Charles University in Prague

Faculty of Mathematics and Physics

BACHELOR THESIS

Adam Blažek

Signature-Based Video Browser

Department of Software Engineering

Supervisor of the bachelor thesis:	RNDr. Jakub Lokoč, Ph.D.
Study programme:	Informatics
Specialization:	Programming

Prague 2014

Hereby, I would like to thank my supervisor Jakub Lokoč for his valuable advice and suggestions, for the extraordinary amount of time he reserved for me and for keeping me motivated all the time.

I shall also mention Tomáš Skopal who enabled me to present this work at international conferences.

Furthermore, I am grateful to Adam Vyškovský, Filip Matzner and other colleagues for constructive criticism and to Tereza Daňhelovská and my family for supporting me during my studies.

Název práce: Hledání ve videu pomocí signatur

Autor: Adam Blažek

Katedra: Katedra softwarového inženýrství

Vedoucí bakalářské práce: RNDr. Jakub Lokoč, Ph.D.

Abstrakt: V této bakalářské práci uvádíme efektivní přístup pro hledání ve videu využívající takzvané signatury – flexibilní deskriptory obrázků založené na distribuci barev. Signatury poskytují vhodnou reprezentaci klíčových snímků a zároveň umožňují uživatelům jednoduše kreslit barevné skeče hledaných scén. V práci detailně popíšeme inovativní model pro hledání ve videu a také probereme a pečlivě optimalizujeme všechny jeho parametry. Pro dosažení vysoké rychlosti modelu navrhneme několik vhodných indexovacích technik a jejich výkon vyhodnotíme v experimentech s reálnými daty. Součástí práce je také implementace navrhovaného modelu v programovacím jazyce C# s jednoduchým a intuitivním uživatelským rozhraním umožnujícím interaktivní prohledávání až desítek hodin videa.

Klíčová slova: Feature signatures, video retrieval, similarity search, feature extraction, Known-Item Search

Title: Signature-Based Video Browser

Author: Adam Blažek

Department: Department of Software Engineering

Supervisor: RNDr. Jakub Lokoč, Ph.D.

Abstract: In this thesis, we present an effective yet efficient approach for Known-Item Search in video data. The approach utilizes so called feature signatures – simple and flexible visual descriptors based on color distribution. The feature signatures enable to efficiently represent video key-frames and at the same time allow users to intuitively draw simple colored sketches of the searched scenes. In the thesis, we describe in detail a novel video retrieval model and also discuss and carefully optimize it's parameters. To achieve high efficiency, we expose several indexing techniques suitable for the model and empirically evaluate their performance in the experiments. The described model is implemented in C# programming language with simple and intuitive user interface enabling users to interactively browse up to tens of hours of video.

Keywords: Feature signatures, video retrieval, similarity search, feature extraction, Known-Item Search

Contents

Introduction

The volume of video data has been increasing rapidly over the last years which poses a challenge to the state-of-the-art video management and retrieval systems. Independently on the volume and the nature of the data, users still expect fast and accurate responses as well as simple user interfaces to specify a query and to intuitively browse the results. These demands are making designing a system for video indexing and retrieval a true challenge.

A large amount of attention is paid to the systems based on semantic annotation [11, 15] allowing users to specify textual queries. To deal with the lack of annotation, complex concept and event detectors are being employed, but despite the progress made in the last years, the *semantic gap* [1] still persists. If we consider just the reliably detectable common concepts (e.g., human faces [3, 33] or cars [14, 31]), we may end up with zero annotation, thus we cannot rely on them exclusively.

For these reasons the general-purpose content-based methods are getting more popular. Many visual descriptors [9, 27, 29] were introduced to enable fast extraction, indexing and searching in large scale video archives. The systems [24] based on these descriptors usually demand an example as a query; however, such an example may not be always available. In such cases the user has to put an effort into obtaining the example, say using Google Images, which can be time consuming or even impossible in some cases. Let us give an example: We are searching for a clip with a particular TV studio interior, filmed from an unusual angle while do not have an example. If we try to find the example in an independent image database with the phrase "TV studio" we will probably retrieve plenty of relevant results and it might be hard to find such image that is visually close to the searched scene.

This scenario matches the problem of the so-called Known-Item Search (KIS), where the user "knows" what object (i.e., a video clip) she is searching for (by imagination and/or textual description), however, she has no example to run a traditional query (an example video shot/key-frame in our case). By allowing users to specify the searched clip directly, for example with a sketch [7, 10], the uncomfortable need of example can be eliminated. It is crucial, however, to keep the user interface as simple as possible. Such descriptor shall be utilized that is descriptive enough, is understandable to users and can be easily specified.

Respecting the mentioned demands, we utilize the feature signatures [26], where a video key-frame is represented by a set of it's color regions. Such simple representation enables users to specify these regions directly in simple sketches. Unlike the fixed grid features, the feature signatures are able to capture even less significant color regions and adapt to the complexity of a key-frame. For instance, the representation of a single-colored key frame comprises only few color regions while in the case of a more complex scene the representation is also more complex. Moreover, the resulting 5-dimensional feature space together with Euclidean distance makes the retrieval process efficient.

[1]The gap between raw pixel data and the semantic meaning — easy to overcome for humans yet a tremendous problem for machines.

In this thesis, we describe in detail a novel video retrieval model based on the feature signatures enabling an effective KIS in video data. We also present an award winning [2] tool implementing the proposed model – Signature-Based Video Browser (SBVB) – which is capable of indexing and searching of up to tens of hours of video content.

Structure

At first, we gently introduce basic concepts used in the similarity searching and discuss key features being employed in the current state-of-the-art tools (Chapter 1). Then, the feature signatures based video retrieval model is presented (Chap. 2) paying special attention to the possibilities of indexing the feature space (Chap. 3). Since the user interface plays a crucial role in the overall performance of SBVB, we present and vindicate our solution in detail (Chap. 4). Finally, we find the optimal setup of the model, ranging from various model parameters to indexing techniques, via experiments with real data (Chap. 5). The results are then summarized in the conclusion.

An important and integral part of this thesis is the attached CD with an implementation of the proposed model in C# programming language – SBVB. Besides the complete sources and compiled ready-to-use binaries, the CD also contains all the logs from the experiments and an already indexed video file suitable for the first inspection of the tool. Since we have not encountered any significant problem during the implementation, we just point out and discuss the interesting parts of the code in a HTML document on the CD instead of devoting a whole chapter for this purpose.

Key Contributions

The key contributions of this thesis can be listed as follows:

- We propose and describe in detail a novel video retrieval model based on feature signatures suitable for visual and textual KIS in video.
- We analyze users behavior, possibilities of indexing the feature space and the model parameters and propose the optimal setup of the model.
- We present SBVB – an award winning tool for the KIS in video which implements the proposed model.

[2] The tool was presented at Video Browser Showdown (VBS) [1] and clearly outperformed the tools of other participants. We report more details in Chapter 5.

Preliminaries

In this chapter, we shortly summarize necessary basic concepts used in the field of similarity search. Namely, we define a *Metric space* along with common query paradigms and a *Vector space* over the real numbers as an example. We also describe in principle the *k-means clustering* widely used in the processing of visual data.

Metric and Vector Spaces

A *Metric space* $\mathbb{M}$ is a pair $\mathbb{M} = (\mathbb{D}, d)$, where $\mathbb{D}$ denotes a set of objects and d is a distance function $\mathbb{D} \times \mathbb{D} \to \mathbb{R}$ ($\mathbb{R}$ is the set of real numbers) satisfying following conditions:

$$\begin{aligned}
&d(x,y) \geq 0 && \text{(non-negativity)}\\
&d(x,y) = 0 \quad \text{iff} \quad x = y && \text{(identity)}\\
&d(x,y) = d(y,x) && \text{(symmetry)}\\
&d(x,y) + d(y,z) \geq d(x,z) && \text{(triangular inequality)}
\end{aligned}$$

A *Metric space* forms a similarity model together with the query-by-example searching paradigm, where a query is formed by an object example $q \in \mathbb{D}$ and an additional constraint defining a subset of $\mathbb{D}$ to be returned. Basic types of queries are *range query* (q, r) returning all the objects within the range r (i.e., $\{x \in \mathbb{D} \,|\, d(q,x) \leq r\}$) and *k nearest neighbor query* (q, k) where the set of k nearest objects to q from $\mathbb{D}$ is returned [34].

As the distance function may be costly to compute or the *Metric space* can contain a large amount of objects, an indexing technique has to be introduced for efficient query processing. Some datasets are of course harder to index than others. To capture this property, we define the *intrinsic dimensionality*[5] based on a basic statistical analysis (1), where μ and σ denotes the mean and variance respectively of all the pairwise distances between the objects from $\mathbb{D}$. The higher the *intrinsic dimensionality* is, the less clusters [3] the dataset contains which indicates less possibilities to prune the space efficiently.

$$\text{idim}_{\mathbb{D}} = \frac{\mu^2}{2\sigma^2} \tag{1}$$

As an example of a *Metric space*, we present the d-dimensional *Vector space* [4] over the real numbers $\mathbb{R}^d$ along with the L_p metric (for $p \geq 1$) defined as

$$L_p(x,y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{\frac{1}{p}} \tag{2}$$

where p is typically 1 (*Manhattan distance*), 2 (*Euclidean distance*) or ∞ (*Maximal distance*). Once particular dimensions of a *Vector space* have different meanings,

[3] A group of objects close to each other and far from the objects outside the group.

[4] From now on, we assume that a *Vector space* is over the real numbers unless stated otherwise.

we can introduce weights w_i to the distance function (3) in order to reflect these differences.

$$L_p^{'}(x,y) = \left(\sum_{i=1}^{d} w_i |x_i - y_i|^p \right)^{\frac{1}{p}} \tag{3}$$

For example, with $p = 2$ we get a *weighed Euclidean distance.*

Note that *Vector spaces* have many important properties enriching the variety of possible indexing techniques in comparison with general *Metric spaces.* For example, since *Vector spaces* have defined operations such as addition of two vectors or multiplication of a vector and a scalar, new vectors can be "created" using these operations. Also, an arbitrary finite set of vectors can be easily bounded, for example, with a sphere or cuboid.

K-means Clustering

The *k-means clustering* is a kind of cluster analysis — grouping a set of objects to clusters so that within a cluster, objects are more similar to each other than to objects in other clusters. More specifically, it operates over a *Vector space* and for a given set of points and k, the goal is to estimate k cluster centers (means) so that the sum of distances of the points to the nearest mean is minimal possible.

This problem is actually NP-hard [8] and thus heuristic algorithms are being employed. We present the original algorithm [16] based on the iterative process of within-cluster sum of distances optimization.

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.

$$S_i^{(t)} = \{x \in X \mid L_2(x, m_i^{(t)}) \leq L_2(x, m_j^{(t)}) \, \forall j = 1 \dots k\} \tag{4}$$

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

$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x \in S_i^{(t)}} x \tag{5}$$

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.

1. Related Work

Tools for the KIS in video are being evaluated at various multimedia retrieval events, like the VBS workshop at the Multimedia Modeling (MMM) conference series. In this particular case, the usage of a textual query is prohibited; thus, participants are forced to introduce innovative and interactive interfaces to their tools which makes the event even more interesting. Visual as well as textual KIS tasks are evaluated in both single video and video archive scopes. We shortly describe the tools of all the participants of VBS 2014.

David Scott et al. [28] participated with a tool based on the automated semantic annotation of both audio and video data. In particular, occurrences of 60 visual concepts were identified and indexed using the current state-of-the-art methods such as Scale-Invariant Feature Transform (SIFT), Support Vector Machine (SVM) and Bag of Visual Words (BoVW). The tool enabled a visual similarity search and a face browsing, where all the faces found in the video were presented and users could list the shots in which selected faces appeared.

Similar approach was followed by Anastasia Moumtzidou et al. [21]. More than 300 concepts were detected via the current state-of-the-art methods and visual similarity search was supported with MPEG-7 and Speeded Up Robust Features (SURF). Agglomerative hierarchical clustering of the segmented shots were employed in order to provide a hierarchical view of the results.

A very innovative tool introduced by Claudiu Cob Arza et al. [6] exploited the advantages of collaborative search. In contrast to the previous approaches, only simple descriptors such as MPEG-7 color layout and motion histogram [27] were extracted. Users could specify the desired scene (via dominant color, background and foreground movement, scene duration etc.) simultaneously on several devices such as tablets or smart-phones. Promising results could be marked for further examination by any of the collaborators.

A tool benefiting from both concept detectors and simple color descriptors was presented by the team from NII and UIT [22]. The training data for concept classifiers were obtained from Google Images; in addition, a simple 4x3 grid of dominant colors for each video segment was calculated. Users could specify a sequence of patterns comprising concept occurrences and grid-like color sketch to filter out the irrelevant segments of a video. Results were presented in a coarse-to-fine manner ensuring optimal space utilization and easy browsing.

Werner Bailer et al. [2] introduced a video browsing tool originally created for media production where a high redundancy is expected. Low-level features such as a global color distribution, camera motion and object trajectories were extracted and aggregated into MPEG-7 descriptors in addition to SURF descriptors. Both used to cluster the video segments and to enable a visual similarity search.

Finally, we joined VBS with the tool described in this thesis – SBVB. The basic principle of the tool from the user as well as the algorithmic point of view was already published [17, 19], however, this thesis goes in much more detail.

Let us summarize the key features appearing repetitively in the tools of other participants:

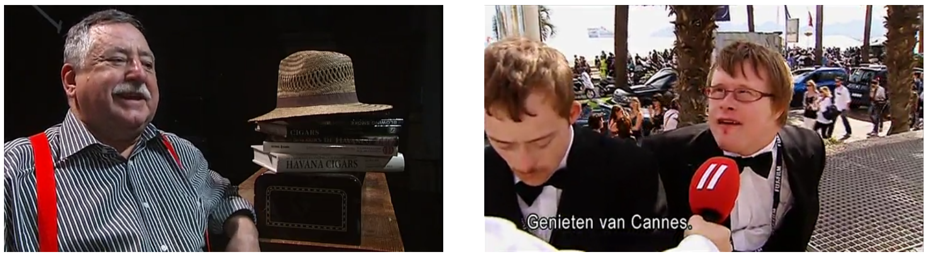
1. Concept detection [21, 22, 28]
2. Low-level visual descriptors [2, 6, 22]
3. Enhanced results presentation [6, 21, 22]
4. Visual similarity search [2, 21, 28]

In our approach, we employ low-level visual descriptors – feature signatures and a visual similarity search based on this descriptor. We also propose a technique how to include the concept-based filtering in our tool. Furthermore, the problem of the results presentation is addressed in Chapter 4.

2. Retrieval Model

In this chapter, we describe the employed retrieval model for searching video clips of user interest. Since humans are naturally able to process visual stimuli very quickly and to remember the color and position of distinct color regions in the observed scene, we believe that extracting these color regions and enabling users to specify them, is sufficient to form an effective video retrieval model.

Let us show two examples (Fig. 2.1) with occurrences of distinct color regions which are, however, not dominant (in terms of the occupied area). The color together with the position of such color region is very descriptive and can be easily remembered and later specified. In order to capture even less dominant regions, we employ the feature signatures [26] as described in the following section.



(a) The red braces. **(b)** The red microphone.

Figure 2.1 Examples of distinct however not dominant color regions.

2.1 Key-frame Representation

For the reasons mentioned above, we focus on position-color feature signatures that can flexibly aggregate and simply represent the color distribution of the contents of the key-frames. In order to extract a feature signature from a given key-frame, the extraction algorithm maps all pixels of the key-frame into 5-dimensional feature space [1] $\mathbb{F} \subset \mathbb{R}^5$ and then performs an adaptive variant of the k-means algorithm [12].

The algorithm results in the set of *centroids* of detected clusters (ideally centers of distinct color regions in the key-frame) forming feature signature $\mathbb{FS} \subset \mathbb{F}$, where the initial set of centers for the k-means is distributed uniformly. Due to the adaptive nature of the utilized k-means algorithm, the feature signatures vary in the number of *centroids* respecting the complexity of the key-frames. Besides the color and position of *centroids*, the weight (i.e., the number of pixels contributing to the cluster) could be extracted; however, in this work we do not utilize this information in the retrieval model. We will discuss this problem in the conclusion.

An example of a feature signature is shown in Fig. 2.2, where we may observe that the utilized feature signatures can be simply and intuitively interpreted as a rough approximation of the original image. Furthermore, such simple colored

[1]The feature space is formed by coordinates (x, y, L, a, b), where x, y denotes the position of the pixel and L, a, b represents its color in the CIE LAB color space [32].

circles can be directly entered by users trying to define their query intent which can substitute the uncomfortable need of an example query image.

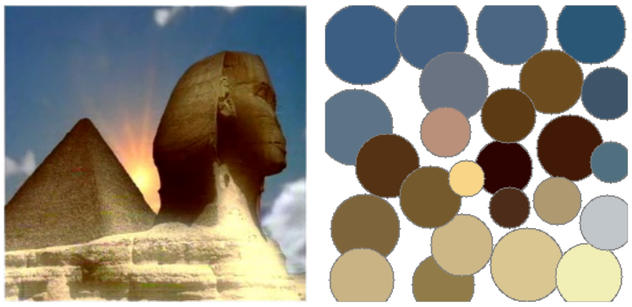


Figure 2.2 A key-frame and the visualization of the feature signature. For each *centroid* a circle with the corresponding color and position is drawn. The weight of the centroid is depicted as the diameter of the circle.

The video retrieval techniques also try to reduce the number of key-frames by selecting only the representative ones. However, employing a scene detection and representing the detected scenes with only one key-frame would bring a possibility of not capturing some of the less significant color regions. For this reason, we decided to select simply every k-th frame for the feature signatures extraction. Although this method may introduce a noticeable overhead, it gives the desired robustness to the retrieval model. Furthermore, the extraction is still reasonably fast allowing to extract the feature signatures from an hour of video within a few minutes. One feature signature comprises tens of *centroids* and it is sufficient to reserve only 3 bytes for the color and 4 bytes for the position. As a result, the memory demands are reasonably low [2].

2.2 Retrieval Algorithm

Let us assume that we have already extracted the feature signatures $FS_i = \{r_{ij}\}$ for the selected key-frames F_i where r_{ij} denotes the j-th *centroid* of the i-th feature signature. Users are enabled to specify a simple sketch of the viewed scene with colored circles(i.e. user-defined *centroids*) which allows to represent a query in the same way as 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 $FS_u = \{r_{uv}\}_{v=1}^{m}$ the ranking for each key-frame is calculated according to (2.1).

$$rank_{ui} = \operatorname*{avg}_{\forall r_{uv} \in FS_u} \frac{(dist_{uvi} - \min D_{uv})}{(\max D_{uv} - \min D_{uv})} \quad \text{where} \tag{2.1a}$$

$$dist_{uvi} = \min_{\forall r_{ij} \in FS_i} L_2(r_{ij}, r_{uv}) \quad \text{and} \quad D_{uv} = \{dist_{uvi} \,|\, \forall i\} \tag{2.1b}$$

The searched clip may consist of more than one visually discriminative scene and a sequence of sketches would be more descriptive. For this reason, users are enabled to specify two (and possibly more) time-ordered query sketches. The overall ranking for a two-sketch query FS_u followed by FS_w in the user-defined range ε is calculated according to (2.2).

[2]In practice, it is sufficient to process 2 frames per second. Assuming the proposed 7-byte representation and 25 *centroids* per frame, an hour of video produces less than 2Mb of meta-data.

$$rank_i = rank_{ui} + \min_{k=i+1}^{i+\varepsilon+1} rank_{wk} \tag{2.2}$$

Once we obtain the overall rankings, it is desirable to merge near-duplicate results generated by the dense key-frame representation. We accomplish that with the Alg. 1, where the function Neighbor returns all the key-frames from the predefined neighborhood, say 10 seconds around the popped key-frame. Note that the algorithm produces properly sorted results.

Algorithm 1 Results merging

```
QUEUE ← overall rankings        ▷ A queue ordered with respect to the ranking
MERGED ← empty                  ▷ A list of the results after merging
while QUEUE.NonEmpty() do
    RESULT ← QUEUE.Pop()
    MERGED.Add(RESULT)
    QUEUE.Remove(Neighbor(RESULT))
end while
return MERGED
```

The model presented so far demands the database to be fully scanned. Although the computation of the distances is not expensive, the overall processing of a query is not feasible once the dataset grows significantly. Moreover, it is clear that some *centroids* in the database are completely irrelevant for the query ranking. For example, if a user specifies a yellow *centroid* in the left-bottom corner of the sketch, a black database *centroid* situated in the right-top corner of a key-frame will not contribute to the query ranking and thus can be omitted.

Once we appropriately prune the database for a particular query *centroid*, the ranking of a key-frame having all the *centroids* pruned is undefined according to (2.1). In such case, we set the ranking to 1 meaning the worst match. It is clear, that correct definition of database *centroid* relevancy is crucial for maintaining the effectiveness of the model. We will show in the experiments (Chap. 5) that the relevant database *centroids* for a query *centroid* are defined with a *range query*.

2.3 Further Concept-Based Filtering

The concept detection is becoming a common procedure and the dataset may be even already annotated. Therefore, we also propose how to enhance the retrieval model with further concept-based filtering. For concepts C_j and key-frames F_i, we assume that the concept detections result into a set of probabilities $C_{ji} \in [0; 1]$ meaning that the key-frame F_i contains the concept C_j with the probability C_{ji}.

Users are enabled to specify whether a particular concept is present in the searched scene or not. The ranking $rank_{ui}$ defined by (2.1) is updated according to (2.3), where C_{def} is the set of concepts which occurrences were specified by a user, and α is user-defined constant defining the weight of the filters. Note that $rank^*_{ui}$ is again within the $[0, 1]$ interval.

$$rank^*_{ui} = (1-\alpha)rank_{ui} + \alpha\frac{1}{|C_{\mathrm{def}}|}\sum_{C_j \in C_{\mathrm{def}}} C^*_{ji} \qquad \text{where}$$

$$C^*_{ji} = \begin{cases} C_{ji} & \text{if } C_j \text{ shall occur} \\ 1 - C_{ji} & \text{if } C_j \text{ shall not occur} \end{cases} \tag{2.3}$$

2.4 Time complexity

In this section, we determine the time complexity of the proposed algorithm in case that the user specified a query with two sketches $FS_u = \{r_{uv}\}_{v=1}^m$ and $FS_w = \{r_{wv}\}_{v=1}^n$. Let the set of relevant *centroids* in the database for a particular query *centroid* r_{uv} be R_{uv}. Every R_{uv} defines a set F_{uv} of the relevant key-frames such that their feature signature contains at least one relevant *centroid*. The query *centroid* with the highest amount of the relevant *centroids* will be denoted by $r_{\max}$. Finally, let us define the overall relevant key-frames as $F_{\mathrm{rel}} = \bigcup_{r_x \in FS_u \cup FS_w} F_x$.

The algorithm can be divided into the following steps:

1. ***Centroid* ranking.** Since $|F_{uv}| \leq |R_{uv}|$, the work done by ranking F_{uv} (2.1b) is $\mathcal{O}(|R_{uv}|)$. Thus, the total work done in this step can be bounded with $\mathcal{O}((m+n)|R_{\max}|)$

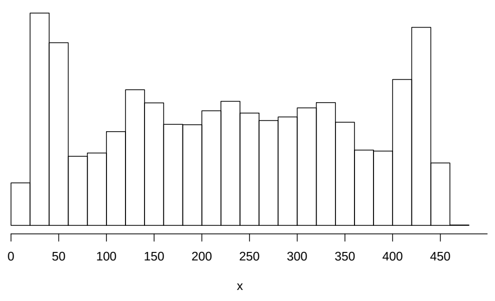
2. **Ranking aggregation** The time needed for aggregating (2.1a) the *centroid* rankings is again $\mathcal{O}((m+n)|R_{\max}|)$ while for determining the overall ranking (2.2) is $\mathcal{O}(\varepsilon|F_{\mathrm{rel}}|)$.

3. **Results merging** Following the Alg. (1), it is clear that each relevant key-frame is inserted into the priority queue and either popped or removed only once and thus the total work done in this step is $\mathcal{O}(|F_{\mathrm{rel}}|\log|F_{\mathrm{rel}}|)$.

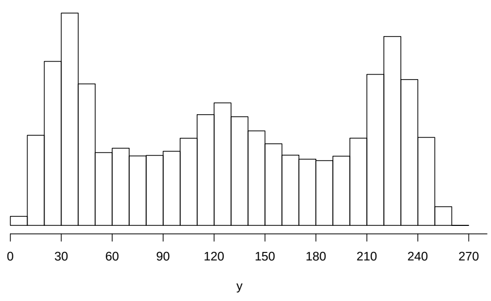
Finally, we get $\mathcal{O}(|F_{\mathrm{rel}}|\log|F_{\mathrm{rel}}| + \varepsilon|F_{\mathrm{rel}}| + (m+n)|R_{\max}|)$, where we expect ε to be less than 20 seconds and $R_{\max}$ to be less than 5% of the database. The pitfall is that in practice, $F_{\mathrm{rel}} = F$, i.e., all the key-frames are relevant for at least one query *centroid*.

3. Indexing the Feature Space

In order to efficiently process *range queries* in the utilized 5-dimensional *Vector space* (x, y, L, a, b) using the Euclidean distance, we investigate both spatial and metric indexing approaches, each represented by a suitable method. Since the employed feature extraction does not favor any key-frame region, the distribution of the position coordinates show high degree of uniformity (Fig. 3.1) and thus we have selected a Grid Index as the representative of spatial indexing methods. As a *Metric space* method, we have selected the current state-of-the-art technique – the M-Index [23]. As both techniques are well described in the literature, we will just review their most important properties.



(a) The histogram of x coordinates.



(b) The histogram of y coordinates.

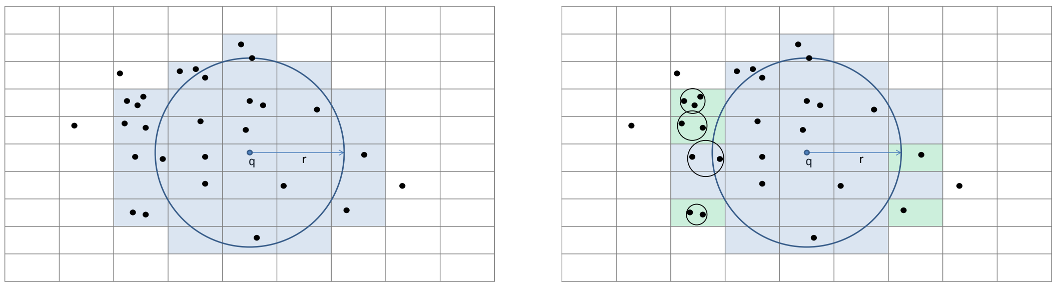
Figure 3.1 The histogram of spatial coordinates of the *centroids* extracted from the dataset used in the experiments (Chap. 5).

3.1 Grid Index

The Grid Index divides the feature space into uniform cuboid-like bins, where the number of bins grows exponentially with the space dimension which limits this approach only for low-dimensional spaces. The advantage of the uniform Grid Index is that the bin where an indexed *centroid* belongs can be directly computed. When the index is queried with a *range query* (q, r), every bin having non-empty intersection with the sphere defined by q and r has to be examined (blue-colored bins in Fig. 3.2a).

3.2 M-Index

The M-Index is a member of the permutation-based index family where the pivots (selected objects from the *Metric space*) help to dynamically cluster the feature space with respect to the data distribution. More specifically, the M-Index uses the repetitive Voronoi-based partitioning to define the cluster tree structure used for efficient *range query* processing where all possible metric filtering principles are combined (for more details, see [23]). The cluster tree is extended dynamically when an amount of objects in a leaf cluster exceeds the cluster size parameter. Besides the original M-Index, we employ the *Cut-region extension* [18](M-Index CR) enabling additional filtering already in the upper levels of the cluster tree.



(a) A basic Grid Index. **(b)** A Grid Index enhanced with BSs.

Figure 3.2 2D Grid Index queried with a *range query* (q, r). The database objects are depicted as black points.

Since the parameters of the M-Index, such as the pivot count or cluster size are crucial for the M-Index performance, we carefully optimize these parameters for every M-Index variant separately. Furthermore, the performance is affected significantly by the pivot selection technique as described in the following section.

3.2.1 Pivot selection techniques

In was shown that some pivots cluster the space better than others [4] which makes the pivot selection technique a point of interest. In particular, we can achieve the same performance with fewer pivots or improve the performance using the same number of pivots when we select them properly. The number of pivots affects also the memory demands of the M-Index and the indexing time. Clearly, it is desirable to optimize the pivot selection technique; therefore, we evaluate the performance with the following techniques in the experiments (Chap. 5).

Data random The standard selection technique where the pivots are selected randomly among the indexed objects.

Space random As the feature space is in fact a bounded *Vector space* and the objects are distributed more or less uniformly, selecting the pivots from the space randomly and independently of the data may be meaningful.

Heuristic This technique is based on the assumption that so called *outliers* (objects far from others) are better pivots [4]. Although finding [1] these *outliers* is computationally expensive, it may bring the desirable performance gain in the querying phase.

3.3 Bounding Sphere Constraint

Since we work with a *Vector space*, we can improve the filtering power of the utilized indexes by tight Bounding spheres (BSs) [2] evaluated dynamically for each bin/cluster separately. The motivation is that additional bounding can describe the region more tightly than grid bin or Voronoi-based cell cut-off by rings centered in global pivots.

[1]We adopted the *incremental* method proposed in [4].
[2]Bounding spheres are also referred as enclosing balls or ball-regions.

For a bin/cluster A with a BS $B_A(c_A, r_A)$ where c_A and r_A denote the center and radius of the BS respectively and a *range query* (q, r), we can formulate the *Bounding Sphere Constraint*:

$$d(q, c_A) > r + r_A \tag{3.1}$$

If (3.1) holds, A can be omitted during the evaluation of the query since every object in A is outside the query range (proof trivially implied by the *Metric space* postulates).

The usage of BSs together with the Grid Index(Grid Index BS) is demonstrated in Fig. 3.2b where the green bins can be skipped as their BSs do not intersect the query sphere (q, r) (i.e., 3.1 holds). Note that we did not depict all the BSs in order to keep the figure lucid. The *Bounding Sphere Constraint* can be also utilized in the M-Index (M-Index BS) and M-Index CR (M-Index BS + CR).

3.3.1 Computing Bounding Spheres

Nevertheless, the creation and maintenance of BSs (finding centers and radii) can be a costly indexing overhead and thus we employ the approximate Ritter's BS algorithm [25] suitable also for dynamic indexing. The algorithm, popular for its efficiency and simplicity, starts with a small sphere, iterates over the given points to be bounded and expands the sphere when needed. The process is depicted in Fig. 3.3 where the sphere $(c^{(t)}, r^{(t)})$ will be expanded to $(c^{(t+1)}, r^{(t+1)})$ (dashed) so that it bounds the current sphere as well as the point p_{t+1}.

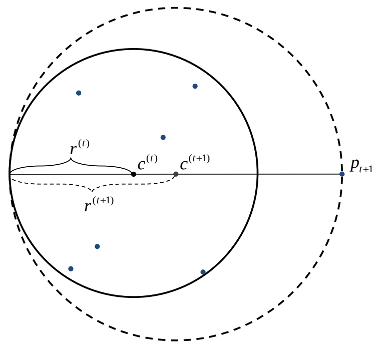


Figure 3.3 A step of Ritter's BS algorithm.

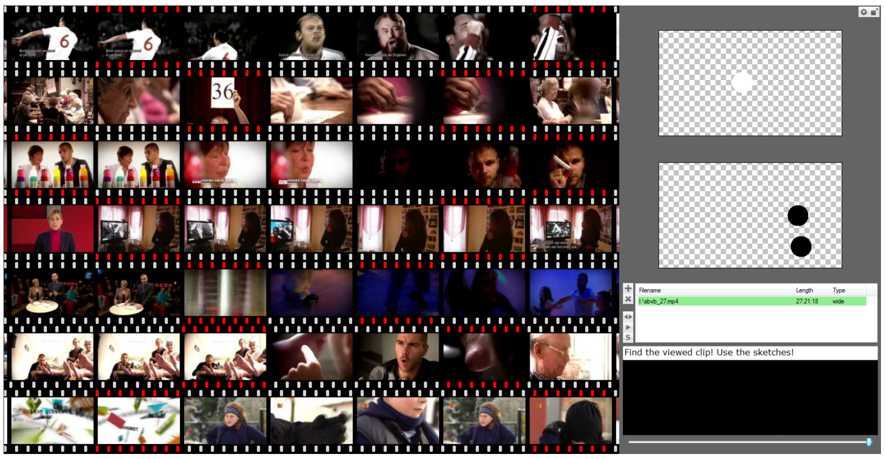
Needless to say, the algorithm gives only coarse results in comparison with the optimal BSs or other (more complicated) approximate algorithms [13, 30] and thus the experimental results related to BSs should be interpreted as a proof-of-concept only.

4. User Interface

The User interface (UI) is a crucial part of SBVB as a cumbersome one would make the tool ineffective. In this chapter, we describe and vindicate the UI implemented in the tool available on the DVD. Our primary goal was the simplicity and intuitiveness of the UI. We expect users to get familiar with the tool without need of any user manual; nevertheless, reading this chapter should be sufficient in case that something would be unclear.

4.1 Overall Layout

The UI depicted in Fig. 4.1 can be divided into two main areas where a query is specified (right) and the results are presented (left). The latter dominates since it is desirable to offer as much space as possible for the results presentation enabling easy identification of the desired clip. Assuming a wide screen (which is more common nowadays), we organized these two areas horizontally.



Filename	Length	Type
I:\sbvb_27.mp4	27:21:18	wide

Find the viewed clip! Use the sketches!

Figure 4.1 The overall layout of SBVB.

To utilize the whole available screen area, we suppressed the window frame and SBVB runs in the full-screen mode by default. Since we target also less skilled users, SBVB can be controlled utterly with a computer mouse leaving the keyboard shortcuts, that might speed up the searching, for experienced users.

4.2 Control Panel

In this section we describe all the controls available for users to specify the files to be searched, the query itself and also additional settings. We comment the depicted controls (Fig. 4.2) in the order in which the controls would be used in a typical usage scenario. The components highlighted in the figure are referred with capital bold letters **A**–**G**.

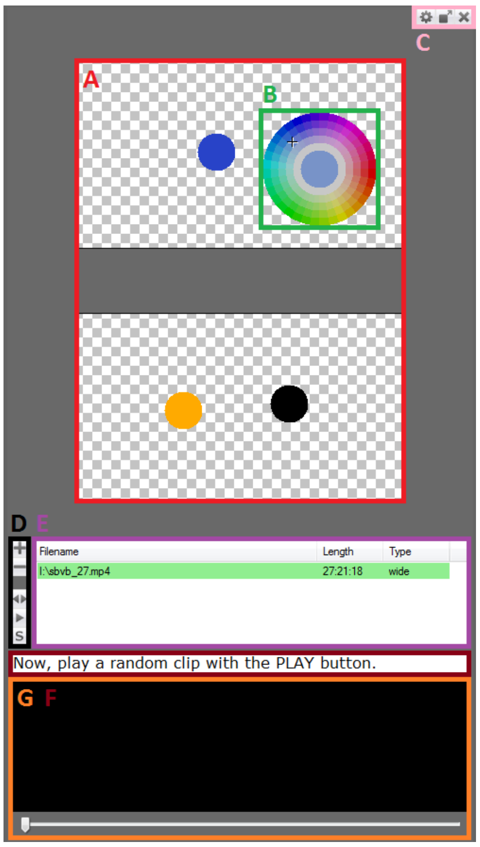
Although the settings of the tool are optimized by default and users shall not need to change them, some basic settings can be accessed with ⚙ (**C**). The full-screen mode can be toggled on and off with ⤢ and ⤡ respectively (**C**).

4.2.1 Selecting Files to be Searched

Prior to the search itself, the files to be searched must be specified and loaded. The currently loaded files are displayed in the table **E** having the self-explanatory header. To load a file, the user can click on + (**D**) and select it in the standard dialog. A file selected for the search first time has to be indexed which may take several minutes depending on the size of the file.

A file can be unloaded by selecting it in the table **E** and clicking on − (**D**).

We distinguish between wide (ca 16:9) and narrow (ca 4:3) video files. The desired video type can be selected with ◂▸ (**D**). Note that a loaded file is marked with the green background (**E**) if it matches the selected video type and with the red background (**E**) if not. The proportions of the sketches (**A**) reflect this selection as well.



Filename	Length	Type
I:\sbvb_27.mp4	27:21:18	wide

Now, play a random clip with the PLAY button.

Figure 4.2 The SBVB controls.

4.2.2 Playing a Random Clip

Once at least one file is loaded, the status line (**F**) advises users to play a random clip. To do so, the user has to click on ▶ (**D**). A window with a video player appears, then the clip playback starts automatically, and the window disappears after the playback ends.

After that, the user can draw simple sketches (**A**) to specify and hopefully retrieve the viewed clip.

4.2.3 Drawing Sketches

The user is enabled to draw up to two sketches (**A**) in order to retrieve the results as described in Sec. 2.2. The first (uppermost) is obligatory while the second one is optional.

Defining a Color Region

A color region can be created simply with a left-click anywhere on the sketch. A colored circle (i.e., a user-defined *centroid*) representing the color region appears at the clicked position together with an enhanced color picker (**B**).

Color Picking

We designed an enhanced color picker (**B**) allowing easy, intuitive and fast color picking. Instead of placing the color picker on a fixed position or to a special dialog window, it is displayed ad-hoc at each colored circle. The colors are organized concentrically around the colored circle in the HSV color space [32]. The user can access different value levels, i.e., lighter and darker colors, with the mouse-wheel having the cursor over the color picker. The desired color can be selected with a left-click on it and the user can cancel/induce the color picking with a left-click on the colored circle itself.

Removing and Moving a Color Region

Any colored circle can be removed from the sketch by right-clicking it. Also, it can be moved in the usual drag-drop manner in the scope of the sketch.

4.3 Results Presentation

The output of the retrieval algorithm is in fact a list of matched key-frames (or pairs of key-frames in case of a two-sketch query). In order to make the identification of the searched clip easier, each result is displayed on a separate row together with a sequence of preceding and following key-frames from the original video as depicted in Fig. 4.3. A displayed key-frame represents 2 seconds of video and thus the user can also see how long a particular scene lasted. A result row can be drag-dropped to the left or right in order to explore additional preceding or following key-frames.

The results are organized in pages according to their ranking in the ascending order (i.e., the best matches first). Initially, the 1st page is displayed and the following pages can be accessed by moving the cursor over the results and scrolling the mouse wheel.

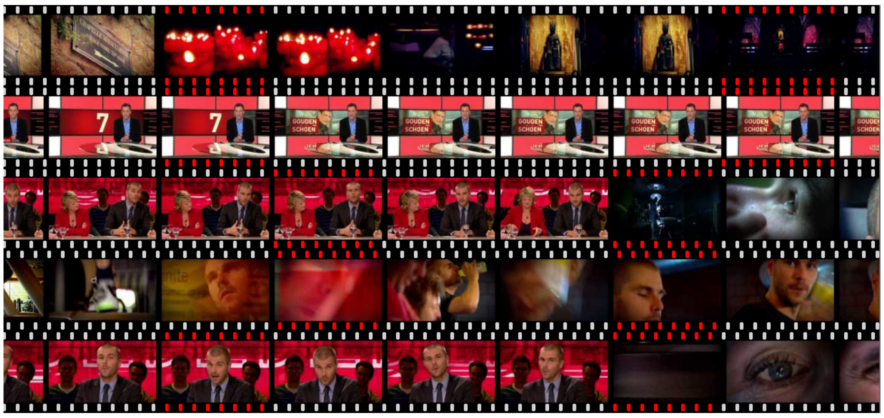


Figure 4.3 Top 5 result rows displayed after a two-sketch query. The matched key-frames are marked with red decorations.

A result can be selected by double-clicking on an arbitrary key-frame. If the selected key-frame is within the previously viewed clip, the selection is considered as correct. Anyway, the exact position of the selected key-frame is displayed in the status line (Fig. 4.2 – **F**) and also the selected clip is played in the result player (Fig. 4.2 – **G**).

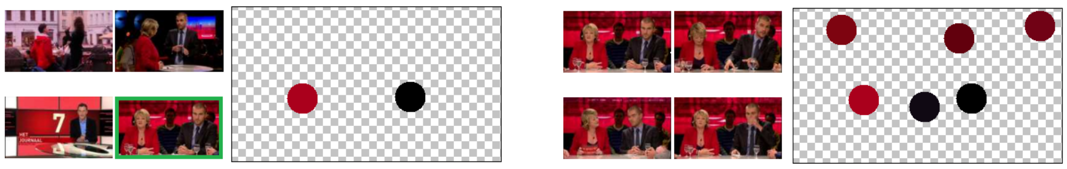
The results are updated on-fly immediately after every query modification making the search indeed interactive.

4.4 Query by Example

It is common practice to utilize a retrieved scene or key-frame, visually similar to the searched one, as further specification of the query. This may lead to the success very quickly and therefore we intend to employ this query-by-example paradigm as well. Commonly, additional descriptors are utilized such as SIFT or MPEG-7; however, extracting, storing and loading these descriptors costs noticeable computational time as well as extra memory.

For this reason, we introduce the following query-by-example procedure which utilizes the feature signatures and thus fits well to the so far presented model and UI. Moving the cursor over the results displays the *centroids* in the retrieved key-frames. The user can pick-up any displayed *centroid* to a sketch simply by clicking on it. In this manner, the user can pick up *centroids* even from key-frames dissimilar to the searched scene yet having a particular color region in common. The picked-up *centroids* can be adjusted in the same way as the user-defined ones, e.g., moved to different position.

The effectiveness of such query specification is demonstrated in Fig. 4.4, where we intended to find scenes similar to the green-framed one in Fig. 4.4a.



(a) Before *centroids* pick-up. **(b)** After *centroids* pick-up.

Figure 4.4 A query-by-example employing the feature signatures. The sketches (right) and the four best matching key-frames (left) before and after picking-up several *centroids*.

This feature can be toggled on and off with S (4.2 - **D**). After a *centroid* pick-up, the results update is a bit delayed giving the user a chance to pick-up more *centroids* at one time.

5. Experiments

In this chapter, we discuss and optimize the important aspects of the proposed model. More specifically, we determine:

- How the color and spatial coordinates should be scaled
- How to prune the database preserving the effectiveness of the model
- Optimal parameters for each index variant
- The efficiency of the indexing techniques

All the experiments were performed with a real dataset and user-defined sketches. Furthermore, we summarize our participation at VBS pointing out the effectiveness of SBVB.

5.1 Settings

The experiments were carried out on an EBU MIM-SCAIE video dataset[20] from which we selected 27 hours of diverse video content. The resulting database of 4.8 millions of *centroids* has the *intrinsic dimensionality* 3.23 which should imply relatively easy indexing.

More than 40 users were told to find a randomly selected short clip within five minutes. Almost 100 successfully found clips along with user-defined sketches were gathered and employed in the model optimization (Sections 5.2 and 5.3).

In order to evaluate the performance of the indexing techniques (Sections 5.4 and 5.5), 300 user-defined query *centroids* were collected and used for querying the index. Since we have utilized just cheap Euclidean distance, we have focused mainly on the overall time needed to process a query. The measurements were performed on an Intel Xeon CPU @2.80 GHz in a single thread. The data structures (excluding the key-frame images) occupied a reasonable amount of memory and were kept in the RAM memory.

5.2 Importance of Position and Color

In this section, we focus on the relation between the color and position coordinates, i.e., what is more important for finding the relevant key-frames. This relation is typically modeled by additional weighs for the color and position coordinates, for example, using weighted Euclidean distance. However, for fixed weights, the same effect can be achieved by scaling the feature space [1] prior to the indexing and searching. In the following text, we fix the color coordinates and scale only the position coordinates with a position-color ratio (PCR) [2].

To determine the optimal value of the PCR, we have performed following steps. For every searched clip c, we have queried the system with the user-defined query

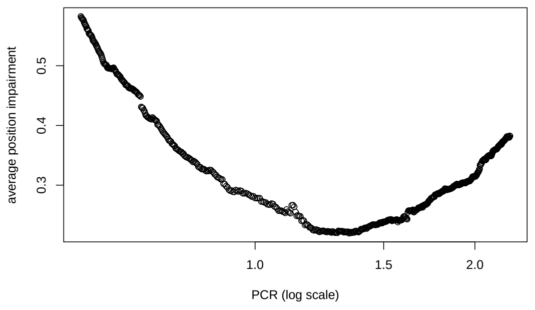
[1]In this way, the distance function will be less computationally expensive which is desirable.

[2]Note that the distances computed in (2.1) are scaled which allows us to optimize only the PCR.

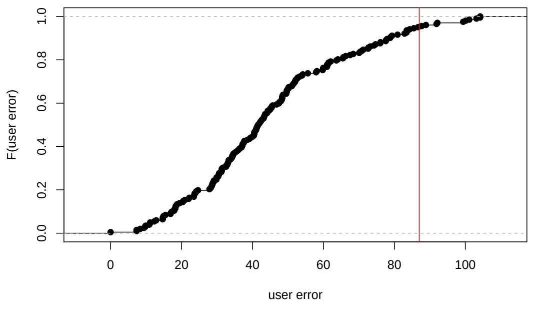
sketch under different values of the PCR and tracked the position of the clip c in the results. For PCR $= x$, the impairment ranking was obtained according to (5.1), where P_{cx} denotes the position of the searched clip in the results (less is better). In this way, imp_{cx} is the relative impairment of the position of the searched clip with respect to the best and worst possibility within the evaluated interval of the PCR.

$$imp_{cx} = \frac{(P_{cx} - \min P_c)}{(\max P_c - \min P_c)} \quad \text{where} \quad P_c = \{P_{cx} \,|\, \forall x\} \tag{5.1}$$

The average position impairment from all the user searches is depicted in Figure 5.1a. The optimal PCR found in this process depends of course on the initial setup. In our case, the color coordinates were scaled to the interval $[0, 255]$ (i.e., one byte) and the position coordinates followed the proportions of the key-frames – 350 x 200 (width x height). Finding the optimum at 1.35, the position coordinates are scaled by this number from now on.



(a) The dependency of the position impairment on the PCR.



(b) The empirical distribution function of the *user errors.*

Figure 5.1 The position – PCR dependency and the distribution function of *user errors.*

5.3 Relevant Neighborhood of a Query *Centroid*

It is desirable to delimit the minimal neighborhood of a query *centroid* in the database that has to be examined in order to retrieve the wanted clip. Users are of course inaccurate in specifying query *centroids* (e.g., the color of the intended region may be in fact darker) and the model has to tolerate these inaccuracies. We define the *user error* for a query *centroid* as the distance to the nearest database *centroid* from the searched clip. More formally, for a query *centroid* r specified by the user searching for a clip c the *user error* e_r is calculated according to (5.2), where FS_c stands for all the feature signatures extracted from the searched clip c.

$$e_r = \min_{\forall r_{ij} \in FS_c} L_2(r, r_{ij}) \tag{5.2}$$

The empirical distribution function of *user errors* is depicted in Fig. 5.1b, where the vertical red line marks the 95% quantile which is ca 87 in our case. Please note that this number may vary with different datasets and users. We can see that if we omit the database *centroids* beyond the range of 87 (i.e., a *range query*) the ranking of the searched clip remains unaffected in most of the cases.

5.4 Index Parameters

Prior to the comparison of the performance of the proposed index variants, we of course optimized the parameters of each separately. Taking the 300 user-defined query *centroids*, the whole database of 4.8 million *centroids* and the query range of 87, we measured the average overall time needed to process a *range query* under a particular setup. Each measurement was repeated 5 times and averaged afterwards.

The situation is quite simple in the case of the Grid Index — the only parameter to be optimized is the number of bins per dimension. Note that the total number of bins in the Grid Index grows exponentially with the number of bins per dimension; thus, it is not surprising that this parameter has a significant impact on the performance having a clear optimum in 16 (Fig. 5.2). It shall be pointed out that the optimal setup was found for a concrete database size and the performance of the Grid Index with this fixed setup might be suboptimal under different database sizes. We will address this problem in the conclusion.

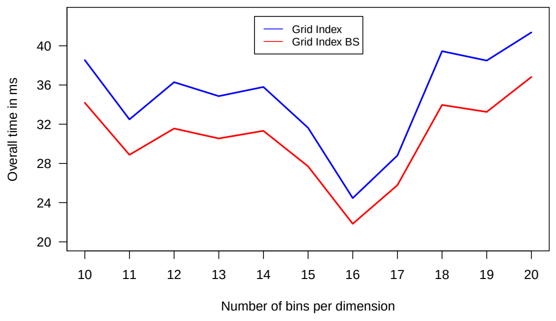
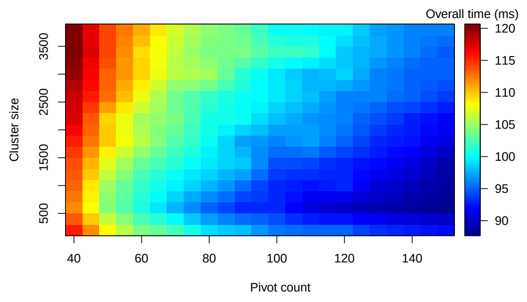


Figure 5.2 The performance of the Grid Index and Grid Index BS with varying number of bins per dimension.

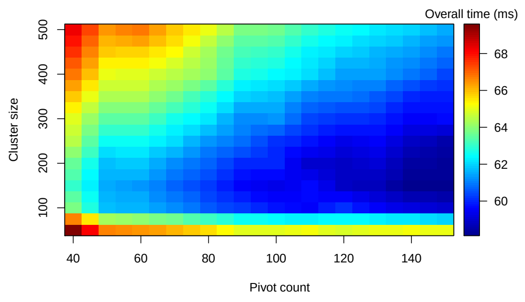
The M-Index, unfortunately, has more parameters requiring optimization. Namely the pivot count [3] and cluster size affects the performance greatly as depicted in Fig. 5.3 for the *heuristic* pivot selection. The results with the *random space* and *random data* pivot selection are depicted in Attachment C – M-Index Parameters Optimization. Here, we only summarize the performance of the best setup for each M-Index variant and pivot selection (Fig. 5.4), where we can see that the best results are achieved with the *heuristic* pivot selection.

The *heuristic* pivot selection, however, costs extra computing time and also requires more pivots for the optimal setup than the *random data* pivot selection. As a result, the indexing time and the memory demands are affected negatively which may be uncomfortable in the actual usage. Nevertheless, we employ it for the comparison of the index variants in the following section.

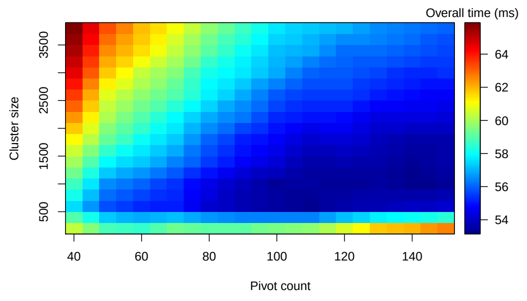
[3] As the pivot count defines the memory demands as well as the indexing time, it cannot be extensively increased. Also, since all the pivot selection techniques contains randomness, too little pivots would cause unstable performance. We empirically identified the reasonable interval to be 40–150.



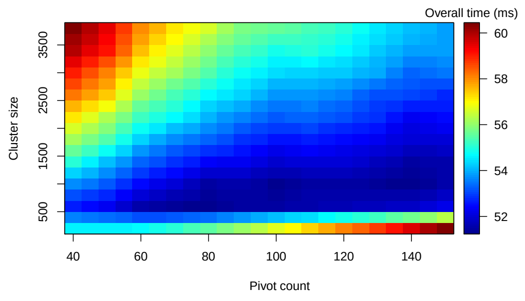
(a) M-Index.



(b) M-Index BS.



(c) M-Index CR.



(d) M-Index BS + CR.

Figure 5.3 The performance of the M-Index variants with the *heuristic* pivot selection under varying parameters.

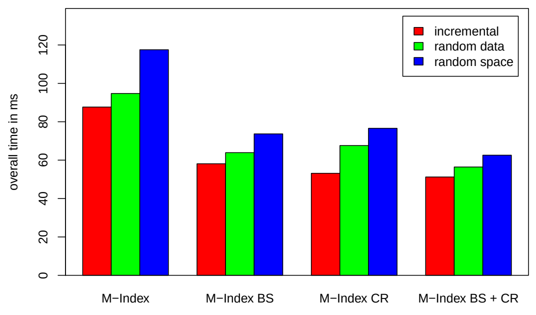


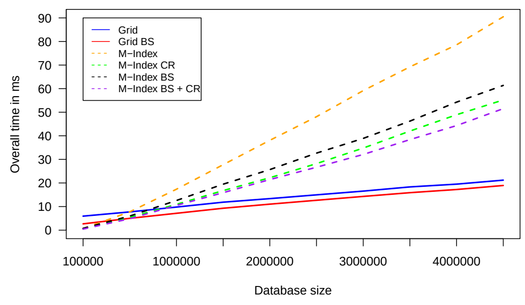
Figure 5.4 The comparison of the pivot selection techniques.

5.5 Index Performance

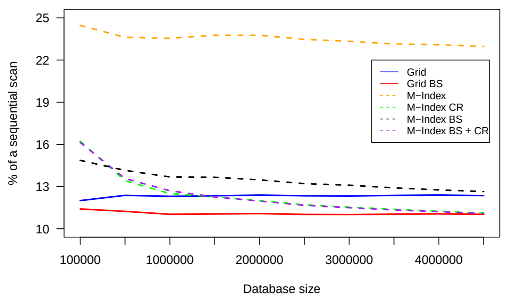
The efficiency of the index variants was evaluated under the growing database of up to 4.8 millions of *centroids*. The performance in the terms of the average overall time needed to process a *range query* is depicted in Fig. 5.5a. Note that the *Bounding Sphere Constraint* brings decent performance improvement to the Grid Index as well as to the M-Index and M-Index CR.

Since the Grid Index uses fixed data structures independently on the database size, visiting bins demands a constant time. This is not the case of the M-Index where the number of buckets grows with the database size causing the cost of their traversing to reflect this trend. As a result, for smaller databases with less than 500 000 *centroids* (i.e., almost 4 hours of video) the M-Index BS + CR performs better than the Grid Index despite the need of more distances to be computed (Fig. 5.5b) while for larger ones the Grid Index BS takes the lead.

As experienced users may be able to specify query *centroids* more precisely, i.e., with a minor *user error*, the query range can be possibly lowered without affecting



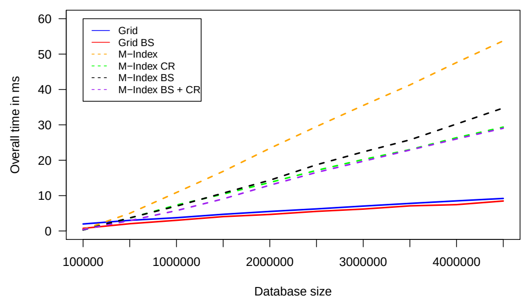
(a) The overall time.



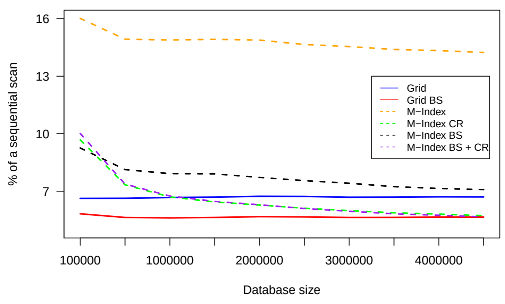
(b) The number of distance computations.

Figure 5.5 The performance of the index variants under growing database.

the search effectiveness. For this reason, we examined the index performance under 75% of the originally determined query range. The results are depicted in Fig. 5.6a and Fig. 5.6b. Differently from the full range, the *Bounding Sphere Constraint* provided only a little or no performance boost to the M-Index CR and Grid Index. Furthermore, the Grid Index BS is no longer outperformed by the M-Index in small databases.



(a) The overall time.



(b) The number of distance computations.

Figure 5.6 The performance of the index variants under 75% of the original query range.

5.6 Video Browser Showdown Participation

We participated at VBS in all the categories which were visual KIS in a single video/video archive and textual [4] KIS in single video/video archive. The length of a single video ranged from 10 minutes to 2 hours, and the video archive comprised total of 14 hours of video content [5]. The tools were evaluated in interactive and competitive way. The participants should perform the KIS tasks at the same time within 3 (single video) or 6 (video archive) minutes. The real-time results

[4] An Example of such textual description follows: *Three men are about to swim in a polar see. One of them has very colorful swimming suit.*

[5] As we were able to distinguish between wide and narrow screened video, we could split the video archive into two sub-archives comprising 6 and 8 hours of video for the visual KIS tasks.

were presented on a publicly visible screen (Fig. 5.7). It should be stressed out that the tool was controlled by an experienced user (in fact by the author of this thesis).

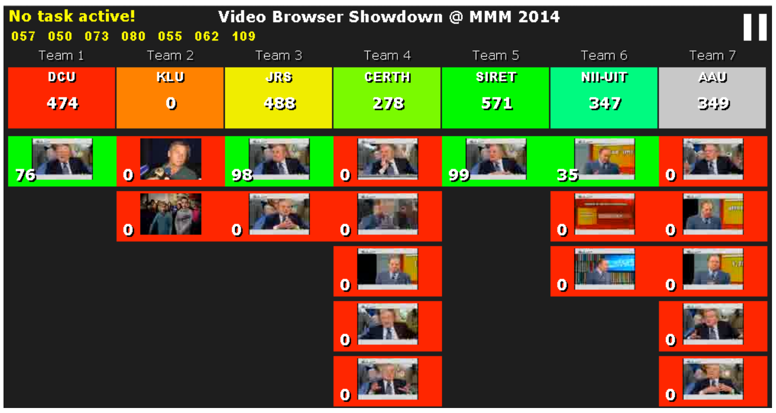


Figure 5.7 VBS – the interactive and competitive evaluation.

The performance of SBVB at VBS is summarized in Table 5.1. Clearly, the single video scope is not a problem for SBVB. For example, in the visual KIS within single video category, we were able to find the desired clip usually even before the playback ended. The video archive scope and especially a textual KIS is, however, more challenging. Luckily, all the textual descriptions contained some color clues enabling us to find the wanted clip in 6 out of 7 cases which is still quite impressive.

Category	Scope	Success	Avg. time
Visual KIS – single video	10 - 120 minutes	10/10	less than 1s
Textual KIS – single video	10 - 120 minutes	10/10	24.8 s
Visual KIS – video archive	6/8 hours	6/7	55.3 s
Textual KIS – video archive	14 hours	6/7	77.5 s

Table 5.1 The summary of the performance of SBVB at VBS.

Winning 3 out of 4 categories, SBVB clearly outperformed all the tools described in Chap. 1. We proudly enclose the award as Attachment A – VBS Award.

Speaking about effectiveness, we should also mention that during the collection of user-defined sketches and *centroids*, the novice users [6] were successful in ca 64% of visual KIS tasks within 2-8 hours of video.

[6]More than 40 first-time users were at first instructed and then also occasionally advised.

6. Conclusion

At first, we discussed the current state-of-the-art approaches for the video retrieval focusing on visual and textual KIS. Frequently, very complex procedures comprising SIFT or SURF features extraction, concept classification etc. are employed. These computationally expensive methods, however, do not ensure high effectiveness since for example, the concept classification, despite the progress, is not very precise in a general case.

In this thesis, we proposed a novel feature signatures based video retrieval model which simplicity contrasts with the previously discussed approaches. Video key-frames are represented with sets of color regions forming a simple 5-dimensional color-position feature space. This representation, also meaningful for humans, enables users to specify the color regions of the desired scene directly in simple colored sketches.

The important aspects of the model were carefully optimized; therefore, the model remains efficient even when dealing with tens of hours of video content. We paid special attention to the indexing techniques exploiting the nature of the feature space; more concretely, we evaluated the performance of the fixed Grid Index and several variants of the M-Index with a real dataset and user-defined queries.

Thanks to the low dimensionality and decent uniformity of the feature space, the Grid Index showed high performance especially in larger datasets even though it was optimized for a fixed dataset size. We believe that the Grid Index with gradual partitioning (i.e., dynamic Grid Index) might perform even better.

It may seem that the M-Index is not suitable for our model since it is outperformed by the Grid Index. Nevertheless, if we enhance the feature space as discussed in the following section, the Grid Index would be probably unusable (since the number of bins grows exponentially with the space dimension) while the M-Index would be still efficient.

We also demonstrated that even roughly computed BSs can provide additional filtering power to the processing of *range queries* in both the Grid Index and M-Index.

The proposed model was implemented in C# programming language and thanks to the optimizations, the tool can offer attractive and interactive UI, where every modification of the query causes immediate update of results. The performance of SBVB was evaluated at VBS. Despite the fact that SBVB clearly outperformed the tools of other participants, it is questionable whether the model would be effective in a general case. For example, it will probably utterly fail in providing effective searching within a black-and-white movie.

Anyway, we believe that the proposed model forms a solid basis for video retrieval systems focusing on KIS. This statement is supported by the successful participation at VBS. Also, the model can be probably further improved and we will now discuss the possibilities.

6.1 Future work

Video data contain way more information than the color distribution in static key-frames. For instance, the motion of the background and foreground can be possibly detected and associated with particular *centroids*. It is also desirable to utilize the weights of *centroids*. In our opinion, this information may play an important role and we plan to incorporate it in the model. Furthermore, the textures might be employed in the feature signatures as well. We should, however, keep on mind that the UI shall be maintained as simple as possible.

Next, we believe that the effectiveness of SBVB can be greatly improved with advanced results browsing. For example, results might be clustered based on visual similarity showing only one representative for each cluster and thus saving space for more results. In addition to that, results can be possibly organized in a different way, e.g., according to the files from which they are.

Bibliography

[1] Werner Bailer and Klaus Schoeffmann. Video search showcase (formely video browser showdown)@ONLINE, May 2014. URL `http://www.videobrowsershowdown.org/`.

[2] Werner Bailer, Wolfgang Weiss, Christian Schober, and Georg Thallinger. Browsing linked video collections for media production. In Cathal Gurrin, Frank Hopfgartner, Wolfgang Hurst, Håvard Johansen, Hyowon Lee, and Noel O'Connor, editors, *MultiMedia Modeling*, volume 8326 of *Lecture Notes in Computer Science*, pages 407–410. Springer International Publishing, 2014. ISBN 978-3-319-04116-2. doi: 10.1007/978-3-319-04117-9_47. URL `http://dx.doi.org/10.1007/978-3-319-04117-9_47`.

[3] Yuseok Ban, Sang-Ki Kim, Sooyeon Kim, Kar-Ann Toh, and Sangyoun Lee. Face detection based on skin color likelihood. *Pattern Recognition*, 47(4):1573 – 1585, 2014. ISSN 0031-3203. doi: http://dx.doi.org/10.1016/j.patcog.2013.11.005. URL `http://www.sciencedirect.com/science/article/pii/S003132031300455X`.

[4] B. Bustos, G. Navarro, and E. Chavez. Pivot selection techniques for proximity searching in metric spaces. In *Computer Science Society, 2001. SCCC '01. Proceedings. XXI Internatinal Conference of the Chilean*, pages 33–40, 2001. doi: 10.1109/SCCC.2001.972629.

[5] Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José Luis Marroquín. Searching in metric spaces. *ACM Comput. Surv.*, 33(3):273–321, September 2001. ISSN 0360-0300. doi: 10.1145/502807.502808. URL `http://doi.acm.org/10.1145/502807.502808`.

[6] Claudiu Cobârzan, MarcoA. Hudelist, and Manfred Fabro. Content-based video browsing with collaborating mobile clients. In Cathal Gurrin, Frank Hopfgartner, Wolfgang Hurst, Håvard Johansen, Hyowon Lee, and Noel O'Connor, editors, *MultiMedia Modeling*, volume 8326 of *Lecture Notes in Computer Science*, pages 402–406. Springer International Publishing, 2014. ISBN 978-3-319-04116-2. doi: 10.1007/978-3-319-04117-9_46. URL `http://dx.doi.org/10.1007/978-3-319-04117-9_46`.

[7] J.P. Collomosse, G. McNeill, and L. Watts. Free-hand sketch grouping for video retrieval. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4, Dec 2008. doi: 10.1109/ICPR.2008.4761466.

[8] Sanjoy Dasgupta. *The hardness of k-means clustering*. Department of Computer Science and Engineering, University of California, San Diego, 2008.

[9] A. Dyana and S. Das. Mst-css (multi-spectro-temporal curvature scale space), a novel spatio-temporal representation for content-based video retrieval. *Circuits and Systems for Video Technology, IEEE Transactions on*, 20(8): 1080–1094, Aug 2010. ISSN 1051-8215. doi: 10.1109/TCSVT.2010.2051367.

[10] Rui Hu and J. Collomosse. Motion-sketch based video retrieval using a trellis levenshtein distance. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 121–124, Aug 2010. doi: 10.1109/ICPR.2010.38.

[11] Nakamasa Inoue, Yusuke Kamishima, Toshiya Wada, Koichi Shinoda, and Shunsuke Sato. Tokyotech+ canon at trecvid 2011. In *Proceedings of NIST TRECVID Workshop*, 2011.

[12] Martin Kruliš, Jakub Lokoč, and Tomáš Skopal. Efficient extraction of feature signatures using multi-gpu architecture. In Shipeng Li, Abdulmotaleb Saddik, Meng Wang, Tao Mei, Nicu Sebe, Shuicheng Yan, Richang Hong, and Cathal Gurrin, editors, *Advances in Multimedia Modeling*, volume 7733 of *Lecture Notes in Computer Science*, pages 446–456. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-35727-5. doi: 10.1007/978-3-642-35728-2_43. URL `http://dx.doi.org/10.1007/978-3-642-35728-2_43`.

[13] Thomas Larsson. Fast and tight fitting bounding spheres. In *Proceedings of The Annual SIGRAD Conference*, pages 27–30, 2008.

[14] Leissi Castañeda León and Roberto Hirata Jr. Car detection in sequences of images of urban environments using mixture of deformable part models. *Pattern Recognition Letters*, 39(0):39 – 51, 2014. ISSN 0167-8655. doi: http://dx.doi.org/10.1016/j.patrec.2013.10.028. URL `http://www.sciencedirect.com/science/article/pii/S0167865513004200`. Advances in Pattern Recognition and Computer Vision.

[15] Suzanne Little, Iveel Jargalsaikhan, Rami Albatal, Cem Direkoglu, Noel E O'Connor, Alan F Smeaton, Kathy Clawson, Min Jing, Brian Scotney, Hui Wang, et al. Savasa project@ trecvid 2013: semantic indexing and interactive surveillance event detection. 2013.

[16] Stuart Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137, 1982.

[17] Jakub Lokoč, Adam Blažek, and Tomáš Skopal. Signature-based video browser. In Cathal Gurrin, Frank Hopfgartner, Wolfgang Hurst, Håvard Johansen, Hyowon Lee, and Noel O'Connor, editors, *MultiMedia Modeling*, volume 8326 of *Lecture Notes in Computer Science*, pages 415–418. Springer International Publishing, 2014. ISBN 978-3-319-04116-2. doi: 10.1007/978-3-319-04117-9_49. URL `http://dx.doi.org/10.1007/978-3-319-04117-9_49`.

[18] Jakub Lokoč, Juraj Moško, Přemysl Čech, and Tomáš Skopal. On indexing metric spaces using cut-regions. *Information Systems*, 43(0):1 – 19, 2014. ISSN 0306-4379. doi: http://dx.doi.org/10.1016/j.is.2014.01.007. URL `http://www.sciencedirect.com/science/article/pii/S0306437914000258`.

[19] 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`.

[20] Mike Matton, Alberto Messina, Werner Bailer, and Jean-Pierre Évain. The ebu mim-scaie content set for automatic information extraction on broadcast media. In *Proceedings of the 5th ACM Multimedia Systems Conference*, MMSys '14, pages 13–17, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2705-3. doi: 10.1145/2557642.2563676. URL http://doi.acm.org/10.1145/2557642.2563676.

[21] Anastasia Moumtzidou, Konstantinos Avgerinakis, Evlampios Apostolidis, and et al. Verge: An interactive search engine for browsing video collections. In Cathal Gurrin, Frank Hopfgartner, Wolfgang Hurst, Håvard Johansen, Hyowon Lee, and Noel O'Connor, editors, *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.

[22] ThanhDuc Ngo, VuHoang Nguyen, Vu Lam, Sang Phan, Duy-Dinh Le, DucAnh Duong, and Shin'ichi Satoh. Nii-uit: A tool for known item search by sequential pattern filtering. In Cathal Gurrin, Frank Hopfgartner, Wolfgang Hurst, Håvard Johansen, Hyowon Lee, and Noel O'Connor, editors, *MultiMedia Modeling*, volume 8326 of *Lecture Notes in Computer Science*, pages 419–422. Springer International Publishing, 2014. ISBN 978-3-319-04116-2. doi: 10.1007/978-3-319-04117-9_50. URL http://dx.doi.org/10.1007/978-3-319-04117-9_50.

[23] D. Novak and M. Batko. Metric index: An efficient and scalable solution for similarity search. In *Similarity Search and Applications, 2009. SISAP '09. Second International Workshop on*, pages 65–73, Aug 2009. doi: 10.1109/SISAP.2009.26.

[24] S. Padmakala, G. S. AnandhaMala, and M. Shalini. An effective content based video retrieval utilizing texture, color and optimal key frame features. In *Image Information Processing (ICIIP), 2011 International Conference on*, pages 1–6, Nov 2011. doi: 10.1109/ICIIP.2011.6108864.

[25] Jack Ritter. An efficient bounding sphere. In Andrew S. Glassner, editor, *Graphics Gems*, pages 301 – 303. Morgan Kaufmann, San Diego, 1990. ISBN 978-0-08-050753-8. doi: http://dx.doi.org/10.1016/B978-0-08-050753-8.50063-2. URL http://www.sciencedirect.com/science/article/pii/B9780080507538500632.

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

[27] K. Schoeffmann, M. Lux, M. Taschwer, and L. Boeszoermenyi. Visualization of video motion in context of video browsing. In *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*, pages 658–661, June 2009. doi: 10.1109/ICME.2009.5202582.

[28] David Scott, Zhenxing Zhang, Rami Albatal, and et al. Audio-visual classification video browser. In Cathal Gurrin, Frank Hopfgartner, Wolfgang

Hurst, Håvard Johansen, Hyowon Lee, and Noel O'Connor, editors, *Multi-Media Modeling*, volume 8326 of *Lecture Notes in Computer Science*, pages 398–401. Springer International Publishing, 2014. ISBN 978-3-319-04116-2. doi: 10.1007/978-3-319-04117-9_45. URL `http://dx.doi.org/10.1007/978-3-319-04117-9_45`.

[29] T. Sikora. The mpeg-7 visual standard for content description-an overview. *Circuits and Systems for Video Technology, IEEE Transactions on*, 11(6): 696–702, Jun 2001. ISSN 1051-8215. doi: 10.1109/76.927422.

[30] Bo Tian. *Bouncing Bubble: A fast algorithm for Minimal Enclosing Ball problem.* GRIN Verlag, 2012.

[31] F.H.C. Tivive and A. Bouzerdoum. A car detection system based on hierarchical visual features. In *Computational Intelligence for Multimedia Signal and Vision Processing, 2009. CIMSVP '09. IEEE Symposium on*, pages 35–40, March 2009. doi: 10.1109/CIMSVP.2009.4925645.

[32] Marko Tkalcic, Jurij F Tasic, et al. Colour spaces: perceptual, historical and applicational background. In *Eurocon*, 2003.

[33] Junjie Yan, Xuzong Zhang, Zhen Lei, and Stan Z. Li. Face detection by structural models. *Image and Vision Computing*, (0):–, 2013. ISSN 0262-8856. doi: http://dx.doi.org/10.1016/j.imavis.2013.12.004. URL `http://www.sciencedirect.com/science/article/pii/S0262885613001765`.

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

Glossary

BoVW Bag of Visual Words. 7

BS Bounding sphere. 14, 15, 26, 33

Feature signatures An adaptive and flexible representation of an image or video key-frame based on the color distribution. For more information see Section 2.1 or Figure 2.2. 3, 4, 8–10, 19, 26, 27, 33

Grid Index A *Vector space* indexing technique dividing the space into uniform cuboid-like bins. 13, 22–24, 26, 32, 33

Grid Index BS The Grid Index enhanced with the *Bounding Sphere Constraint* defined in Section 3.3. 15, 22–24

Known-Item Search (KIS) Kind of information search where we are familiar with at least a part of the searched document (item) – a video clip in our case. The video clip can be previously viewed (visual KIS) or just described textually (textual KIS). 3, 4, 7, 24–26, 32

M-Index The current state-of-the-art technique in indexing *Metric spaces*. More details are provided in Section 3.2. 13–15, 22–24, 26, 33, 36

M-Index BS The M-Index enhanced with the *Bounding Sphere Constraint* defined in Section 3.3. 15, 23, 36

M-Index BS + CR The M-Index enhanced with the *Cut-region Extension*[18] and the *Bounding Sphere Constraint* defined in Section 3.3. 15, 23, 36

M-Index CR The M-Index enhanced with the *Cut-region Extension*[18]. 13, 15, 23, 24, 36

MPEG-7 A set of standards for description and search of multimedia content. We name color layout, edge histogram, camera motion and motion trajectory as examples of defined visual descriptors. 7, 19

SBVB Signature-Based Video Browser. 4, 7, 16, 17, 20, 25–27, 33, 35

SIFT Scale-Invariant Feature Transform. 7, 19, 26

SURF Speeded Up Robust Features. 7, 26

SVM Support Vector Machine. 7

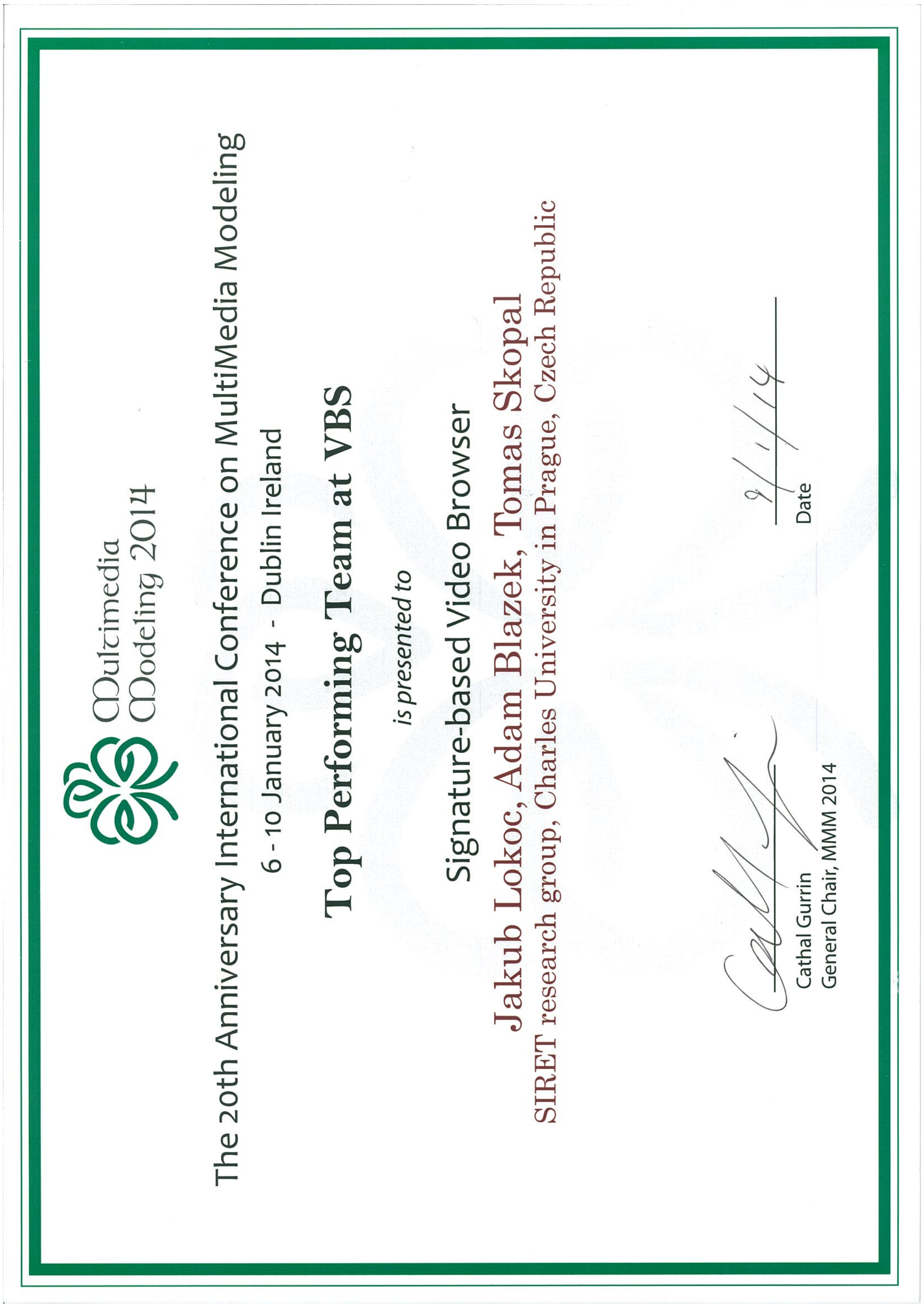
UI User interface. 16, 19, 26, 27

VBS Video Browser Showdown. 4, 7, 20, 24–26, 33

List of Figures

Attachments

A – VBS Award



Multimedia Modeling 2014

The 20th Anniversary International Conference on MultiMedia Modeling

6 - 10 January 2014 - Dublin Ireland

Top Performing Team at VBS

is presented to

Signature-based Video Browser

Jakub Lokoc, Adam Blazek, Tomas Skopal

SIRET research group, Charles University in Prague, Czech Republic

[illegible]

Cathal Gurrin
General Chair, MMM 2014

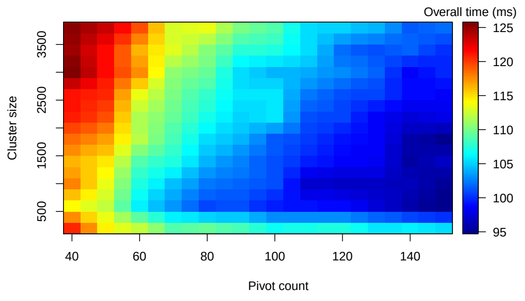
9/1/14
Date

B – CD

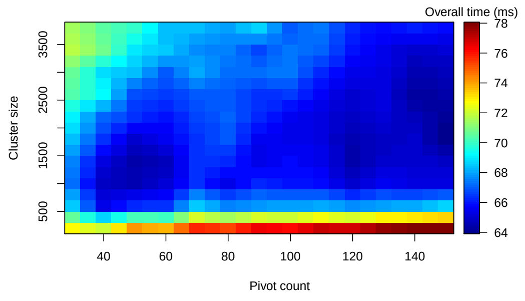
The attached CD contains the following:

- HTML documents describing its contents
- The SBVB sources and compiled binaries
- A list of the available SBVB settings and keyboard shortcuts
- An already indexed video file
- A brief discussion about the interesting parts of the SBVB code

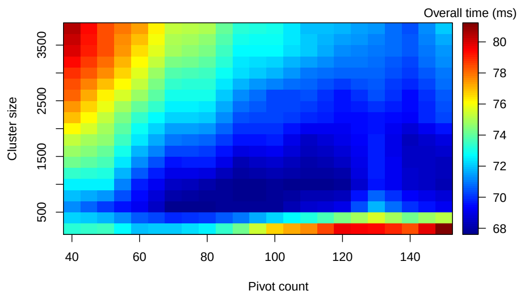
C – M-Index Parameters Optimization



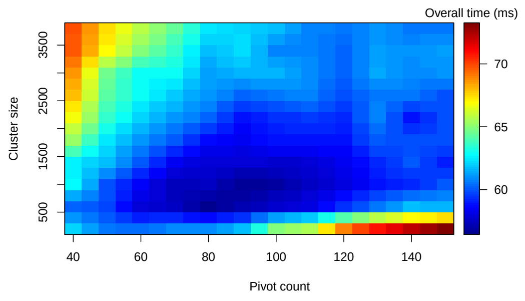
(a) M-Index.



(b) M-Index BS.

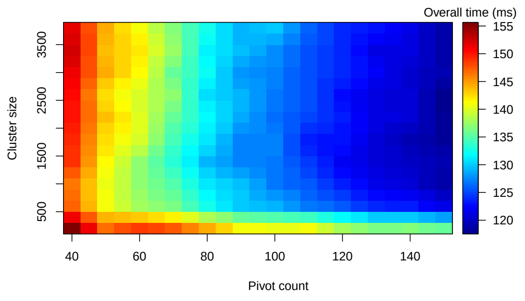


(c) M-Index CR.

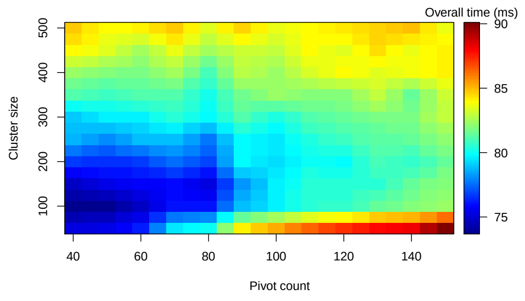


(d) M-Index BS + CR.

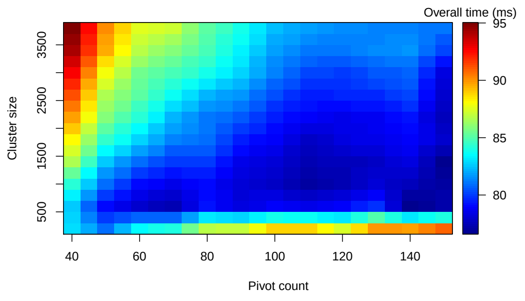
Figure C.1 The performance of the M-Index variants with the *random data* pivot selection under varying parameters.



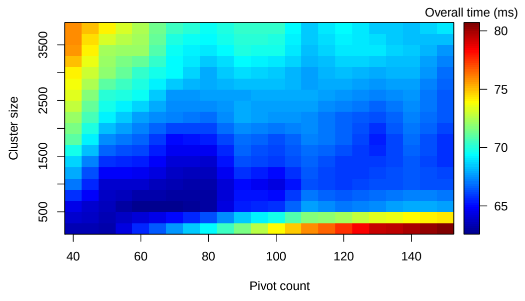
(a) M-Index.



(b) M-Index BS.



(c) M-Index CR.



(d) M-Index BS + CR.

Figure C.2 The performance of the M-Index variants with the *random space* pivot selection under varying parameters.