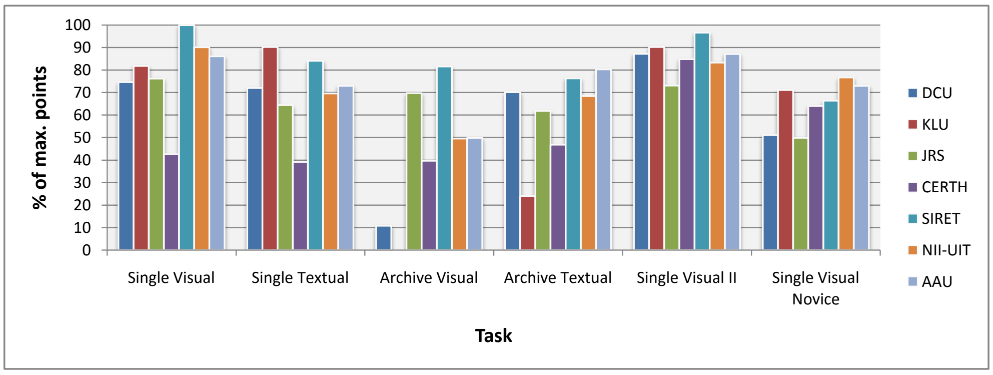


Figure 8.3 Percentage of maximal possible points for each task category acquired by the teams participating on VBS 2014.

points obtained by all the teams were roughly the same for comparable Visual and Textual categories.

Clearly, the degree of user expertise has noticeable impact on the performance as in the novice category the teams achieved on average 25% points less than in Single Visual II category. A noteworthy exception to this is NII-UIT team achieving only 8% less (our SBVB experienced loss of 31%). The novices had only little time to familiarize with UIs; thus, we argue that intuitiveness of UIs played an important role here.

At VBS 2014, we demonstrated that a tool effectively utilizing a single descriptor based solely on color distribution can outperform other state-of-the-art methods on given datasets. We confirmed the strong role of color based descriptors in specific KIS video retrieval scenarios. Actually, no other descriptors are necessary for the scope of tens hours of BBC video content.

8.2 VBS 2015

The dataset for VBS 2015 comprised approx. 100 hours of video content, i.e., 4 times more than the dataset for VBS 2014. The content was again diverse mixture of TV programmes. In the light of the results from VBS 2014, organizers did not include single video tasks. The categories, numbers of tasks etc. are summarized in Table 8.2. Total of 9 teams participated: DCU [88], IMOTION [63], KLU-TJ [29], NGVB [13], NII-UIT [56], SIRET (**us**) [8], HTW [4], UU [31] and VERGE [53].

8.2.1 VBS 2015 Results

The points obtained by each team are depicted in Figure 8.4, where we can see that only 6 out of 9 teams actually scored some points. It was due to various technical problems; nonetheless, it is perhaps the significantly larger dataset that induces a number of technical challenges. Although we won VBS 2016, it is no longer clear victory. In fact, 2nd team IMOTION obtained only 10 points less which is no more than 1% of our points. Team UU ended up 3rd loosing 58 points to us. The portion of max. points obtained by the teams are summarized in Figure 8.5.

Contrary to other years, it was allowed to capture the screen during the visual tasks. This rule was unfortunately announced only about a month prior to the competition; thus, most of the teams including us could not exploit this option.

Table 8.2 VBS 2015 task categories.

Category	**Scope**	**User**	**KIS**	**Tasks**	**Limit**
Archive Visual	video archive	expert	Visual	9	6 min.
Archive Textual	video archive	expert	Textual	6	6 min.
Archive Novice	video archive	novice	Visual/Textual	6	6 min.

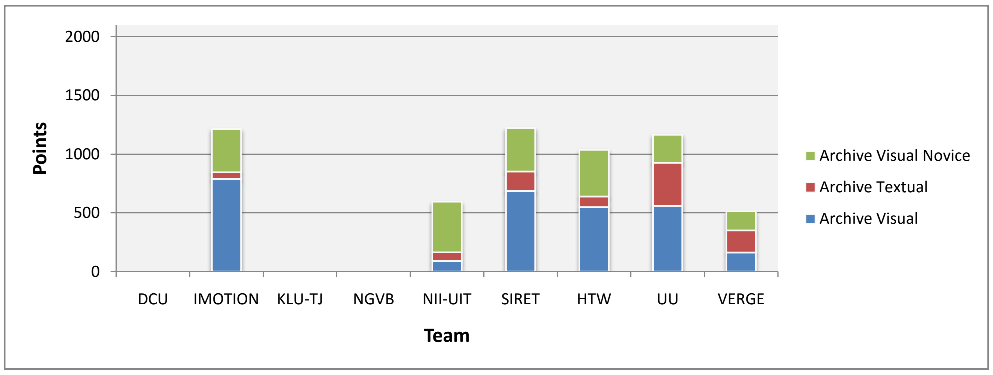


Figure 8.4 Total points acquired by the teams participating on VBS 2015.

The top 3 teams performed equivalently well yet each quite differently. The IMOTION dominated the expert visual category. It was perhaps with the help of screen capturing that IMOTION team managed to utilize properly. Even more distinct outlier can be found in the expert textual category. The UU team obtained more than two times more points than any other team.

As a mater of fact, UU team was the biggest surprise of VBS 2015. Their tool was nothing more than storyboard browser. During the search time, users simply sequentially scanned the key-frames (one key-frame per second) with no additional filtering being utilized. Effective use of human computation achieved the same results or even outperformed other state-of-the-art techniques.

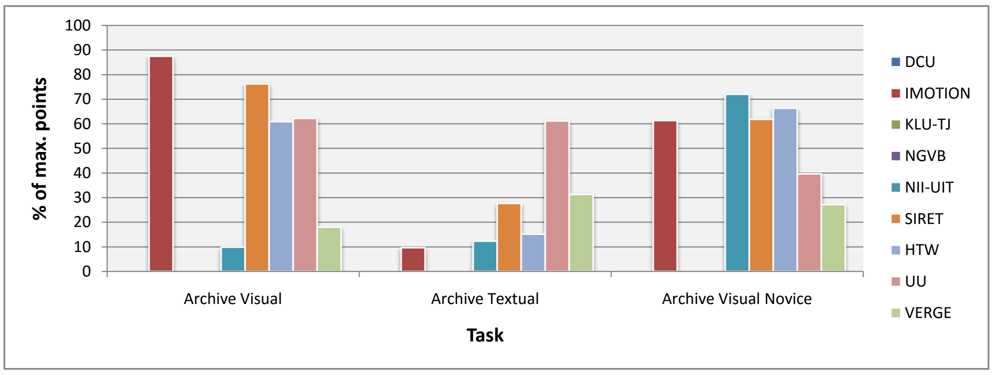


Figure 8.5 Percentage of maximal possible points for each task category acquired by the teams participating on VBS 2015.

We may argue that the expert memorized majority of the database and could skip certain videos knowing that they do not contain the searched clip; nonetheless, this more or less applies to the other teams as well. Furthermore, even UU novices scored reasonable amount of points.

Our tool showed stable performance over all the categories and in the end, scored the most points; however, very small margin suggest that other approaches are viable as well. As the dataset gets larger we no longer see convincing results and no clear winner arise from VBS 2015.

In-depth analysis of the results from VBS 2015 is available in [14].

8.3 VBS 2016

It seems that the dataset size follows the Moore's law, i.e. it doubles every year. More precisely, the dataset for VBS 2016 comprised 250 hours of video content, that is almost 10 times more than in 2014. Although the session retained novice tasks, they are no longer included in the final scoring. The task categories of VBS 2016 are summarized in Table 8.3. Total of 9 teams participated: IMOTION [64], iAutoMotion [65], JRS [3], VERGE [54], KLU-UU [30], SIRET (**us**) [39], DCU [89], HTW [5] and UoS [25]. The iAutoMotion team was completely autonomous tool and included optical character recognition algorithms to deal with the textual tasks.

Table 8.3 VBS 2016 task categories.

Category	**Scope**	**User**	**KIS**	**Tasks**	**Limit**
Archive Visual	video archive	expert	Visual	10	6 min.
Archive Textual	video archive	expert	Textual	10	6 min.

8.3.1 VBS 2016 Results

The resulting points obtained by the participating teams are depicted in Figure 8.6. The scale of the y-axis was intentionally set to the maximal number of points that could be acquired. It is clearly visible that with larger dataset, the tasks became noticeably more challenging. The competition was won by HTW team, followed by KLU-UU team with loss of 42 points. Our tool ended up 3rd with 55 points loss to HTW team. The portion of max. points obtained by the teams are summarized in Figure 8.7.

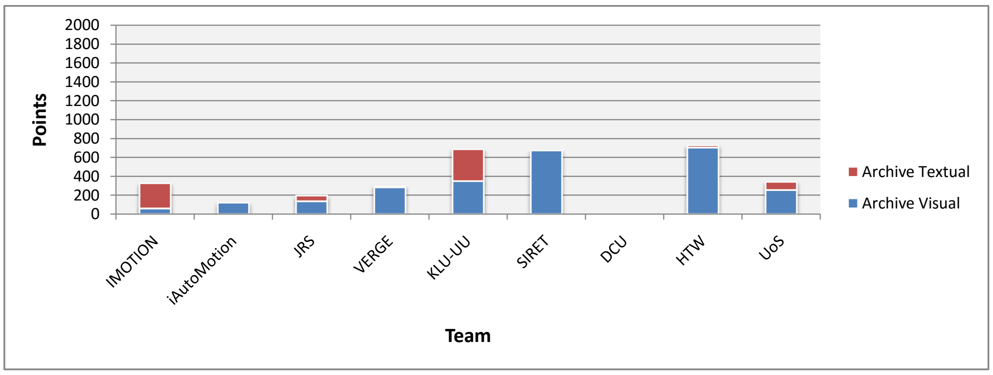


Figure 8.6 Total points acquired by the teams participating on VBS 2016.

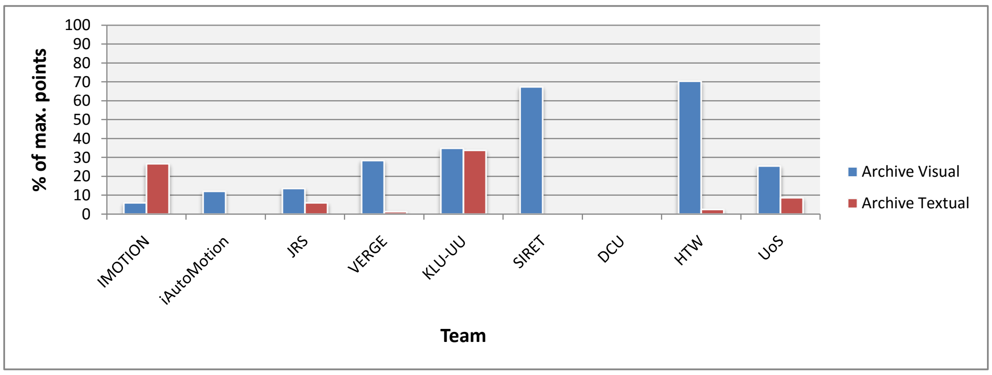


Figure 8.7 Percentage of maximal possible points acquired by the teams participating on VBS 2016.

The large dataset effectively divided the teams in two groups. One "successful" with teams HTW, KLU-UU and SIRET each having more than 600 points and "unsuccessful" where teams have less than 300 points, i.e., two times less. From our experience, it might be seemingly trivial things that may drag the overall performance down. For this reason, the techniques utilized by the teams from the "unsuccessful" group should not be considered as inefficient. Perhaps they only require more fine tuning.

Interestingly, both SIRET and HTW teams seem to struggle with textual KIS tasks. Both teams combined have only one correct textual KIS submission. In contrast to that, this does not apply to KLU-UU team which obtained roughly the same amount of points from both textual and visual KIS tasks.

We conclude that VBS and its goal remain highly relevant. No distinct winner came out of VBS 2016 and it is clear that exchange of ideas and combination of approaches may push the field even further.

8.4 Summary

Since its first versions, SBVB reached the performance of the state-of-the-art in the field of video retrieval and KIS in video. Some teams, inspired by our tool, included similar sketching canvas and retrieval model to their tools [5, 56]. Thus, we are able to say that we played our small part in pushing the frontier a bit further.

Despite the effort, the problems we aim are far from being satisfactory solved. This confirms the importance of events such as VBS.

The results are summarized in Table 8.4 including our remarks for each year we participated.

Table 8.4 Summary of the VBS results from years 2014–2016.

Year	**Scope**	**Our rank**	**Our loss**	**Remarks**
2014	26 hrs	1st	-	Single video KIS solved We won by a significant margin Feature Signatures established as a baseline color descriptor
2015	100 hrs	1st	-	A tool based on human computation demonstrated competitive performance Very tight results
2016	250 hrs	3rd	55 pts	Dataset size becomes challenging Tight results

9. User Studies

In this chapter, we present results of two user studies. In particular, we analyze an expert user behavior from data logged at VBS 2014 and evaluate performance of two textual search approaches introduced in Section 6.3. Interestingly, the studies were carried out two years apart.

9.1 Analysis of SBVB at VBS 2014

In this section, we analyze users behavior at VBS competition 2014. Both novices and experts performed visual and textual KIS tasks using the original version of SBVB. Task categories, participating teams and the overall results are summarized in Section 8.1.

All the user interactions with the tool were recorder including sketch manipulations, results browsing etc. In fact, the searching options and browsing capabilities were quite limited at that time. Either one or two sketches could be specified and the best matching key-frames were displayed.

Out of 54 KIS tasks, the expert successfully solved 52 within the time limit reaching impressive 96% success rate. On the other hand, the novice user was successful at only 7 out of 10 KIS tasks. This suggest significant difference between novice and expert users performance.

The search times [1] for the successful tasks are plotted in Figure 9.1.

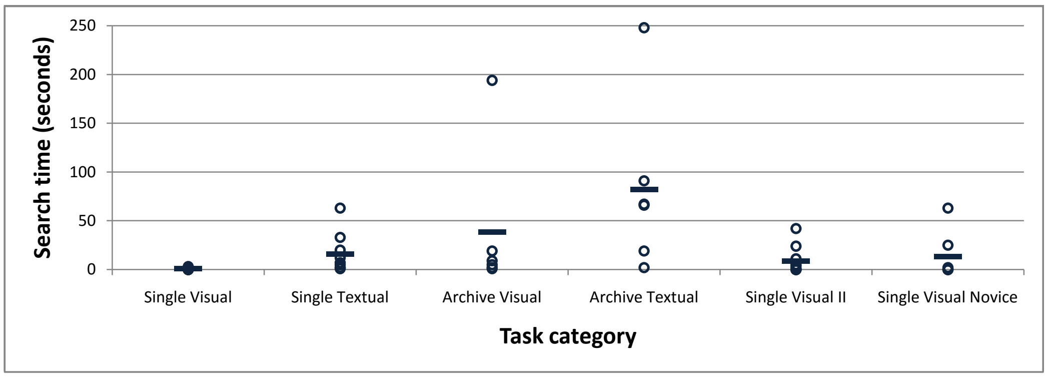


Figure 9.1 Search times of the 59 successful KIS tasks from VBS 2014.

Extraordinary results were achieved in the first task category where all the tasks were solved within one or two seconds from the end of the video playback. Actually, this holds for the most of the visual KIS tasks in single video even in the Single Visual Novice category. Search times are noticeable longer in textual and archive categories which is not surprising.

We identified two types of interactions – results browsing and query specification. Results browsing includes examining wider neighborhood of a particular result or listing additional results. Although we incorporated a video player in our tool it was not used at all. Query specification is in fact drawing of the sketches, i.e., positioning colored circles, color picking etc.

[1]For visual KIS, time is measured from the end of the playback. For textual KIS the time is measured from the first appearance of the textual description.

The relative amount of the search time spent browsing is depicted in Figure 9.2.

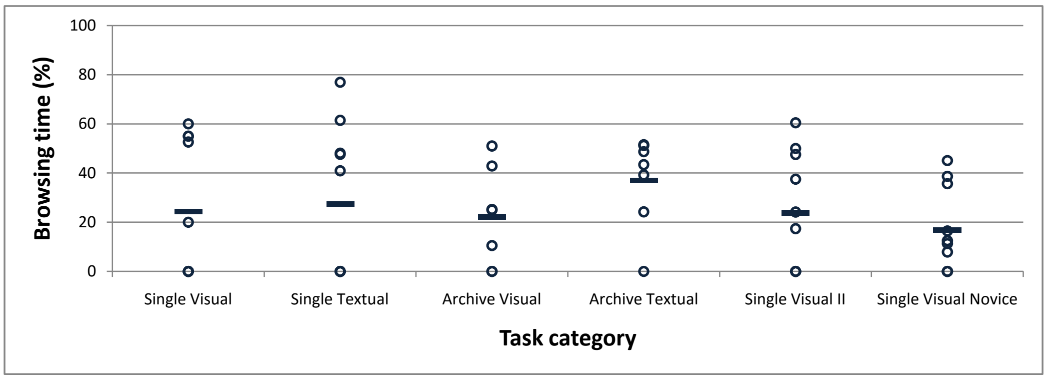


Figure 9.2 Relative amount of the search time spent by browsing in the 59 successful KIS tasks from VBS 2014.

We may see that within all the categories users spend roughly 30% of time with results browsing. Considering the variance, we can not identify any (statistically) significant differences between the task categories.

Furthermore, we identified the number of centroids in the final sketch for each KIS task. Histograms of the number of sketch centroids are depicted in Figure 9.3 where Single Video summarizes Single Visual and Single Textual tasks and Video Archive summarizes Archive Visual and Archive Textual tasks.

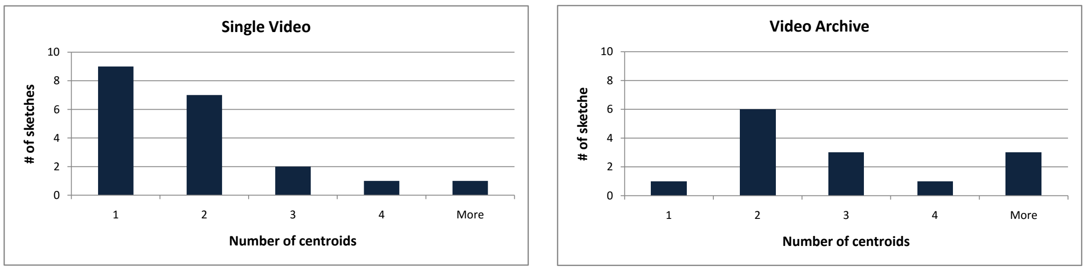


Figure 9.3 Distributions of the numbers of centroids in the final sketches for single (left) and archive (right) task categories.

Apparently, it is often sufficient to specify two sketch centroids to locate the target segment in a single video. This explains how we achieved the minuscule search times for the single KIS tasks. Average sketch of the target segment can be drawn within seconds and the results are displayed nearly instantly.

In accordance to our expectations, the archive KIS tasks required more elaborate sketches. Typically, the final sketch comprised two or more centroids. Additionally, in 35% of the archive tasks, the user utilized also the second sketch contrary to 15% for the single video tasks. We conclude that in the archive scope, the searched scene has to be described in more detail to facilitate a successful search.

9.2 Search Initialization with Textual Queries

In section 6.3, we proposed two orthogonal textual search approaches. In fact, the textual search module is the only part that was not tested at the VBS competition.

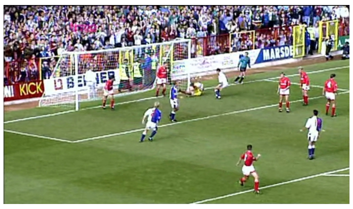
Hence, we carried out a simple user study in order to determine which of the proposed approaches is more effective. A total of 21 novice participants were briefly introduced to the tool and asked to find 6 different video segments presented via playback with a limit of 3 minutes for each task. The database contained almost 30 hours of diverse video content including, TV shows, sports, indoor and outdoor activities, etc. Example key-frames from the searched video segments are displayed in Fig. 9.4.

(a) Task 1

(b) Task 2

(c) Task 3

(d) Task 4


(e) Task 5

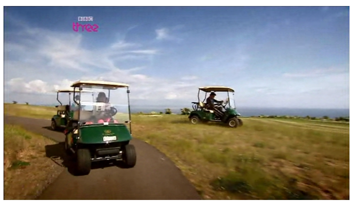

(f) Task 6

Figure 9.4 Example key-frames from the searched video segments.

The two textual search approaches presented in this paper are referred to as Google (Section 6.3.1) and ImageNet (Section 6.3.2). For each task, participants were randomly divided in three groups, where each group was enabled to use either Google, ImageNet or both approaches.

Out of 102 individual searches, 96 included a textual query, and 33 of them consecutively lead to a success (34.4 %). The numbers of successful searches are captured in Fig. 9.5. In 13 out of the successful 33 searches, participants did not use any feature other than textual search. The data collected do not demonstrate statistically significant differences between success rates — for Google vs ImageNet p-value is 0.731014, for Google vs both p-value is 0.291958 and for ImageNet vs both p-value 0.261518. In the case of the group enabled to use both techniques, total number of Google queries was 108, as opposed to 72 ImageNet queries. However, if we restrict this to the successful tasks, there were 17 Google queries and 20 ImageNet queries.

ImageNet turned to be effective search initialization for scenes, that contain a particular, easily identifiable object, e.g. a *harvester* in Task 3 or a *golf cart* in Task 6. Regarding complex scenes containing plurality of concepts (Tasks 2, 4 and 5), users struggled to select the one actually detected by ImageNet.

Arguably, the main limitation of the Google approach is that the retrieved images frequently do not fit users expectations. Although they may contain the searched object, the context happens to be quite dissimilar to the searched scene. We observed that inexperienced users were using these unfitting images instead of refining the query. On the other hand, we are able to retrieve suitable images with a seemingly unrelated query independently on the content of our database.

Apparently, Tasks 6 and 3 were rather easy as both approaches lead to the

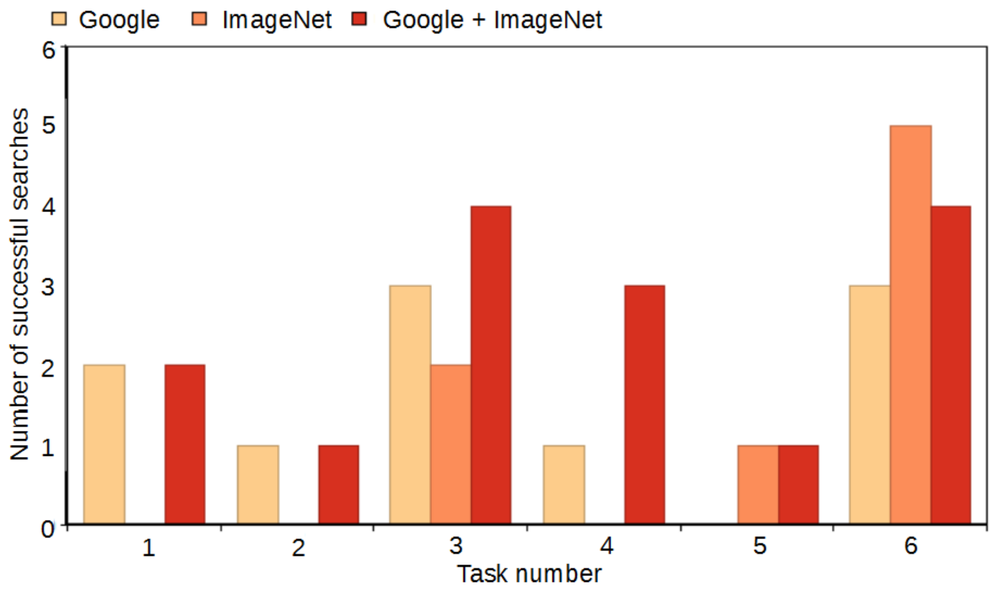


Figure 9.5 Numbers of successful searches for each task.

success almost instantaneously. We contribute this to the fact that distinct keywords were available to begin the search with (e.g. *golf cart*, *harvester*). Task 1 could have been solved just by using the sketch-based techniques. Task 4 was very hard for users limited to ImageNet as none of the obvious concepts (*pumpkin*, *goat*) were actually detected. Task 5 required to follow the obvious text query *football* with additional browsing techniques. Task 2 was rather confusing as some of the apparent search words such as *military* or *kitchen* provided misleading results.

We conclude that both approaches provide a viable way to search for a known-item in certain scenarios. It was also revealed that the textual queries are preferred by novice users, as a third of the successful searches was carried out without any other modalities, such as color and edge sketches.

9.3 Summary

In this chapter, the results of two user studies were provided. In the first study, we analyzed the behavior of users when only the feature signature model was available. It was revealed that a sketch with units of centroids may be sufficient to find the searched segment. In the second study, we tested two orthogonal textual search approaches. Both appeared viable even for novice users.

10. Conclusions

This thesis approaches the problem of large-scale video retrieval and browsing. In particular, we address the Known-item Search scenario wherein users search for a short video segment known either visually or by textual description. The topic requires to combine algorithms and apparatuses from a number of research areas such as machine learning, computer vision, database systems and cognitive science.

We have shown that the state-of-the-art solutions to KIS problem are indeed diverse yet none clearly outperforms the others. More importantly, none of the solutions provides steady and satisfactory level of performance.

In our bachelor thesis, we proposed a novel approach to the known-item search problem based on color feature signatures descriptor, where users define their search intent in a form of simple colored sketches. Although, compared with other approaches at VBS 2014 our tool SBVB came out as a winner, we saw room for improvements.

In this work, we adapted additional search modalities and enhanced SBVB with a number of browsing options. In particular, we incorporated multi-modal (color and edge) sketches, similarity search with DeCAF descriptors, new visualization/browsing methods and two orthogonal models for textual search. Hereby, SBVB evolves to Enhanced SBVB with which we underwent a comparison with other state-of-the-art tools at VBS 2015 and 2016. As a matter of fact, we won VBS 2015 and ended up 3rd at VBS 2016.

The results shown that our approach is not only viable but also able to outperform other methods. Our feature signatures video retrieval model was adapted and further extended by other teams competing at VBS. Furthermore, we enclosed additional user studies which demonstrated the applicability of our methods even for novice users.

10.1 Future Work

Although the overall performance of ESBVB is among the state-of-the-art, we still see room for improvements. We would like to investigate various novel image/video descriptors and similarity models (including up-to-date DCNN models). Furthermore, we register a significant performance gap between novice and expert users. One factor is that the user interface offers a number of features which might be confusing at the first glance. Nonetheless, it is also the knowledge about the behavior of the underlying retrieval system which may be effectively exploited by experts.

In our future work, we would like to further study behavior of users in our system and investigate what is the difference in novice and expert usage patterns. Is it possible to aid novice users and minimize the performance gap? Ultimately, we would like our system to be adaptive to a particular user.

Nonetheless, we believe that to significantly push the performance further, we must better understand the actual video content. More precisely, machines have to. It was shown that human aided computation may perform surprisingly well. Perhaps mimicking human cognitive abilities is the way to go.

Bibliography

[1] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. *Neural Codes for Image Retrieval*, pages 584–599. Springer International Publishing, Cham, 2014. ISBN 978-3-319-10590-1. doi: 10.1007/978-3-319-10590-1_38. URL http://dx.doi.org/10.1007/978-3-319-10590-1_38.

[2] Werner Bailer, Wolfgang Weiss, Christian Schober, and Georg Thallinger. *Browsing Linked Video Collections for Media Production*, pages 407–410. Springer International Publishing, Cham, 2014. ISBN 978-3-319-04117-9. doi: 10.1007/978-3-319-04117-9_47. URL http://dx.doi.org/10.1007/978-3-319-04117-9_47.

[3] Werner Bailer, Wolfgang Weiss, and Stefanie Wechtitsch. *Selecting User Generated Content for Use in Media Productions*, pages 388–393. Springer International Publishing, Cham, 2016. ISBN 978-3-319-27674-8. doi: 10.1007/978-3-319-27674-8_38. URL http://dx.doi.org/10.1007/978-3-319-27674-8_38.

[4] Kai Uwe Barthel, Nico Hezel, and Radek Mackowiak. *Graph-Based Browsing for Large Video Collections*, pages 237–242. Springer International Publishing, Cham, 2015. ISBN 978-3-319-14442-9. doi: 10.1007/978-3-319-14442-9_21. URL http://dx.doi.org/10.1007/978-3-319-14442-9_21.

[5] Kai Uwe Barthel, Nico Hezel, and Radek Mackowiak. *Navigating a Graph of Scenes for Exploring Large Video Collections*, pages 418–423. Springer International Publishing, Cham, 2016. ISBN 978-3-319-27674-8. doi: 10.1007/978-3-319-27674-8_43. URL http://dx.doi.org/10.1007/978-3-319-27674-8_43.

[6] A. Del Bimbo and P. Pala. Visual image retrieval by elastic matching of user sketches. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):121–132, Feb 1997. ISSN 0162-8828. doi: 10.1109/34.574790.

[7] Adam Blažek, Jakub Lokoč, and Tomáš Skopal. Video retrieval with feature signature sketches. In *International Conference on Similarity Search and Applications*, pages 25–36. Springer, 2014.

[8] Adam Blažek, Jakub Lokoč, Filip Matzner, and Tomáš Skopal. *Enhanced Signature-Based Video Browser*, pages 243–248. Springer International Publishing, Cham, 2015. ISBN 978-3-319-14442-9. doi: 10.1007/978-3-319-14442-9_22. URL http://dx.doi.org/10.1007/978-3-319-14442-9_22.

[9] Adam Blažek. Sketch-based video browser, 2014. URL https://is.cuni.cz/webapps/zzp/detail/146017?lang=en.

[10] Gary Bradski et al. The opencv library. *Doctor Dobbs Journal*, 25(11): 120–126, 2000.

[11] Thomas Brox and Jitendra Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE transactions on pattern analysis and machine intelligence*, 33(3):500–513, 2011.

[12] Claudiu Cobârzan, Marco A. Hudelist, and Manfred Del Fabro. *Content-Based Video Browsing with Collaborating Mobile Clients*, pages 402–406. Springer International Publishing, Cham, 2014. ISBN 978-3-319-04117-9. doi: 10.1007/978-3-319-04117-9_46. URL `http://dx.doi.org/10.1007/978-3-319-04117-9_46`.

[13] Claudiu Cobârzan, Manfred Del Fabro, and Klaus Schoeffmann. *Collaborative Browsing and Search in Video Archives with Mobile Clients*, pages 266–271. Springer International Publishing, Cham, 2015. ISBN 978-3-319-14442-9. doi: 10.1007/978-3-319-14442-9_26. URL `http://dx.doi.org/10.1007/978-3-319-14442-9_26`.

[14] Claudiu Cobârzan, Klaus Schoeffmann, Werner Bailer, Wolfgang Hürst, Adam Blažek, Jakub Lokoč, Stefanos Vrochidis, Kai Uwe Barthel, and Luca Rossetto. Interactive video search tools: a detailed analysis of the video browser showdown 2015. *Multimedia Tools and Applications*, pages 1–33, 2016. doi: 10.1007/s11042-016-3661-2. URL `http://dx.doi.org/10.1007/s11042-016-3661-2`.

[15] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005.

[16] Manfred Del Fabro and Laszlo Böszörmenyi. *AAU Video Browser: Non-Sequential Hierarchical Video Browsing without Content Analysis*, pages 639–641. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-27355-1. doi: 10.1007/978-3-642-27355-1_63. URL `http://dx.doi.org/10.1007/978-3-642-27355-1_63`.

[17] Manfred Del Fabro and Laszlo Böszörmenyi. *AAU Video Browser: Non-Sequential Hierarchical Video Browsing without Content Analysis*, pages 639–641. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-27355-1. doi: 10.1007/978-3-642-27355-1_63. URL `http://dx.doi.org/10.1007/978-3-642-27355-1_63`.

[18] Manfred Del Fabro, Bernd Münzer, and Laszlo Böszörmenyi. *AAU Video Browser with Augmented Navigation Bars*, pages 544–546. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-35728-2. doi: 10.1007/978-3-642-35728-2_64. URL `http://dx.doi.org/10.1007/978-3-642-35728-2_64`.

[19] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *CoRR*, abs/1310.1531, 2013. URL `http://arxiv.org/abs/1310.1531`.

[20] Mathias Eitz, Kristian Hildebrand, Tamy Boubekeur, and Marc Alexa. A descriptor for large scale image retrieval based on sketched feature lines. In *SBM*, pages 29–36, 2009.

[21] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[22] Digital River GmbH. Media monkey, 2016. URL `http://www.mediamonkey.com/`.

[23] H. Han and A. K. Jain. Tattoo based identification: Sketch to image matching. In *2013 International Conference on Biometrics (ICB)*, pages 1–8, June 2013. doi: 10.1109/ICB.2013.6613003.

[24] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[25] Jun He, Xindi Shang, Hanwang Zhang, and Tat-Seng Chua. *Mental Visual Browsing*, pages 424–428. Springer International Publishing, Cham, 2016. ISBN 978-3-319-27674-8. doi: 10.1007/978-3-319-27674-8_44. URL `http://dx.doi.org/10.1007/978-3-319-27674-8_44`.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[27] Marti A. Hearst, Susan T Dumais, Edgar Osman, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28, 1998.

[28] Rui Hu and John Collomosse. A performance evaluation of gradient field hog descriptor for sketch based image retrieval. *Computer Vision and Image Understanding*, 117(7):790–806, 2013.

[29] Marco A. Hudelist and Qing Xu. *The Multi-stripe Video Browser for Tablets*, pages 272–277. Springer International Publishing, Cham, 2015. ISBN 978-3-319-14442-9. doi: 10.1007/978-3-319-14442-9_27. URL `http://dx.doi.org/10.1007/978-3-319-14442-9_27`.

[30] Marco A. Hudelist, Claudiu Cobârzan, Christian Beecks, Rob van de Werken, Sabrina Kletz, Wolfgang Hürst, and Klaus Schoeffmann. *Collaborative Video Search Combining Video Retrieval with Human-Based Visual Inspection*, pages 400–405. Springer International Publishing, Cham, 2016. ISBN 978-3-319-27674-8. doi: 10.1007/978-3-319-27674-8_40. URL `http://dx.doi.org/10.1007/978-3-319-27674-8_40`.

[31] Wolfgang Hürst, Rob van de Werken, and Miklas Hoet. *A Storyboard-Based Interface for Mobile Video Browsing*, pages 261–265. Springer International Publishing, Cham, 2015. ISBN 978-3-319-14442-9. doi: 10.1007/978-3-319-14442-9_25. URL `http://dx.doi.org/10.1007/978-3-319-14442-9_25`.

[32] Wolfgang Hürst, Rob van de Werken, and Miklas Hoet. *MultiMedia Modeling: 21st International Conference, MMM 2015, Sydney, NSW, Australia, January 5-7, 2015, Proceedings, Part II*, chapter A Storyboard-Based Interface for Mobile Video Browsing, pages 261–265. Springer International Publishing, Cham, 2015. ISBN 978-3-319-14442-9. doi: 10.1007/978-3-319-14442-9_25. URL `http://dx.doi.org/10.1007/978-3-319-14442-9_25`.

[33] Daniel P. Huttenlocher, Gregory A. Klanderman, and William J Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on pattern analysis and machine intelligence*, 15(9):850–863, 1993.

[34] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3304–3311. IEEE, 2010.

[35] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[36] J. R. Kender and Boon-Lock Yeo. Video scene segmentation via continuous video coherence. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pages 367–373, Jun 1998. doi: 10.1109/CVPR.1998.698632.

[37] J. R. Kender and Boon-Lock Yeo. Video scene segmentation via continuous video coherence. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pages 367–373, Jun 1998. doi: 10.1109/CVPR.1998.698632.

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[39] David Kuboň, Adam Blažek, Jakub Lokoč, and Tomáš Skopal. *Multi-sketch Semantic Video Browser*, pages 406–411. Springer International Publishing, Cham, 2016. ISBN 978-3-319-27674-8. doi: 10.1007/978-3-319-27674-8_41. URL `http://dx.doi.org/10.1007/978-3-319-27674-8_41`.

[40] Fredrik L. Fast video cataloger, 2016. URL `http://videocataloger.com/`.

[41] Christoph H Lampert, Matthew B Blaschko, and Thomas Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[42] Duy-Dinh Le, Vu Lam, Thanh Duc Ngo, Vinh Quang Tran, Vu Hoang Nguyen, Duc Anh Duong, and Shin'ichi Satoh. *NII-UIT-VBS: A Video Browsing Tool for Known Item Search*, pages 547–549. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-35728-2. doi: 10.1007/978-3-642-35728-2_65. URL `http://dx.doi.org/10.1007/978-3-642-35728-2_65`.

[43] Wee Kheng Leow and Rui Li. The analysis and applications of adaptive-binning color histograms. *Computer Vision and Image Understanding*, 94(1): 67–91, 2004.

[44] D. Lin, S. Fidler, C. Kong, and R. Urtasun. Visual semantic search: Retrieving videos via complex textual queries. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2657–2664, June 2014. doi: 10.1109/CVPR.2014.340.

[45] T. Lindeberg. Scale Invariant Feature Transform. *Scholarpedia*, 7(5):10491, 2012.

[46] Jakub Lokoč, Adam Blažek, and Tomáš Skopal. *Signature-Based Video Browser*, pages 415–418. Springer International Publishing, Cham, 2014. ISBN 978-3-319-04117-9. doi: 10.1007/978-3-319-04117-9_49. URL `http://dx.doi.org/10.1007/978-3-319-04117-9_49`.

[47] Jakub Lokoč, Adam Blažek, and Tomáš Skopal. *Signature-Based Video Browser*, pages 415–418. Springer International Publishing, Cham, 2014. ISBN 978-3-319-04117-9. doi: 10.1007/978-3-319-04117-9_49. URL `http://dx.doi.org/10.1007/978-3-319-04117-9_49`.

[48] Jakub Lokoč, Adam Blažek, and Tomáš Skopal. On effective known item video search using feature signatures. In *Proceedings of International Conference on Multimedia Retrieval*, ICMR '14, pages 524:524–524:526, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2782-4. doi: 10.1145/2578726.2582617. URL `http://doi.acm.org/10.1145/2578726.2582617`.

[49] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[50] Wei-Ying Ma and Hong Jiang Zhang. Benchmarking of image features for content-based retrieval. In *Signals, Systems &amp; Computers, 1998. Conference Record of the Thirty-Second Asilomar Conference on*, volume 1, pages 253–257. IEEE, 1998.

[51] Anastasia Moumtzidou, Konstantinos Avgerinakis, Evlampios Apostolidis, Vera Aleksić, Fotini Markatopoulou, Christina Papagiannopoulou, Stefanos Vrochidis, Vasileios Mezaris, Reinhard Busch, and Ioannis Kompatsiaris. *VERGE: An Interactive Search Engine for Browsing Video Collections*, pages 411–414. Springer International Publishing, Cham, 2014. ISBN 978-3-319-04117-9. doi: 10.1007/978-3-319-04117-9_48. URL `http://dx.doi.org/10.1007/978-3-319-04117-9_48`.

[52] Anastasia Moumtzidou, Konstantinos Avgerinakis, Evlampios Apostolidis, and et al. Verge: An interactive search engine for browsing video collections. In *MultiMedia Modeling*, volume 8326 of *Lecture Notes in Computer Science*, pages 411–414. Springer International Publishing, 2014. ISBN 978-3-319-04116-2. doi: 10.1007/978-3-319-04117-9_48. URL `http://dx.doi.org/10.1007/978-3-319-04117-9_48`.

[53] Anastasia Moumtzidou, Konstantinos Avgerinakis, Evlampios Apostolidis, Fotini Markatopoulou, Konstantinos Apostolidis, Theodoros Mironidis, Stefanos Vrochidis, Vasileios Mezaris, Ioannis Kompatsiaris, and Ioannis Patras. *VERGE: A Multimodal Interactive Video Search Engine*, pages 249–254. Springer International Publishing, Cham, 2015. ISBN 978-3-319-14442-9. doi: 10.1007/978-3-319-14442-9_23. URL `http://dx.doi.org/10.1007/978-3-319-14442-9_23`.

[54] Anastasia Moumtzidou, Theodoros Mironidis, Evlampios Apostolidis, Foteini Markatopoulou, Anastasia Ioannidou, Ilias Gialampoukidis, Konstantinos Avgerinakis, Stefanos Vrochidis, Vasileios Mezaris, Ioannis Kompatsiaris, and Ioannis Patras. *VERGE: A Multimodal Interactive Search Engine for Video Browsing and Retrieval*, pages 394–399. Springer International Publishing, Cham, 2016. ISBN 978-3-319-27674-8. doi: 10.1007/978-3-319-27674-8_39. URL `http://dx.doi.org/10.1007/978-3-319-27674-8_39`.

[55] Thanh Duc Ngo, Vu Hoang Nguyen, Vu Lam, Sang Phan, Duy-Dinh Le, Duc Anh Duong, and Shin'ichi Satoh. *NII-UIT: A Tool for Known Item Search by Sequential Pattern Filtering*, pages 419–422. Springer International Publishing, Cham, 2014. ISBN 978-3-319-04117-9. doi: 10.1007/978-3-319-04117-9_50. URL `http://dx.doi.org/10.1007/978-3-319-04117-9_50`.

[56] Thanh Duc Ngo, Vinh-Tiep Nguyen, Vu Hoang Nguyen, Duy-Dinh Le, Duc Anh Duong, and Shin'ichi Satoh. *NII-UIT Browser: A Multimodal Video Search System*, pages 278–281. Springer International Publishing, Cham, 2015. ISBN 978-3-319-14442-9. doi: 10.1007/978-3-319-14442-9_28. URL `http://dx.doi.org/10.1007/978-3-319-14442-9_28`.

[57] David Novak and Michal Batko. Metric index: An efficient and scalable solution for similarity search. In *Proceedings of the 2009 Second International Workshop on Similarity Search and Applications*, pages 65–73. IEEE Computer Society, 2009.

[58] David Novak, Michal Batko, and Pavel Zezula. Large-scale similarity data management with distributed metric index. *Information Processing & Management*, 48(5):855–872, 2012.

[59] David Novak, Michal Batko, and Pavel Zezula. Large-scale image retrieval using neural net descriptors. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1039–1040. ACM, 2015.

[60] Paul Over, Jon Fiscus, Greg Sanders, David Joy, Martial Michel, George Awad, Alan Smeaton, Wessel Kraaij, and Georges Quénot. Trecvid 2014–an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *Proceedings of TRECVID*, page 52, 2014.

[61] Bo Gun Park, Kyoung Mu Lee, and Sang Uk Lee. Color-based image retrieval using perceptually modified hausdorff distance. *EURASIP Journal on Image and Video Processing*, 2008(1):1, 2007.

[62] Dong Kwon Park, Yoon Seok Jeon, and Chee Sun Won. Efficient use of local edge histogram descriptor. In *Proceedings of the 2000 ACM workshops on Multimedia*, pages 51–54. ACM, 2000.

[63] Luca Rossetto, Ivan Giangreco, Heiko Schuldt, Stéphane Dupont, Omar Seddati, Metin Sezgin, and Yusuf Sahillioğlu. *IMOTION — A Content-Based Video Retrieval Engine*, pages 255–260. Springer International Publishing, Cham, 2015. ISBN 978-3-319-14442-9. doi: 10.1007/978-3-319-14442-9_24. URL `http://dx.doi.org/10.1007/978-3-319-14442-9_24`.

[64] Luca Rossetto, Ivan Giangreco, Silvan Heller, Claudiu Tănase, Heiko Schuldt, Stéphane Dupont, Omar Seddati, Metin Sezgin, Ozan Can Altıok, and Yusuf Sahillioğlu. *IMOTION – Searching for Video Sequences Using Multi-Shot Sketch Queries*, pages 377–382. Springer International Publishing, Cham, 2016. ISBN 978-3-319-27674-8. doi: 10.1007/978-3-319-27674-8_36. URL `http://dx.doi.org/10.1007/978-3-319-27674-8_36`.

[65] Luca Rossetto, Ivan Giangreco, Claudiu Tănase, Heiko Schuldt, Stéphane Dupont, Omar Seddati, Metin Sezgin, and Yusuf Sahillioğlu. *iAutoMotion – an Autonomous Content-Based Video Retrieval Engine*, pages 383–387. Springer International Publishing, Cham, 2016. ISBN 978-3-319-27674-8. doi: 10.1007/978-3-319-27674-8_37. URL `http://dx.doi.org/10.1007/978-3-319-27674-8_37`.

[66] Yossi Rubner and Carlo Tomasi. *Perceptual metrics for image database navigation*, volume 1. Springer, 2000.

[67] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.

[68] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[69] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[70] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[71] Hanan Samet. *The design and analysis of spatial data structures*, volume 199. Addison-Wesley Reading, MA, 1990.

[72] Jorge Sánchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. Image classification with the fisher vector: Theory and practice. *International journal of computer vision*, 105(3):222–245, 2013.

[73] K. Schoeffmann. A user-centric media retrieval competition: The video browser showdown 2012-2014. *IEEE MultiMedia*, 21(4):8–13, Oct 2014. ISSN 1070-986X. doi: 10.1109/MMUL.2014.56.

[74] Klaus Schoeffmann and Laszlo Boeszoermenyi. Video browsing using interactive navigation summaries. In *2009 Seventh International Workshop on Content-Based Multimedia Indexing*, pages 243–248. IEEE, 2009.

[75] Klaus Schoeffmann, David Ahlström, and Laszlo Böszörmenyi. *Video Browsing with a 3D Thumbnail Ring Arranged by Color Similarity*, pages 660–661. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-27355-1. doi: 10.1007/978-3-642-27355-1_70. URL `http://dx.doi.org/10.1007/978-3-642-27355-1_70`.

[76] Klaus Schoeffmann, Marco A. Hudelist, and Jochen Huber. Video interaction tools: A survey of recent work. *ACM Comput. Surv.*, 48(1):14:1–14:34, September 2015. ISSN 0360-0300. doi: 10.1145/2808796. URL `http://doi.acm.org/10.1145/2808796`.

[77] David Scott, Jinlin Guo, Cathal Gurrin, Frank Hopfgartner, Kevin McGuinness, Noel E. O'Connor, Alan F. Smeaton, Yang Yang, and Zhenxing Zhang. *DCU at MMM 2013 Video Browser Showdown*, pages 541–543. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-35728-2. doi: 10.1007/978-3-642-35728-2_63. URL `http://dx.doi.org/10.1007/978-3-642-35728-2_63`.

[78] David Scott, Zhenxing Zhang, Rami Albatal, Kevin McGuinness, Esra Acar, Frank Hopfgartner, Cathal Gurrin, Noel E. O'Connor, and Alan F. Smeaton. *Audio-Visual Classification Video Browser*, pages 398–401. Springer International Publishing, Cham, 2014. ISBN 978-3-319-04117-9. doi: 10.1007/978-3-319-04117-9_45. URL `http://dx.doi.org/10.1007/978-3-319-04117-9_45`.

[79] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL `http://arxiv.org/abs/1409.1556`.

[80] Alan F. Smeaton, Paul Over, and Wessel Kraaij. Evaluation campaigns and trecvid. In *MIR '06: Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*, pages 321–330, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-495-2. doi: http://doi.acm.org/10.1145/1178677.1178722.

[81] Cees G. M. Snoek, Marcel Worring, and Arnold W. M. Smeulders. Early versus late fusion in semantic video analysis. In *Proceedings of the 13th Annual ACM International Conference on Multimedia*, MULTIMEDIA '05, pages 399–402, New York, NY, USA, 2005. ACM. ISBN 1-59593-044-2. doi: 10.1145/1101149.1101236. URL `http://doi.acm.org/10.1145/1101149.1101236`.

[82] Cees GM Snoek, Spencer Cappallo, Daniel Fontijne, David Julian, Dennis C Koelma, Pascal Mettes, KEA Sande, Anthony Sarah, Harro Stokman, R Blythe Towal, et al. Qualcomm research and university of amsterdam at trecvid 2015: Recognizing concepts, objects, and events in video. 2015.

[83] Irwin Sobel and Gary Feldman. A 3x3 isotropic gradient operator for image processing. *a talk at the Stanford Artificial Project in*, pages 271–272, 1968.

[84] T. Volkmer and A. Natsev. Exploring automatic query refinement for text-based video retrieval. In *2006 IEEE International Conference on Multimedia and Expo*, pages 765–768, July 2006. doi: 10.1109/ICME.2006.262951.

[85] Feng Wang, Yu-Gang Jiang, and Chong-Wah Ngo. Video event detection using motion relativity and visual relatedness. In *Proceedings of the 16th ACM International Conference on Multimedia*, MM '08, pages 239–248, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-303-7. doi: 10.1145/1459359.1459392. URL `http://doi.acm.org/10.1145/1459359.1459392`.

[86] Jin Yuan, Huanbo Luan, Dejun Hou, Han Zhang, Yan-Tao Zheng, Zheng-Jun Zha, and Tat-Seng Chua. *Video Browser Showdown by NUS*, pages 642–645. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-27355-1. doi: 10.1007/978-3-642-27355-1_64. URL `http://dx.doi.org/10.1007/978-3-642-27355-1_64`.

[87] Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. *Similarity search: the metric space approach*, volume 32. Springer Science & Business Media, 2006.

[88] Zhenxing Zhang, Rami Albatal, Cathal Gurrin, and Alan F. Smeaton. *Interactive Known-Item Search Using Semantic Textual and Colour Modalities*, pages 282–286. Springer International Publishing, Cham, 2015. ISBN 978-3-319-14442-9. doi: 10.1007/978-3-319-14442-9_29. URL `http://dx.doi.org/10.1007/978-3-319-14442-9_29`.

[89] Zhenxing Zhang, Wei Li, Cathal Gurrin, and Alan F. Smeaton. *Faceted Navigation for Browsing Large Video Collection*, pages 412–417. Springer International Publishing, Cham, 2016. ISBN 978-3-319-27674-8. doi: 10.1007/978-3-319-27674-8_42. URL `http://dx.doi.org/10.1007/978-3-319-27674-8_42`.

List of Figures

List of Tables

Attachment – DVD

We attach a DVD with the following contents:

- ESBVB application in binary form compiled for 64bit Windows operating system equipped with .NET Platform version 4 or higher.
- ESBVB application sources in a form of MS Visual Studio 2015 project.
- User video guide to ESBVB.