Charles University in Prague

Faculty of Mathematics and Physics

DOCTORAL THESIS

Mgr. Juraj Moško

Exploration of Multimedia Collections

Department of Software Engineering

Supervisor of the doctoral thesis:	doc. RNDr. Tomáš Skopal, Ph.D.
Study programme:	Computer Science
Specialization:	Software Systems

Prague 2015

First of all, I would like to thank my supervisor doc. RNDr. Tomáš Skopal, Ph.D. for his valuable guidance, and leading during my whole doctoral study and also for his advices, which always came at critical moments when I needed them most.

I would like to thank my consultant RNDr. Jakub Lokoč, Ph.D. for his inspirative ideas and that he consistently worked with me until the end. Without him, I would never have been able to finish my study.

I would like to express my gratitude to the other co-authors, Tomáš Bartoš, Přemysl Čech and Tomáš Grošup for cooperation on the papers.

I am very grateful for the endless support from my girlfriend Katka, who was with me during the final writing of the thesis and who always stand by me.

I would like to thank my family and especially my parents for their support and my brother Matej for small corrections.

Finally, I would like to thank my friends and all those people, who supported me and who prayed for me.

V prvom rade ďakujem svojmu školiteľovi doc. RNDr. Tomášovi Skopalovi, Ph.D. za to, že ma viedol počas celého doktorského štúdia, za jeho usmernenia, ktoré prišli vždy v kritických chvíľach, keď som ich najviac potreboval.

Ďakujem aj svojmu konzultantovi RNDr. Jakubovi Lokočovi, Ph.D. za jeho inšpirácie aj za to, že so mnou vytrvalo spolupracoval až do konca, bez neho by som doktorát nikdy nedokončil.

Ďakujem aj ostatným spoluautorom, Tomášovi Bartošovi, Přemyslovi Čechovi a Tomášovi Grošupovi za spoluprácu na článkoch.

Som veľmi vďačný za veľkú podporu od mojej priateľky Katky, ktorá bola pri mne pri záverečnom spisovaní práce, a ktorá stále stojí pri mne.

Ďakujem aj svojej rodine, hlavne rodičom za ich podporu a svojmu bratovi Matejovi za korektúru textu.

V neposlednom rade ďakujem aj svojim priateľom a všetkým tým, ktorí ma podporovali a ktorí sa za mňa modlili.

Annotation

Title:
Exploration of Multimedia Collections

Author:
Mgr. Juraj Moško
email: mosko@ksi.mff.cuni.cz

Department:
Department of Software Engineering
Faculty of Mathematics and Physics
Charles University in Prague

Supervisor:
doc. RNDr. Tomáš Skopal, Ph.D.
Department of Software Engineering
email: skopal@ksi.mff.cuni.cz

Abstract:

Multimedia retrieval systems are supposed to provide the method and the interface for users to retrieve particular multimedia data from multimedia collections. Although, many different retrieval techniques evolved from times when the search in multimedia collections firstly appeared as a research task, not all of them can fulfill specific requirements that the multimedia exploration is determined for. The multimedia exploration is designated for revealing the content of a whole multimedia collection, quite often totally unknown to the users who retrieve data. Because of these facts a multimedia exploration system has to solve problems like, how to visualize (usually multidimensional) multimedia data, how to scale data retrieval from arbitrarily large collections and how to design such an interface that the users could intuitively use for the exploration.

Taking these problems into consideration, we proposed and evaluated ideas for building the system that is well-suited for the multimedia exploration. We outlined the overall architecture of a multimedia exploration system, created the Multi-Layer Exploration Structure (MLES) as an underlying index structure that should solve problems of efficient and intuitive data retrieval and we also proposed definitions of exploration operations as an interactive and intuitive interface that the users can use for the multimedia exploration.

We integrated the MLES into the web-based multimedia exploration system and enhanced its visualization framework with our defined exploration operations. On this exploration system we performed the evaluation of properties of the MLES and the multimedia operations engaging the real users so we were able to discuss results from this evaluation in the extensive user study.

Keywords:
multimedia databases, similarity search, similarity indexing, multimedia exploration, approximate search, MLES

Anotace

Název práce:
Explorácia multimediálnych kolekcií

Autor:
Mgr. Juraj Moško
email: mosko@ksi.mff.cuni.cz

Katedra:
Katedra softwarového inženýrství
Matematicko-fyzikální fakulta
Univerzita Karlova v Praze

Školitel:
doc. RNDr. Tomáš Skopal, Ph.D.
Katedra softwarového inženýrství
email: skopal@ksi.mff.cuni.cz

Abstrakt:

Vyhľadávacie systémy nad multimediálnymi databázami sú určené k poskytnutiu metód a užívateľského rozhrania pre užívateľov, ktorý chcú získať konkrétne dáta z multimediálnej kolekcie. Od čias, keď sa vo vede prvý krát objavila problematika získavania dát z multimediálnych kolekcií, vzniklo veľa rôznych vyhľadávacích techník, ale nie všetky z nich sú vhodné pre špecifické podmienky, ktoré predpokladá multimediálna explorácia. Multimediálna explorácia je určená na odhaľovanie obsahu celej multimediálnej kolekcie, veľmi často úplne neznámej užívateľom, ktorý chcú získať informácie o tom, čo kolekcia obsahuje. Na základe týchto faktov, multimediálny exploračný systém musí riešiť problémy ako zobraziť (zvyčajne viacdimenzionálne) multimediálne dáta, ako škálovať problematiku získavania dát z ľubovoľne veľkých kolekcií a ako navrhnúť také užívateľské rozhranie, ktoré by užívatelia mohli intuitívne využiť pri explorácii multimediálnych kolekcií.

Ako odpoveď na tieto problémy sme navrhli a vyhodnotili naše idey pre vytvorenie vhodného systému pre multimediálnu exploráciu. Načrtli sme celkovú architektúru (všeobecného) multimediálneho exploračného systému, vytvorili sme viacvrstvovú exploračnú štruktúru Multi-Layer Exploration Structure (MLES) ako indexovú štruktúru, ktoré by mala vyriešiť problémy efektívneho a intuitívneho získavania dát z multimediálnych kolekcií, a tiež sme ponúkli definície exploračných operácii ako podporu interaktívneho a intuitívneho rozhrania, ktoré môžu užívatelia použiť pri explorácii.

Vytvorenú štruktúru MLES sme integrovali do webového multimediálneho exploračného systému a zároveň sme obohatili jeho zobrazovaciu komponentu nami definovanými exploračnými operáciami. Pomocou tohoto exploračného systému sme vykonali evaluáciu vlastností exploračnej štruktúry MLES a aj exploračných operácii so zapojením reálnych užívateľov, takže sme boli schopní vyhodnotiť výsledky z evaluácie v rozsiahlej užívateľskej štúdii.

Klíčová slova:

multimediálne databáze, podobnostné vyhľadávanie, podobnostné indexovanie, multimediálna explorácia, aproximatívne vyhľadávanie, MLES

Contents

Preface

The enormous amount of data collections spread over the Internet has given an impulse for research of ideas how to store and access such data. These data were produced in different ways and from various devices, they had not a common structure, but if we want to include all kind of data under some specific designation, these data are referred to as *multimedia data*.

We can meet different sort of collections containing multimedia data everyday. They come to us, especially, as some form of entertainment in a television, cameras, social networks, but they are also a part of our everyday and work life as medical records, video records from security cameras or histories of share prices from stock markets. However, the multimedia collections are not bound to these areas and in future, we can suppose continuous growth of the multimedia collections of any sort and that leads to the larger and generally unknown collections. These facts and the latter assumption arise into questions how to store, access and search such collections. A common character of these collections is that they are generally unstructured, as opposed to data stored in relational databases or document management systems, so there is need for different access methods than, e.g., structured query languages. One traditional approach for accessing data in the multimedia collection is so called *query by example* scenario. In this scenario, a user who wants to access data in the collection, formulates his[1] intent with a query object and he expects to get relevant (usually similar) data to this query object. But in some multimedia collections, where their content is rather general than specific, e.g., images from a social network, it is hard to express the querying intent using a query object, because the user does not know what kind of objects the collection contains. Hence, it is more intuitive to *browse* such a collection, where the query object provided by the user in advance is not necessary.

Information used for browsing in multimedia collections can be stored next to data objects in some text annotations or in any other meta-data, but as these are optional the only information that browsing systems often have is the content of data itself. In such a case, it is necessary to create a model of the multimedia collection, which defines relations between particular data objects. Through the relations in the modeled multimedia collection the user can browse from one view of the collection to another. The relations between the data objects are usually based on some kind of differences or similarities between particular objects of the collection.

We can say that in our research topic, browsing, or better term, the *exploration*, is a continuous process of revealing the content of an explored collection.

[1]A user can be 'he' or 'she', however, for better readability we will refer to a user as to 'him' in the whole thesis.

Thus, there is nothing like a final result of the retrieval process, the user is getting the results in form of new information that he obtains in each step of the exploration. Due to this fact, the methods how the user is performing his exploration steps are more important than precision of the exploration process. Methods for the exploration steps, we refer to them as to *exploration operations*, should navigate the user through the collection in a direction he wants to go – from more general to more specific concepts or vice-versa, or to navigate to (similar) objects within the same level of details.

Summary of Contributions

In our research, we study properties of retrieving information from a multimedia collection, where a user does not generally know about its content. More specifically, we try to provide the answers to the user who stays in front of the tasks: to extract information, or to get a global overview of such a collection its content he previously does not know about.

We also contribute with a survey of exploration and browsing systems and techniques that were proposed so far.

We propose the architecture of a general multimedia exploration system, where we study data flow from user requests to a data layer where the data collection resides. Next we study how to evaluate these requests and also partly how to schedule them.

We define the multilayer model over an explored collection, the *MLES*, a hierarchical exploration structure that enables the user to get the overview of a specific area in the multimedia collection within different levels of details. In this model we define exploration operations *Zoom-In*, *Zoom-Out* and *Pan* that allow the user to navigate through the collection.

The user is usually browsing/exploring the collection using some multimedia exploration system, so we implement our multilayer model into a multimedia exploration system, where we study problems of visualizing and mapping a multidimensional space into 2 dimensions of a computer screen and we study also the user interaction with the system.

Structure of Thesis

The thesis is divided into 6 chapters. The first chapter overviews basic terms and methods of multimedia retrieval and similarity search. Chapter 2 introduces principles of the multimedia exploration and compares it to the traditional scenario of query by example. Following the introduction of the multimedia exploration, in Chapter 3 we put down our ideas about the architecture of a multimedia exploration system. The principles of our general solution for the multimedia exploration are outlined in Chapter 4. Experimental evaluation and user study are summed up in Chapter 5, where experiments on the proposed solution are evaluated. The conclusion of the thesis is summarized in the Chapter 6.

Chapter 1

Multimedia Retrieval

When speaking about multimedia data, we refer to any unstructured data like text documents, image collections from social networks, medical records - images, videos or also cardiographs, movies, videos from security cameras, stock market records, music/speech records, and so on. A common characteristic of these data collections is the fact that unlike relational data, multimedia data are inherently *unstructured*. The only information that multimedia data generally have is their content, the raw data itself. Therefore, multimedia databases are studied in a specific research area which can use this content information in processing of the multimedia collections, the *content-based multimedia retrieval (CBMR)*.

The CBMR, as its name says, includes retrieval techniques that are primarily based on the content of raw data and not on additional information, like text annotations. If we imagine the content of unstructured multimedia data as raw bytes, there have to be suitable data structures and methods developed, that can process this low-level raw information. For example, in a photo collection the raw information is a matrix of color pixels, or in a video collection we can also expect some additional information about the camera, like its motion and position. Subsequently, in the case of an image collection, a bit higher-level information could be extracted from the matrix of color pixels like the distribution of edges or texture information. Then, this extracted information - so-called data *descriptors* are used in content-based retrieval instead of raw data itself. The format of some general descriptors has been standardized in the MPEG-7 standard [1], but many more complex descriptors are still being proposed, trying to improve performance of retrieval, e.g., feature signatures [2], or descriptors extracted by deep neural networks [3], which become a groundbreaking solution in recent years.

As the multimedia collections are generally unstructured, it is hard to query them with some structured query language which is a traditional way in relational databases. Instead, a model representing the space of specific multimedia data is created from a particular collection, allowing the retrieval of objects from the collection. For the collection represented by the model the exact match, popular in relational databases, is not very suitable, due to the absence of a query language, strong-type data structure, and thus also data semantics. Hence, new query approaches are required for querying the collections like, for example, searching based on *similarity*, which is the main concept of research area referred to as the *similarity search* [4].

1.1 Similarity Search Paradigm

As we mentioned in the previous paragraphs, raw data of an unstructured multimedia collection are transformed into data descriptors of features, where this transformation from an object space Obj to a descriptor space $\mathcal{U}$ is called *feature extraction*:

$$\varphi : Obj \rightarrow \mathcal{U} \tag{1.1}$$

The extracted descriptors usually have a format of a numerical vector, but other formats are also possible as it is, for example, in case of strings used to describe the primary structure of DNA molecules.

1.1.1 Similarity Modeling

The format of extracted descriptors and the definition of a descriptor space represents the first component of a model in the similarity search. The other component consists of the definition of a proximity measure. Retrieval in the similarity search is based on a proximity search, where the proximity is understood here as either a *similarity* between objects in a collection or a *dissimilarity* between them, also known as a *distance*. This proximity measure determines in what relation two objects of the feature space are, where magnitude of this relation is determined by the proximity function δ:

$$\delta : \mathcal{U} \times \mathcal{U} \rightarrow \mathbb{R} \tag{1.2}$$

In the case when the proximity measure is the similarity, a higher value means higher similarity and vice versa for the distance. In further reading, we will use *distance function* as it allows to intuitively illustrate the search problem as a geometrical problem (searching certain region in space).

The most used classification of models in similarity search is division into those based on the theory of metric spaces, referred to as metric space models and the others, which are referred simply as non-metric space models. When the theory of metric spaces is applied in similarity modeling, it presumes a *metric space* $(\mathcal{U}, \delta)$ consisting of a descriptor domain $\mathcal{U}$ and a distance function δ, which has to satisfy the following metric axioms:

non-negativity	$\forall x, y \in \mathcal{U}, \delta(x, y) \geq 0$
symmetry	$\forall x, y \in \mathcal{U}, \delta(x, y) = \delta(y, x)$
identity	$\forall x, y \in \mathcal{U}, x = y \iff \delta(x, y) = 0$
triangle inequality	$\forall x, y, z \in \mathcal{U}, \delta(x, z) \leq \delta(x, y) + \delta(y, z)$

A knowledge about the metric axioms can be exploited in creation of the similarity model and afterwards, in the process of retrieval. As two different models can be created over the same collection, they can be compared with regard to their retrieval performance. When it is considered which model is more effective, we refer to their *effectiveness* - how is the model precise in comparison to the reality. The second useful measure is *efficiency*, which considers how fast can model be in the retrieval process. Sometimes, the metric axioms tend to be very strict, therefore to improve effectiveness, the non-metric models [5, 6, 7, 8, 9], which do not satisfy all metric axioms can be also useful. They usually try to relax the metric axioms to reach some level of trade-off between efficiency and effectiveness.

1.2 Query by Example

As we mentioned before, an exact search is not very suitable for searching in a space that is based on similarity model. Extracted features are usually too complex and often in such a format that is not understandable by a human. Therefore, it is hard to state a query with use of any query language as we are accustomed from searching in relational databases.

As the answer to these facts, similarity modeling specifies a suitable alternative – *similarity querying* – a retrieval that is based on similarity and a specific *query by example* scenario. As its name indicates, searching in the query by example scenario requires an input element, an example query object, which can, but does not necessarily have to be from a queried collection. In fact, a search where the query object is not from the queried collection appears to be intuitively more useful, because the user gets only new information from the result of his search as the query object itself is not present in the result. Beside the query object, the next search parameter in the query by example is a restrictive condition which is put on the objects that the search returns. In the following paragraphs we outline basic types of similarity queries and their variants.

For the following definitions we presume a descriptor space $\mathcal{U}$, its subset $\mathcal{S} \subseteq \mathcal{U}$ as the queried collection and the query object $q \in \mathcal{U}$.

1.2.1 Range Query

A range query puts the restrictive condition on the value of distance from the query object, it returns all objects, of which their distance from the query object q is at most as the value of the query radius r. Formally, the range query for the query object q and the query radius $r \in \mathbb{R}$ is defined as:

$$\mathcal{R}(q, r) = \{o \in \mathcal{S}, \delta(o, q) \leq r\} \tag{1.3}$$

A useful application of the range queries can be used, for example, in the text search area, to solve type of problems like: *Give me the english words that are not more different from the word 'word' than in change/insert/delete of a single letter.* The range query returns among others also the words: 'world', 'sword', 'wood'.

From the above definition it is clear that the number of the returned objects is not known in advance until the whole search is evaluated. That is not very practical in many applications, hence to solve this problem the following type of the similarity query is an alternative.

1.2.2 Nearest Neighbor Query

A nearest neighbor query just returns an object from the queried collection that is the nearest one to the query object. However, more useful is its variant of k-nearest neighbors ($k\mathcal{NN}$) query, which also put the restrictive condition on the query result similarly as the range query, but instead of the query radius, the number of objects k in the final query result is stated. Formally, the $k\mathcal{NN}$ query for the query object q and the number k of wanted objects in the result is defined as:

$$k\mathcal{NN}(q) = \{\mathcal{R} \subseteq \mathcal{S}, |\mathcal{R}| = k \wedge \forall x \in \mathcal{R}, y \in \mathcal{S} - \mathcal{R} : \delta(q,x) \leq \delta(q,y)\} \qquad (1.4)$$

Applications of the $k\mathcal{NN}$ queries can be, for example, e-shops, which recommend similar products to the one currently shown, in such a task it is desired to have a fixed number of recommended objects.

One of consequences of the restriction which is put on the number of objects in the query result is the fact that there can still exist objects not returned by the $k\mathcal{NN}$ query with the same distance as some of the returned objects. Applications that employ the $k\mathcal{NN}$ queries should take this fact into account.

1.2.3 Other Query Approaches

Range and $k\mathcal{NN}$ queries are basic types of the similarity queries that are used in the query by example scenario, but other query approaches are derived from them. For example, both types of queries can be combined into the variant where the limitation for the number of objects in the result is used beside the restriction on the query radius [4]. Another interesting variant of the $k\mathcal{NN}$ query is an incremental similarity search [10, 11, 12], where the search algorithm is designed in the way to return more nearest neighbor objects also after the first k objects in the result. It can be useful for applications when a user is not satisfied with the first returned objects and want to see the next, possibly less similar, ones.

1.3 Indexing

In the previous, we mentioned two measures that reflect a quality of similarity queries, the first one, effectiveness, is connected to the precision of a query and the second one, efficiency, is linked to the speed of query evaluation. For improving effectiveness it is usually necessary to improve the similarity model, while the basic method for improving efficiency of querying is indexing. As indexes in relational databases improve performance of query evaluation, similarly we can use indexes to improve similarity querying in multimedia databases.

If you think about the evaluation algorithm of a range query, you will realize that for making a decision if a particular object is or it is not within the query radius, a distance between this object and the query object should be computed. Hence, for complete evaluation of the range query such a *distance computation* (DC) should be determined for each object in the queried collection. This fact leads to the computationally expensive evaluation, unless some of distances can be determined from the ones which were evaluated previously. And that is exactly the basic principle of *similarity indexing*.

An indexing technique that is extensively used in the similarity search is *metric indexing* [13, 4], which utilizes the existence of metric axioms. The fundamental principle of metric indexing is *estimation of lower bounds* of real inter-object distances with use of some precomputed distances, as depicted in Figure 1.1. The smaller circle represents the query ball where all objects meeting a query condition reside, $q \in \mathcal{U}$ is the query object, $r \in \mathbb{R}$ is the query radius, $o \in \mathcal{S}$ is some object from the collection compared with the query condition and $p \in \mathcal{U}$

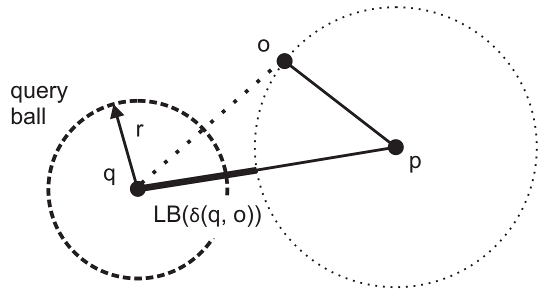


Figure 1.1: The lower-bounding principle.

is a reference object, selected in advance. The whole principle is based on the previously computed real distances between the query object q and the reference object p and also between the database object o and the reference object p. With knowledge of these distances we can compute the triangular lower bound distance

$$LB(\delta(q,o)) = |\delta(q,p) - \delta(o,p)| \tag{1.5}$$

of the real distance $\delta(q,o)$ between the query object q and the database object o. If this lower bound distance is greater than the query radius r, the real distance does not have to be computed, because the object o is definitely out of the query ball. Obviously, this kind of filtering can be very useful in cases when the computation of the lower bound distance is computationally less expensive than the computation of the real distance as it is, for example, in the cases of the Signature Quadratic Form Distance [14, 15] and the Earth Mover's Distance[16].

1.3.1 Metric Access Methods

Advantages of metric indexing are exploited in the index structures called *metric access methods* (MAMs) or *metric indexes*. Their general purpose is to minimize costs of query evaluation, where beside well-known I/O costs, they are designed to minimize also the number of distance computations, usually with use of the principles mentioned in the previous paragraph. The MAMs usually rely only on the distance function δ, without knowledge of the exact structure of indexed features. Many MAMs for different applications were designed in last two decades, we return back to them later in the next chapter in Section 2.2, where we examine some of well-known MAMs.

1.4 Motivation for Exploration

So far, we discussed the query by example scenario, but multimedia retrieval does not have to be used only in the scheme *formulate query -¿ get results*. Enormous growth of mobile devices, especially those with touch screens, leads to more continuous and interactive retrieval scenarios. The mobile devices offer a simple way to create multimedia data, e.g., with a built-in camera people can take a photo, or capture a video. Such created data can be simply viewed directly on the mobile device or they can be uploaded to some collection on the Internet, e.g., to social networks. Since multimedia collections can be nowadays created so fast and so simply, typical requirements for multimedia retrieval have also changed. In the next four points we try to review the requirements for a new retrieval scenario, the *multimedia exploration.*

User Interface

Firstly, due to rapid growth of social networks and mobile devices, the user of a retrieval system can be almost anyone. Hence, the system should provide a *simple intuitive user interface* to satisfy any kind of a user from a technique expert to a computer laymen. Similar suggestions offers the survey by Lew et al. [17] aimed on a future direction of content-based retrieval, their conclusions among others also include *human-centric methods* and an *interactive search.*

Starting Point of Retrieval

Secondly, the user who performs retrieval in a multimedia collection typically does not know in advance what he is searching for, the collection can be totally unknown to him. In such a case, the user can hardly state any query object, instead, he should be provided with some options that give him *a direction where to start.* This starting suggestion should be a representative sample of the whole collection as the retrieval system cannot generally suppose what the user is interested in.

Continuous Re-retrieval

Since the retrieval task in the multimedia exploration consists of uncovering what the whole collection contains, it cannot be usually satisfied with evaluating one single query, especially in case when the collection is very large. Thus, it is likely that the *process* of retrieval *will continue* when first query results are returned, by that time already with some knowledge about the objects from the first results.

Efficiency

The previous requirement of continuous retrieval assumes *efficient evaluation of similarity queries* in order to not discourage the user with long responses between particular steps of the search. In a single-step-like retrieval scenario, the user forgives a little delay of getting the results, but when it happens repeatedly in consecutive steps of continuous retrieval it can be very intimidating.

Taking these requirements into consideration, we proposed the general architecture of a *multimedia exploration system* [18], we describe it in more detail in Chapter 3. Before that, in the next chapter we continue examining assumptions of the multimedia exploration started in the previous paragraphs and look at the solutions of multimedia exploration systems proposed so far. Finally, in Chapter 4 we propose a multimedia exploration structure, the *MLES* [19], capable of the multimedia exploration in different levels of granularity.

Chapter 2

Related Work

The previous chapter ends with the motivation for a different scenario from the query by example that satisfies new requirements on multimedia retrieval. We mentioned that one of presumptions for such a scenario is the different character of a user retrieval task, the user does not want to retrieve concrete data as a response to his stated query. Instead, the user wants to explore, to find out what the multimedia collection contains and what kind of multimedia data are hidden in the collection. According to exploring behavior of the retrieval scenario, such a type of retrieval has got the name *multimedia exploration*.

The idea of the multimedia exploration was raised two decades ago and one work that was directly involved in was the paper by Santini and Jain with a rendering title *Beyond Query by Example* [20]. As the title indicates the authors of the work deal with limitations which a traditional query by example scenario has. They demonstrate those limitations on an exemplary search scenario, thus we demonstrate it in the same way. Try to imagine the following scenario: you have the task in your mind to search image collection for the houses with a gable red roof. You have an example object, so you query the collection by this example image. In a result, the retrieval system returns you a collection of images where some of them are relevant, they are the houses with the gable red roof, but many of them are not, they are, for example, desert dunes, tents or red houses with a flat roof. As Santini et al. remark this happens because the retrieval system uses the content of the images during the query evaluation, but it is not able to work with their semantics. The authors see the source of this problem in the different perception of an image content between the user and the retrieval system. In the previous exemplary scenario, the system returns the images with the desert dunes because they have similar shape to the gable roof. Similarly, the red houses with a flat roof were returned in the result, because the red color was dominant in both, the result and the query image - all images in the result are semantically similar to the example image according to the perception of the system, but according to user's perception, they are not. Hence, the author propose the solution, not to create the system that sees and understands all semantics in an images, but to build the system which provides the user with right tools enabling the exploration of the perceptual space of the database.

In the spirit of the previous observation, in the following paragraphs we discuss the particular limitations that the query by example scenario has.

Need of an *Example*. The very first problem of the query by example scenario is in its fundamentals, in the example query object. In many cases a user is not able to provide the example object, because he does not have a clear query intent in his mind, instead the user wants to find out what data the explored collection contains. For example, in the real life scenario, doctors are examining a collection of roentgen images with the non-clear concept in their minds, they do not know what they are looking for, but everything is made clear when they find *it*.

One-Dimensional Expression of Results. In most cases of querying systems, the query result is presented as a list of objects ordered with regards to their distance to the query object. Such ordering does not express the relationships between the particular items in the result collection. With knowing of connections between them, the user would better understand the system's perception of the queried collection.

Counterintuitive Relevance Feedback. In order to give feedback to the retrieval system, the users usually have some options to express which items of the result are relevant and which are not. For example, some objects can be explicitly marked as relevant, or the system can offer techniques for adjusting the used similarity function, e.g., in image collections, the users are able to put more emphasis on the color and less on the shape. But, if the user does not have an idea what kind of objects are stored in the collection, such adjustments are not intuitive for him. In such a scenario, better for him would be to continue searching in some direction with regards to one or more objects from the current result.

Non-Interactivity. This limitation is related to the previous ones. In the query by example scenario the user interacts with the system first by providing an example object and afterwards, when he evaluates relevance of the results, by rating particular images or adjusting underlying similarity. But for achieving more interactive feeling, it would be better to truly interact with the underlying similarity model, by providing tools for examination of the model and also tools for changing the model.

Considering the limitations of query by example, more and more ideas were transformed into proposals of browsing and exploration scenarios. Beside the individual browsing and exploration proposals, there were introduced surveys that try to summarize pros and cons of the proposed systems. In the significant survey by Smeulders et al. on the topic of content-based image retrieval [21], the authors address ability of a user to interact with a retrieval system together with visualization of an underlying similarity model – in the survey designated as the *query space*. Although the survey reviews also other concepts of content-based image retrieval, it is primarily aimed on query by example systems.

Six years later, in 2006, the work of Lew et al.[17] concludes five major challenges of content-based multimedia information retrieval and two of them include the *interactive search* and also the *experiential multimedia exploration systems which allow users to gain insight and explore media collections.*

Later, in 2008, next two surveys appeared, aimed more or less on the systems for browsing multimedia collections. First of them, the work of Datta et al. [22] is primarily focused on the systems for automatic annotation, but the authors discuss all aspects of image retrieval. They themselves consider their work as the successor of the work by Smeulders et al. [21] from the beginning of the decade. In Datta et al. [22] the authors believe that the intent of a user can act as a guideline for system design. They subdivide the user by clarity of his intent into three main categories:

- **Browser:** The user that browses the collection with no clear goal in his mind. In such a case the user session consists of more unrelated searches.
- **Surfer:** The user that browses the collection with no clear goal in the beginning, but with subsequent searches the clarity of what the surfer wants from the system tends to increase.
- **Searcher:** The user that searches the collection with very clear goal in his mind. The session of the searcher is typically short and leads directly to the end-result.

The authors conclude their categorization with the interesting idea: the image retrieval systems can gain wide acceptance if they will implement the *human-centered* perspective.

The second survey from the year 2008, by Heesch [23], can be considered as the first survey fully-oriented on multimedia browsing and exploration systems. On behalf of defending browsing, the author gives the interesting idea of the *mental query*, meaning that the very best version of the query exists only in the mind of the user. When the user tries to formulate it, he is limited by *means of expression* of a retrieval system. In the case of browsing, the query does not have to be formulated explicitly, the search of the user is driven by the *virtual query* formulated only in the mind of the user. In some cases, the user even does not have the query in his mind in the moment when he starts to search. Thereafter, the query intent is getting its shape only after the user gradually interacts with the retrieval system.

In 2011, Beecks et al. [24] state new challenges for the multimedia exploration of very large collections. They aim for performance of an exploration system with adaptability on one side and scalability on another. They think that high scalability of the exploration can be achieved by a *near-real-time index support* for evaluation of similarity queries. The authors conclude this idea with the statement that the interplay between user-centric similarity queries invoked by actions of the user and data-centric indexing methods should be optimized.

2.1 Exploration and Browsing Systems

2.1.1 Exploration versus Browsing

The term *exploration* is not commonly used as a name for the scenario of content-based retrieval, because its concepts are often confused with the term *browsing*. But from our point of view, there is a difference between browsing and exploration.

During browsing, the users typically have the idea in their minds, what they want to find, while in the exploration they do not. Therefore, browsing is categorized as a direct search[1], similarly like query by example, while the exploration is an indirect search. Similar differentiation of the term browsing from other concepts is contained in the definition of the term browsing in *BussinessDictionary.com*:

> **browsing** - Exploration of the World Wide Web by following one interesting link to another, usually with a definite objective but without a planned search strategy. In comparison 'surfing'[2] is exploration without a definite objective or a search strategy, and 'searching' is exploration definite in both objective and strategy [25].

Nevertheless, browsing and exploration have many properties in common. In the consecutive text we discuss ideas from both of these retrieval scenarios.

In the following, we introduce the survey of existing browsing and exploration systems beside the overview of browsing and exploration concepts that we consider crucial for the multimedia exploration. We focus on three aspects of the exploration systems:

- **Exploration structure**, how it is created and how it supports the exploration itself.
- **User interface**, especially on the options that the user has for interaction with the system.
- **Visualization**, how the explored collection is mapped to the space of the visualization.

2.1.2 Visualization, Mapping and Exploration Structures

The corner-stone of the multimedia exploration is the *visualization* of an explored collection. If the visualization is intuitive and truly reflects the reality it helps the user orient easier in the explored collection, and on the contrary if the visualization is inaccurate, the best of all performance properties of such a system happen to be useless. In the excellent work by Nguyen and Worring [26] about similarity-based visualization, the authors define three general requirements that are put on the system which visualize a multimedia (image) collection:

- **Overview requirement:** The visualization should give a faithful overview of the distribution of images in the collection.
- **Structure preservation requirement:** The relations between images should be preserved in the projection of the information space[3] to the visualization space.

[1]See Footnote 2 and a corresponding text in the following paragraph.

[2]Notice, that in comparison to categorization of users by Datta et al. [22] from the beginning of this chapter the terms surfing (exploration) and browsing are swapped. Since for content-based retrieval, the term exploration is more suitable than the term *surfing*, we will use the term exploration in the rest of the thesis.

[3]The information space here is meant the descriptor space.

- **Visibility requirement:** All displayed images should be visible to the extent that the user can understand the content of each image.

The all three requirements are not independent, i.e., when one of them is accomplished in a good degree the fulfillment degree of another can decrease. Hence, to design good, the reality demonstrating visualization means to find balance between these three requirements. As the authors propose, balancing can be achieved, for example, by defining a cost function for each of three requirements and then by optimizing the linear combination of these cost functions. For more information about the cost functions, you can revisit the proposal of the authors.

The fundamental question that should be answered when visualizing a result collection of some query or the whole multimedia database is the *mapping from a descriptor space to a space of visualization*. The visualization space is typically two- or three-dimensional, while the descriptor space is typically multidimensional. Hence, almost in every exploration system, visualization design leads to the problem of *dimensionality reduction*. A lot of research was done and many techniques were proposed on this subject, in the next paragraphs we introduce some of them.

One of the algorithms determined for dimensionality reduction is the *FastMap* [27], which is able to project objects from n-dimensional space to k-dimensional space, so in case when k is 2 or 3, it made it the suitable technique for visualization of the data collection. The main idea of the algorithm is based on projection of data objects on the geometric line between two reference objects which are far from each other. With consecutive applications of the idea the n-dimensional space can be reduced to the k-dimensional, where $k < n$. The advantage of the *FastMap* is in the fact that the algorithm is completely based on the distances between objects and not on the representation of the objects itself.

Another well-known technique used for projection from the descriptor space to the visualization space is the *Principal Component Analysis (PCA)* [28, 29]. The PCA algorithm is based on the mapping to objects the new orthogonal system where the variance along axis in the new system is maximized (from the largest variance to the smallest). In case when the dimensionality of the new system is lower then the initial one, the dimensionality reduction occurs. The PCA does not adhere to the requirement of structure preservation (see categorization above) because from its nature it is not preserving the mutual distances from the original space, it simply ignores the less important dimensions.

Unlike the PCA, the *Multidimensional scaling (MDS)* [30] is an algorithm that tries to preserve the mutual distances from the original space to the highest degree. Whereas computation of optimal mapping for large collections is expensive, the *incremental MDS* proposed by Basalaj [31] is less expensive, thus more suitable for the interactive application. The core principle of the incremental MDS is the cluster analysis of a mapped collection, which helps to avoid computations of distances between too similar objects. Another mapping used for dimensionality reduction, which is based on the MDS, is the *Sammon's mapping* [32]. While another one is the *self-organizing map (SOM)* [33], which is based on an artificial neural network and the MDS.

A visualization idea from the area of graph drawing, the *force-directed placement* [34], has also its origin in the MDS. The idea is based on a spring network, where the anchors are particular objects from a visualized collection and physical properties of the springs, the attractive and the repulsive forces between two anchors, are based on the similarity and the dissimilarity between the connected objects. The algorithm contains the optimization part utilizing different optimization methods, for example, the modifications of the original algorithm that use the optimization method called the *simulated annealing* is offered in the works by Fruchterman et al. [35]. and by Davidson et al. [36].

Nguyen and Worring [26] prefer methods for non-linear mapping like the *isometric mapping (ISOMAP)* [37], the *local linear embedding (LLE)* [38] and the *stochastic neighbor embedding (SNE)* [39]. The authors claim that these methods perform better in fulfilling the requirement of structure preservation than the PCA or the MDS and they use them in their exploration system, see Figure 2.1.

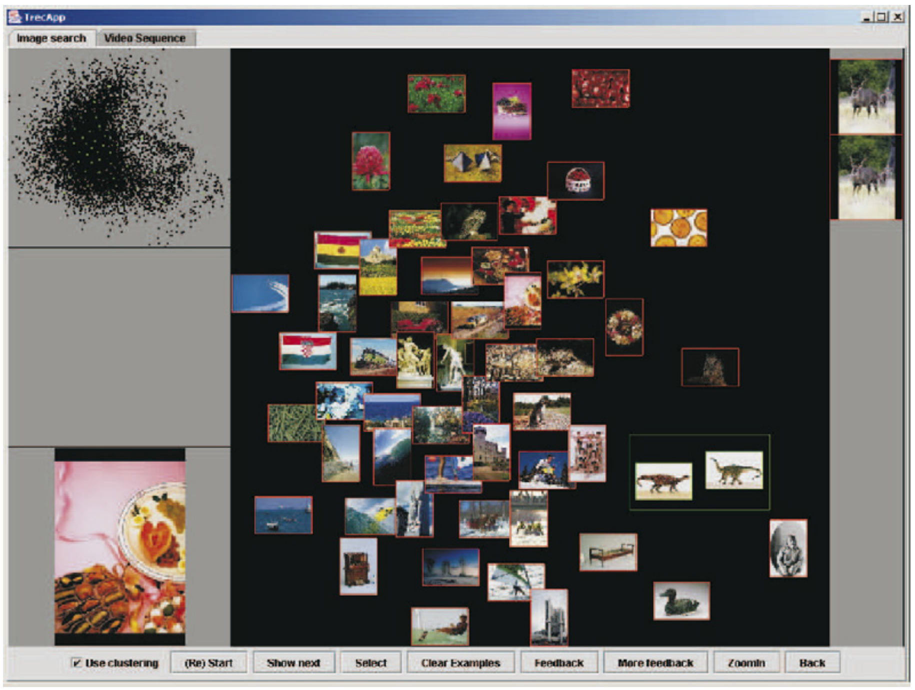


Figure 2.1: The layout of the exploration system by Nguyen et al., in the upper-left corner is depicted distribution of a whole collection in a visualized space, while the main screen displays images from a particular part of the space.

In the work by Pečenović et al. [40], the authors introduce the retrieval system *CIRCUS*, which integrates browsing and querying subsystem in the single user interface. The CIRCUS visualizes the underlying descriptor space represented as the precomputed visual map. The multidimensional descriptor space is projected into the 2-D map using the Sammon's mapping, where performance of the Sammon's algorithm is improved by the preprocessing step that prepares the initial configuration for the algorithm by using PCA projection. To show images in more detail the CIRCUS is supported by a hierarchical tree-based structure, created by repetitive k-means clustering. The constructed tree is kept balanced using a limit for the sizes of the clusters, where the limit is determined heuristically.

After mapping, the next issue that the visualization systems have to challenge is the *overlapping of visualized objects*. Nguyen and Worring [26], the authors of the visibility requirement from the beginning of this section, represent each rectangular image by a circle and try to eliminate overlapping of the circles. For simplification, they presume square images with the same height h of the image as its width w, i.e., $w = h$. Then the visible part of the image (the representing circle with its middle in the middle of the image and with the radius equals to $w/2$) covers $\pi/4 \simeq 80\%$ of the image, which is sufficient for visibility.

The question of overlapping is also answered in the visualization systems based on the force-directed placement. Desired overlapping is achieved by adjusting the repulsive forces between individual images. In the work by Rodden et al. [41], the authors use for visualization the force-directed placement next to the incremental MDS. They analyze the impact of similarity-based visualization on the speed of locating a target image. In their experiments, they compare similarity-based visualization with random distribution of images in a 2-D grid and the results of the similarity-based visualization results are better. But the interesting side result of the experiments is finding that the participants of the experiments prefer non-overlapping regular placement in the 2-D grid over more overlapping visualization because of the readability of the images displayed on the screen. On behalf of this observation, the authors propose the method that fits the force-directed placement in the regular 2-D grid, the differences can be seen in Figure 2.2.

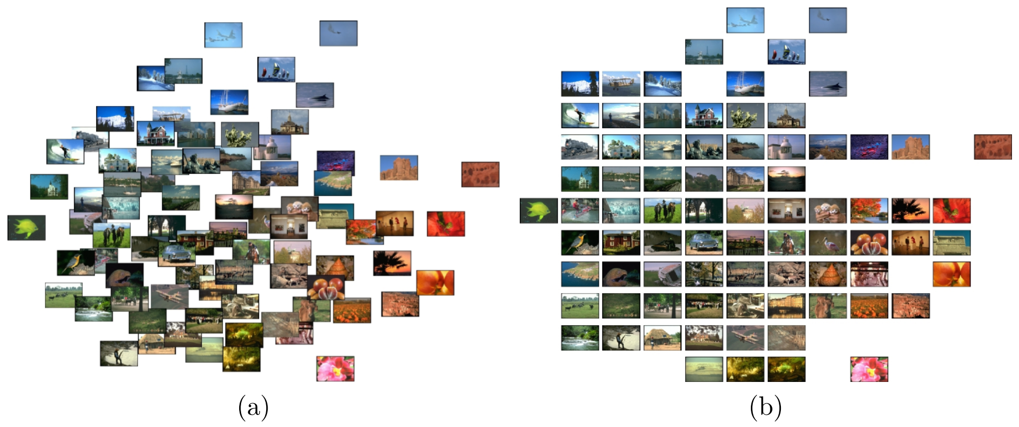


Figure 2.2: a) Visualization of 80 random images based on the force-directed placement b) the same collection rearranged in the 2-D grid in the post-processing step, the structure of the force-directed placement is largely preserved

Another system that use the force-directed placement in its visualization is the *SIR* [42] and its successors [43, 44, 45]. The attractive forces between the images are based on the similarities between these images and affect only neighboring images. The repulsive forces are set globally, they affect all images and they are not based on the similarity, they are modeled in the way to prevent overlapping of the images (see Figure 2.3).

Visualization based on the force-directed placement uses also the authors of the NN^k *Networks* [46], but only in the detail mode (when one image is in the center of the screen), see Figure 2.5. In the underlying structure, their solution connects one image to the images that are the nearest neighbors in at least one of the multiple descriptor spaces in which the images are represented. Each image is represented by a vector of weighted descriptors and the algorithm that the authors propose tries to find the neighbors of the image while it changes this weighting vector. All nearest neighbors are connected in the network, while further clustering is applied for visualizing only the representatives of some clusters.

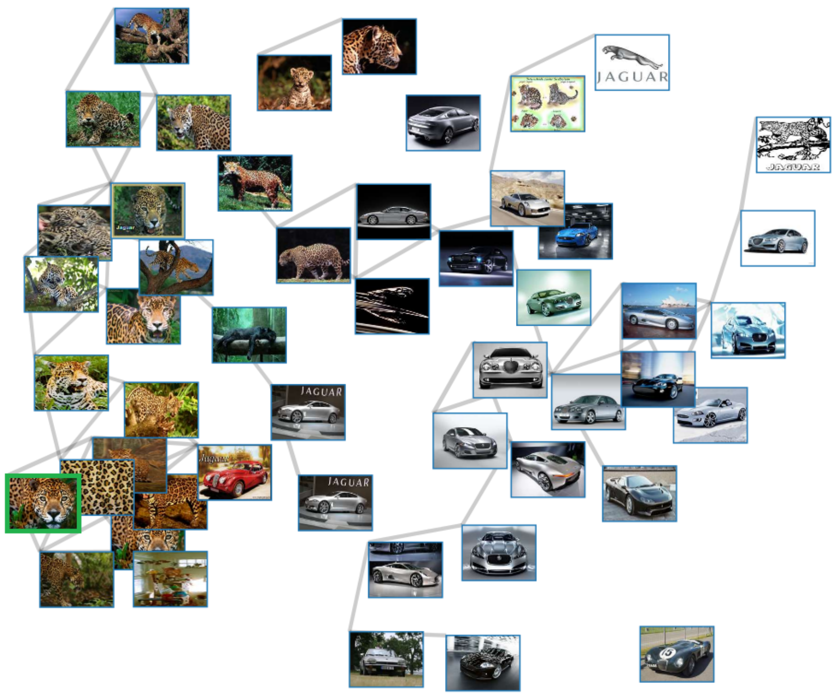


Figure 2.3: A force-directed placement layout of the SIR engine.

For clustering, they use the graph-based *Markov clustering* algorithm proposed in [47], see Figure 2.4.

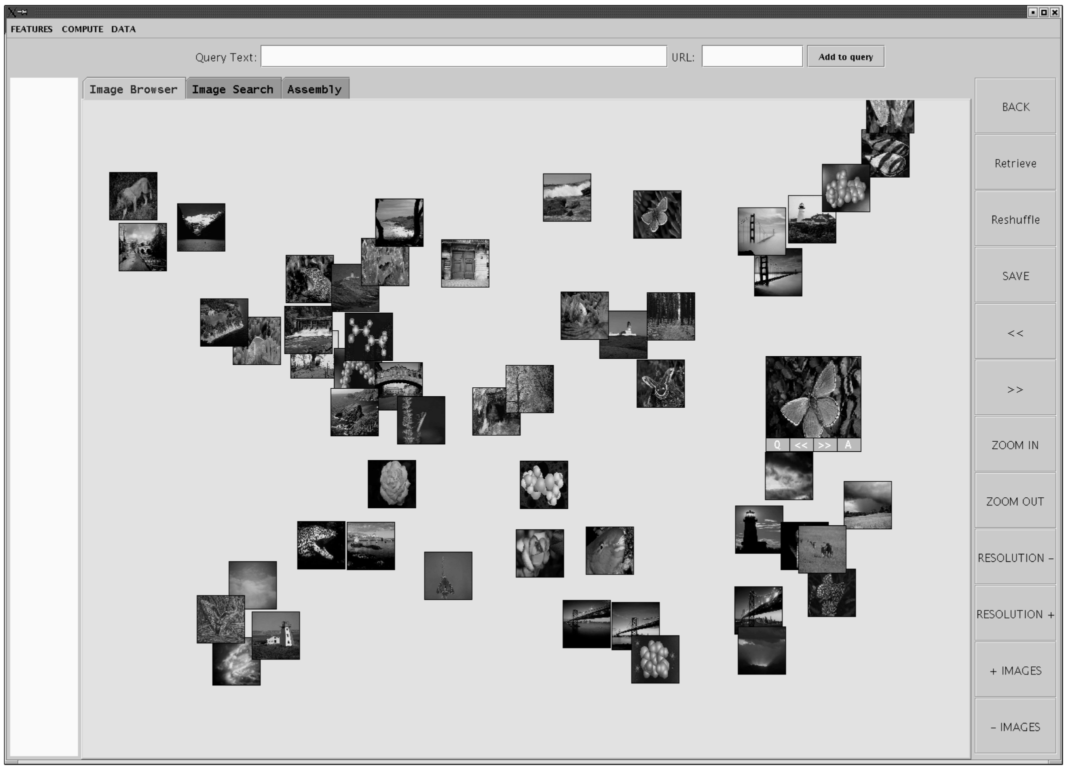


Figure 2.4: A visualized image collection clustered with the Markov Clustering algorithm.

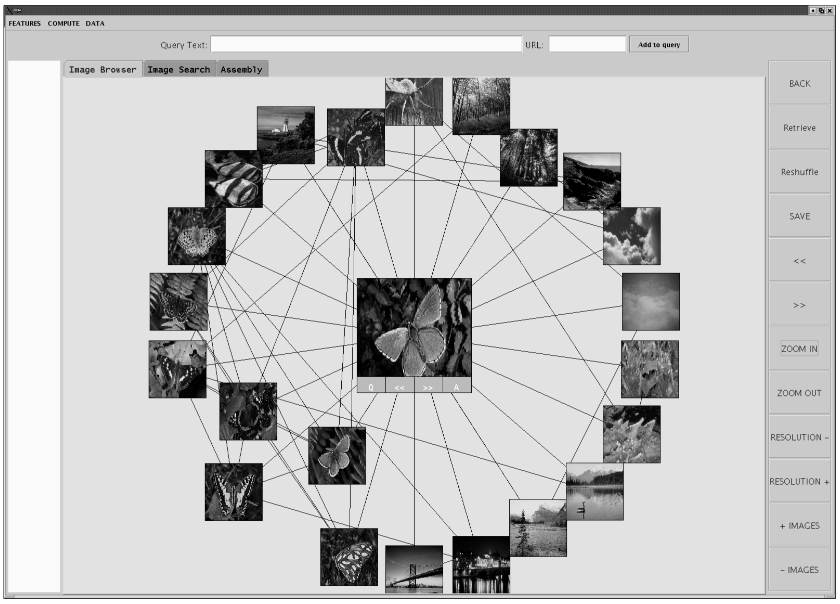
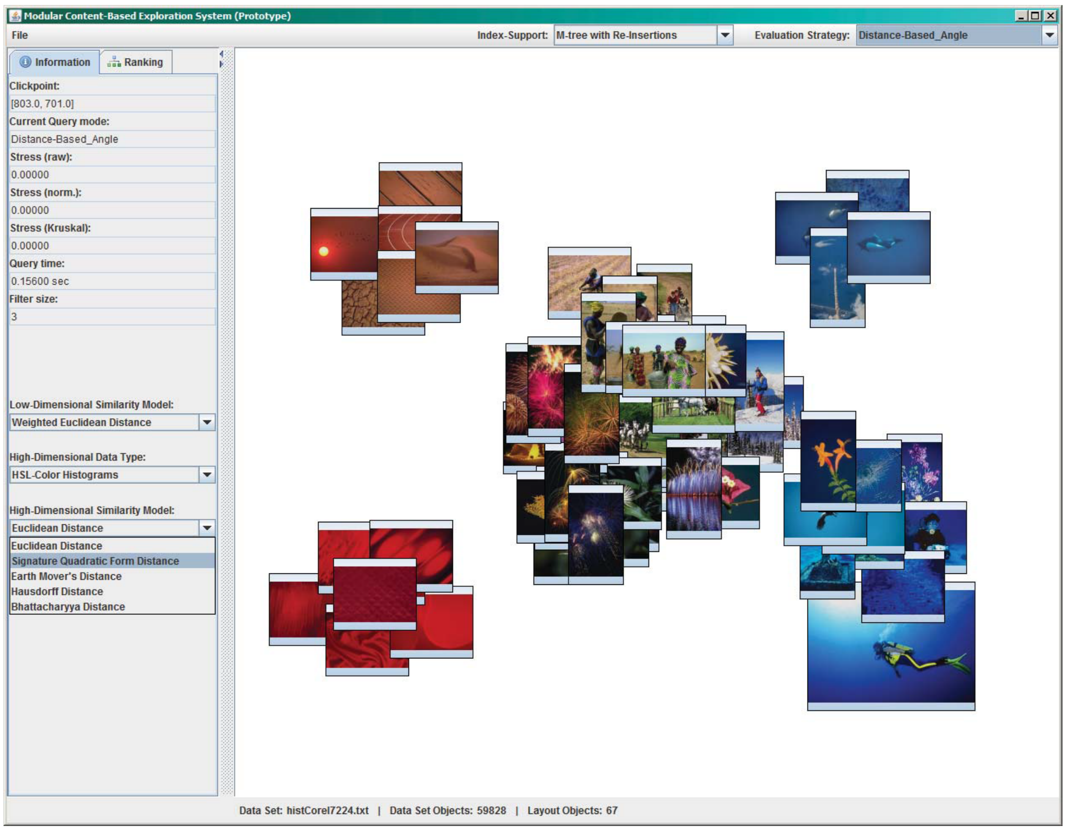


Figure 2.5: The local NN$^{\mathrm{k}}$ network around the chosen butterfly image in the middle.

The next work, by Liu et al. [48], proposes the visualization of images that are the result of some web search. With omission of the web environment, it can be generally considered as the visualization component of a querying system. The authors of the work suggest to map the images into a 2-D space involving the MDS and they contribute also with a research over image overlapping. Their research results in the visualization component where users are able to dynamically change the overlapping ratio and even to display the image collection in the regular grid.

Beecks et al. [49] describe the process of the multimedia exploration in four entities: the first is the *multimedia database* consisting of data themselves, the second is the *initial mapping process* that maps the multidimensional space to the space that has the number of dimensions which visualization can work with (usually two or three dimensions), the third one is the *visualization* itself and the fourth is the *user who is interacting* with the exploration system. In the consequent works by Beecks et al. [50, 24, 51], the authors face up challenges of the exploration of large collections and propose their prototype of the exploration system, see Figure 2.6. The system is based on the modularity of its components, where beside the exploration operations like zoom in or zoom out, the users can choose the similarity model, which is used for the visualization and the querying, e.g., feature histograms or feature signatures. The users can also choose the query strategies that improve the effectiveness of evaluating similarity queries. The authors suggest two different strategies, the first one is the approach based on the well-known filter-refinement paradigm and the second one involves supporting evaluation of the queries with some similarity index like the M-tree [52].

The interesting visualization idea that goes beyond more than the pure 2 dimensions is introduced by the Hue Sphere Image Browser[53, 54]. The objects from the database are visualized on the globe, placed according to their values of the hue and the value in the HSV color space. The visualization space is divided regularly in the multiple sections, each section displays one image ensuring that the images do not overlap.



Index-Support: M-tree with Re-Insertions
Evaluation Strategy: Distance-Based_Angle

Clickpoint: [803.0, 701.0]
Current Query mode: Distance-Based_Angle
Stress (raw): 0.00000
Stress (norm.): 0.00000
Stress (Kruskal): 0.00000
Query time: 0.15600 sec
Filter size: 3

Low-Dimensional Similarity Model: Weighted Euclidean Distance
High-Dimensional Data Type: HSL-Color Histograms
High-Dimensional Similarity Model: Euclidean Distance

Euclidean Distance
Signature Quadratic Form Distance
Earth Mover's Distance
Hausdorff Distance
Bhattacharyya Distance

Data Set: histCorel7224.txt | Data Set Objects: 59828 | Layout Objects: 67

Figure 2.6: The prototype of the exploration system by Beecks et al. [51].

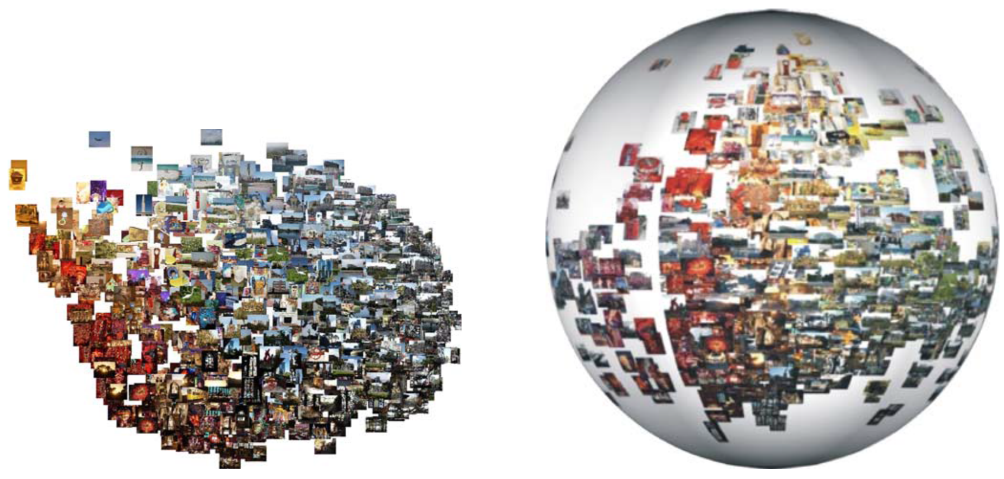


Figure 2.7: The comparison of visualizations a) using multidimensional scaling b) using the Hue Sphere

Lokoc et al. [44] indirectly categorize the exploration according to the used structures. They distinguish between two approaches, the difference is in the method that exploration systems use when they explore space:

- **Iterative querying**, which is the sequence of consecutive similarity queries that users use for getting closer to their (possibly formerly unknown) query intent. When the users interact with some object, the similarity query, usually the $k-NN$ query, is performed in a background and similar objects to the target of the interaction are returned back to the users.

- **Iterative browsing**, which is the sequence of consecutive user actions that the users use for navigating through objects in the explored space by following a hierarchically organized structure, which was created in advance.

The difference between these two approaches is also in the initial phase, before the users start to explore. The systems based on the first approach evaluate the similarity queries during the exploration phase, hence for the efficient evaluation, the explored space should be *indexed with some similarity index*. On the contrary, the systems based on the iterative browsing actually do not evaluate similarity queries in the exploration phase, instead, they traverse the created structure, hence in the pre-exploration phase such a *hierarchical structure* has to be created.

The browsing system *PIBE* [55] is based on the idea of following the hierarchy of a clustered structure called the *Browsing Tree*. The Browsing Tree uses the repetitive *k-means* algorithm to create the clustered hierarchy of a visualized collection and for visualization of the images on the screen the authors of the PIBE adopt the MDS. The PIBE allows the user to personalize his visualization by performing personalizing actions that can modify the Browsing Tree, without the need to fully reorganize the visualized structure. On behalf of basic browsing operations, the user can browse *vertically* along the created hierarchy. Beside that, the authors define the new *horizontal* operation for displaying descendants of more visualized clusters at once.

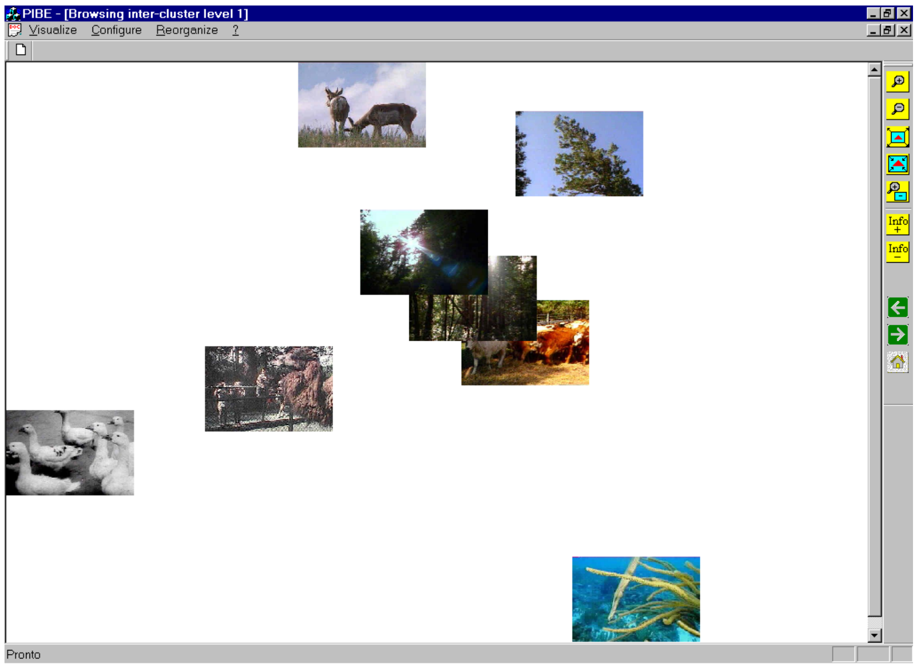


Figure 2.8: The PIBE browsing system using a hierarchical visualization structure. The system allows the horizontal browsing by selecting a point in the "white space" between images and also the vertical browsing by selecting some image to display descendant images in the clustered hierarchy.

The next visualization layout that browses a hierarchical structure is proposed in the work by Chen et al.[56], the authors call it the *similarity pyramid*. Each level of the pyramid is organized as a 2-D grid, where similar data are near to each other. The structure of the pyramid is built by employing an efficient version of hierarchical clustering, the *fast-sparse clustering*, which the authors also propose. The result of the fast-sparse clustering, the clustered binary tree, is subsequently transformed into the Quad Tree [57], whose each level is mapped into one level of the pyramid. Since the pyramid is created as the clustered hierarchy, each level represents the different level of details of the visualized space.

Another exploration system from the category of hierarchical browsing is the *ImageMap* proposed in [58]. The ImageMap allows to visually explore and search millions of images as it is proven in the web-application *Picsbuffet*[59], see Figure 2.11. The underlying visualization structure of the ImageMap is, in a similarly way to the solution from the previous paragraph, a hierarchical clustered pyramid (as depicted in Figure 2.10) sorted by visual and semantics similarities. The bottom level of the hierarchical pyramid structure is based on the self-organizing maps (SOM), while the higher levels are created from the bottom one on the principles of the Quad Tree [57].

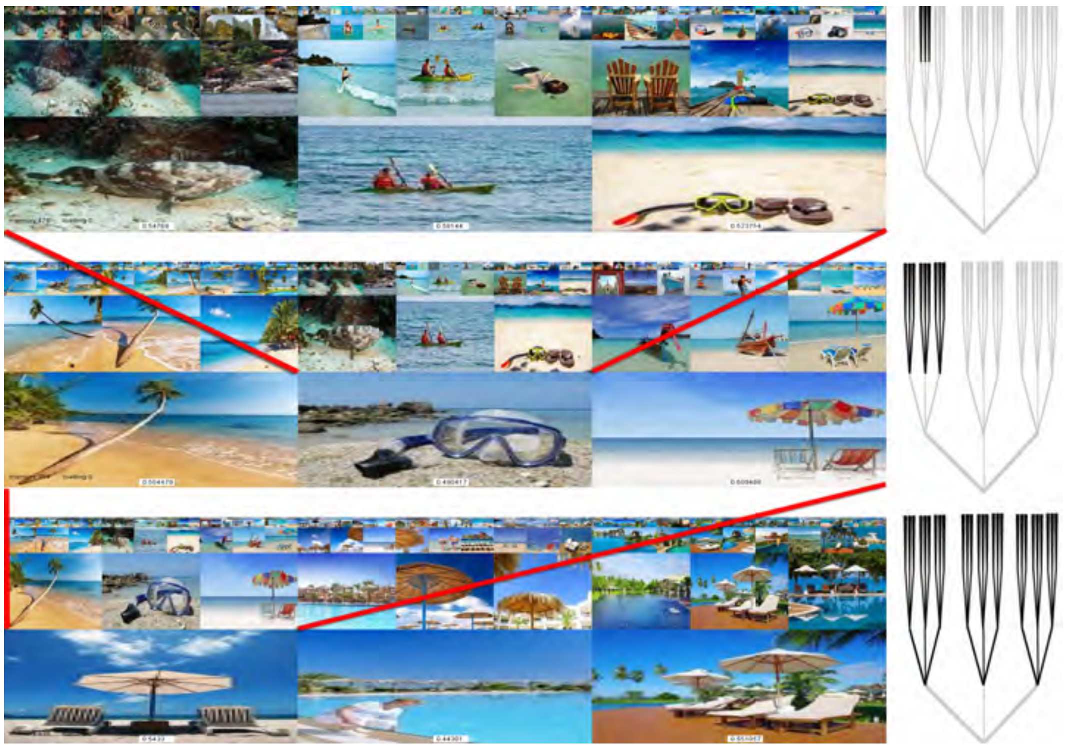


Figure 2.9: The display mode the Fractal tree map in the ImagMap exploration system.

The basic idea of the ImageMap comes from *map services* like the Google Maps. In visualization, the user sees a so-called *viewport* that in the lowest level shows a part of the whole image map, while in the higher levels it shows a part of the particular representatives selected from clusters created in the lower levels. For the sake of the intuitive and continuous exploration, the use of the map-based approach in the area of multimedia retrieval has to follow some rules. Thus beside the necessity of a hierarchical structure, the authors state two requirements that have to be satisfied:

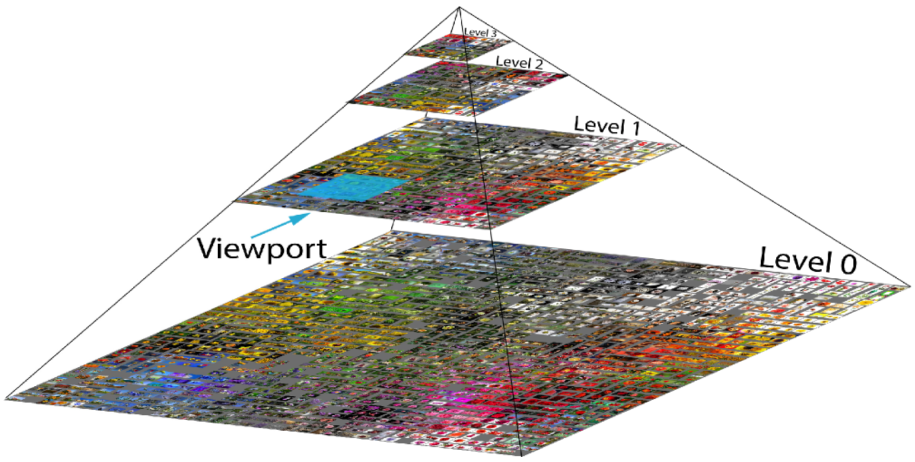


Figure 2.10: The ImageMap - the example pyramid with four levels.

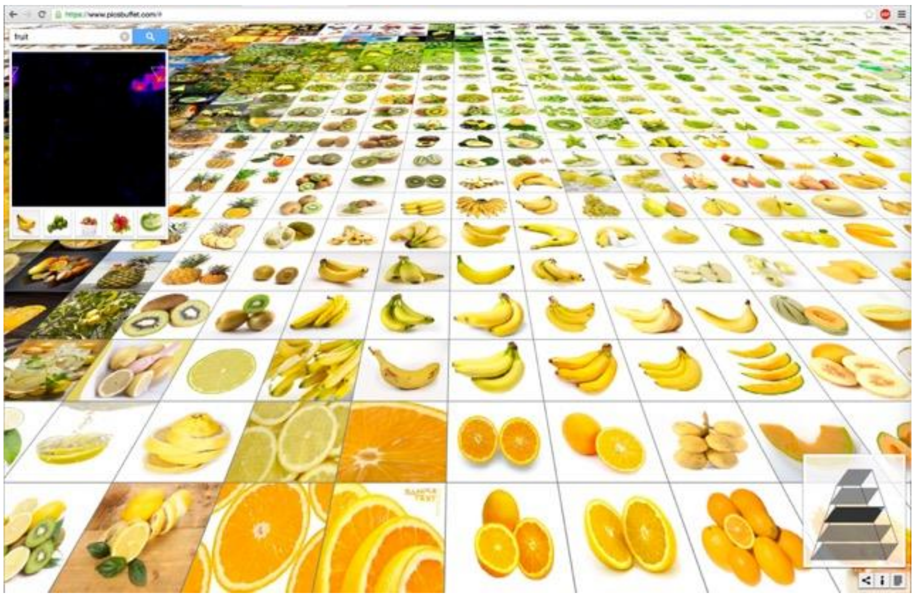


Figure 2.11: The Picsbuffet - the web-application based on the ImageMap exploration system.

- Images on the lowest level of the image pyramid need to be arranged in such a way that similar images are placed close to each other.
- Images of the higher levels need to be chosen as the good representatives of corresponding images from the lower levels.

In the sequel work by Barthel et al. [60], the authors solve the problem how to organize the underlying 2-D structure of a visualized database. The problem is in dimensionality reduction, the high dimensional relationships between images could not be represented in the two-dimensional SOM, so they are lost. The authors propose the solution, the structure is visualized as the hierarchical graph where the vertices are the images and the edges represent the relations between the vertices that are similar to each other. But as there is none obvious way to display such a graph with similarity-based relations, the authors propose the new visualization mode, the *Fractal tree map*, see Figure 2.9, which allows fast navigation through the graph.

The self-organizing maps are used as a mapping approach for visualization of image collection in a 2-D map also in the work by Strong et al. [61]. The authors use the Graphics processing unit (GPU) for evaluation of difficult computations to increase training of the SOM. They suggest to use the k-means algorithm for *dynamic clustering*, to allow displaying the images with an increased size, the difference can be seen in Figures 2.12a and 2.12b. Beside changing the size of displayed images, the user interface of the visualization system allows users to pan or zoom in the visualized map to see details of the specific part of the database. If necessary, the users can also display all images of the clusters not just their representatives (notice the difference between Figures 2.12c and 2.12d).

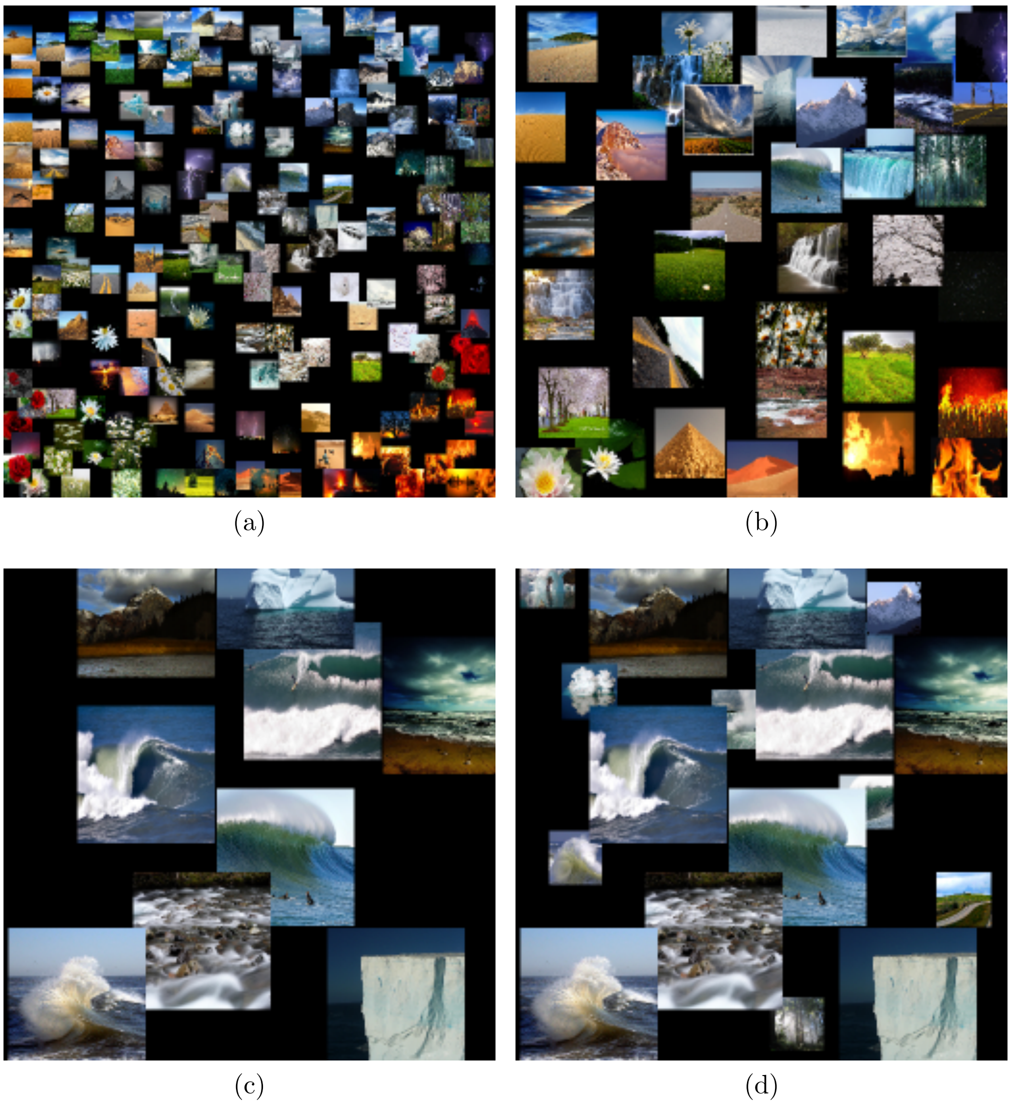


Figure 2.12: The photo collages dynamically generated based on the user interactions: (a) the initial SOM trained using 256 images (b) increasing of the photo display size results in the state where fewer images are shown, each of them represents a cluster (c) zooming into the portion of the SOM where some ocean pictures are mapped to (d) showing remaining images in the clusters of reduced size [61].

2.1.3 User Interface

The intuitive interface is, from the user perspective, one of the fundamentals of an acceptable exploration system. The user interface consists of both, the visualization layout and also of the operations that the system offers for active interaction with it. As the visualization we address in the previous section, in the following we focus on the *exploration operations*.

There are many perspectives, from which the user interaction with the visualized collection can be seen. One of them, proposed in the work by Plant et al. [62], offers two categories of browsing in retrieval systems according to the direction of browsing. The first category, the *horizontal browsing*, allows the operations with images placed on a *visualization plane*, such operations could be *panning* (2-dimensional moving on the plane), *zooming*[4] (here it means focusing on some part of the visualization plane for seeing more details of some images), *magnification* (focusing on one image on the plane to see more details of it, see Figure 2.13) and *scaling* (dynamically resizing the resolution of some images on the visualization plane). The second category is the *vertical browsing*, which allows users to navigate between different levels of hierarchical visualization. As the authors conclude, the operations that the hierarchical browsing introduces could be practically used in every visualization which displays images on a single plane. Whereas the operations of the vertical browsing are limited only to the systems with hierarchically organized visualization.

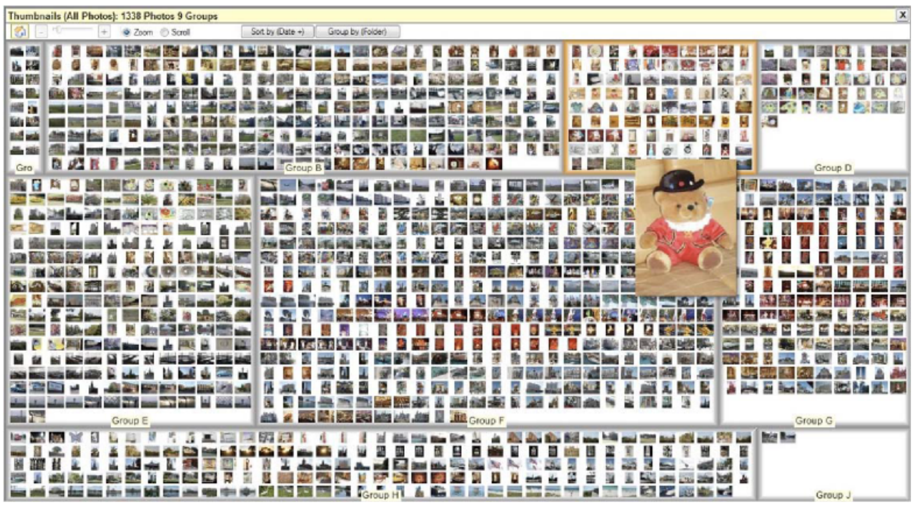


Figure 2.13: The demonstration of the magnifying operation in horizontal browsing.

Another perspective on the exploration operations is introduced in 3-D like visualizations, e.g., placement on a globe provides to the user the intuitive interface, see the Hue Sphere visualization [53, 54] in Figure 2.14. With *rotating* the globe (the operation panning), the user browses the image collection in accordance with different hue values and by *tilting* the globe the user can see more darker or more brighter images. In case when the database is large, more images are internally assigned to one section on the globe and only the representative image of that section is displayed. While exploring, the user is able to open such

[4]In the latter text we connect zooming to the vertical, not the horizontal browsing.

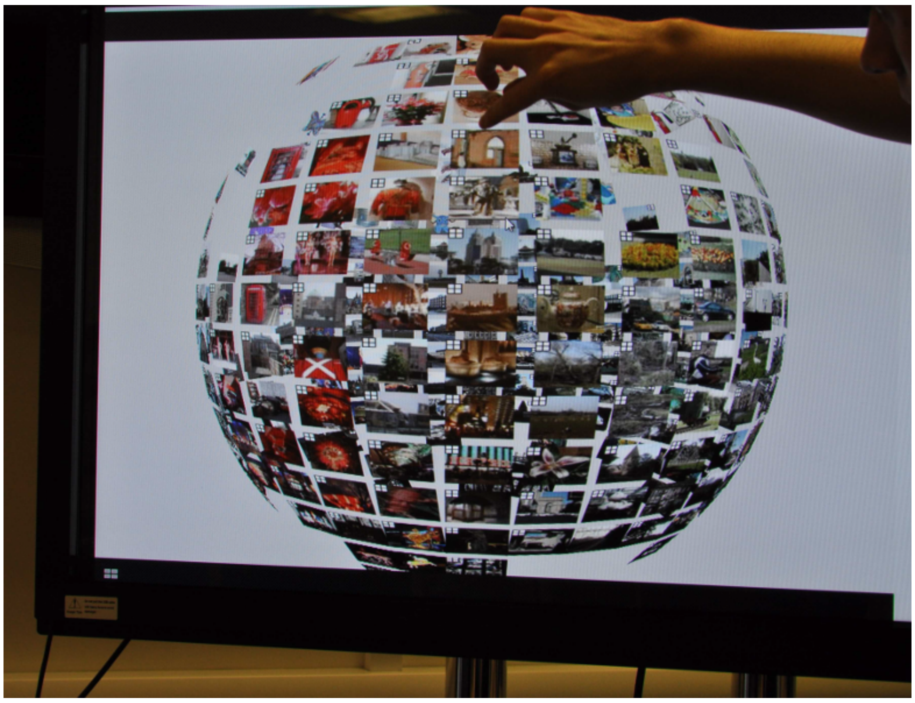


Figure 2.14: The Hue Sphere Image Browser on a multi-touch screen.

a section and its content is visualized applying the same rules (changing hue from left to right and changing brightness from top to bottom) that are applied for visualization of the whole collection.

With the rapid growth of mobile devices, the techniques for usual user interaction with visualized collection are changing. In the work by Schoeffmann et al. [63], the authors study different ways how to visualize image collections within the limited space that the screens of mobile devices typically have. With considering the third dimension in the visualization, the systems are able to display much more images on the screen at once. The solutions by Schoeffmann et al. consist of the 3-D visualization on the 2-D screen as it is also in case of the 3-D Cylinder visualization [64] (see Figure 2.15). The images are placed in the similar manner as those in the Hue Sphere, based on the hue value from the *HSV* color system. The 3-D cylinder can be rotated to show more images from another part of the visualized database. The authors try to utilize specifics of mobile devices, e.g., by employing touch gestures, *swiping* for rotating and *pinching* for zooming. Moreover, an accelerometer integrated in the devices is also exploited, *horizontal tilting* rotates the cylinder and *vertical tilting* switches the vertical perspective of the whole cylinder.

The principles of the multi-touch 3-D Cylinder were joint with the principles of the Hue Sphere (see Figure 2.14) the authors of both systems cooperate and they propose the new multi-touch 3-D Globe [65], you can see it in Figure 2.16. The 3-D Globe utilizes the visualization on a sphere from the Hue Sphere and the user interface for mobile devices from the 3-D cylinder.

For helping the user to see details of some part of the visualized space but still in a global context, the *fisheye view* [66] can be used. The visualization system by Liu et al. [48] uses this technique for highlighting some images when the user interacts with the visualization through a mouse click.

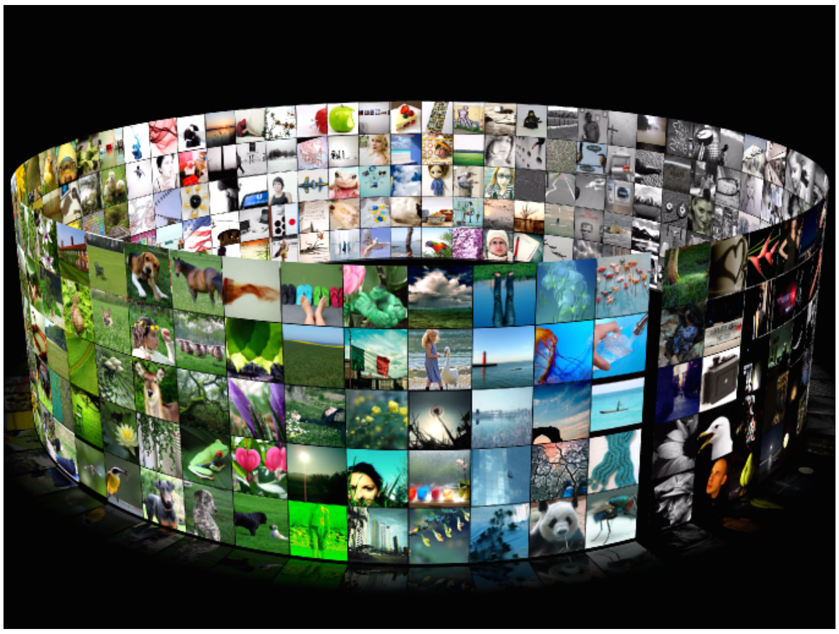


Figure 2.15: Multi touch visualization on the 3-D Cylinder.

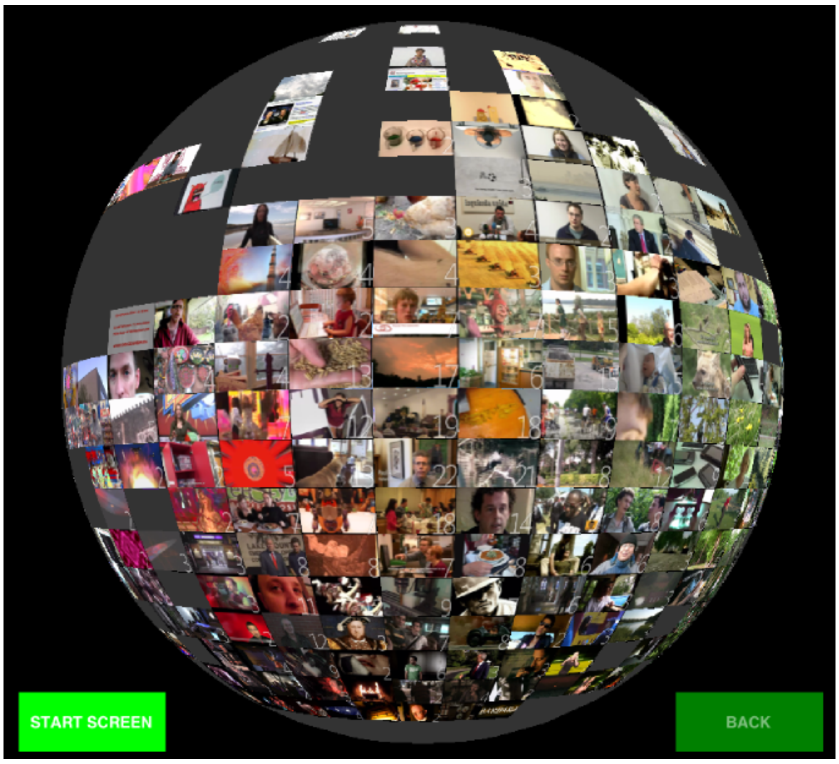


Figure 2.16: Multi-touch visualization on the 3-D Globe - combination of the multi touch 3-D Cylinder and the Hue Sphere.

In the system proposed by Chen et al. [56], the users are able to follow the hierarchy of a hierarchical structure called the similarity pyramid. Following the hierarchy down or up involves the operations *zooming in* or *zooming out*, where the next level (lower or upper) is displayed centered with regards to the spatial position in the hierarchy of the previous view. Moreover, if the currently viewed level of the pyramid is too large to display on a single screen, the users can use direction arrows for *panning* to explore hidden parts of the current level also horizontally.

Some systems provide the user interface that combines searching techniques of the query by example scenario in addition to the explorations operations. For example, the authors of the *MediaMill* search engine [67, 68] use in their proposal a combination of four query techniques - *query by concept*, *query by example*, *query by keyword* and *user interaction*. But their system (see Figure 2.17) is primarily designed for a targeted interactive search and not for an indirected exploration. The typical retrieval scenario in their system starts with the user defined query and after obtaining the results, the user rephrases the previous queries to retrieve the results that are more accurate than the first ones.

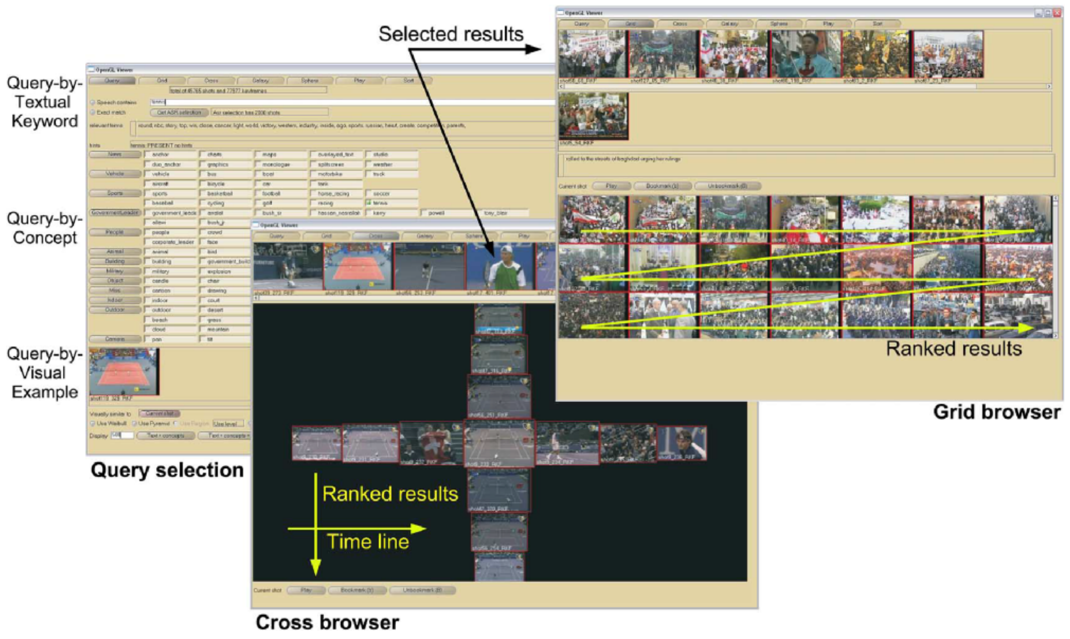
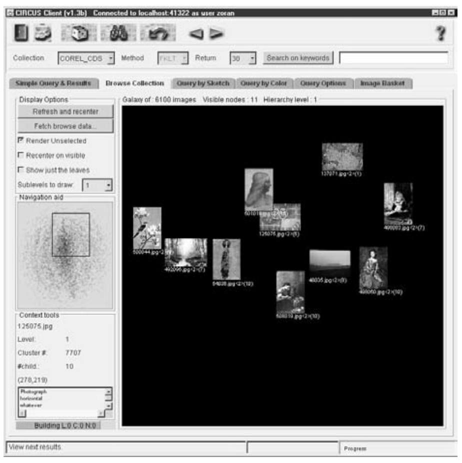


Figure 2.17: The interface of the MediaMill semantic video search engine.

Similarly, the hierarchical tree-based *CIRCUS* system [40] provides the user interface (see Figure 2.18) for two content-based retrieval modalities - querying and browsing. Beside *query by example*, the users performing querying are able to *query by text*, *color*, or *painting*, or even use some combination of these query types. The browsing operations include *panning* and *zooming* in the visualized space. At each level of the hierarchy the users see the representative images of clusters and the several sub-levels, which can be also displayed optionally within the current level. The user interface of the CIRCUS provides a so-called *semantic zoom*, i.e. at a particular zooming level, some additional information (e.g., the annotation or meta-data) is visualized next to the own images. The system displays the entire space also in the smaller overview component, for better orientation within the visualized space (see Figure 2.18).

2.1.4 More Unconventional Approaches

Beside the approaches we mentioned in the previous sections, there exist uncommon visualization and exploration systems based on some special, typically cross-disciplinary, ideas. In the works by Chen et al. [69, 70], the authors adopt


(a)

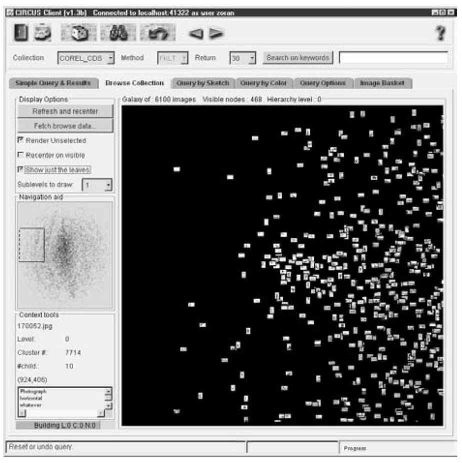

(b)

Figure 2.18: The search engine of the exploration system CIRCUS a) visualization of one cluster b) the higher overview of a visualized collection at a given level of the hierarchy

the *Pathfinder networks* used in the psychology [71] to visualize image and video collections, see Figure 2.20. Their visualization can be based on different metrics - color, layout or texture, while with the change of the metric the visualized structure also changes. The whole multimedia collection is visualized at once and beside exploring, the users can also search with the traditional query by example scenario. The layout of their searching engine you can see in Figure 2.19.

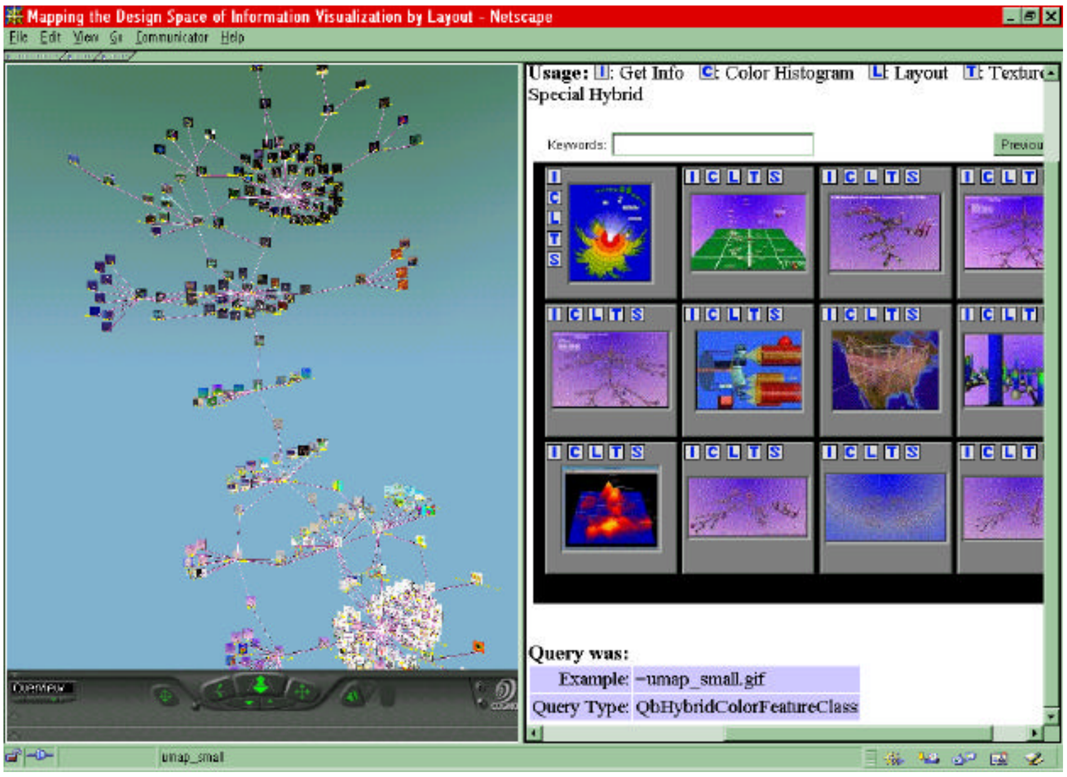

Mapping the Design Space of Information Visualization by Layout - Netscape
Usage: I: Get Info C: Color Histogram L: Layout T: Texture
Special Hybrid
Keywords:
Query was:
Example: ~umap_small.gif
Query Type: QbHybridColorFeatureClass

Figure 2.19: The searching engine of Pathfinder network visualization.

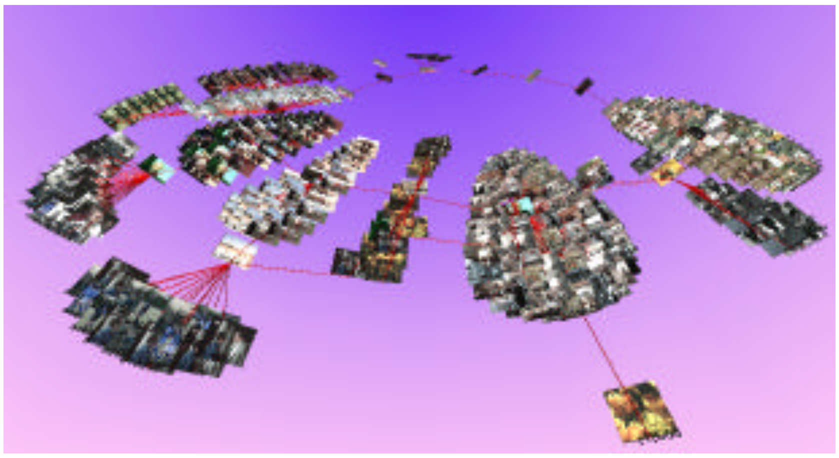
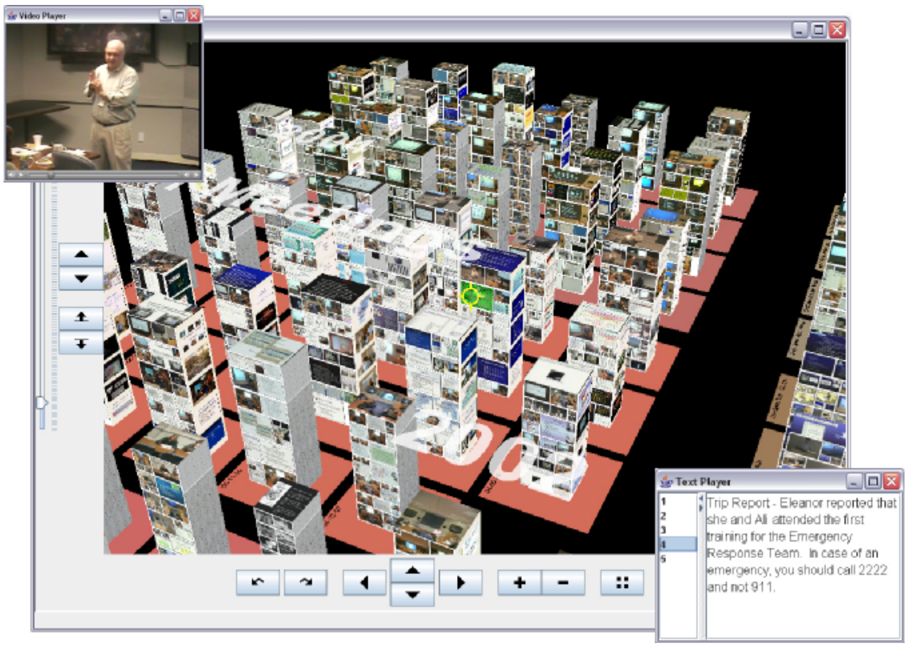


Figure 2.20: Pathfinder network visualization of a video collection.

The very complex visualization and the user interface are offered in the application *MediaMetro* [72], which uses the *3-D city metaphor* for visualizing and browsing the document collections, see Figure 2.21. The MediaMetro is useful for the visualization of documents organized in a directory hierarchy (subdirectories are visualized in a repetitive way), where each directory is visualized as a building in the virtual city. The buildings have the images instead of their windows that represent the documents contained in the directory, while on the top of each building is a single representative frame which shows the directory storyboard. The visualized documents can be of any type, from images or texts to presentation slides and videos, which can be selected for media playback. The users of the application are able to explore the city in the 3-D manner, well known from computer games, the authors liken it to the movement like *flying around in a helicopter*.



Trip Report - Eleanor reported that she and Ali attended the first training for the Emergency Response Team. In case of an emergency, you should call 2222 and not 911.

Figure 2.21: The MediaMetro - browsing a documentary collection in the city-like visualization.

2.2 Index Support of Multimedia Exploration

Heesch in his survey [23] mentioned in the beginning of this chapter is giving into parallel the searching and the indexing of the multimedia browsing systems with the World Wide Web (WWW). He states that a browsing structure can be navigated very quickly when it is already built, as it happens in the WWW. This observation emphasizes the necessity for building the scalable structure that subsequently supports the actions of the user when he is exploring the multimedia collection.

But, the principles of the WWW cannot be directly transposed into the multimedia exploration systems. The majority of the exploration systems we outlined in the previous sections create the structures that can be quickly followed during the exploration phase, but these proposals were tested on the relatively small collections with hundreds of thousands of images. Only the *ImageMap* [58] has proven to explore huge collections with over millions of images. With the growing size of the indexed collection, it is legitimate to ask if these proposals are able to perform the responsive exploration on respectively large collections.

The majority of the outlined systems more or less follow some hierarchy created from the explored collection during the exploration phase, but such an approach is not the only one used in the multimedia exploration systems. The concept of *iterative querying* [44] we mentioned in the Section 2.1.2 does not directly require the hierarchical structure, it assumes the support of evaluating the *exploration operations based on similarity queries* employing some indexing methods. The metric access methods (MAMs), we mentioned in Section 1.3.1, can be used in such a scenario to improve the efficiency of query evaluation.

2.2.1 Metric Indexing

Many different MAMs and other indexing methods [73] were proposed in the last two decades to improve the efficiency of the query evaluation. In the following paragraphs, we outline general principles of some of them.

One of the most efficient (yet simple) MAM is the *pivot table* [74], originally introduced as the *LAESA* [75]. Basically, the structure of the pivot table is the simple $n \times m$ matrix of distances $\delta(o_i, p_j)$ between n database objects $o_i \in \mathcal{S}$ and the pre-selected static set of m reference points, called the pivots $p_j \in \mathcal{P} \subset \mathcal{S}$. For querying, the pivot table allows us to perform cheap lower bound filtering by computing the maximum lower bound (see Equation 1.5) to the real distance $\delta(q, o)$ using all the pivots as we described it in Section 1.3.

As we mentioned in Section 1.3, the fundamental principles of metric indexing, which are used in the MAMs, utilize the metric axioms. Beside the principle of *lower bound estimation* based on the *triangle inequality*, described in Section 1.3, the similar principle based on the *Ptolemy's inequality* called the *ptolemaic indexing* [76] is for example used in the *Ptolemaic Pivot Table* [77]. The Ptolemy's inequality states that for any quadrilateral with points q, o, u, v (as depicted in Figure 2.22) holds:

$$\delta(q, v).\delta(o, u) \leq \delta(q, o).\delta(u, v) + \delta(q, u).\delta(q, v) \tag{2.1}$$

If we consider the situation, where q is the query object, o is the database object and u, v are two pivots, from the above inequality we can deduce the

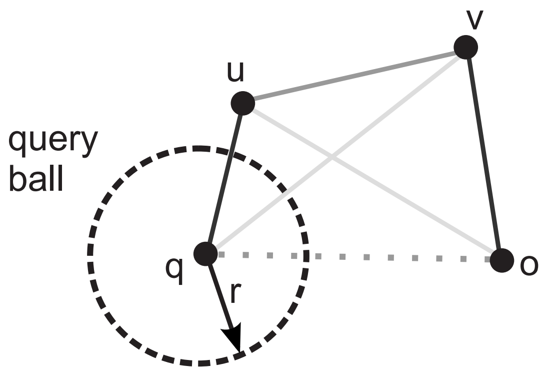


Figure 2.22: The ptolemaic lower-bounding principle

ptolemaic lower bound distance of the real distance $\delta(q, o)$ between the query object q and the database object o as

$$LB_{Ptolemy}(\delta(q,o)) = \frac{|\delta(q,v).\delta(o,u) - \delta(q,u).\delta(o,v)|}{\delta(u,v)} \tag{2.2}$$

Similarly, like in the case of the triangle lower bound (see Section 1.3) if the ptolemaic lower bound distance is greater than the query radius r, the real distance does not have to be computed. Beside the Ptolemaic Pivot Table, the ptolemaic lower bounds can be also employed in the indexing principles of other MAMs [77].

Next MAMs, the *M-tree* [52], the *PM-tree* [78, 79] and their variants [80, 81, 82] are dynamic index structures that provide good performance in the secondary memory (i.e. in database environments). The M-tree is a hierarchical index where some of the data objects are selected as the centers (local pivots) of the ball-shaped regions, while the remaining objects are partitioned among these regions in order to build up a balanced and compact hierarchy of data regions, see Figure 2.23. Each of these regions (the subtree) is indexed recursively in the B-tree-like [83] (bottom-up) way of the construction.

The range and the $k\mathcal{NN}$ queries are implemented by traversing the tree starting from the root, where only those nodes are accessed, whose parent regions are overlapped with the query ball (q, r). In case of the $k\mathcal{NN}$ query the radius r is not known beforehand, so the additional process is employed to dynamically decrease the radius during the search algorithm (initially set to ∞). The $k\mathcal{NN}$ algorithm performs the best-first traversal of the index, where the regions are accessed in order of the increasing lower bound distance to the query object q.

Next MAM similar to the pivot table is the *permutation index* proposed in [84]. The index also consists of the simple $n \times m$ matrix, but instead of the objects to pivots distances, the permutation of the pivots is stored for each object o_i from the indexed collection. This permutation is ordered with regards to the distances between the database/query object and the particular pivots. The main principle of the permutation index is based on the idea that similar objects see the pivots in the underlying space from the same perspective, hence their permutations of the pivots are also similar.

While querying the permutation index, first the permutation for the query object is computed and then the objects from the queried collection are ordered in descending order according to the similarity of their pivot permutations to

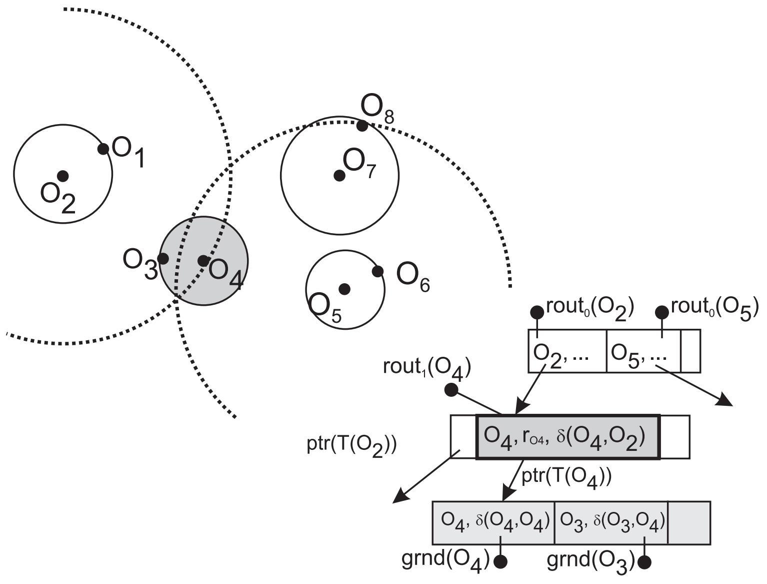


Figure 2.23: The M-tree - hierarchical space decomposition and the tree structure

the permutation for the query object. Next, in that order, the query-to-object distances are evaluated until the stop condition based on some heuristic is fulfilled (e.g., when the fraction of the whole collection is evaluated). Since it is not ensured that all objects whose query-to-object distance was not evaluated yet[5] are refined with the querying condition, this querying method belongs to the category of the *inexact (approximate) search*.

Another index that employs the idea of the ordered permutations of pivots is the *M-index* [85, 86, 87]. The M-index consists of the hierarchically organized *cluster tree* which organizes the underlying data storage represented by the *B-tree* [83]. For data clustering, the cluster tree repetitively uses the Voronoi-like partitioning for dividing the objects into non-intersecting clusters. Each cluster is identified by the pivot permutation that is computed for each object in the cluster on the same principle as in the permutation index from the previous paragraph. In addition, the clusters from the cluster tree are numbered using the specific principle used in the *iDistance* [88], where these numbers together with the distance to the nearest pivot are used as keys to the B-tree storage (see in Figure 2.24).

The querying algorithm of the M-tree starts in the root of the cluster tree and traverses the tree in a breadth-first manner to those subtrees which are represented by the prefixes of the permutation for the query object. While traversing the tree, the known metric space postulates [4] are applied for pruning non-relevant clusters. Finally, the iDistance-based identifiers are used for retrieving relevant objects from the non-pruned clusters and the collection of these result objects is subsequently refined by computing the real distances to the query object.

2.2.2 Non-Metric Indexing

As we mentioned in Section 1.3, the metric axioms help to build efficient indexing structures, but the metric indexing has also some limitations as it is described in

[5]in the moment when the searching algorithm stops

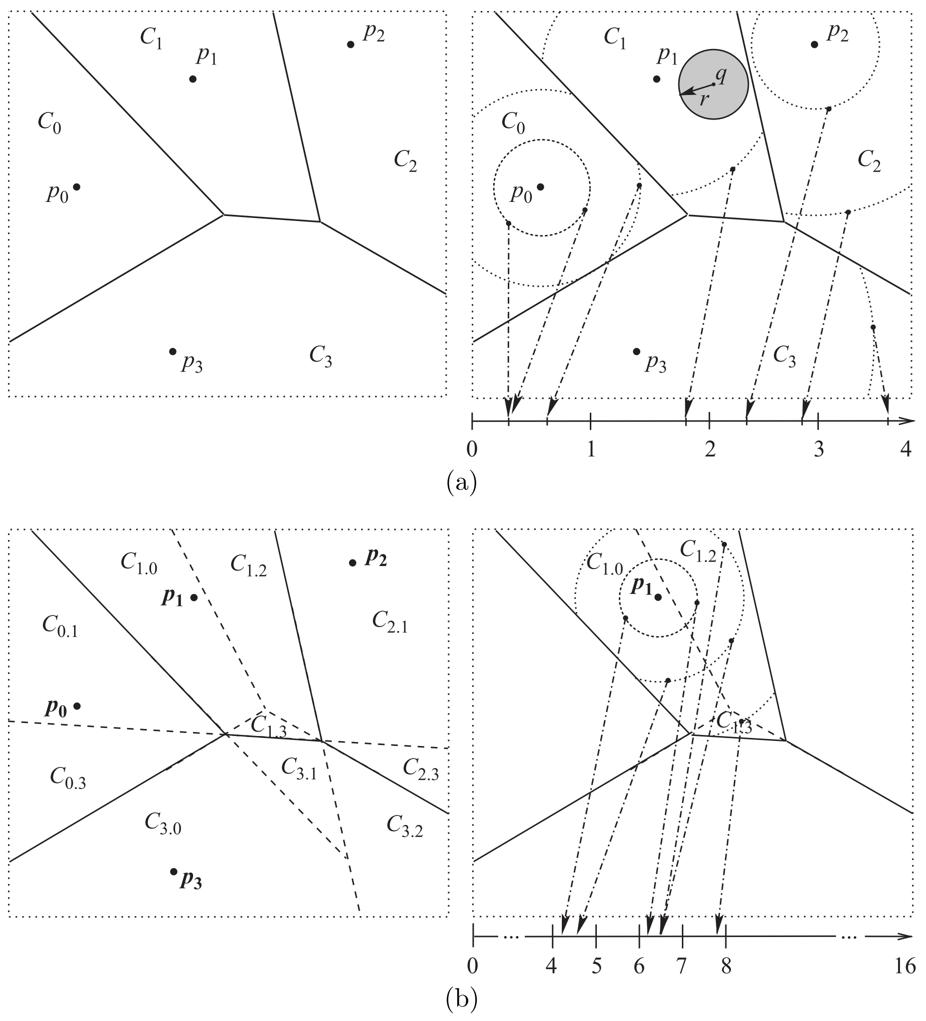


Figure 2.24: The principles of the M-Index Voronoi-like partitioning (left) and the iDistance-like mapping (right) a) on the first level b) repetitively on the second level [86].

the work by Skopal [5]. Skopal mentions that in many cases the metric axioms can be very restrictive for the process of modeling similarity whereas the similarity in real-world is perceived with the human perception. On the contrary, more general similarity can be achieved when the *non-metric similarity function* is used, such one which does not satisfy some of metric axioms. But it is not for free, with relaxing some metric axioms the new problem is introduced in indexing: the triangle lower bound principle (see Section 1.3) cannot be directly used, hence it is necessary to find the new principles for *non-metric indexing* that improves the indexing efficiency. One of possible solutions introduces Skopal in his work [5], he proposes the *TriGen* algorithm which can transform the semi-metric similarity function (fulfilling only nonnegativity and reflexivity of the metric axioms) into the metric one, hence the triangle lower bounding principle can be still used. The real-world usage of the TriGen principles in MAM is, for example, used in the NM-tree [89], the non-metric M-tree.

But there are also other techniques of non-metric indexing, different from those based on the technique of relaxing some metric axioms. For example, in our research over the similarity modeling [90, 91], we tried to find new axioms in

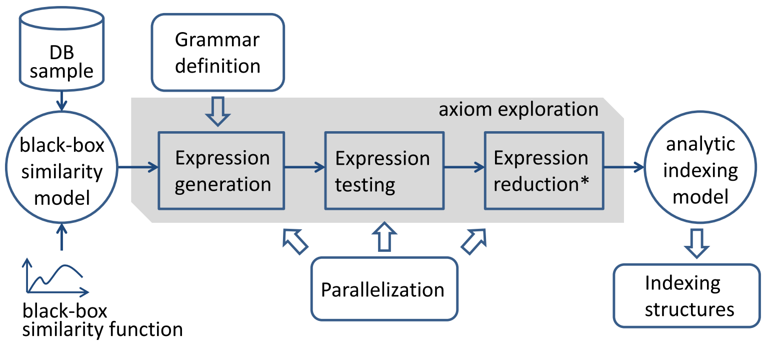


Figure 2.25: Axiom exploration in the SIMDEX Framework - the high-level overview.

a created model by exploring the space of axioms (see Figure 2.25). By applying them in the retrieval process, we examined their effectiveness and efficiency, and proved their usefulness.

2.2.3 Approximation Search

Beside the exact search, when the query result exactly corresponds to the query conditions and modeled reality, there exists also the *approximate search* which allows the presence of some error in the query result. The approximate search is based ob the idea of exchanging the query precision (effectiveness) for the query speed (efficiency). Typically, the user adjusts this trade-off at the query time using various approximation parameters, while these differ from one approximation method to another. The parameters can be of a different kind, for example, the probability of an error in the query result, the maximal allowed number of distance computations, or the threshold on improvement of the query result.

When considering the previous observations, the interactive preconditions of the multimedia exploration can truly exploit from the advantages of the approximate search. In the following, we present four of many proposed scenarios for the approximate search based on different principles. Although the following list is by far not complete, the examples illustrate various ways how a user can manipulate the trade-off between the effectiveness and the efficiency of retrieval.

Example 1 – Proprietary Methods Bounded on the Particular MAM. In the work by Zezula et al. [7], the authors present three techniques for the approximate nearest neighbor search in the M-tree. All the proposed techniques guarantee the high degree of precision, when all objects of the approximate query result are almost as good (to some extent given by the approximation parameter) as those in the exact result. However, because of the strict guarantee, the speedup is not as large as the one achieved when some of probabilistic techniques from the next example are used, where the guarantee is not so strict.

In the first technique (presented also in the work by Arya [8]) the user-specified parameter ε represents the *relative distance error*. The relative distance error states how much larger can be the distance from the query object q to the returned (approximate) nearest neighbor o^A than the true nearest neighbor o^N of the exact search. It is computed as $\varepsilon = \frac{d(o^A,q)}{d(o^N,q)} - 1$. To employ this technique in a searching algorithm, the query radius should be decreased by the factor of $1/(1+\varepsilon)$.

The second technique takes into consideration the *relative distance distribution* $F_q(x)$, which represents the fraction of database objects that are no more distant from the query object q than in particular distance x. While searching, the stop condition is fulfilled when the distance distribution $F_q(d(q, O_A^k))$ is lower than the user-defined parameter ρ, where O_A^k is the k-th candidate object in the current result.

The main idea of the third technique is based on observation that the *refinement progress* of the $k\mathcal{NN}$ candidates on the result[6] slows down with running time of the query evaluation. Hence, this approximation technique stops the evaluation of the query at the moment when the collection of actual $k\mathcal{NN}$ candidates is changing very slowly.

Example 2 – General Probabilistic Methods. Next approximation approaches are from the class of the probabilistic methods which introduce the expected *probability of an error* – the user-defined parameter, for the query result. This parameter gives a user the possibility to get the result either faster or more precise. Unlike the methods in the first example, the probabilistic methods lead to much faster query processing, however, for the price of the qualitatively weaker guarantee that the query result that it contains the correct objects (i.e., some query result may be completely wrong while the other one may be fully correct).

In the work by Chavez et al. [9], the authors present the probabilistic technique for the approximate search when using the decreasing search radius – the query radius is shrunk by the factor β derived from the user-defined probability of an error.

The TriGen algorithm [5], we mentioned above, can be also utilized in the approximate search. The special functions called the T-modifiers are used not only for turning semi-metric distances into the metric ones, but also vice versa. When utilized by some MAM, the evaluation of such a modified distance function can result in approximate behavior. The T-modifiers can be also specified as the user-defined parameter and modified at query time thus the different levels of approximation can be achieved.

Example 3 – Transformation Methods. A quite different approximate approach is based on the transformation of a source metric space into another (mostly vector) space. For example, the *Fastmap* [27], originally proposed as the technique for mapping a metric space into an euclidean space, can be also used as the approximation method. Other approaches [84, 85] use the permutations of pivots for mapping a metric space into a space of ordered permutations. The problem of transformation methods is that they do not guarantee the query time and moreover they even do not provide any user-defined probability of the retrieval error.

[6]The principle of the $k\mathcal{NN}$ search is based on the progressive refinement of the collection of those objects which potentially can be in the final result. This collection is changing during the search, but this change is approximative with regard to the final result and the progress of the change is typically decreasing.

Example 4 – Iterative Improvements. The most suitable approximation methods for the multimedia exploration seem to be ones that continuously improve the query result, allowing the user or the exploration system to stop the refinement process whenever the results are quite sufficient. A one of such approximation methods is the proposal of Ferhatosmanoglu et al. [92] which combines two approaches on the continuous refinement step. The first one is based on the so-called *retrieved set reduction*, when data are clustered with the k-means clustering algorithm and the clusters are subsequently accessed in order from the nearest to the furthest ones with regards to the query object. The second approach the authors classified as the *representative size reduction*, when the dimensions of the descriptor vector are not considered for computation of the distance fully at once but rather partially in order from the most important ones. Considering more and more dimensions evidently improves the accuracy, while accessing more clusters also improves the accuracy, since the query algorithm can consequently choose from more candidate objects.

One of the inspiring approaches from the classification of the approximate search scenarios by Patella et al. [6] is the RC_{ES} which stands for *reducing comparisons* and *early stopping*. Like the approach mentioned in the previous paragraph, the search is terminated whenever is necessary, while the partial results are returned.

Another relevant technique of this category is the concept of the incremental similarity search [93] mentioned already in Section 1.2.3. It works with tree-based index structures based on the idea of assigning higher priority to more promising nodes of the tree. Based on this priority, the unprocessed nodes are processed within the tree until the search process is stopped, which happens whenever the most promising node in the processing queue guarantees worse value than currently the "worst" node in the result set (e.g., the greater distance to the query object than the k-th nearest neighbor). This technique was later proven to be range-optimal [11], firstly designed for spatial databases [10] and also optimal for disk page accesses [12].

2.3 Real-Time Queries

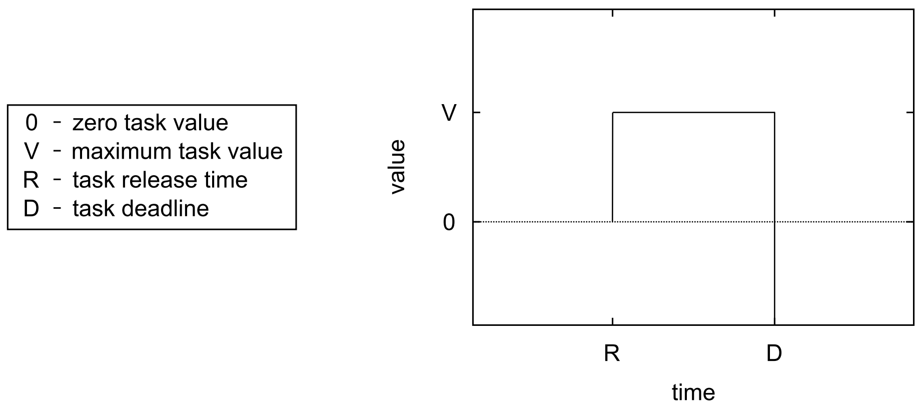
If we consider the multimedia exploration generally as the continuous process of revealing and examining the content of some multimedia collection, it is necessary to perform the individual exploration steps in a *non-discouraging way* for the user who performs exploration. The main factor of this non-discouraging way, which influences the user satisfaction with the interactive exploration system, is the *system response time* on user actions. In the historic survey by Shneiderman [94] on the research over a human reaction and response time, the author tries to find out how long will users wait for the response of the system before they become discouraged of its delays. Although the survey is now three decades old and from those times the response time of computer systems grows rapidly, some of its general results still hold. Shneiderman firstly studies the expectations on the response time of the users themselves. He observes that the previous experience with a user task critically shapes their expectations, i.e. if the user uses the system repeatedly he expects that it responses in like manner. This is related to the other observation, the users are highly adaptive, they get used to long delays, but their satisfaction is likely to suffer. And in the last observations, Sheiderman states that there is the enormous variation in the response time expectations among particular users and also user tasks. One limitation statement that results from the survey is the *upper limit 1 second* of a generally satisfying response time, of course, if the user task is technically feasible of it.

In another study by Thorpe et al. [95], the authors examine the speed of image processing in the human visual system. They conclude their research on this topic with the statement that in the visual processing, the time needed for performing a highly demanding task can be achieved under *150 milliseconds.* This limitation can be used as the *lower limit* for the response time, whereas, if the multimedia exploration is performed in a reasonable way, one exploration step could be hardly performed faster than it is the time which the user needs for absorbing information from its result. This measurement is confirmed by the simple benchmark on the human reaction time [96], where in the very simple user task the average reaction time of twenty two and half million tests is ≈ 260 milliseconds.

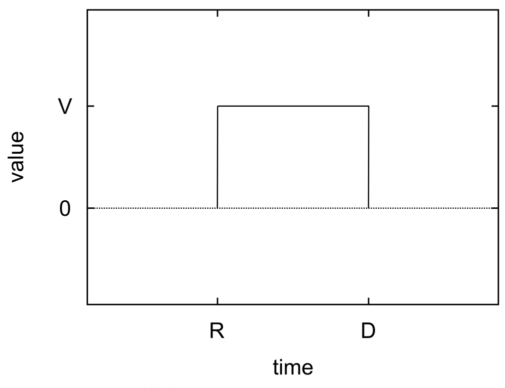
The necessity for the near-real time query evaluation remarks also the authors of the exploration vision study by Beecks et al. [24]. They state that the database indexing structures were primarily designed for the efficient evaluation of simple (single) similarity queries. But in the case of real world applications, as the multimedia exploration systems are, one exploration step can invoke multiple similarity queries and therefore the indexing methods should be optimized for the near-real time data access at a large scale.

We can see the parallel between the user discouragement of long query responses and the task deadlines in real-time computations used in the theory of real-time systems [97]. In the deadline theory of the real-time systems, the tasks usually have the timing constraints which specify some task parameters like, for example, the task release time when the computation begins and also the time in which its computation should finish. Different tasks have different expectations on their deadlines and according to consequences that can happen if the task does not meet its deadline, the real-time tasks are generally categorized in those with hard and soft deadlines. You can observe the value functions of such tasks in Figure 2.26.

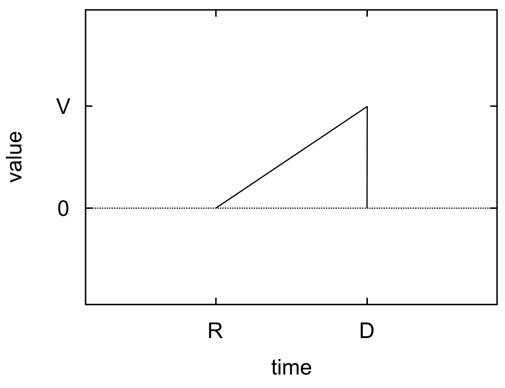
In Figure 2.26c we can see that the task value is increasing during the task execution. The same impression we can get from the evaluation of such a similarity query which precision of the result is increasing with its execution time. Moreover, the decreasing character of the task value after it reaches its deadline in Figure 2.26d can be put in the parallel with the reduction of user satisfaction with the exploration system and its long response time. We tried to summarize these facts in Figure 2.27. The figure depicts three queries (a), (b), (c) with different behavior, and two thresholds that represent the state when the result is good enough (d) and the point when the user starts to be annoyed with long query evaluation (e). The query A represents the evaluation with slow improvement on the query result. Such an evaluation is not very suitable for the interactive manner of the multimedia exploration, because it takes very long time to get some satisfying partial result. On the contrary, the query C is very suitable, because its query execution gets the approximate result at the high level of precision very early and for the rest of the execution the result is just lightly improved. The



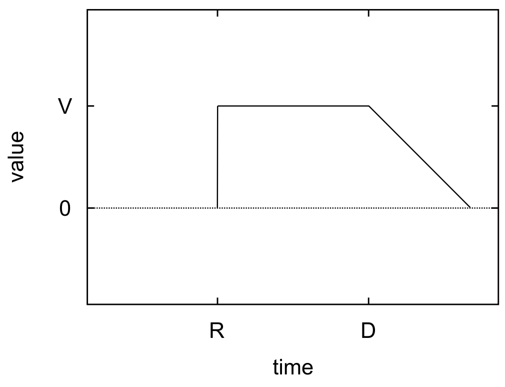
(a) Hard (Catastrophic) Deadline.



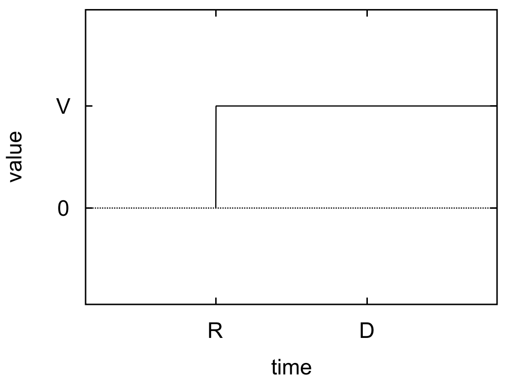
(b) Hard Deadline.



(c) Ramped Hard Deadline.



(d) Soft Deadline.



(e) Non-real-time.

Figure 2.26: Value functions during task execution.

query B depicts the situation when the task get the exact result much earlier than in the case of the query C, but the evaluation of the query B improves its result in the similar slow way as the query A, hence it is also not suitable for the interactive search in the multimedia exploration.

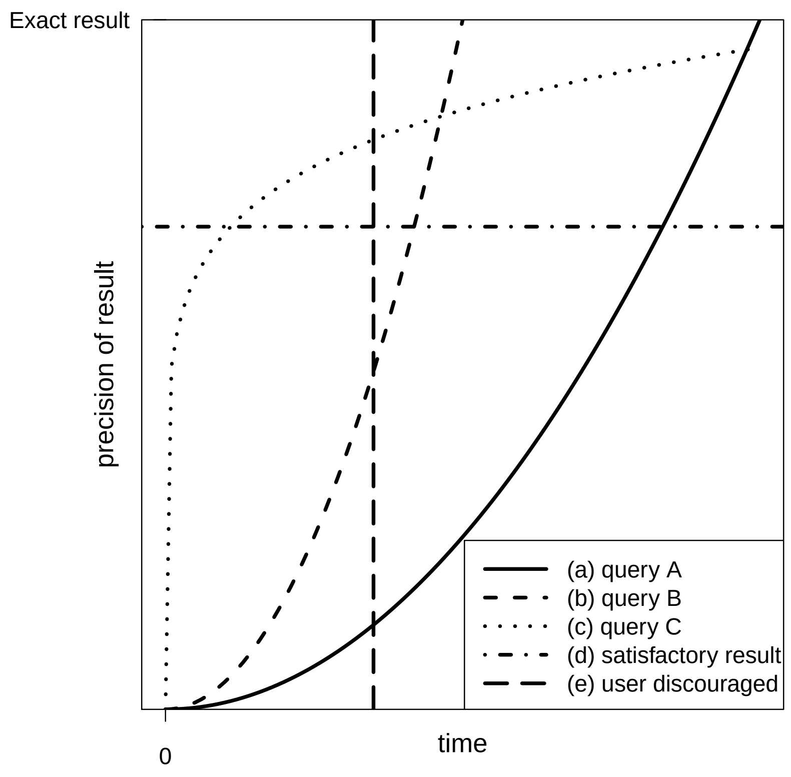


Figure 2.27: Precision of the query result during query execution.

2.4 Discussion and Summary

In the previous section we summarized the research over the topic of the multimedia exploration and related areas as we saw it before starting our research and also during the time when we worked on it. Within that summarization we also outlined a direction where our research, described in the following chapters, aims.

As we look at our survey of existing browsing and exploration systems, they are generally based on some mapping principles. The whole database is firstly somehow organized into a structure, this period of time we can designate as the *construction phase*, where later in a so-called *exploration phase* this structure is traversed with the help of various user interfaces. But in case of very large collections, mapping the whole database into the traversable structure can take a very long time and it does not matter if it is used the MDS, the SOM, or any other mapping method. In such a situation, the more promising way seems to be the *iterative querying* reported by Lokoc et al. [44], because it does not expect the underlying structure for performing the exploration operations. Such an approach solves the problem of the long times for building the structure in the construction phase, but when there is no structure to follow, it seems that the problem is only moved to the exploration phase. Therefore the exploration phase should be additionally supported with some structure after all and there the iterative querying utilizes the similarity indexes.

The similarity indexes support the efficient evaluation of similarity queries and also, as we propose in the the following chapters, the exploration operations.

But when the indexed collection is very large, the evaluation of the exact query can take a very long time and that is not suitable for the interactive nature of the multimedia exploration, the approximation queries should be considered instead. The approximation techniques we reviewed in the previous section usually rely on some approximation parameter and guarantee some level of precision. However, the interactivity of the multimedia exploration requires also the guarantee on the efficiency or the speed of evaluation, thus in the end of the previous section we outlined parallelism between evaluation of the exploration operations and the tasks in real-time systems.

When it comes to the visualization, beside the grid-based solutions, which are very similar to the visualizations of results from querying systems, also more progressive ones were used in the examined systems, e.g., the force-directed placement. While the grid-based visualizations do not have to solve the problem with overlapped visualized objects, the non-grid visualizations better preserve the structure of relations between the objects from the descriptor space.

The user interface is another research topic in which the exploration systems from our survey were reviewed. The common feature widely used in many of the examined systems was two directions of the exploration operations – vertical and horizontal. They are based on the user interfaces used in the map services, what seems to be a reasonable choice as the map services are used by millions of users every day.

Chapter 3

Designing Multimedia Exploration System

On behalf of the multimedia exploration overview presented in the previous chapter, in this chapter we introduce the general multimedia exploration system which is able to support the interactive multimedia exploration in arbitrary conditions. In the first part of the chapter we describe the *architecture* of such a system and in the second one we overview details of the *instant queries*, the similarity queries that can be terminated whenever it is necessary.

3.1 Motivation

Although some of the multimedia exploration systems from the previous chapter consider the index support and the approximate search to improve the efficiency of the multimedia exploration, to the best of our knowledge, none of the related techniques consider the real-time responses of the exploration process (i.e., decrease of lagging) in context of the large multimedia datasets and the mobile environment. We believe that guaranteeing the real-time response is one of the most crucial requirements on a successful multimedia exploration system, even more important than steadily high precision of the retrieval process. Hence, in the design of a multimedia exploration system, we focus on the approximate search techniques guaranteeing the user-defined response time and allowing occasional inaccuracies for long running queries. Furthermore, the response time can be controlled by the user, who adjusts the trade-off between the precision and the efficiency for various exploration tasks. For example, for the casual exploration of an unknown collection, the user would probably prefer the quick response time. While when already searching for some specific objects (e.g., at the end of their exploration), the user more likely accepts longer responses for a qualitative better result.

3.1.1 Real-Time Similarity Queries.

Before we introduce the proposed architecture of an exploration system, we provide the necessary background for the *near real-time similarity queries* performed in a limited time frame.

Very popular similarity queries are the k nearest neighbors (kNN) queries, which return a collection of k most similar objects to the query object. With such a design, they appear to be more suitable for the time-limited queries, because unlike the range queries they guarantee the fixed number of result objects from the very early point of their evaluation. Therefore, for the rest of the thesis, we consider the $k\mathcal{NN}$ queries only.

Our main intention for the similarity exploration queries is to *limit* their *response time*. Hence, to do so, we define $k\mathcal{NN}(t)$ as the *time-limited* $k\mathcal{NN}$ *query* which returns up to k most similar objects obtained from such a subset of the database collection which is accessed during the query processing within the restricted time frame t. In our design, we suppose that these time-limited queries occur in batches initiated by the user interaction with the exploration system. We refer to the set of queries of such a batch as the *query stream*.

3.2 Architecture of Multimedia Exploration System

Following the chapter introduction, in this section, we provide detailed information about the high-level architecture of a multimedia exploration system proposed in [18], we reference to the system as to the *RTExp*. Beside that, we also outline the functionality of the individual system layers in the architecture and describe the data-flow of the whole system with the typical use cases. In the proposed design of the RTExp, we follow these three invariants with respect to the principles of the multimedia exploration:

- the system displays the visualized part (which corresponds to the currently explored portion) of the whole underlying multimedia collection, the visualized part is specific to the user session
- each user action (zoom, pan, select) generates the stream of similarity queries, which are required (for the sake of scalability) to be evaluated within a limited time period
- resulting objects are visualized according to their similarities – the objects that are close to each other in the visualized space are more similar than those located distantly (see Section 3.2.5)

The useful exploration system has to be provided with the intuitive and comfortable user interface that is easy to control. Moreover, as outlined in Section 2.3, the user interface should be also responsive, otherwise the user performing the exploration will be discouraged with the response time of the system. Therefore, it is also needed to ensure the instant processing of all (underlying) operations and, ideally, to distribute intermediate query results immediately. If the user interface will be designed in this way, the users will not experience any lagging or delays.

From the technical point of view, the RTExp combines the ideas from existing exploration systems (see Section 2.1) such as the approximate similarity search or the intuitive user interface. In addition to those principles, we include our innovations in the form of the *timely-limited similarity queries* and the *query streams* (see beginning of this chapter) using a *modular approach*. In the design of the RTExp, we subdivide the whole system in smaller parts which correspond to independent modules with predefined interfaces and which provide a specific functionality. This enables us to better reflect the future change requests from the users, to increase the support for the scalability, and to easily implement enhancements in order to keep track with the state of the art technologies. From the more general view, the architecture is divided into three layers: the *presentation*, the *logic*, and the *data* layer with one extra system layer for *communication* purposes. In the following sections, each of these layers is described into details in the top-down direction as they are depicted in Figure 3.1.

3.2.1 Presentation Layer

The presentation layer represents the graphical user interface visible and accessible to the end users via a web browser or by a native application in mobile devices. As it provides the only access point to the system intended for the end users, it necessarily has to be intuitive and well-designed. Clear separation from the underlying layers allows us to easily build the client application, no matter if the client is a web, a desktop, or a mobile application.

Each user action performed in the visualized exploration space (e.g., panning, zooming, more details about these *exploration operations* will follow in Section 3.2.5) is submitted to the logic layer, where this action is subsequently transformed into the query stream (see Section 3.1.1). If the user decides to navigate his exploration in other directions while the system is still actively processing the query stream, we send the request to terminate evaluation of this query stream and the new one is generated based on the most recent user action.

Note that the presentation layer only handles the user interaction with the system and provides the interface to work with the system. It does not contain any logic of how to convert the user actions into the query streams, how to evaluate individual queries, or when to return new results. This functionality is transparently delivered by the application logic layer.

3.2.2 Logic Layer

The application of middle layers (i.e. an application logic) between the end user interface and the data layer is not new and exists in various systems. Authors either declare it explicitly, as it is in the case of the *mediator* component [98], or include the layer implicitly based on the functionality requirements without any special attention. There are two main tasks performed by the logic layer in the architecture of the RTExp:

- to transform the exploration operations to the query streams
- to deliver the partial/final results from the data layer back to the user.

In the first of these tasks, we take the user input encapsulated in the exploration operations and analyze it within the *Query analyzer* component. The analysis should take into consideration the query streams that are currently evaluated together with the direction of the exploration. While possibly in the future, it can also consider the query costs and prioritize those queries which results add more value to the current exploration context. Next, according to the result of

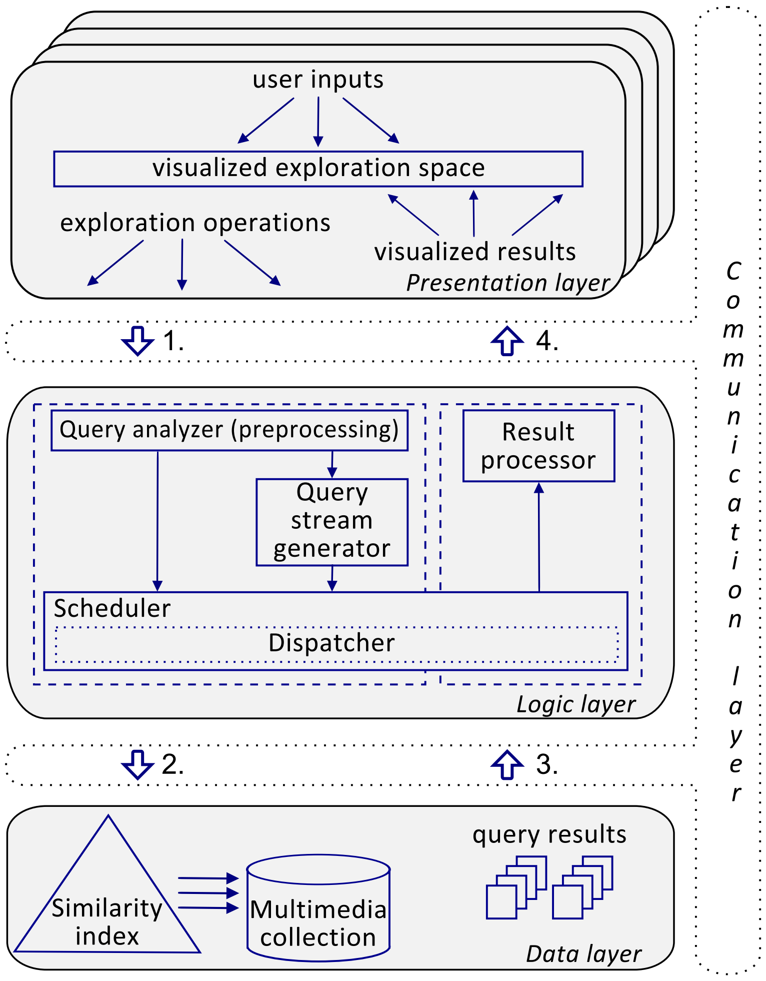


Figure 3.1: The multimedia exploration system overview - the high-level architecture.

this analysis, it is decided, whether it is necessary to terminate some of the running similarity queries (or the query streams), or whether the new query stream will be created. Our research concerning the ideas how to stop the running query stream is covered in Section 3.3, while generating the new query stream involves the research about the exploration operations, which is described in Section 3.2.5. In our architecture depicted in Figure 3.1, the transformation of the exploration queries is covered by the additional logic component, the *Query stream generator* .

When the query streams are generated, all requests are transfered to the *Scheduler*, another component of the logic layer that schedules and controls the query processing by employing a prioritized queue. The concept of a scheduler is well-established in real-time operating systems [97], from which we adopt the prioritizing principle of the task execution and use it for the query streams – they are executed in the order which is based on the user preferences (see more details in the following paragraphs).

Once the query stream from the processing queue is ready to be executed and has not been terminated yet, it is dispatched through the *Dispatcher* to the underlying data layer where the dedicated sequence of the similarity queries are evaluated. Afterwards, when the similarity queries are evaluated, the *Result* *processor* decides which query results are propagated to the presentation layer. But first, they have to be retrieved from the data layer.

3.2.3 Data Layer

The data layer is the most important layer, because it consists of the *multimedia collection* itself, and the guarantee of the efficient retrieval, the similarity access method, is also located here. Because we expect the evaluation of numerous similarity queries from the query streams, we prefer the access method that provides the best efficiency vs. effectiveness ratio. And while efficiency is usually the question of an approximation search, we evaluate and compare several metric access methods (MAMs) accompanied with the approximation technique that we propose in Section 3.3. The results of our comparison are provided in Section 3.3.4.

We can say that two factors can influence the real-time capabilities of similarity access methods in the RTExp architecture. The first one is the different query execution plans for computationally *cheap* and *expensive* similarity queries. We assume that the *cheap* similarity query can be executed "as is", however the *expensive* one should be approximated in some way. For example, in the M-tree [52] we can use the top-down search strategy for the cheap queries, however for the expensive ones we should apply the bottom-up approach (fast traversing to the leaf level followed by the enhancement of the first results), with regards to the observations we get from Figure 2.27 in Section 2.3. The second idea is the instant evaluation of similarity queries (the *instant queries*), already mentioned in the introduction of this chapter. The main idea is to ensure that any running query can be terminated from 'outside' (the logic layer in our case) whenever it is necessary. We believe that benefits of the instant queries can be utilized in many scenarios:

- When the user requires to get the result of a similarity query very quickly, while the relevancy of the obtained results is not so important.
- When the result of a similarity query is obsolete and not required anymore. In the exploration scenario, it happens when the user decides to explore other parts of the database.
- When the overhead of a similarity query influences the scalability of the whole system. System scalability can be violated also when the number of simultaneous queries rapidly increase.

Further details of the instant queries follow in Section 3.3.

Beside the research on the approximate technique that utilizes the multimedia exploration, we study the user interaction with the exploration system and propose the exploration structure that directly supports the exploration operations. More details about this *multi-layer exploration structure* are provided in the next chapter.

3.2.4 Communication Layer

The *communication layer* establishes the virtual connection between individual layers within the system and ensures that the messages (requests/responses) are sent/delivered to the appropriate recipients (modules). The typical data flow is in Figure 3.1 denoted by the sequence of numbers next to the appropriate arrows – from the user actions translated into the requests for the query analysis (1), their subsequent transformation into the query streams (2), to the appropriate partial/final responses of the similarity queries (3), and their inclusion within the visualized exploration space (4).

Connecting the presentation layer with the logic layer is simple and straightforward. The requests holding the exploration operations translated from the user inputs are sent from the presentation layer and the system logic layer processes them accordingly. The corresponding objects from the multimedia collection are returned as a response to be included within the visualized exploration space. Together with them is returned the list of objects that should be removed from the current visualization.

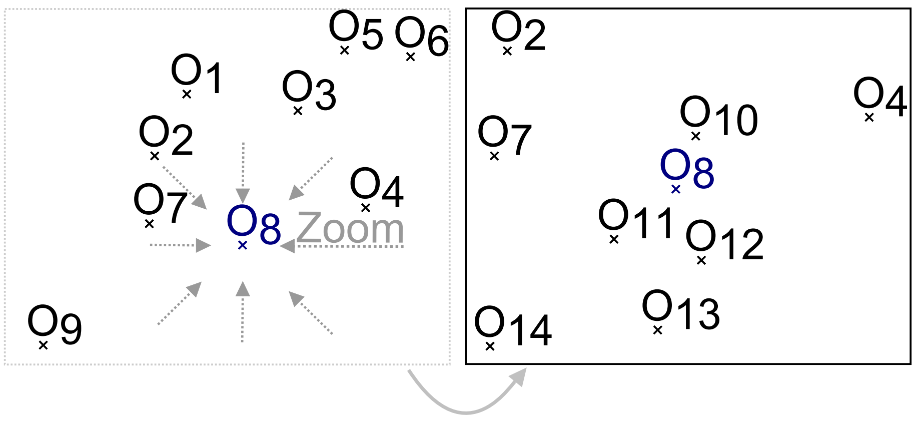
3.2.5 User Interface

The applicability of a multimedia exploration system largely depends on the user interface as we already mentioned in Section 2.1.3. The more intuitive and ergonomic interface the client application provides, the more likely the end users will use the system and benefit from it. As it results from Chapter 2 of the related work, two most used user actions in the exploration process are *zooming* (Section 3.2.5) and *panning* (Section 3.2.5). We select these two user actions as the ideal candidates for the basic exploration operations because they are intuitive and have been known to billions of users as the basic operations used in the web mapping services. The list of these supported operations can be naturally extended in the future, e.g., if the visualization of the system displays a 3D space instead of the supposed 2D space, several other well-known operations could be also used such as, for example, *rotating*.

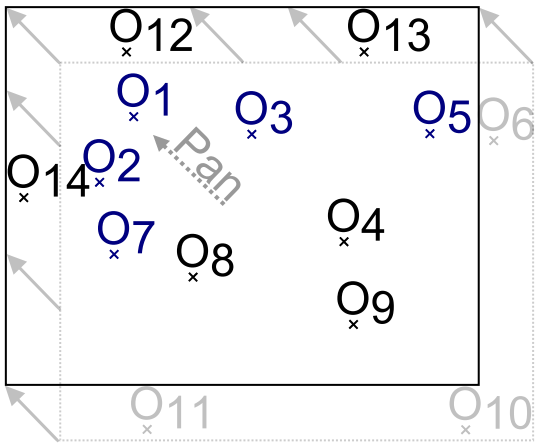
As we already outlined in the description of the presentation layer (see Section 3.2.1), all *user actions* captured as inputs from a touch device, a mouse, or a keyboard are transformed into the *exploration operations*, which represent the adequate requests for the lower layers within the RTExp system. These user actions are changing the current state of the active user session, which is represented by the visualization of the exploration space on the user screen. In the following sections, we describe the supported user actions into more details and explain how we transform them into the actionable exploration operations.

Zooming

This operation is just a naive implementation of a zoom in/out mechanism, the further research on this topic is necessary. Imagine that the visualized set of objects from the multimedia collection represented by the objects O_i as depicted in Figure 3.2a is the current *exploration state*. In this state, we are interested in the context of objects similar to the specific object O_8 and we *zoom in* to this object. The action of zooming in on the target, depicted with the arrows, subsequently reveals the previously invisible objects O_{10}, $O_{11} \ldots O_{14}$. The newly discovered objects are typically more similar than the previously shown objects or provide the visualization of more specific object clusters. Notice that beside the appeared objects, some of the previously displayed objects (e.g., O_1, O_6)



(a) Zooming



(b) Panning

Figure 3.2: Exploration user actions

disappear from the visualization as they are not present anymore in the part of the exploration space that is currently visualized (i.e., not present in the new exploration state).

The opposite action, the *zooming out*, on the contrary returns the new objects that are less similar to the previous ones. Furthermore, to preserve the impression of the vertical exploration, some too specific objects should disappear as they are replaced by those new appeared objects, which come from more general clusters.

Whenever the user executes a zooming action, the adequate exploration operation is created. It consists of (a) the target zooming point, (b) the object closest to the target zooming point, and (c) the objects closest to the boundaries of the visualization component. The zoom in initiates the single similarity query for which the query object is the closest object to the zooming target point, while in the case of the action zoom out the query stream is generated consisting of multiple similarity queries, where the query objects for them are used those objects that are closest to the visualization boundaries.

Panning

The *panning* action advances the exploration in the specific horizontal direction and thus also changes the current state similarly as the zooming action. The action is depicted in Figure 3.2b, the dotted rectangle represents the current

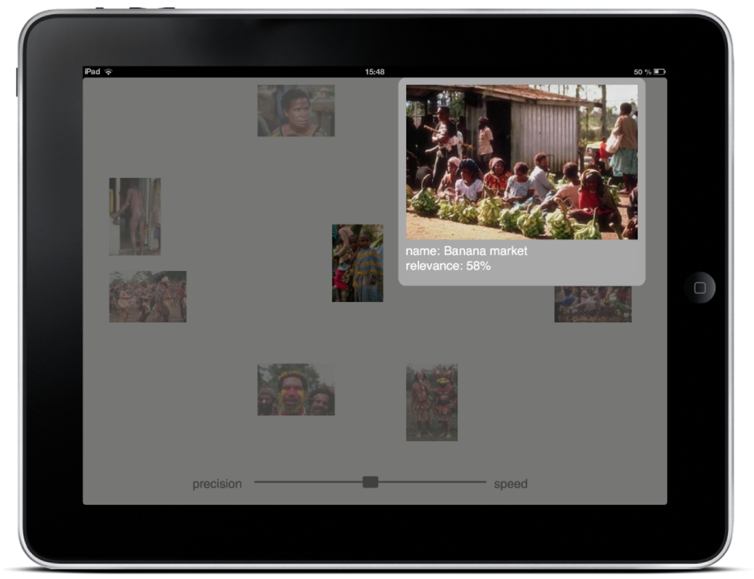


Figure 3.3: RTExp exploration system

state before the panning action is applied and the arrows outline the direction of the action. After this operation is performed, we get the new set of visualized objects, in the figure bounded by the solid black rectangle, which represents the new exploration state.

The appropriate exploration operation is created for the panning action, it includes several parameters such as (a) the panning direction and its size, (b) the objects closest to the visualization boundaries following the panning direction, and (c) the objects that get outside the visualization boundaries of the new state. Then, the query stream of multiple similarity queries is generated where the query objects are those objects which are closest to the visualization boundaries following the panning direction and its size. As depicted in Figure 3.2b, the panning action should also consider the exclusion of those objects of the current state that reside in the opposite direction (with regards to the center of visualization) to the objects which the panning action follows.

RTExp Presentation Layer for Mobile Devices

To validate the feasibility of the proposed exploration system architecture, we implemented the first prototype, the RTExp, together with the presentation layer for mobile devices.

The initial user interface for iPad tablets as depicted in Figure 3.3 gives the overview of the RTExp as it is presented to the user. In every moment of the user session, the screen displays the current state of the exploration process according to all performed user actions and operations since the beginning of this session. For better orientation, the objects from the underlying databases (i.e., images) are displayed with different opacity values. These values represent the similarities of the individual depicted objects relevant to the query object(s) of the previous similarity query(ies). The more the object is similar to the query object with respect to the corresponding similarity query, the smaller the opacity value is.

This similarity value is displayed for each object as the *relevance* value whenever the particular object is selected by the user. Additionally, we display other metadata values of the currently selected object such as its name, as can be seen in the figure for the 'Banana market' object[1].

The powerful slider control at the bottom of the screen (see Figure 3.3) enables the user to adjust the evaluation process of all similarity queries within the context of his session. Sliding to the left gives more precise (but potentially slower) results, while putting the slider to the right returns faster (yet not necessary relevant) results. This setting is transparent to the user as the system itself transforms this value to the individual exploration query settings, such as the parameter for the time-limited $k\mathcal{NN}$ query, mentioned in Section 3.1.1.

3.3 Instant Similarity Queries

Following the general architecture, in this section we focus on the one specificity that is the crucial part of the proposed design, the time-limited similarity queries. As their design is based on the termination of them, which can be performed almost instantly, we refer to them as to the *instant similarity queries*.

3.3.1 Motivation

The approximate techniques presented in the end of the previous chapter provide some user-specified parameter that more or less controls how 'good' or 'bad' the query result should be. The perfectly tuned approximation parameter can lead to more efficient queries, however, there is still no guarantee that the correctly set parameters result in the desired efficiency, i.e., the query is evaluated in such a query time which the user prefers.

The motivation for our work is firstly based on the basic requirements of a multimedia exploration system, which is typically tightly connected to the GUI where the highly interactive 'dialog' between the user and the system is expected. Considering this fact, the requirement on the precisely controlled real time of the underlying similarity queries becomes critically important. Since the number of queries generated by the user interaction at one particular moment is strongly affected by the user behavior (clicking, panning, zooming) and/or by the number of simultaneously working users, the main requirement on the query evaluation is the guarantee of the query termination in expected time in arbitrary conditions. The second motivation arises from the previous one. As the similarity queries are continuously generated by the exploration system itself (while the user is zooming, panning, etc.), such an approximate method is preferred that does not require any user-defined parameter.

3.3.2 General Requirements

When taking all presumptions into consideration, the most suitable solution is the model of the instant termination of a similarity query, i.e., the query that is being processed might be terminated at any time and the partial results obtained till that time must be immediately consolidated and returned. However,

[1]The name of the object is displayed only if it exists in its metadata, and it is not used as the supportive information for the similarity queries.

the completely *any-time* query termination puts some requirements on the corresponding similarity index (or the querying algorithm, respectively). The first requirement concerns the results of the similarity query, the query should produce the *monotonically approximative* [99] (intermediate) results. In our case, this means that the intermediate result in the time $t + 1$ cannot be worse than the result available in the time t. The second requirement is a technical one, the runtime environment supporting the instant termination should run at least in two threads, one for 'the terminator' (client) and one for 'the evaluator' (server).

We implemented the instant query termination into some well-known metric access methods to validate its benefits. The next section describes general constructs as well as the particular implementations.

3.3.3 Instant Termination in MAMs

The idea of the query termination in specific MAMs is quite simple and straightforward. During the processing of the query, someone from 'the outside' has to notify the similarity index (or its querying algorithm, respectively) to immediately stop the query evaluation. Following this scenario, we have used two threads – the first one represents the system performing the *notification of the instant termination* and the second one is used for the query evaluation itself. It is obvious that for the query evaluation is possible to use more threads than single one, but for our demonstrative purposes, it is sufficient to use just one.

The algorithm of the similarity query evaluation consist mostly of processing the objects from the database within some loop, or in a recursion. Presuming such cycling behavior, we can check for the notification of the termination in the beginning of each cycle (or the recursion level), so when the notification occurs we can break the cycle (or the recursion) and immediately return the results computed so far. Actually, the query result cannot be usually returned exactly immediately, because all MAMs perform some necessary operations (i.e., the *finalization phase*) before returning the result, e.g., the cleanup of the temporary structures, the re-ordering of the result objects, etc. Nevertheless, it is quite obvious that such indexes are preferred for which the time of the finalization phase is as low as possible (milliseconds or at most tens of milliseconds).

Sequential Scan

The sequential scan can be considered as the primitive MAM, and the extension of its query algorithm with the instant termination is quite straightforward, see Algorithm 3.3.1. At the very moment of the instant termination, the algorithm is breaking its main loop and the result computed so far is returned (the additional operations are not required).

Pivot Table

The next MAM whose query algorithm we extended with the instant query termination is the pivot table [73], based on the LAESA [75] variant. Moreover, beside the original metric pivot table, we have implemented the instant termination also into the Ptolemaic Pivot Table [77]. The extended range query algorithm of the

Algorithm 3.3.1: SEQUENTIALSCANRANGEQUERY($\boldsymbol{q}, \boldsymbol{r}$) $\mapsto$ *Result*

```
1: $Result = \emptyset$
2: for each $o$ in $\mathbb{S}$ do
3:     if InstantTermination then
4:         break for {no explicit finalization required}
5:     compute $\delta(q, o)$
6:     if $\delta(q, o) \leq r$ then
7:         add $o$ to $Result$
```

Algorithm 3.3.2: PIVOTTABLERANGEQUERY($\boldsymbol{q}, \boldsymbol{r}, \boldsymbol{mode}, \boldsymbol{k}$) $\mapsto$ *Result*

```
1: $Result = \emptyset$
2: for each $o$ in $\mathbb{S}$ do
3:     if InstantTermination then
4:         break for {no explicit finalization required}
5:     if $mode$ = triangle or $mode$ = triangle+pto then
6:         if TriangleFilter($q, o, r$) $> r$ then
7:             continue for
8:     if $mode$ = ptolemaic or $mode$ = triangle+pto then
9:         if PtolemaicFilter($q, o, r, k$) $> r$ then
10:            continue for
11:    compute $\delta(q, o)$
12:    if $\delta(q, o) \leq r$ then
13:        add $o$ to $Result$
```

pivot table is outlined in Algorithm 3.3.2. Similarly like in the case of the sequential scan, the main loop is immediately broken when the instant termination occurs (without the need of any additional cleanup operations).

(P)M-tree

The last group of MAMs we prepared for the instant query termination are hierarchical MAMs – the M-tree [52] and the PM-tree [79]. Algorithms 3.3.3, and 3.3.4 show the implementation of the range and the kNN query algorithm extended with the instant termination for both MAMs. Whereas the instant termination of the range query is similar to that one in the pivot table implementation, the termination of the $k\mathcal{NN}$ query is little different and we propose two variants, the *strict* and the *lazy* termination level.

Strict termination. The strict termination level is similar to the instant query termination of the sequential scan or the pivot table, the main loop is instantly broken and after some negligible cleanup the intermediate result is returned.

Lazy termination. On the contrary, the $k\mathcal{NN}$ query with the lazy termination level is not terminating instantly. After the termination is notified, the priority queue (used for the unprocessed nodes in the $k\mathcal{NN}$ query algorithm of the (P)M-tree) is checked for the leaf nodes. The entries of each leaf node are then processed one by one (ignoring the inner nodes in the priority queue) and the

Algorithm 3.3.3: PMTreeRangeQuery($\boldsymbol{q}, \boldsymbol{r}$) $\mapsto$ *Result*

```
1: $Result = \emptyset$
2: $Stack = \{root\}$
3: while $Stack <> \emptyset$ do
4:     $node \leftarrow Stack$.Pop
5:     if InstantTermination then
6:         continue while {finalization - emptying of the stack}
7:     for each $entry$ in $node$ do
8:         if InstantTermination then
9:             break for
10:        if FilterObjectOrRegion($q, entry$) $> r$ then
11:            continue for
12:        if IsPMTree and not(RingsIntersect($q, r, entry$) then
13:            continue for
14:        if IsLeaf($node$) then
15:            compute $\delta(q, o)$
16:            if $\delta(q, o) \leq r$ then
17:                add $o$ to $Result$
18:        else
19:            $Stack$.Push($entry$.ChildNode)
```

intermediate result are being improved as it happens during the standard query evaluation. This idea of postponing the termination can significantly improve the approximate result, especially when the query evaluation got almost to its end in the moment of the termination. If we look at the $k\mathcal{NN}$ query in more detail, the processing of the leaf nodes in the priority queue is actually the very moment of improving the intermediate result. It is obvious that the lazy termination is not for free and the overhead of postponing the termination has to be added to the total query time. Unlike the pivot table (or sequential scan), the range or the $k\mathcal{NN}$ queries of the (P)M-tree require some cleanup operations such as emptying the stack (in the case of the range query) or emptying the priority queue (in the case of the $k\mathcal{NN}$ query).

3.3.4 Experiments

In this section we provide and discuss the results of the experiments performed on the MAMs with their querying algorithms extended by the instant termination, as we present it in the previous chapter.

Testing Datasets

We tested the instant queries on the ALOI database [100] composed of 70,000 images, using extracted feature signatures based on the color, the position and the image texture. The second dataset we used was the subset of the CoPhIR [101] consisting of 1,000,000 images, while the feature vectors were represented by the MPEG-7 descriptors – the Scalable Color and the Color Structure [102]. As the distance function we deliberately used one cheap metric, the Euclidean

Algorithm 3.3.4: PMTree$\boldsymbol{k}$NNQuery$(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{termLevel}) \mapsto NNQueue$

```
 1: $NNQueue = \emptyset$
 2: $PRQueue = \{root\}$
 3: while $PRQueue <> \emptyset$ do
 4:     $node \leftarrow PRQueue$.RemoveFirst
 5:     if InstantTermination then
 6:         continue while {finalization - emptying of the priority queue}
 7:     for each $entry$ in $node$ do
 8:         if InstantTermination then
 9:             if not(IsLeaf($node$)) or $terminationLevel$ = strict then
10:                 break for
11:         if FilterObjectOrRegion$(q, entry) > NNQueue$.Last then
12:             continue for
13:         if IsPMTree and not(RingsIntersect($q, NNQueue$.Last, $entry$) then
14:             continue for
15:         $objDist \leftarrow \delta(q, entry.Obj)$
16:         if $NNQueue$.Count $< k$ or $objDist \leq NNQueue$.Last then
17:             $NNQueue$.Update($entry, objDist$)
18:             $PRQueue$.Filter($objDist$)
```

(L_2) distance for the CoPhIR and one expensive metric, the SQFD [14] for the ALOI database.

Experiment Settings

As mentioned in the previous section, we implemented the instant queries into the sequential scan, the pivot table, the Ptolemaic Pivot Table, the M-tree, and the PM-tree. We simulated the notification of the query termination (in a real-world application invoked by the impulse of the user) using the separate thread. The timing of the termination signal in the experiments was set from 1ms to 4096ms. In each of these termination times, the query result was compared to the result of the exact (complete, respectively) search using the Jaccard distance (see Equation 3.1) to determine the approximation error.

$$J_\delta(Exact, Approx) = 1 - \frac{|Exact \cap Approx|}{|Exact \cup Approx|}. \qquad (3.1)$$

Besides the error, the time of the finalization phase was also measured as the time between the moment of the termination notification and the time when the query result was returned.

The testing environment consisted of 2x Intel Xeon X5660 2.8GHz, 24 GB RAM and Windows Server 2008 R2 64-bit. For the better orientation in the figures we summarized all used labels in Table 3.1.

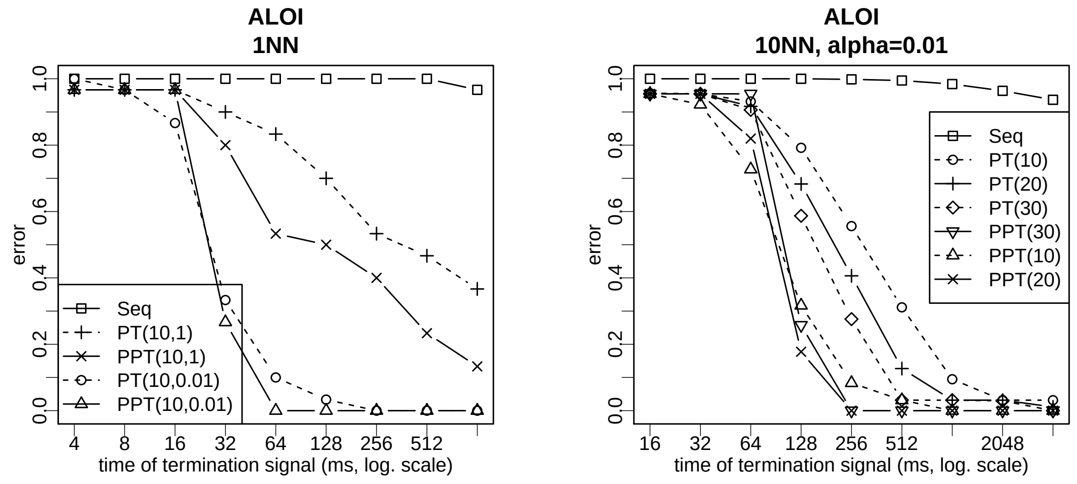
Results

Firstly we present the comparison of the pivot tables with various parameters. As we can see in Figure 3.4a, the parameter α of the SQFD distance[2] has the

[2]For more details about the SQFD parameters see [14].

Label	Description
Seq	sequential scan
PT(n, α)	pivot table
PPT(n, α)	Ptolemaic Pivot Table
MT(n, S|L, α)	M-tree
PMT(n, S|L, T|P, α)	PM-tree
	n is the number of used pivots
	α is the paramter of the SQFD
	S|L stands for Strict or Lazy Termination Level
	T|P stands for triangle or ptolemaic filtering
	RI stands for forced reinsertions
range(k)	range query returning approximately k objects
total time	query time before termination signal
	+ finalization time

Table 3.1: Labels used in figures.



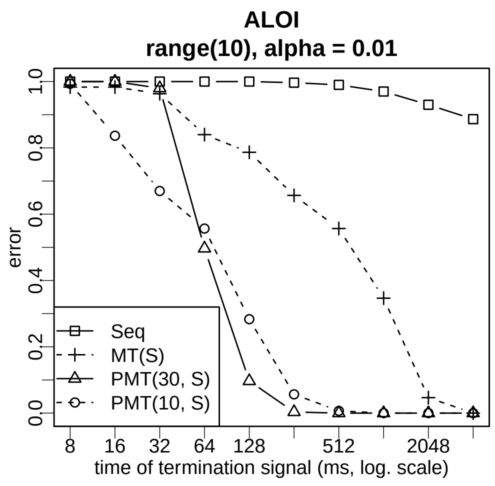
(a) (b)

Figure 3.4: $k\mathcal{NN}$ queries by the pivot table on the ALOI database (a) the impact of the SQFD parameter alpha (b) the impact of the number of pivots.

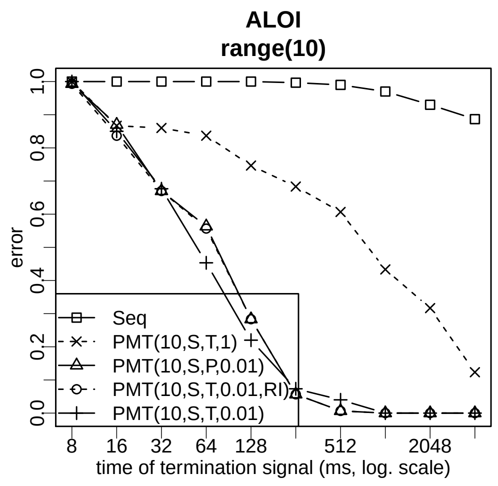
significant influence on the query efficiency. After 32ms the pivot table with $\alpha = 0.01$ shows four times lower error than the pivot table with $\alpha = 1$. Moreover, the configurations of the pivot table with $\alpha = 0.01$ finished after 128ms, while the others were not completed even after 1s. Hence, if not stated otherwise, in the next experiments we used $\alpha = 0.01$ for the SQFD distance.

Figure 3.4b shows the pivot tables and the Ptolemaic Pivot Tables with the varying number of pivots. Here we can observe, how the number of pivots can positively influence the speed of the approximation and also the query time (at least for the classic pivot table). We can also see that the results of the Ptolemaic Pivot Tables are significantly better than the results of the classic variants.

Next, as depicted in Figure 3.5, we compared tree-based structures, the M-tree and the PM-tree. The querying process of the M-tree is not very suitable for

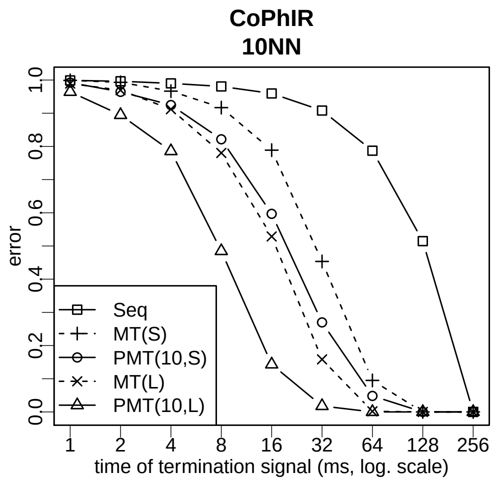


(a)

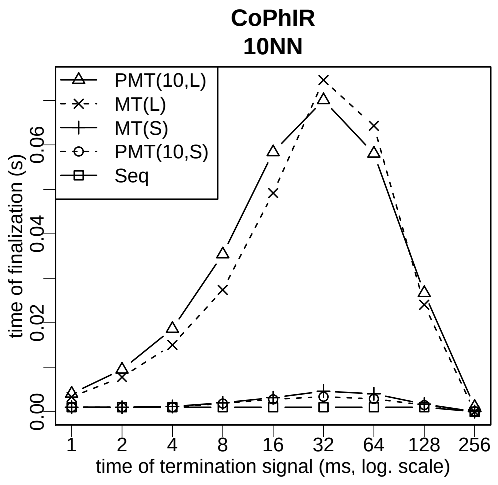


(b)

Figure 3.5: Range queries in tree-based MAMs on the ALOI database. (a) the impact of the number of pivots, (b) various PM-tree parameters.



(a)



(b)

Figure 3.6: $k\mathcal{NN}$ queries by the tree-based MAMs on the CoPhiR database - the impact of the postponed termination on (a) the error in the result (b) the finalization time - in seconds

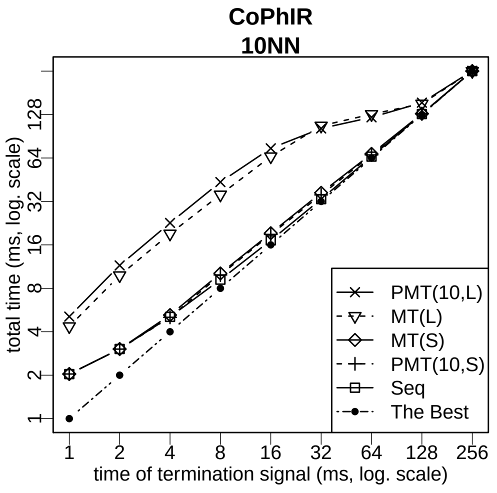
the instant termination as the error was still over 50% after 512ms, what is the time when the PM-tree has already finished. For the latter, the influence of the number of the pivots is quite interesting. While for the short termination times the higher number of the pivots is just an overhead – 100% error with 30 pivots in comparison to just 65% error with 10 pivots – with the increasing termination times the benefits of more pivots take effect and the query finishes with lower error in the result. It is also interesting that different variants of the PM-tree have not significant impact on the approximation process at all, only the high value of α is something that matters (similarly as for the pivot table).

The following tests show the results of different termination levels as they were proposed in Section 3.3.3. In Figure 3.6 we can observe that the postponing of the termination (the case of the lazy termination level) should be beneficial, however, if we analyze it in more depth we shall see that the finalization time increases vastly.

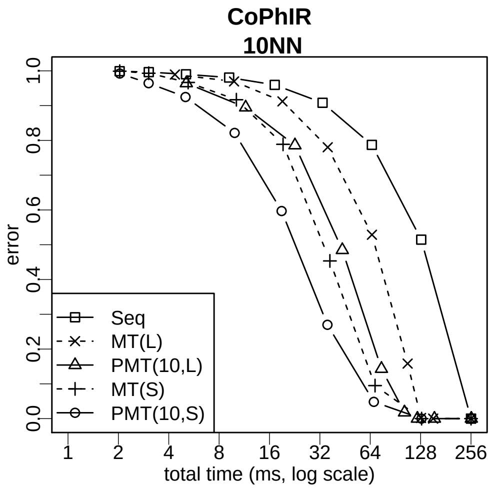
The impact of the excessive finalization time is better shown in Figure 3.7, where the total time (the time of query evaluation + the time of the finalization phase) of the lazy termination level is sometimes even four times larger than the case of the strict termination level. From the results in Figure 3.7b it is clearly obvious that the strict termination level beats the lazy one.

Figure 3.8 shows the comparison of all presented MAMs. While in the case of the dataset with the cheap distance function the PM-tree is the clear winner and the pivot tables behave even worse than the sequential scan, the case of the ALOI database with the expensive SQFD suits better for the pivot tables. We can also see that the simple implementations of the instant query termination as presented in this paper may be not sufficient; the fastest index on the ALOI reaches some reasonable results after 100ms. Hence in the future, the more specific approaches for particular MAMs are needed to design.

In the last figure (Figure 3.9), there are presented the results of the range queries on the ALOI database. The pivot tables won also in this scenario, showing even better results than in the case of the $k\mathcal{NN}$ query (because of the cheaper maintenance of the intermediate result and also because of the quicker finalization phase).



(a)

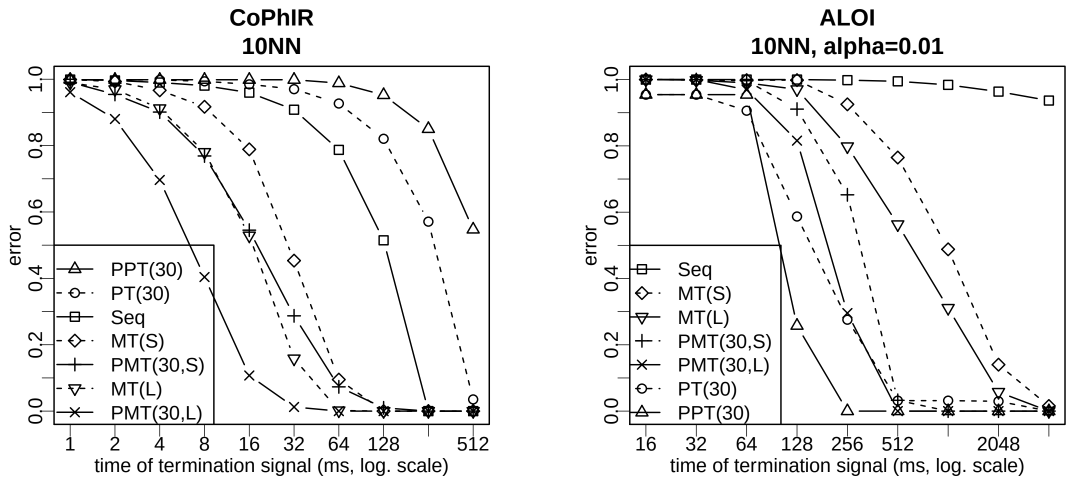


(b)

Figure 3.7: $k\mathcal{NN}$ queries on the CoPhiR database - (a) the impact of the postponed termination on the total query time - in milliseconds (b) the aggregated graph - the error for the total query time

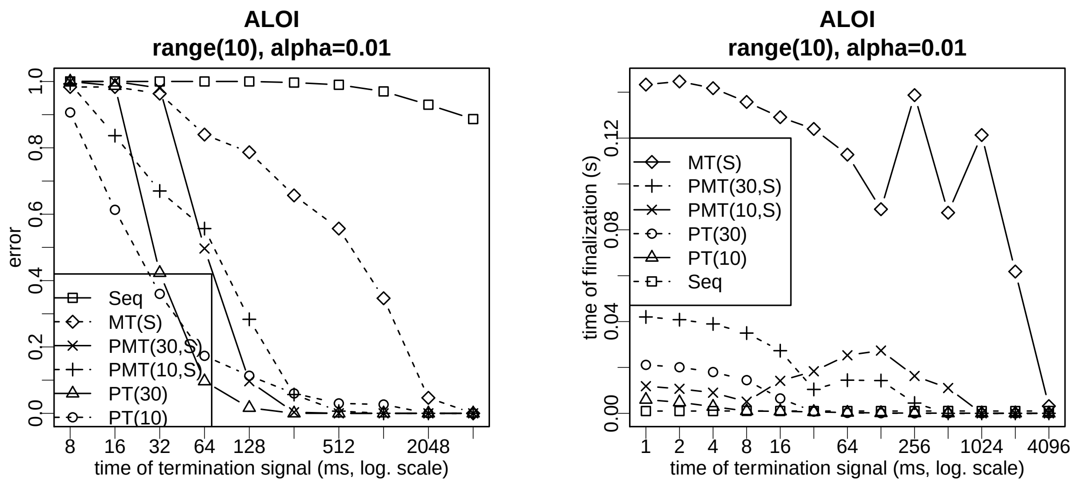
3.3.5 Conclusions on Instant Queries

In this section, we showed the possible new direction in the research of the approximate similarity search based on the instant termination of the similarity queries. We studied the behavior of several adapted MAMs in case that they are



(a) (b)

Figure 3.8: $k\mathcal{NN}$ queries on the CoPhiR and the ALOI database – an inter-MAM comparison



(a) (b)

Figure 3.9: Range queries on the ALOI database – the inter-MAM comparison (a) the error in the result (b) the time of the finalization phase in seconds

terminated before the full query evaluation is completed. It is quite understandable that the simple implementation of the instant termination into the existing MAMs without any further modification of their query algorithms does not guarantee the best possible result, however, the concept of the instant queries which can be terminated whenever it is necessary could be beneficial in many scenarios, especially in the multimedia exploration systems. As said in one provocative survey [103]: "Only the real-time cost is the moment of truth for an end user [...], a MAM is either fast or slow."

3.4 RTExp Challenges

During designing the architecture of the RTExp, we experienced several issues, which we would like to highlight, we suggest possible solutions, and we address them afterwards.

- **Continuous Query Evaluation.** For the continuous performance, the concept *publish/subscribe data delivery* from the real-time cloud systems could be adopted. Whenever the user executes an action, it is translated into the appropriate query stream. Then, the visualization of the underlying database is continuously updated with the partial and the final results, while the system is evaluating similarity queries in the background. The *Result processor* decides whether the obtained (partial, temporal) results are propagated to the presentation layer and which change is the user notified about.
- **Exploration Operations.** We realize that the definitions of the exploration operations we provided in Section 3.2.5 are too high-level with no emphasis on their detail implementation. In the next chapter, we propose a new exploration structure together with the definitions of the exploration operations on it, which are more detailed than the ones provided in this chapter.
- **Visualization of the Multimedia Collection.** So far, we have not addressed the nontrivial problem of mapping the multimedia collection to the visualization space (see Section 2.1.2), because we wanted to address the real-time challenges first. Hence, for the first simple visualization, it can be considered the physical model based on the multidimensional scaling and the simulated annealing [35], successfully used and verified in another exploration system [43].
- **Scalability.** Sooner or later a successful exploration system will be challenged by the growing number of its users, followed by continuous additions of objects to the multimedia collection. To be prepared for such a situation, besides the parallelization of the framework processes, at least the data layer of the RTExp should be implanted in a distributed environment.

Chapter 4

Multilayer Multimedia Exploration

Following the research on the architecture of a multimedia exploration system, in this chapter, we focus on two parts of the design proposed in the previous chapter.

Firstly, we present the new structure intentionally designed for supporting the multimedia exploration [19], and which in the proposed design can be understood as the similarity access method in the data layer. The new structure combines two approaches mentioned in Section 2.1.3, it utilizes the similarity indexes for the *horizontal browsing* and employs the multiple layers enabling also the *vertical browsing.*

Secondly, we present the exploration layout for this new multimedia exploration structure. This layout can be understood as the visualized exploration space, the component from the presentation layer mentioned in the previous chapter.

4.1 MLES - Multilayer Exploration Structure

The multimedia exploration usually starts in the so-called *page zero view*, where the users start the exploration using the limited number of representative objects. Then, the users consecutively zoom in to specific parts of the view, pan to other groups of objects, or zoom out if the actual view is filled with undesired objects, all the time seeing the same (small) number of objects.

From the above exploration use case, we assume that the *exploration structure* should be able to provide the representative distinct objects for the earlier phases of the exploration, while the more related (similar) objects to each other should be retrieved in the later stages, i.e. it should provide some mechanism to search in the *different level of details.* At the same time, whenever the users select the object to zoom in to the details of its neighborhood, this object should be visualized also in the query result to preserve the fluency of the exploration. Based on these assumptions, we define the Multilayer Exploration Structure (see Figure 4.1):

Definition 4.1. *(Multilayer Exploration Structure). Given a dataset $\mathbb{S}$ of objects (descriptors of the respective multimedia objects), the Multilayer Exploration*

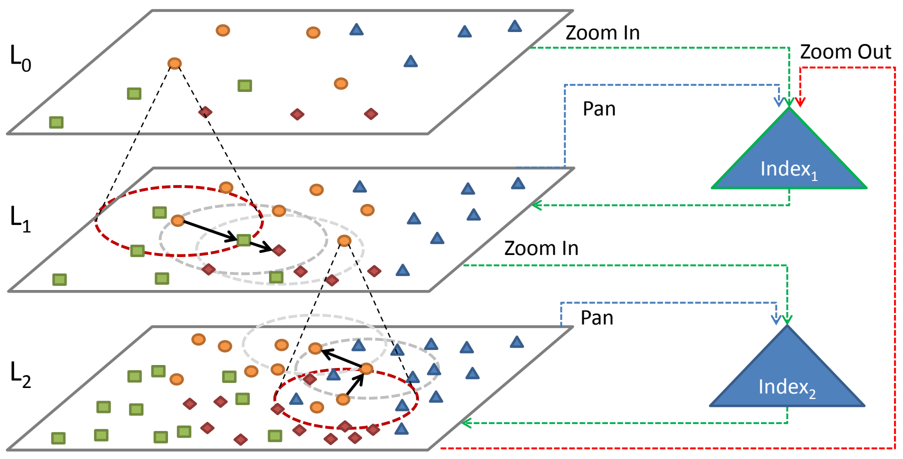


Figure 4.1: The MLES with similarity indexes $Index_1$ for the layer L_1 and $Index_2$ for the layer L_2.

Structure, $MLES(\mathbb{S}, m, v, \phi)$, is a system of $m+1$ subsets $\mathbb{L}_i$ (layers), where the subset condition holds:

$$\mathbb{L}_i \subset \mathbb{L}_{i+1}, \forall i = \{0, .., m-1\}$$

The smallest subset $\mathbb{L}_0$ represents the first depicted v objects (i.e., page zero view) and the proper subset $\mathbb{L}_m = \mathbb{S}$ represents the whole database. Selection of objects for each layer is determined by a selection function $\phi : \mathbb{N} \rightarrow 2^{\mathbb{S}}$, w has to comply with the subset condition.

With such a definition, each layer could be indexed separately by the most suitable similarity index that supports the $k\mathcal{NN}$ similarity query processing. For example, the upper layers containing the small number of objects can use an efficient *memory-based indexes* (e.g., the pivot table [74]), while the lower layers in the MLES hierarchy containing a lot of objects can use an efficient *disk-based indexes* [52, 79, 86] and/or GPU computing [61].

In order to select objects for the particular layers in the MLES, various techniques can be utilized (e.g., the random sampling, the (hierarchical) clustering techniques, or their combination). In the first design of the MLES, we consider just simple random sampling of the subsets[1] which is applicable for huge datasets. The objects in the dataset have to be just randomly mixed and then the subsets corresponding to the layers can be simply defined by using the prefixes of the dataset.

Since the number of displayed objects v (see Definition 4.1) depends on the size of a client device and on the user preferences, we need to answer the question of a various configurations of the zero layer $\mathbb{L}_0$. However, as we use the prefixes of the mixed dataset to define the layers, the page zero view corresponds just to the first prefix of the dataset. Therefore, the page zero view can be changed dynamically by using the prefix with a variable length according to the needs of the device (e.g., two zero layer configurations, one consisting of 10 objects for a small screen and one consisting of 50 objects for a large screen).

[1]Techniques guaranteeing the representativeness of the objects in the top layers are out of the scope of this paper and we leave them for future work.

4.1.1 Exploration Operations

A typical querying task in the similarity search, the k-nearest neighbor ($k\mathcal{NN}$) query (Definition 4.2) described also in Section 1.2.2, can be used also as the supportive operation for the exploration operations. Moreover, for the lower layers of the MLES with a huge number of objects, the approximate $k\mathcal{NN}$ search techniques (see Sections 2.2.3 and 3.3) can be also utilized, to achieve higher efficiency of the exploration. We believe that the multimedia exploration belongs to such a category of retrieval tasks where maximal precision is not required and thus the multimedia exploration can profit from the approximate search.

Definition 4.2. *(k-nearest neighbor query). Given a dataset S of multidimensional objects, a query object $q \in S$ and a query parameter k, the k-nearest neighbor query returns a set $NN_q(k) \subseteq S$ that contains the k most similar objects to the query object, for which the following condition holds:*

$$\forall o \in NN_q(k), \forall o' \in S - NN_q(k) : d(o,q) \leq d(o',q)$$

In the following, we define the exploration operations for the MLES, but before that, we introduce some labeling for better understanding of the described exploration situations. The *current user view* represents the collection of objects currently visualized on the user screen, while the *next user view* represents the collection of objects resulting from some exploration operation. We assume that the *current user view* of k objects is visualized as the static network [23], where only the most similar objects are connected to each other and the positions of the objects are influenced by the similarities between them. For that, we employed the force-directed placement (see Section 2.1.2) introduced in [35]. In the first design of the MLES, we do not focus on the precise positioning of the particular objects in the user views.

We start with the definitions of the operations *Zoom-In* and *Zoom-Out* providing the *vertical browsing* between the layers and allowing to narrow or broaden the user view that is actually explored in the collection.

Zoom-In

In the operation *Zoom-In*, first, the user selects one object from all objects available in the current user view that will be used as the query object. Then, using the query object, the $k\mathcal{NN}$ query is performed on the layer immediately *below* the layer of the current user view.

Definition 4.3. *Given the Multilayer Exploration Structure $E = MLES(\mathbb{S}, m, v, \phi)$, a query object $q \in \mathbb{L}_i$ and parameters k, i, the operation Zoom-In(E, q, k, i) on the layer $\mathbb{L}_i, \forall i = \{0, .., m-2\}$ returns a set of objects being the k-nearest neighbors to the query object q from objects in the layer $\mathbb{L}_{i+1}$.*

In Figure 4.2, the differences between two MLES structures with a different configuration are depicted. The arrow on the left shows the operation Zoom-In performed in the MLES that has only two layers in total, the zero layer and the leaf layer. In such a case, performing the Zoom-In operation is equivalent to the iterative querying mentioned in Section 2.1.2. On the contrary, in the 3-layer MLES (the arrows on the right in the figure), the Zoom-In operation from the same page zero view returns objects from the first (in this case labeled as middle) layer. In this case, the returned objects also represent similar objects to the query object, but in a lower level of details than in the leaf layer, hence the users can select from more diverse set of objects (unlike in the case in the 2-layer MLES). If the user performs another Zoom-In operation with the same query object from this middle layer, the same next user view as in the case of the 2-layer MLES scenario is returned.

Zoom-Out

The operation *Zoom-Out* is similar to the operation Zoom-In except that the $k\mathcal{NN}$ query is performed on the layer immediately *above* the layer of the current user view. As the layer L_i does not contain all objects from the layer L_{i+1}, the query object does not necessary have to be present in the next user view, because of the *the subset condition* in Definition 4.1 of the MLES.

Definition 4.4. *Given the Multilayer Exploration Structure, $E = MLES(\mathbb{S}, m, v, \phi)$, a query object $q \in \mathbb{L}_i$ and parameters k, i, the operation Zoom-Out(E, q, k, i) on the layer $\mathbb{L}_i, \forall i = \{1, .., m-1\}$ returns a set of objects being the k nearest neighbors to the query object q from objects in the layer $\mathbb{L}_{i-1}$.*

The motivation for the operation Zoom-Out is straightforward, *to lower the level of details* in the current user view by navigating vertically to the upper layers of the hierarchy. It is important to realize that the operation Zoom-Out is not just the reverse operation to the previously performed Zoom-In operation. The Zoom-Out can be performed on any actually displayed object and not just the ones residing also in the upper layer. The Zoom-Out also differs from the hypothetical operation of *returning back in history* of the previously performed exploration steps.

The explanatory scenario is depicted in Figures 4.3 and 4.4. In the first four steps of this scenario (Figure 4.3), the user explores through some images of sunsets, while suddenly he decides to explore a different part of the collection (but still similar to the actually displayed objects, e.g., with a sky), hence, he performs the Zoom-Out on the image that mostly differs from the previously explored images of the sunsets (Figure 4.4). The result in the next user view contains the objects that are less similar to the sunsets (but still with the sky), thus the user has more options to choose in what direction his exploration will continue.

Pan

The operation Pan provides the horizontal navigation through the layer, allowing users to locally browse the collection while keeping the actual granularity of the exploration at the same level.

Definition 4.5. *Given the Multilayer Exploration Structure, $E = MLES(\mathbb{S}, m, v, \phi)$, a query object $q \in \mathbb{L}_i$ and parameters k, i, the operation Pan(E, q, k, i) on the layer $\mathbb{L}_i, \forall i = \{0, .., m-1\}$ returns a set of objects being the k nearest neighbors to the query object q from objects in the layer $\mathbb{L}_i$.*

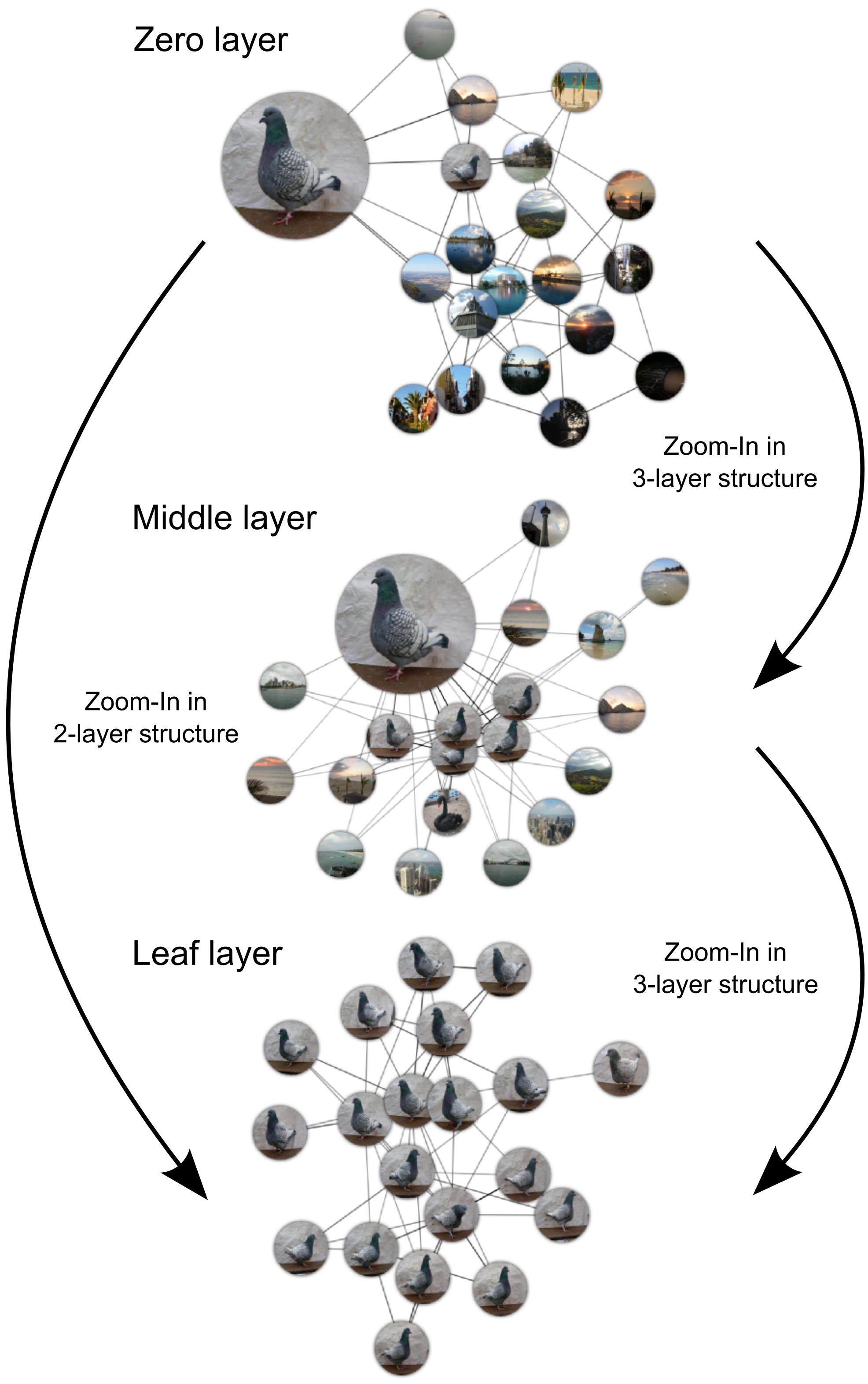


Figure 4.2: The operation Zoom-in performed in two MLES structures with the different number of layers (the enlarged image represents the query object for the following operation).

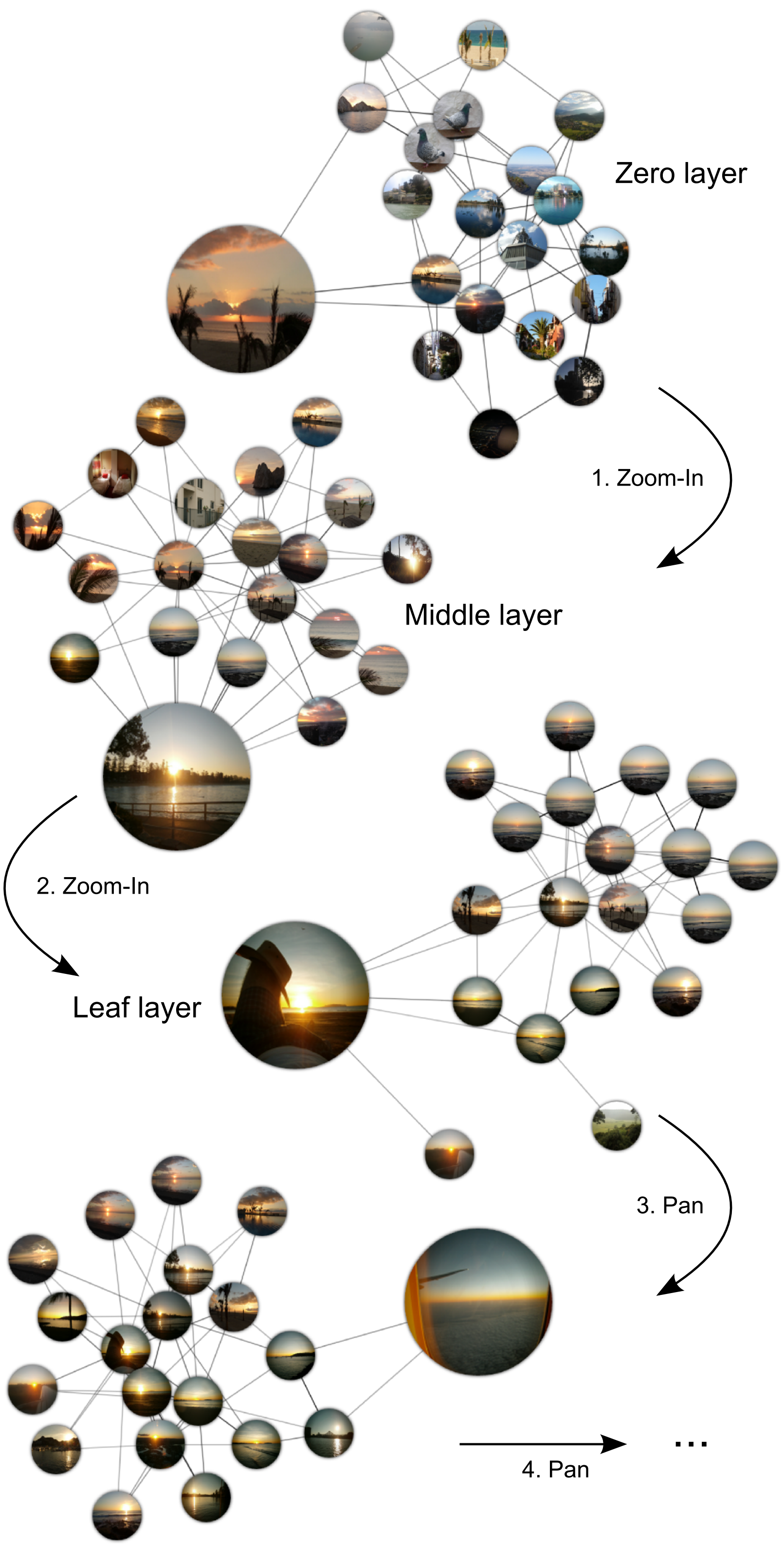


Figure 4.3: The first part of the exploration scenario in the 3-layer MLES involving all exploration operations for navigating horizontally and vertically through all layers. In this part, the user starts exploring the images of sunsets and he consecutively navigates to the leaf layer (the enlarged image represents the query object for the following operation).

This simple definition of the operation *Pan* follows the definitions of the operations Zoom-In and Zoom-Out, except that the $k\mathcal{NN}$ query for the selected query object is evaluated in the *same* layer. The operation Pan allows the user to investigate the neighborhood of some object in the current user view while the current level of details is not changed. In Figures 4.3 and 4.4, the operation Pan is used for two purposes. In the leaf layer (Figure 4.3), the operation is used to retrieve more similar images of a searched concept (e.g., sunsets) while in the middle layer (Figure 4.4), it is used as the method for navigation to other parts of the explored collection.

There is one problem in our definitions of the exploration operations that comes from the non-symmetry of $k\mathcal{NN}$ queries. Lets assume that k is fixed and an object B is in the result of the $k\mathcal{NN}$ query for an object A. Then, the object A does not necessarily has to be in the result of the $k\mathcal{NN}$ query for the object B. The consequence of this fact is that there could be some object which does not have to be reachable in the MLES if the exploration operations are defined with the definitions provided above. For completion of those definitions, we need to ensure that each object is reachable by some sequence of the exploration operations starting from the page zero view. Whereas the definitions of the zooming operations seem to be quite intuitive, we simply extend the definition of the panning operation by adding some random objects from the same layer to the result of the $k\mathcal{NN}$ query. Although, with such a step we add some level of non-determinism into the exploration process, we ensure that each object is with the non-zero probability reachable from the page-zero view. The proportion between the number of the objects that come from the result of the $k\mathcal{NN}$ query and the number of the randomly selected objects can be handled by some parameter of the operation Pan. More complex solution involving detection of outliers is out of scope of this paper, we suggest it for a future work. The randomly selected objects added into the result of the operation Pan solve also another unwanted behavior of the exploration driven by the $k\mathcal{NN}$ queries that is depicted in Figure 4.5. Such an unwanted situation includes the case when all objects in the current user view reside within the same homogeneous cluster and all distances between objects in this cluster are lower than the distance to any objects which are out of that cluster. Let us note that in this scenario the users can also use the operation Zoom-Out to successfully navigate the exploration process out of the cluster. Our further research on the reachability can be found in Appendix A.2.

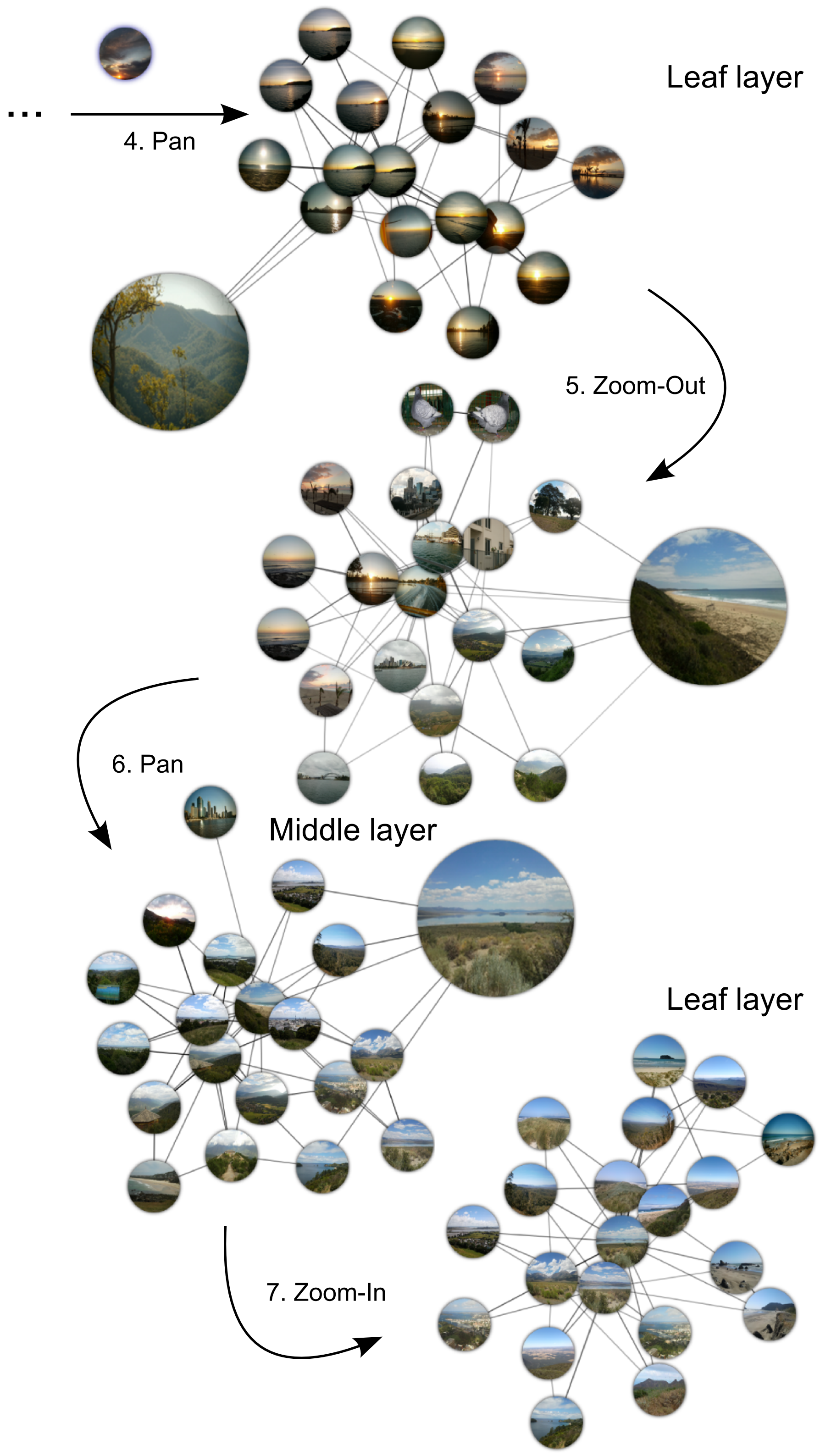


Figure 4.4: The second part of the exploration scenario in the 3-layer MLES involving all exploration operations for navigating horizontally and vertically through all layers. In this part, the user continues from the previous exploration of the sunset images and he consecutively navigates to the images of a nature in daylight (the enlarged image represents the query object for the following operation).

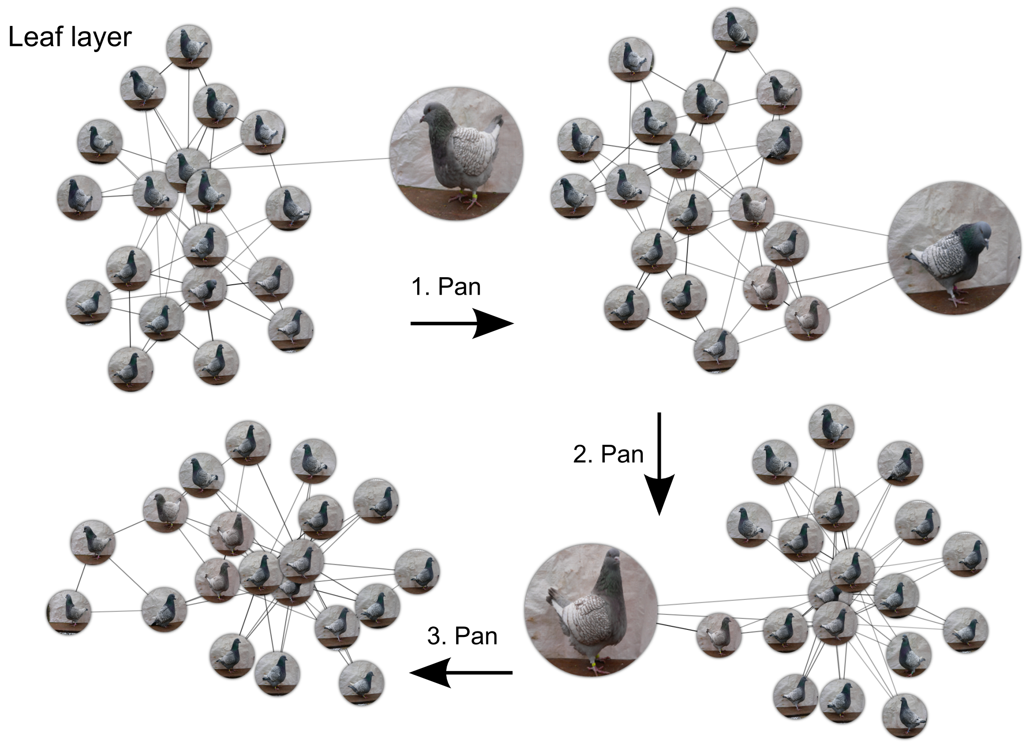


Figure 4.5: The example situation when the user gets stuck in the cluster of pair-wise similar objects (the enlarged image represents the query object for the following operation).

4.2 Exploration Layout

Following the introduction of the MLES, designed as the index structure for the data layer of the provided architecture from the previous chapter, in this section, we describe the component of the presentation layer – the exploration layout visualizing the MLES, together with its implementation in the demo application [104, 105].

We implemented the MLES in the previous version of our demo application used for the multimedia exploration [44]. The previous version of the demo focused on the efficiency of the exploration process, when it demonstrated the exploration supported with some MAMs. That work also discussed two different multimedia exploration techniques, see upper part of Figure 4.6, we already mentioned in Section 2.1.2. For refreshing your memory – in the *iterative querying* the exploration process starts in some initial view and consecutively uses a well-known query based approach as a basic modality. While in the *iterative browsing*, the user queries follow the hierarchy of some exploration structure. As was mentioned in the work [44], both of these techniques have their drawbacks, the iterative browsing depends on the compactness of the underlying structure and the iterative querying depends on the first selected collection for the zero view. On the other side, both ideas have also their advantages. In case, when the hierarchical structure that supports the iterative browsing truly follows the hierarchy of an indexed space, the subsequent steps in the browsing process are supposed to reveal details in more continuous way. But, the problem is in the claim "truly following the hierarchy of an indexed space", since to achieve such an illusion is usually very difficult, at least in an arbitrary space, because creation of such a hierarchical structure can be very expensive. Conversely, the index support for the iterative querying seems to lead to very efficient evaluation of underlying similarity queries. Following these ideas we propose a combination of both these techniques and designate it as the *hierarchical querying.*

Hierarchical Querying

The idea of the *hierarchical querying* takes the concept of the hierarchy from the *iterative browsing*, and the concept of the querying supported with the similarity index and which does not directly follow any structure from the *iterative querying.* At the beginning of the exploration process the user is provided with objects from the *page zero* (the layer L_0 in the MLES), using the very same approach as in the iterative querying. But, when it comes to next querying from this page zero, instead of returning objects from the whole database, only objects from a subset are returned. This subset, we denoted as the *middle page* (here, the layer L_1 in the 3-layer MLES, see it in the bottom of Figure 4.6), is the collection of preselected objects and is created during the phase of constructing the index for the whole exploration environment. The middle page consists of all objects that are in the page zero supplemented with the additional objects selected from the rest of the database. We used randomly selected objects for both, the page zero and the middle page (as mentioned in Section 4.1), but more complex selection methods known from the area of hierarchical clustering or pivot selection can be used. It should be known that the number of objects selected for the middle page also influences the illusion of the hierarchical querying. We derived the object

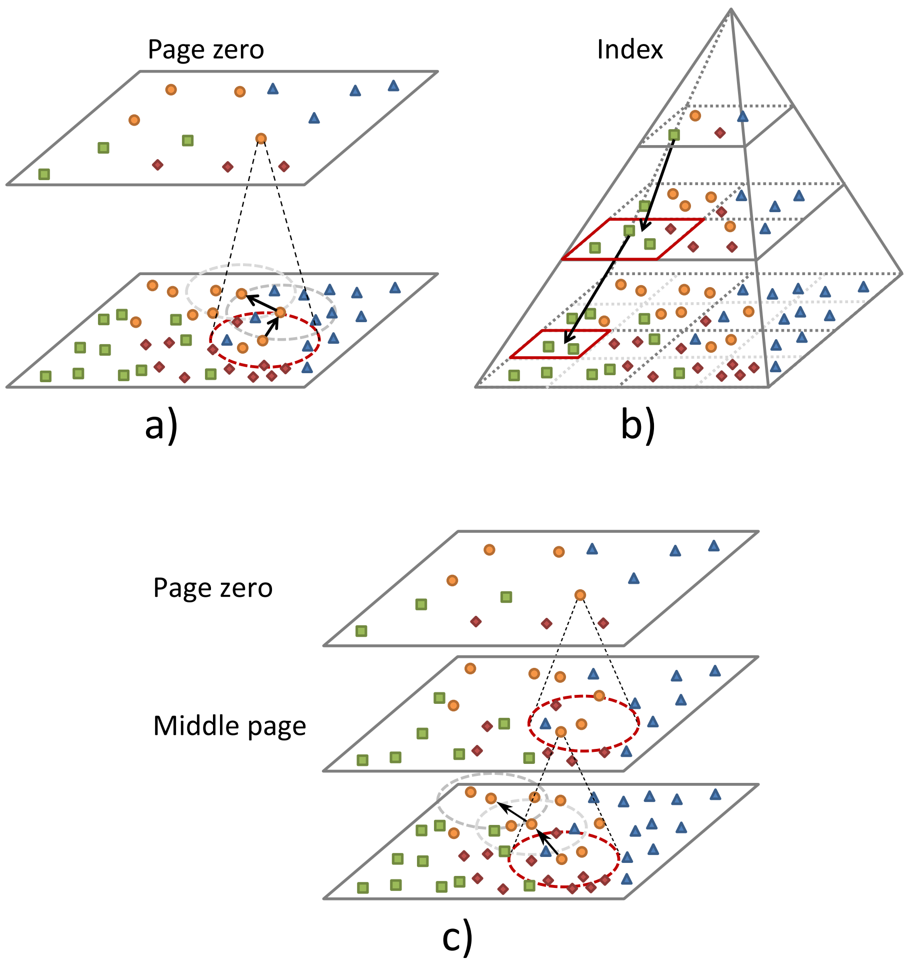


Figure 4.6: a) iterative querying b) iterative browsing c) hierarchical querying

count from the number of objects in the page zero ($objects_{PZ}$) and the variable parameter *pow*, which determines the granularity of details in the middle page, see Equation 4.1, where $|S|$ is the size of the whole collection.

$$objects_{MiddlePage} = \left\lceil objects_{PZ} + (|S| - objects_{PZ}) * \left(\frac{1}{2}\right)^{pow} \right\rceil \tag{4.1}$$

The parameter *pow* directly controls the number of objects in the middle page, in Table 4.1 you can see the counts for few values of *pow* in case when the size of the whole collection is $|S| = 20,000$ and the number of objects in the page zero is 20. We added this possibility to parametrize the number of objects in the middle page because different multimedia collections have internally the different level of granularity when it comes to the classification of objects. For example, if the most of the objects are from the same class, the number of objects in the middle page should be higher than in the case when the objects are distributed in many small classes. Since, it is more desirable to get higher perspective over the collection than just the representatives of the single class.

pow	1	2	3	4	5	6
$objects_{MP}$	10,100	5,015	2,518	1,269	645	333

Table 4.1: The influence of the parameter *pow* on the object counts in the *middle page*

4.2.1 More Continuous Way

Within our demo application, we introduce also the implementation of preserving the user context in the visualization between the individual exploration steps. In other words, the state of the visualization (i.e. the visualized objects and their positioning on the screen) "*before the query is evaluated*" should adhere the consecutive visualization state "*after the query evaluation*" more continuously. The one of requirements for more continuous transition is the binding of the new result set to the previous one. As you can see in Figure 4.7 the process of one exploration step starts with the selection of the query object (by the user) in step I. After the query is evaluated, the back-end of the exploration system sends the new result set to the visualization component, while the old result set still remains on a screen. Both results sets, the new and the old, are compared to each other and the objects from their intersection are marked, they will be preserved in the new visualized state (step II). After the rest of objects from the old result set are removed from the visualization (step III), the rest of the new objects, which are not already visualized, are placed next to the old preserved objects. For the placing method of the new objects, we chose the strategy to place each new object in the position next to the most similar object that is already visualized, as depicted in step IV. There is also possibility to take into consideration more than a single one most similar object and place the new object depending on distances to more similar objects. When all objects from the new result set are placed in the visualization, the process of adjusting the proper position according to the particle physics model (i.e., force-directed placement) is iteratively run (step V) until the position of each object is stabilized (step VI).

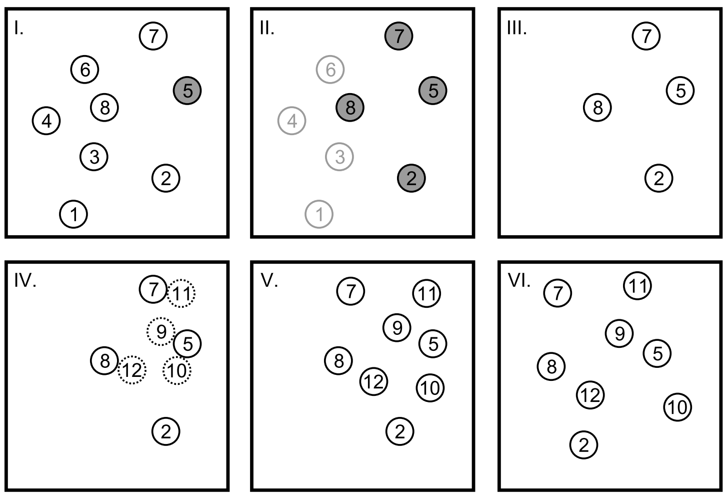


Figure 4.7: The schema of preserving the user context during the single exploration step.

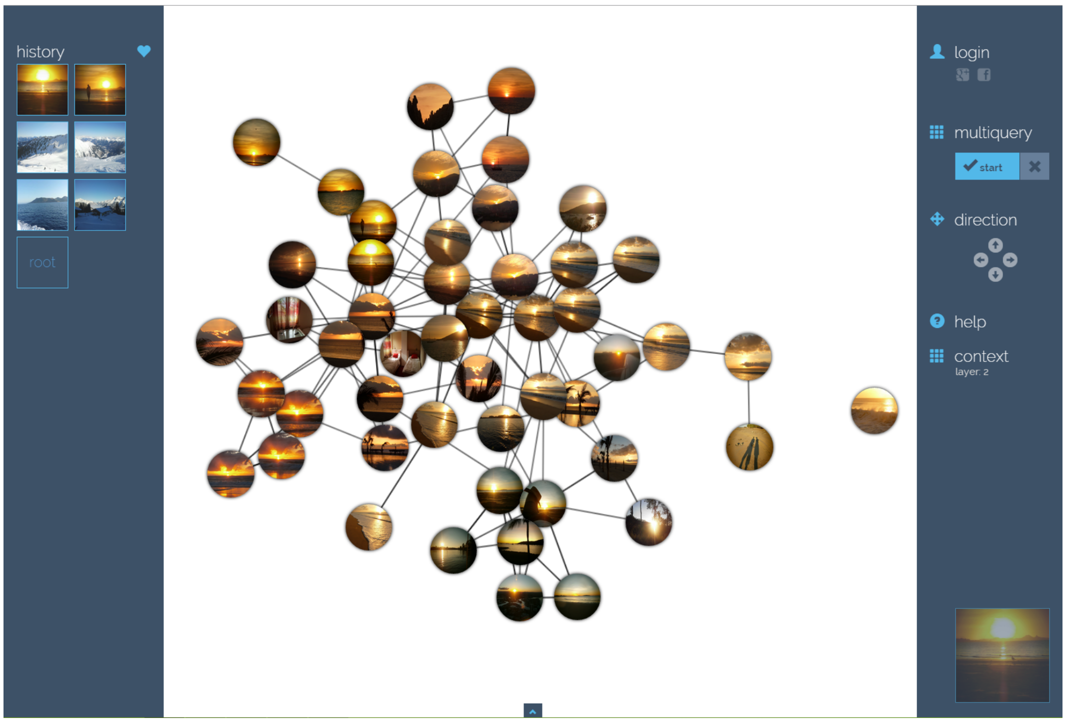


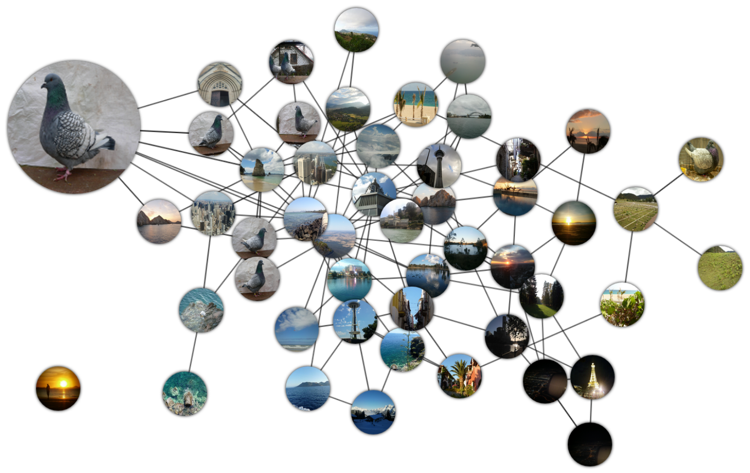
Figure 4.8: The layout of the whole demo application. You can see the history panel on the left side, some information about the current context on the right side and the current exploration view on the main canvas.

4.2.2 Demo Application

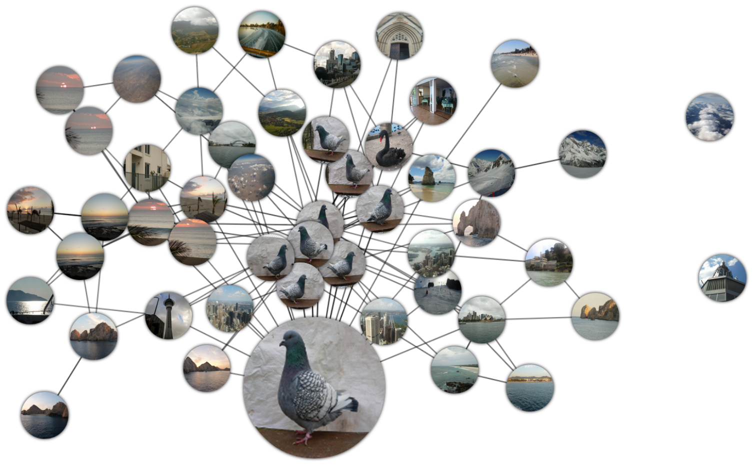
Our demo application is the online software for the content-based exploration of image collections and it is accessible from a web browser [105]. More information about the architecture of the demo application can be found in Appendix A.1.

The layout of the client demo application is shown in Figure 4.8. The user can explore the collection by clicking on some images of his interest. For returning back in the querying history, he can use the history panel in the left part and then continue by exploring a different part of the visualized collection. In the right part, the user can see some information about the current context.

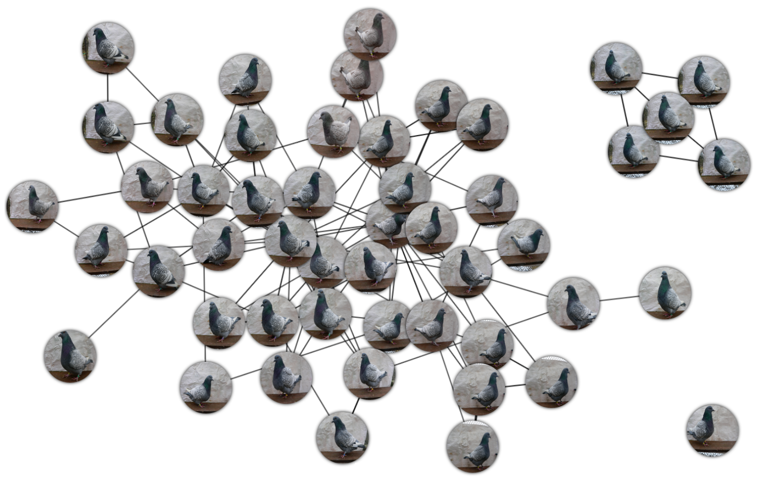
Figure 4.9 shows the exploration process of the hierarchical querying in particular exploration steps, starting from the page zero (a), continuing with the middle page containing only some sample images (b) and ending in the bottom of the hierarchy where all objects from the dataset are stored in (c). For a comparison, the process of performing the same exploration query in the concept of the iterative querying is depicted in Figure 4.10. As you can see, the absence of the middle page in the iterative querying results in the undesired effect, only the most similar objects from the whole dataset are retrieved, and such a very detailed view does not give the user the option to see the results of the query from a bigger perspective.



(a)

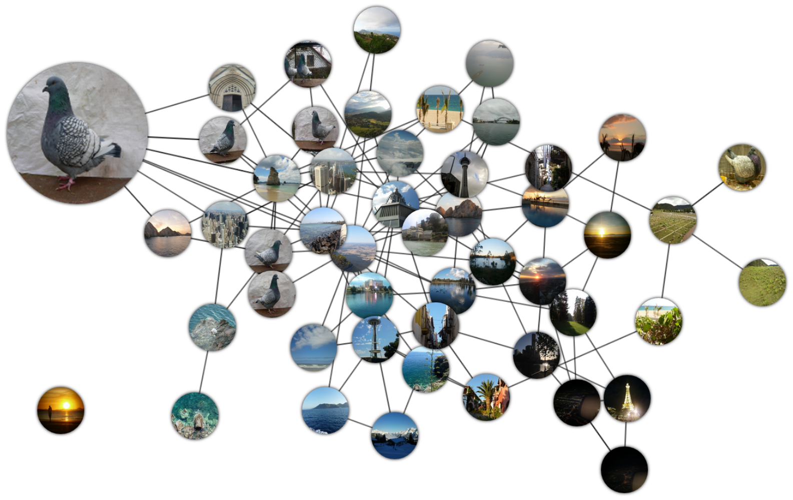


(b)

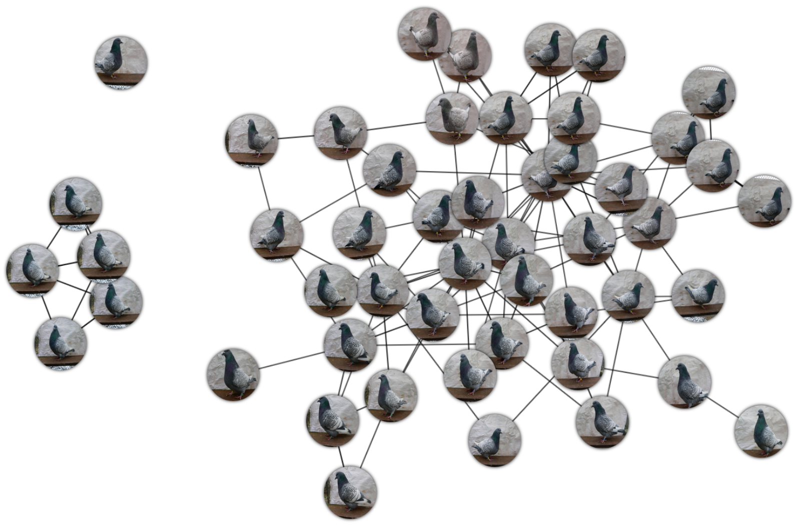


(c)

Figure 4.9: The exploration in the *hierarchical querying*. (a) The initial view of the page zero. (b) The view of the middle page after clicking on the pigeon image. (c) The view from the bottom of the hierarchy after clicking on the pigeon image again in the middle page.



(a)



(b)

Figure 4.10: The process of the *iterative querying*. (a) The initial view. (b) The view after performing the $k\mathcal{NN}$ query with the pigeon image as a query object.

4.3 Index support for the MLES

The crucial part of the MLES structure is its index-presuming evaluation of underlying similarity queries, and thus also the evaluation of the exploration operations. Each layer can be supported with a separate index structure, which can improve the efficiency of the evaluation, and no matter that it is done in a metric, in a non-metric or in an approximate way. In the next sections, we introduce two techniques designed for improving the efficiency of the evaluation, which we proposed in recent years.

4.3.1 Clustered Pivot Table

Our first work oriented on the efficient metric indexing proposes the persistent variant of a pivot table [74] (see Section 2.2.1), the *clustered pivot table* [106] (see Figure 4.11), which focuses on the minimizing of I/O costs when accessing small data blocks (a few kilobytes). The clustered pivot table employs: (a) the preprocessing method utilizing the M-tree [52] (see Section 2.2.1) in the role of a clustering technique, and also (b) the original heuristic for I/O-optimized processing of $k\mathcal{NN}$ queries.

The idea of utilizing the clustering in the clustered pivot table is based on the observation that the objects which it is necessary to process in the filtering phase of the query algorithm can be spread over many database disk pages, while the pre-clustering of these disk pages will lead to the accessing of only the necessary ones. Hence, the clustered pivot table consists of the distance matrix connected to the M-tree, where the vectors in the distance matrix are grouped according to the corresponding M-tree leaf nodes. The main idea behind the efficient query processing in the clustered pivot tables is to minimize the volume of data retrieved from the disk. As the clustered pivot table groups similar objects within the M-tree leaves, many of the data pages could be filtered out because the clusters they represent do not overlap with the query region (see the lower-bounding principle in Section 1.3).

Whereas, the range query processing is nearly the same as for the original pivot table, for the $k\mathcal{NN}$ query we have to consider a completely different heuristic if we want to minimize also the I/O costs. The $k\mathcal{NN}$ algorithm of the classic pivot table is optimal in terms of the minimal distance computations spent, but to prevent multiple accesses to the disk pages, we cannot use the reordering of objects according to their lower-bound distance to the query object (as the original LAESA (pivot table) [75] algorithm does). Moreover, for achieving the quicker decrease of a dynamic $k\mathcal{NN}$ query radius, we keep data in the first few disk pages unsorted and non-clustered, while the rest of the dataset is reordered by the M-tree. Hence, when the $k\mathcal{NN}$ query tries to determine the dynamic query radius, it first processes the objects from the non-clustered part (randomly distributed through the whole space) so the radius is quickly decreased in first cycles of the algorithm.

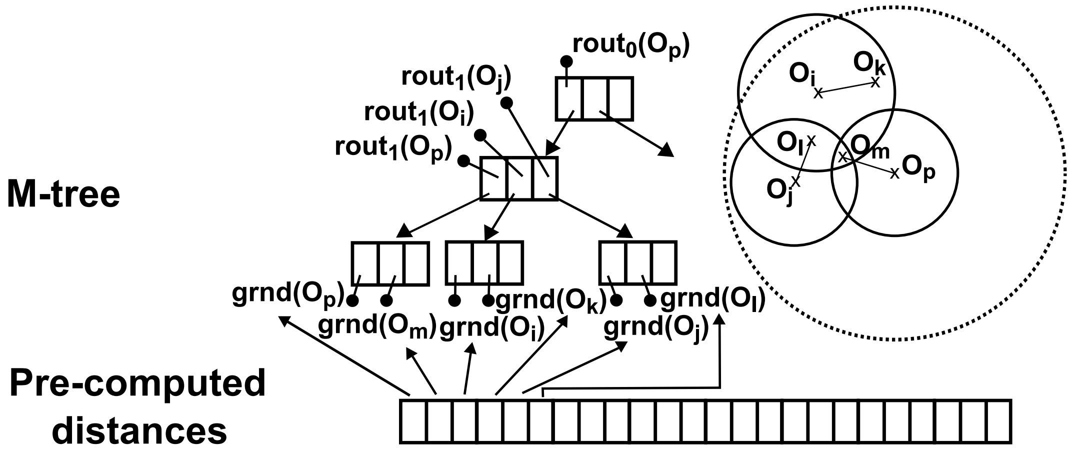


Figure 4.11: The clustered pivot table with data stored within the M-tree leaves.

4.3.2 Cut-Regions

In our second research of improving the efficiency of similarity search techniques [107], we present the concept of *cut-regions*, which could heavily improve the performance of metric indexes that were originally designed to employ simple ball-regions. The shape of the cut-regions is far more compact than that of the ball-regions, while they still preserve the simple and concise representation.

The popular simple ball-regions, defined only by the center object and the covering radius, have one main drawback – they cannot tightly cover the cluster of similar objects in a sparse metric space. However, if the static set of k global pivot objects is employed, the original ball-region can be further cut off by the rings of each pivot (where the ring is an annulus centered in the pivot), forming thus the cut-region. In particular, the ring (or the hyperring respectively, in case of the multidimensional space) for a given pivot is determined by the distances from the pivot to the closest and the farthest objects in the ball-region. The example of the difference between the cut-region and the ball-region is depicted in Figure 4.12 (where *hr* states for a hyperring).

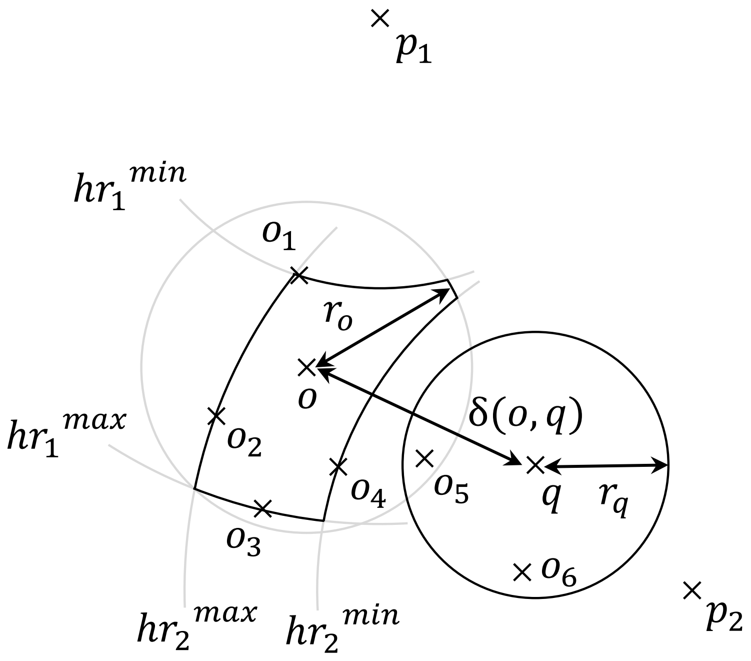


Figure 4.12: The overlap of the cut-region and the ball-region. The ball of the cut-region on the right overlaps the ball-region on the left, but the the cut-region itself does not overlap it because of its boundaries determined by the hr_2 rings of the pivot p_2.

The idea of the cut-regions was first used in the PM-tree [78, 79] (see Section 2.2.1), though there it was not described as the standalone formalism but as the part of the PM-tree structure itself. In addition to the formalism, we present two other redesigned metric indexes the M-index [85, 86] (see Section 2.2.1) and the List of clusters [108] utilizing the cut-regions instead of the ball-regions (see Figure 4.13).

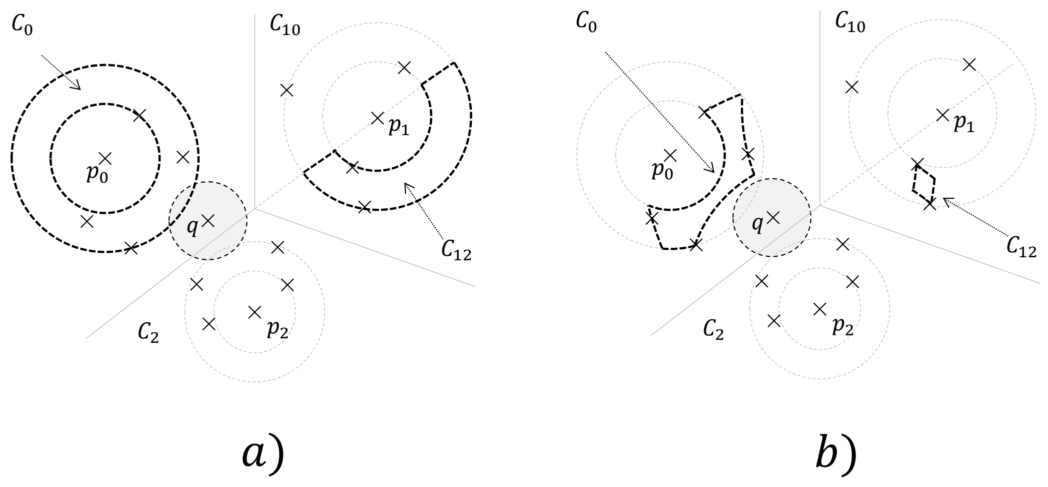


Figure 4.13: The M-Index cluster tree without (a) and with the cut-regions (b).

4.4 Discussion and Future Work

The provided proposal of the MLES can be certainly improved in many directions, in next sections we provide some possibilities for the further improvement of the MLES structure, of its presentation layout and also of its operations.

4.4.1 Employing Clustering

The effectiveness of clustering can be understood in the degree how the partitioning to clusters reflects the distribution of objects in a description space. It is known that if the objects are evenly distributed in the description space, it is hard to partition them into the clusters, regardless how good in the clustering the particular algorithm is. But, in cases when the clustering works, various clustering methods have positive effect on the hierarchic visualization.

It is necessary to realize that the MLES is not primarily visualizing a hierarchic structure, its operations are purely based on the nearest neighbors search. Nevertheless, employing the clustering in the MLES can lead to the better selection of objects for the first layer(s). With replacing cheap random selection, the first layer(s) of the explored collection can become more representative and that can solve the problem of reaching the small distant clusters.

If we would like to to truly visualize a hierarchic structure, the natural option we will have is the agglomerative hierarchical clustering [109]. But, with the growing size of the clustered collection, the hierarchical clustering becomes computationally very expensive. Therefore, in the future, we want to employ more efficient solution of the hierarchical clustering that is suggested in the work by Kull et al. [110] with support of the Epsilon Grid Ordering [111] or the fast-sparse clustering proposed by Chen et al. [56].

4.4.2 Improving User Feeling

As we already mentioned in Section 2.1.3, the growth of mobile devices leads to the new modern user interfaces, especially those allowing touch gestures. In the section mentioned above, we reviewed some exploration systems that already employs modern technologies like the touch screen and the accelerometer in their user interfaces. The further research in this area can lead to the definitions of additional MLES exploration operations (or to the improving of the previous ones) by employing the tilting (a device), the swiping, the pinching, the spreading (stretching), the sliding, the dragging, the flicking, or the other special multi-touch techniques that employ even many fingers [112].

4.4.3 Improving Operations

Besides the improving of the exploration operations from the user's perspective, we can also improve the underlying transformation of the operations into the similarity queries. The $k\mathcal{NN}$ queries generated from the exploration operations are used in the MLES in their standard form, but there exist many variants of the k-nearest neighbor search, which can be utilized in the MLES operations. For example, the *Weighted-$k\mathcal{NN}$ search* considers the potential candidates for the nearest neighbor according to their weights that reflect the distance from the query object. Other variants can be, for example, the *Condensed-$k\mathcal{NN}$* (which omits repetitive and redundant data), the *Reduced-$k\mathcal{NN}$* (omits data that are ineffective in results), the *Clustered-$k\mathcal{NN}$* (clusters data and omits very distant data), the *Continues-$k\mathcal{NN}$* (returns all results for the query objects that are moving continuously along a straight line), or the *Group-$k\mathcal{NN}$* (returns the nearest neighbors for a group of the query objects) search. These techniques are by far not exclusive, beside them, many other $k\mathcal{NN}$ techniques were proposed and some of them are reviewed in the surveys on the $k\mathcal{NN}$ search [113, 114].

4.4.4 User Study

In order to demonstrate the benefits of more layers in the multimedia exploration, many user studies focusing on the various retrieval scenarios have to be performed. Hence, in the following chapter, we provide the evaluation of the MLES behavior in the multimedia exploration performed by real users.

Chapter 5

User Study on the MLES

In this chapter, we present the results of our user study [115] focused on the effectiveness of the multilayer exploration using the MLES. We investigated the behavior of the 3-layer MLES and compared it to the simple 2-layer MLES, which we used for simulating the standard flat k-NN browsing.

In order to fully compare different exploration approaches, various exploration tasks have to be utilized to challenge the performance of the examined structures. In our study, we focused on the *known-item search tasks*, where users receive the simple textual description of the *searched class* (a category of the annotated test collection, e.g., pyramid, bottle, pineapple, pumpkin, handshake, etc.). Then, starting from the *initial page zero view*, the users have to find as many objects of the searched class as possible using just a limited number of exploration operations (in our study we used 15 operations). Let us note that the initial page zero view was the same for all search tasks and it did not contain any object of the searched class. For each of the search tasks, we measured the *number of* necessary *clicks for finding the first object* of the searched class together with the *cumulative number of found objects* for each *exploration step* (i.e., one performed exploration operation). Although that it was not the objective of the search tasks, we also measured the *cumulative number of visited classes* for each exploration step.

We performed this extensive user study on 94 participants from different countries (47 men and 47 women), who altogether completed 1661 search tasks. The participants are students of the University of Finance and Administration attending different study programs (IT, economics) and they were from different student groups. All students of the same group received the instructions from the same single person. In order to diversify the groups even more, some groups were motivated to find as many images of the searched class as possible, while the others received few more keywords, which describe the searched class. In each group, the tasks were distributed uniformly for both compared structures. In the following, we describe the *Find the image* application that was used for testing, next we propose the test settings, following with the results of the executed tests and in the end we discuss the overall results of the performed user study.

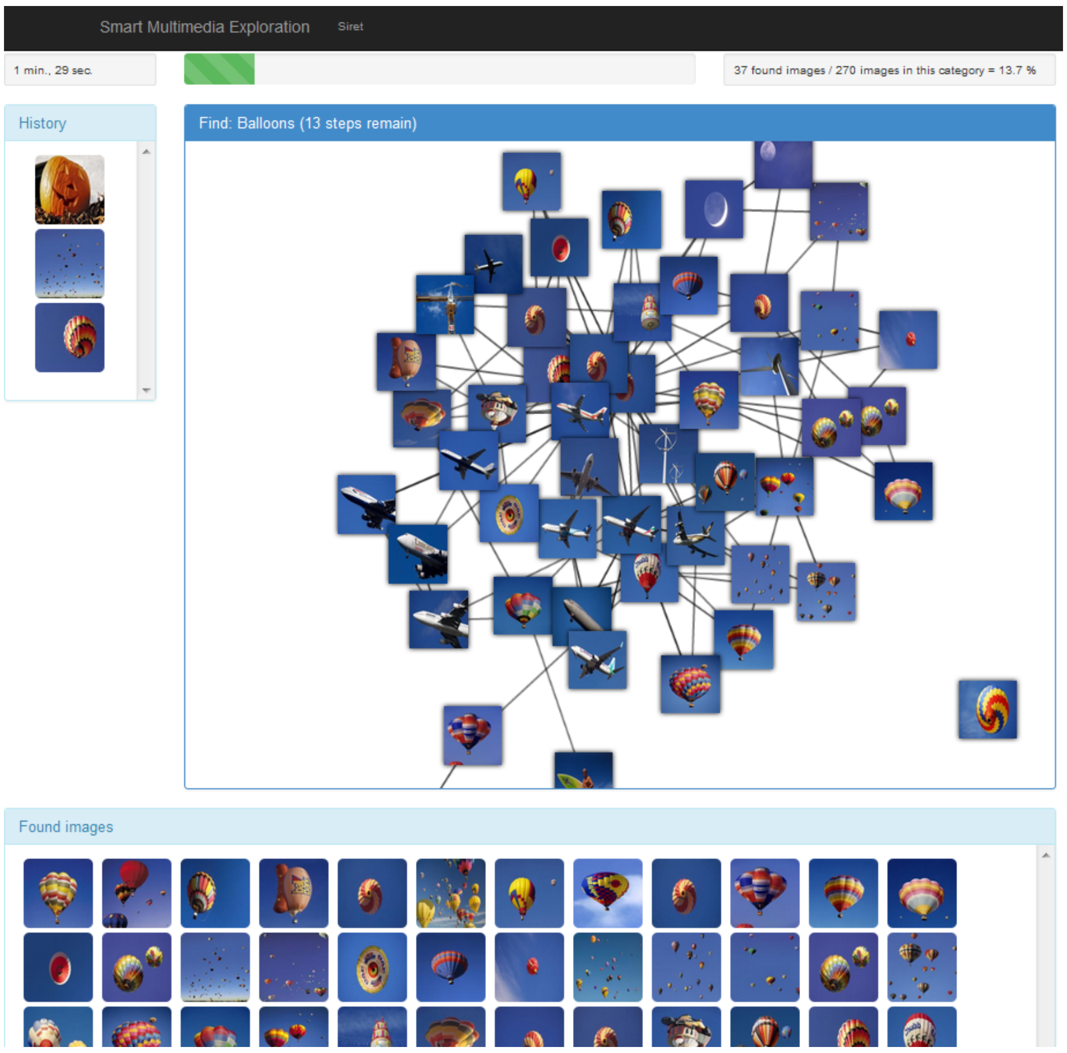

Smart Multimedia Exploration
Siret
1 min., 29 sec.
37 found images / 270 images in this category = 13.7 %
History
Find: Balloons (13 steps remain)
Found images

Figure 5.1: The layout of the Find the image application.

5.1 Find the Image Application

The user study was performed using the *Find the image* web application [44, 116]. It is an open platform for performing the user studies on the image exploration, and it can be used also as a web service [117].

The layout of the application is depicted in Figure 5.1, the users are presented with the current search task and their progress on the top of the screen. The application shows and monitors the wall-clock time, the number of remaining exploration steps and the number and the percentage of found objects fulfilling the current search task. The main part of the screen is dedicated to the visualization of the results of exploration operations. The results are visualized using the force-directed placement which uses the objects as the nodes and the similarities between the individual objects as the weights for the edges between the nodes. Depending on the current level of the MLES, each node provides the interactive Zoom-In, Zoom-Out and Pan operations (see definitions in Section 4.1.1). The Zoom-Out can be performed by double-clicking on the individual depicted images with the right mouse button, while the buttons for the Zoom-In and/or the Pan operation are offered when hovering with the mouse cursor over the image node.

The bottom part of the screen shows the images found so far, in the grid of thumbnails. When the user spends all 15 allowed exploration operations, the application saves the exploration statistics and offers to the user another search task.

5.2 Test Settings

5.2.1 Dataset

As a dataset, the PROFIMEDIA test collection [118] comprising 21993 small thumbnail images divided into 100 classes was employed. The position-color-texture feature signatures [119] and the Signature Quadratic Form Distance (SQFD) [14] were used as a similarity model.

5.2.2 MLES Configuration

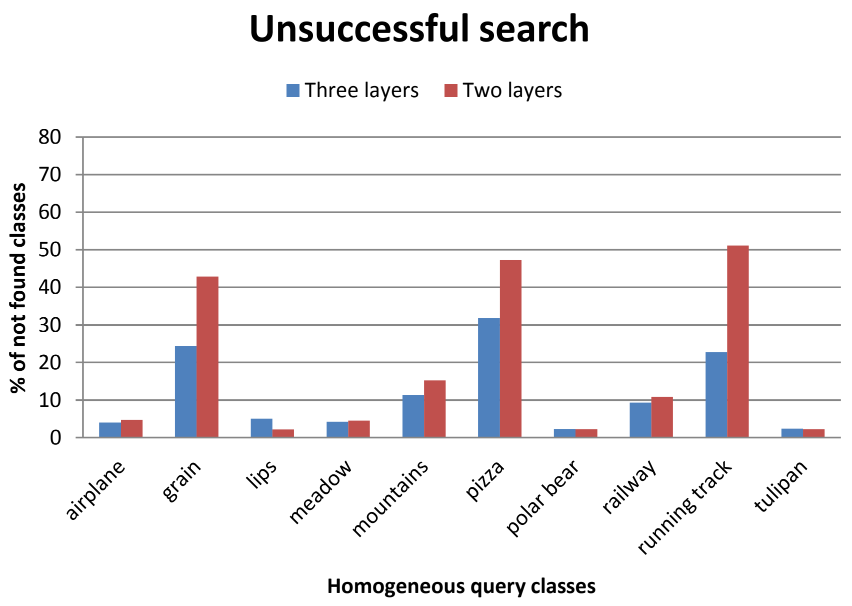
As we mentioned in the introduction of this chapter, we compared 2-layer and 3-layer variants of the MLES. Both variants of the multilayer structures shared the same set of images for the initial zero page view visualizing 50 images, while the last layer comprises the whole dataset. The 3-layer structure used in addition the layer in the middle consisting of one eighth of the objects from the third layer. The PM-Tree index [79] was used to support the evaluation of the $k\mathcal{NN}$ queries (see Section 4.3).

5.2.3 Test Configuration

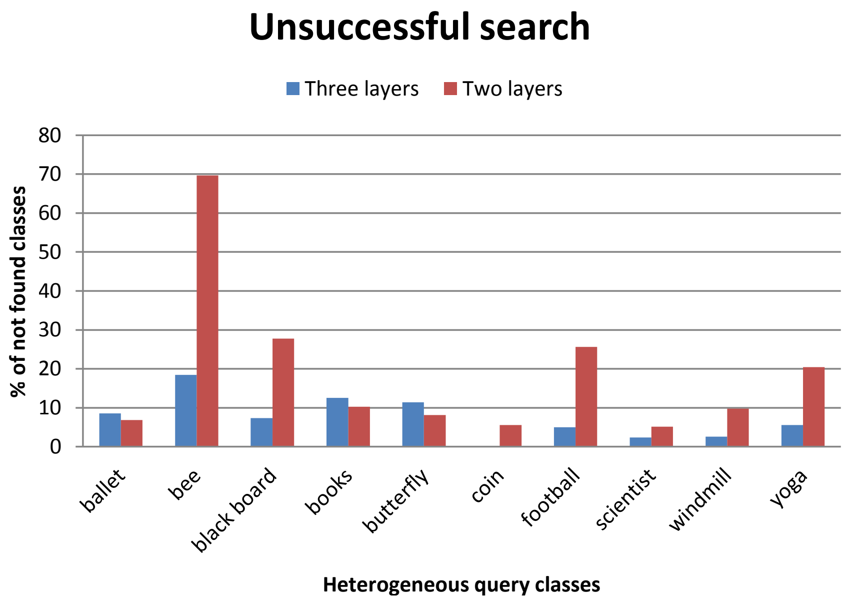
The searched classes for the search tasks were selected from those classes whose images were not present in the initial page zero view. More specifically, 10 homogeneous (images of the same class that are visually similar) and 10 heterogeneous (images of the same class that can be less visually similar) searched classes were manually selected. Each participant received the user ID, where for each user ID there was generated the sequence of 20 search tasks. The sequence always consisted of 10 permuted homogeneous searched classes followed by 10 permuted heterogeneous searched classes. Using such sequences, the participants always started with the easier search tasks and then continued with the more complex ones. The participant could access the next search task only if he finished the actual search task. Before starting the sequence of the tasks, each participant could also perform one test task for each MLES structure with such a simple searched class whose some of images appeared in the initial zero page view. The results of these test tasks were not included in the following overall results of this user study.

5.3 Results

In the first part of this section, we focus on the number of exploration operations that were needed for finding the first objects of the searched class (the initial page zero view did not contain the objects from the searched classes). In some of the search tasks, the users were not able to find any object of the searched class, as depicted in Figure 5.2. We may observe that for the particular searched classes – grain, pizza, running track and bee – it was difficult for the users to find the objects of the searched class when they were given just a short textual description. Finding objects of these four classes was difficult for both compared MLES structures, however, the 2-layer MLES resulted in more unsuccessful searches (in the case of the bee class, only 30% of searches using the 2-layer MLES were successful). Except five query classes, the 3-layer MLES outperformed the 2-layer MLES in this experiment. Overall, there was 10% of unsuccessful searches for the 3-layer MLES and 18% of unsuccessful searches for the 2-layer MLES. Men participants were slightly more successful than women participants for both indexes (this information is not depicted in figures).

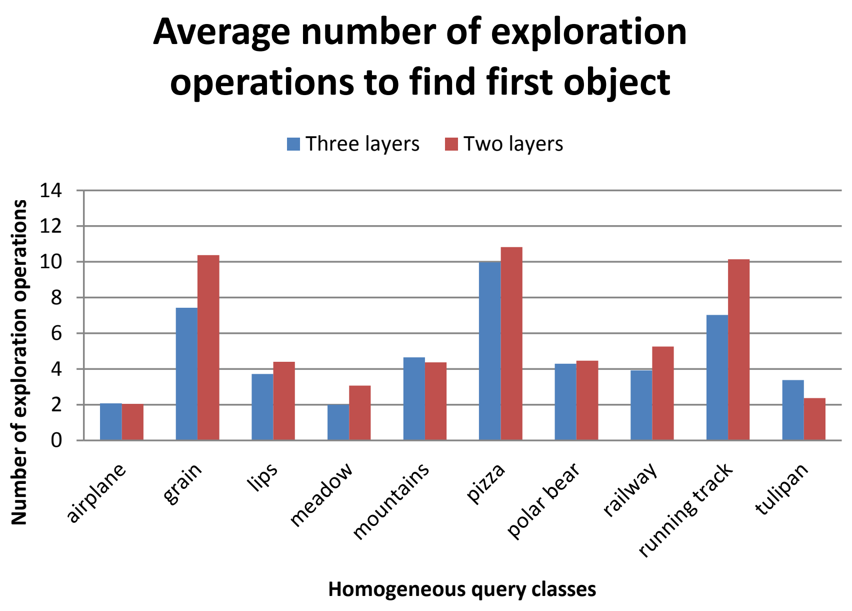


(a)

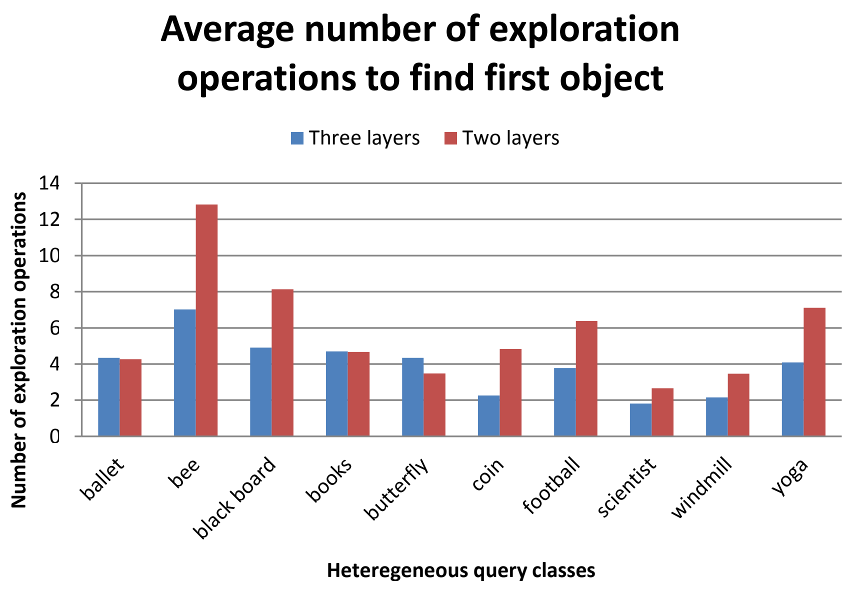


(b)

Figure 5.2: The unsuccessful search.



(a)



(b)

Figure 5.3: The average number of exploration operations needed for finding the first object of the searched class.

In Figure 5.3,there is depicted the average number of exploration operations needed for finding the first object of the searched class. Since the number of such spent exploration operations is unknown for the case of unsuccessful searches, we used the number 16 as the minimal number of operations required for finding the first object in that case. We may observe that in most cases five exploration operations were sufficient. Similarly, like in the previous figures, except few cases the number of exploration operations needed for finding the first object was lower for the 3-layer MLES.

Since the users always started the exploration from the same initial page zero view, the learning effect could take part in the effectiveness of the exploration significantly. In Figure 5.4 is outlined the average number of exploration operations needed for finding the first object of the searched class for each search task. Although the numbers are slightly lower for the search tasks that were performed later, there is not any significant evidence of the learning effect. This is probably caused by the fact that the next searched class was presented only after the previous search task was finished, and also by the fact that the descriptions of the classes were textual. Therefore, any visual information that could be remembered by the user, was not often connected to the actual search task. On the other side, in some cases, the participants reported that the previous experience with the visualized collection helped them to find some searched classes faster.

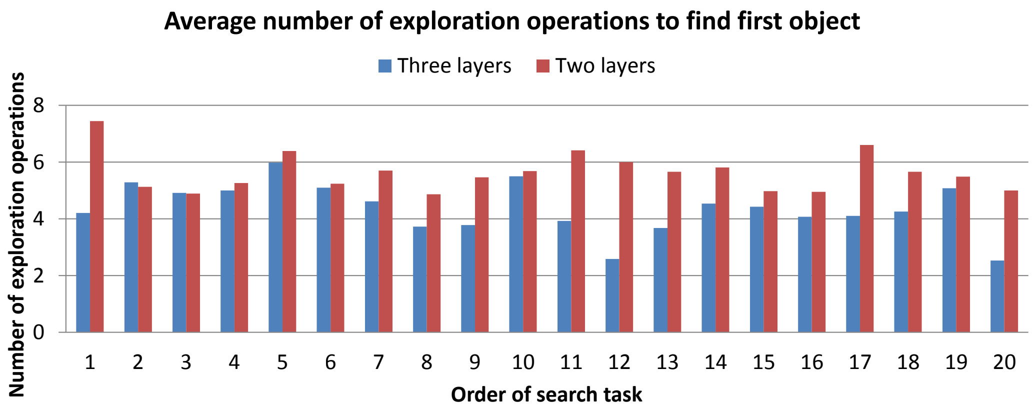


Figure 5.4: The average number of exploration operations needed for finding the first object of the searched class for each search task.

In the second part of this section, we present the graphs which depict how many objects of the searched classes were found using 15 exploration operations. In Figure 5.5, we may observe that searching for objects of the homogeneous searched classes (1. - 10. search task) resulted in higher percentage of found objects than the case of the heterogeneous searched classes (even though the heterogeneous searched classes could benefit from the learning effect). This is probably caused by the fact that with the finding of the first object of a homogeneous searched class the participant accesses a large cluster of visually similar objects, which can be simply explored by the $k\mathcal{NN}$ queries. On the other side, a heterogeneous searched class consists of more visually dissimilar clusters and thus the participant often gets stuck in just one of them during all 15 exploration steps.

Figure 5.6 depicts the number of found objects for individual classes in more detail. Except for the classes – grain, railway and running track, the participants were able to find at least 20% of objects from the homogeneous classes. The reason of these three exceptions is probably the problematic formulation of the mental query (i.e. the imagination of the representative object, see the introduction of Chapter 2) for these classes. On the contrary, the results of the heterogeneous classes show four exceptions that result over 20% of found objects, all others result in lower numbers. The reason of the low numbers we mentioned in the previous paragraph, while the exception classes – coin, football, scientist, windmill, are probably easy to imagine and identify, even in their heterogeneous form.

In Figure 5.7a, there is depicted the average percentage of *found objects* from the searched classes for each exploration step. We may observe that from the second exploration step the 3-layer MLES starts to outperform the 2-layer MLES.

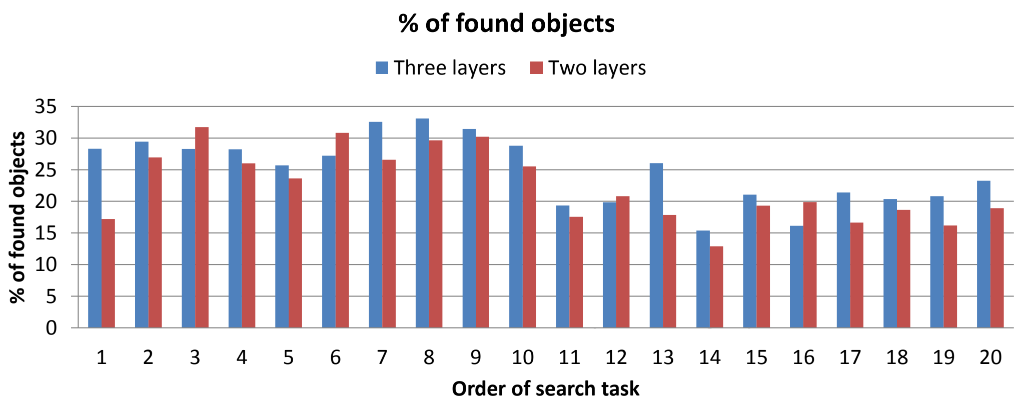
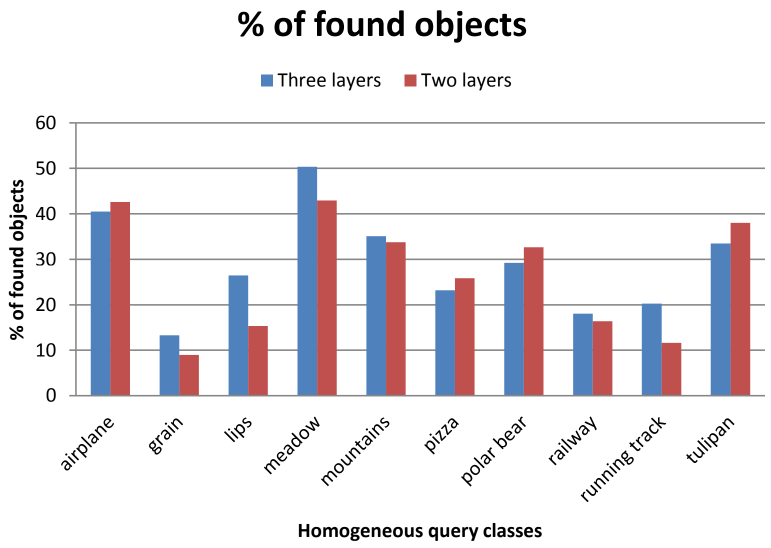
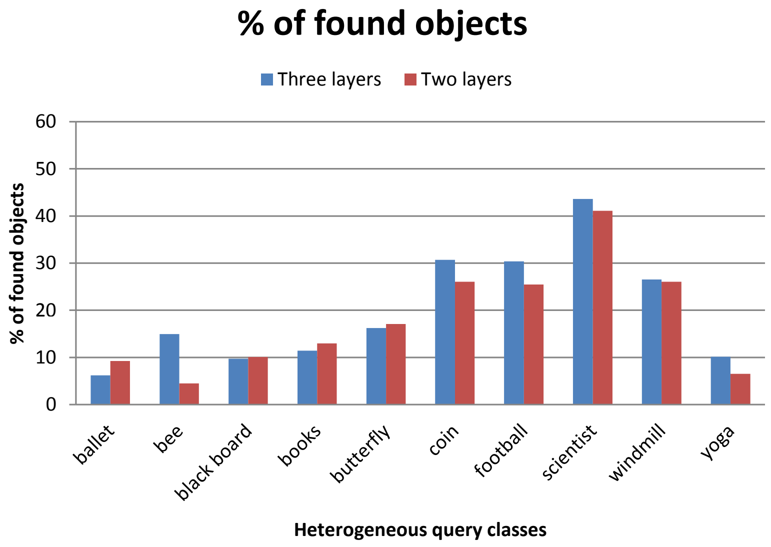


Figure 5.5: The found objects and classes during the exploration – the learning effect has minimal influence on the number of found objects.



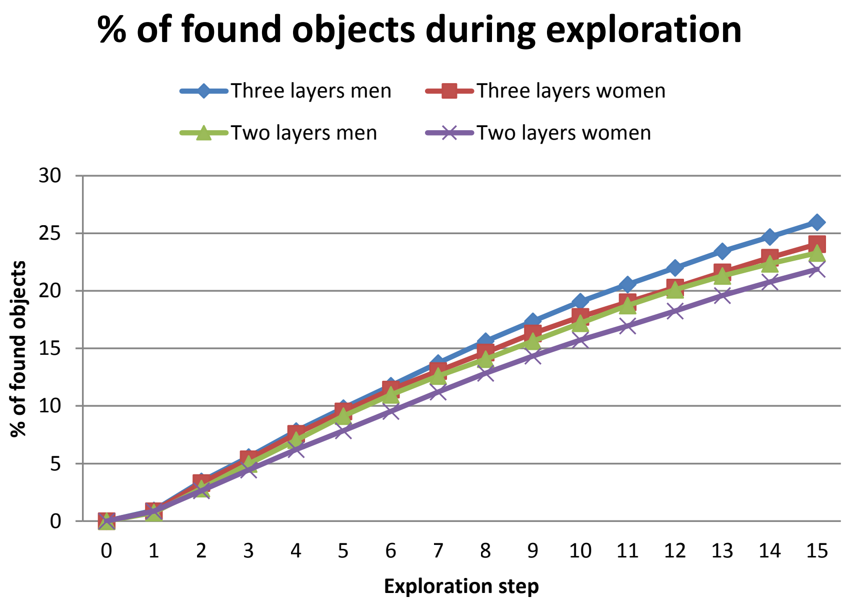
(a)



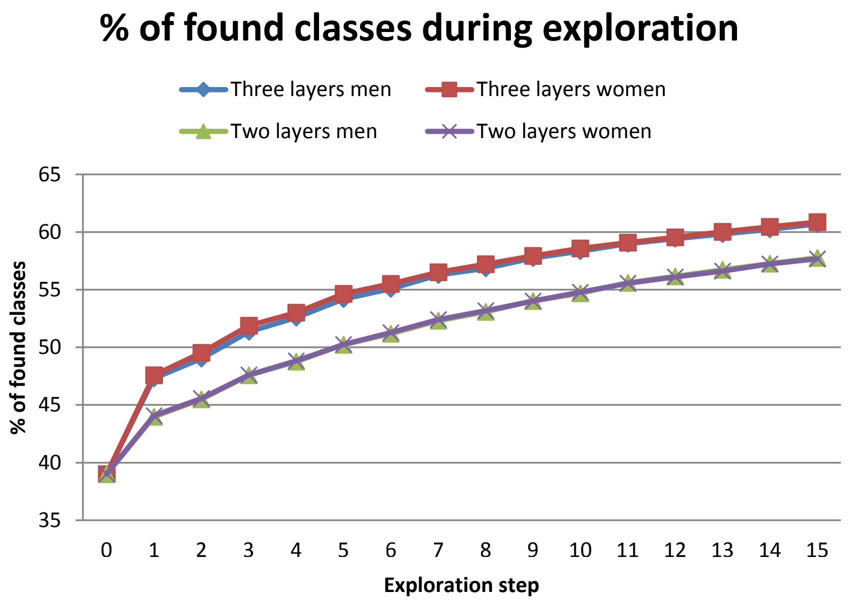
(b)

Figure 5.6: The found objects and classes during the exploration.

We may also observe that in later steps the men participants were able to find more objects of the searched class than the women participants for both structures. In Figure 5.7b are the same results depicted from a different perspective, it shows the average percentage of the *found classes* for each exploration step. We may observe that in each exploration step, the user visited more classes when using the 3-layer MLES. This behavior is caused by the presence of the middle layer in the 3-layer MLES, which provides the higher variability of classes immediately after the first exploration operation (which is always the Zoom-In operation).



(a)



(b)

Figure 5.7: The found objects and classes during the exploration.

5.4 Discussion

The results of the user study show that using more layers in the multimedia exploration could bring better effectiveness to the exploration process. Especially, the finding of the first object of the searched class seems to be more effective when using more than two layers. On the other side, when the first object of the searched class is found, then simple $k\mathcal{NN}$ queries are utilized to explore the cluster of visually similar objects. In this study, we did not primarily focus on the searching for more visually dissimilar clusters of the same class, therefore we cannot conclude if employing more layers could help with such a task. Considering the results presented in Figures 5.6 and 5.7, there is not any significant performance gain of the three layer exploration structure for the heterogeneous searched classes. We can just intuitively guess from the results of Figures 5.3 and 5.4 that if more than 15 exploration operations are used, the users will find the first object from another visually homogeneous cluster of the searched class probably sooner.

The results would show bigger differences between both compared structures if the utilized collection contained an order of magnitude higher number of objects and classes. For example, if given 1 billion images and 1 million searched classes, it would be probably much more complicated to find some object of the searched class when using just the 2-layer MLES.

We also asked the participants to verbally compare both exploration structures. The participants did not see a big difference between effectiveness of both structures, considering the percentage of found objects from the searched class. For some search tasks the participants found the 3-layer MLES to be more useful for finding the first object of the searched class. The participants also often prefer simple user interfaces and therefore their zooming in to the last layer and using just the simple pan operation could neglect the benefits of the 3-layer MLES. The effect of the learning also slightly affected the results, because the participants always started from the same initial page zero view (which was given by the exploration structure). Therefore, in some cases, the participants could quickly find the objects of the searched class using their previous experience. On the other side, the effect of learning was the same for both compared structures and so it improved the results of both structures.

Chapter 6

Conclusion

The main motivation that triggers our research summarized in this thesis was to create the awareness of the *exploration* – the way of multimedia data retrieval that is quite different from the well-known and widely used *querying*. We started to advocate the exploration with stating the drawbacks that the querying suffers from and we revealed the proper solutions which the exploration proposes. All this, we provided together with the survey of existing browsing and exploration systems and exploration principles in Chapter 2. That survey reveals and summarizes common techniques used in the multimedia exploration, but also uncovered the main problems like the scalability of used exploration methods and structures in cases when the explored collection is very large.

When talking about the common techniques, we want to underline the fact that the multimedia exploration is the retrieval aimed for the end users. Thus, driven by such requirements of the end users as the *modern user interface* or the *truly continuous exploration*, we proposed the architecture of the general real-time multimedia exploration system (we refer to it as the *RTExp*) in Chapter 3. The RTExp meets these criteria by providing the scalable multi-layered architecture, applying the real-time similarity exploration queries, and delivering the intuitive user interface. Besides the description of the overall architecture, in Section 3.3 we introduced the approximation technique of the instant similarity queries, which directly supports the requirement for the continuous exploration. The performed evaluation of the instant similarity queries on the several metric access methods showed that not all MAMs are suitable for supporting the continuous exploration with a similarity index. The preferred are those ones, whose query evaluation is *monotonically approximative* and whose evaluation process returns suitable results very early.

The RTExp represents the high-level overview of the multimedia exploration, where we tried to describe general parts of such a system, but beside the instant similarity queries we did not focus on the particular details in it. Therefore, we continued our research by introducing the particular details of the presentation and the data layer of the RTExp architecture in Chapter 4.

Firstly, in the beginning of Chapter 4, we proposed the MLES – the new structure for the multilayer multimedia exploration. Using the MLES, we defined *exploration operations* enabling the *horizontal* and the *vertical browsing* (described in Section 2.1.3) of any multimedia collection. The MLES consists of multiple layers, where each layer can be indexed by a separate similarity index structure, which improves the scalability of the whole structure in case of the growing database. So far, the defined exploration operations are based just on the exact $k\mathcal{N}\mathcal{N}$ queries, however, the further enhancements are possible. For example, the approximate $k\mathcal{N}\mathcal{N}$ queries could be used for the more efficient retrieval in the lower layers of the MLES, or the multiple $k\mathcal{N}\mathcal{N}$ queries or the multiple similarity models could be combined in one exploration operation.

Secondly, in Section 4.2 we aimed on the details in the presentation layer, we introduced the continuity and the hierarchy into the real-world demo application providing the multimedia exploration. Our ideas came out of two previously discussed concepts the *iterative querying* and the *iterative browsing*. We tried to adopt advantages from both of them and proposed the new technique - the *hierarchical querying*. From our perception, when hierarchical querying was implemented into our system the exploration process became more directed by the user. Moreover, by introducing the preservation of the user context, we made the exploration process notably more fluent. But, the demo is still missing the desired improvements like presenting the intermediate results from continuous evaluation of the exploration operations.

In Chapter 5, we performed the extensive user study on the MLES to prove its suitability for the multimedia exploration. The MLES proved to be a more suitable concept for multimedia exploration than the standard flat $k\mathcal{N}\mathcal{N}$ querying (simulated by 2-layer MLES), but we suppose that the 3-layer MLES does not show all advantages of the multiple layers. For the further research, we plan to use more than one middle layer in the MLES hierarchy to evaluate the scalability of the system on very large collections.

Bibliography

[1] Phillipe Salembier and Thomas Sikora. *Introduction to MPEG-7: Multimedia Content Description Interface*. John Wiley & Sons, Inc., New York, NY, USA, 2002.

[2] Andres Dorado and Ebroul Izquierdo. Fuzzy color signatures. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 1, pages I–433–I–436 vol.1, 2002.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[4] Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. *Similarity Search: The Metric Space Approach*, volume 32 of *Advances in Database Systems*. Springer US, 2006.

[5] Tomáš Skopal. Unified framework for fast exact and approximate search in dissimilarity spaces. *ACM Transactions on Database Systems*, 32(4), November 2007.

[6] Marco Patella and Paolo Ciaccia. Approximate similarity search: A multifaceted problem. *Journal of Discrete Algorithms*, 7(1):36–48, March 2009.

[7] Pavel Zezula, Pasquale Savino, Giuseppe Amato, and Fausto Rabitti. Approximate similarity retrieval with M-trees. *The VLDB Journal*, 7(4):275–293, December 1998.

[8] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, 45(6):891–923, November 1998.

[9] Edgar Chávez and Gonzalo Navarro. A Probabilistic Spell for the Curse of Dimensionality. In AdamL. Buchsbaum and Jack Snoeyink, editors, *Algorithm Engineering and Experimentation*, volume 2153 of *Lecture Notes in Computer Science*, pages 147–160, London, UK, 2001. Springer Berlin Heidelberg.

[10] Gísli R. Hjaltason and Hanan Samet. Distance browsing in spatial databases. *ACM Transactions on Database Systems*, 24(2):265–318, June 1999.

[11] Gísli R. Hjaltason and Hanan Samet. *Incremental Similarity Search in Multimedia Databases*. Computer science technical report series. University of Maryland, 2000.

[12] Christian Böhm, Stefan Berchtold, and Daniel A. Keim. Searching in High-dimensional Spaces: Index Structures for Improving the Performance of Multimedia Databases. *ACM Computing Surveys*, 33(3):322–373, September 2001.

[13] Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José Luis Marroquín. Searching in Metric Spaces. *ACM Computing Surveys*, 33(3):273–321, September 2001.

[14] Christian Beecks, Merih Seran Uysal, and Thomas Seidl. Signature Quadratic Form Distance. In *Proceedings of the ACM International Conference on Image and Video Retrieval*, CIVR '10, pages 438–445, New York, NY, USA, 2010. ACM.

[15] Christian Beecks, Jakub Lokoč, Thomas Seidl, and Tomáš Skopal. Indexing the Signature Quadratic Form Distance for Efficient Content-based Multimedia Retrieval. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, ICMR '11, pages 24:1–24:8, New York, NY, USA, 2011. ACM.

[16] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. A Metric for Distributions with Applications to Image Databases. In *Proceedings of the Sixth International Conference on Computer Vision*, ICCV '98, pages 59–66, Washington, DC, USA, 1998. IEEE Computer Society.

[17] Michael S. Lew, Nicu Sebe, Chabane Djeraba, and Ramesh Jain. Content-based Multimedia Information Retrieval: State of the Art and Challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 2(1):1–19, February 2006.

[18] Juraj Moško, Tomáš Skopal, Tomáš Bartoš, and Jakub Lokoč. Real-Time Exploration of Multimedia Collections. In Hua Wang and Mohamed A. Sharaf, editors, *Databases Theory and Applications*, volume 8506 of *Lecture Notes in Computer Science*, pages 198–205. Springer International Publishing, 2014.

[19] Juraj Moško, Jakub Lokoč, Tomáš Grošup, Přemysl Čech, Tomáš Skopal, and Jan Lánský. MLES: Multilayer Exploration Structure for Multimedia Exploration. In Tadeusz Morzy, Patrick Valduriez, and Ladjel Bellatreche, editors, *New Trends in Databases and Information Systems*, volume 539 of *Communications in Computer and Information Science*, pages 135–144. Springer International Publishing, September 2015.

[20] Simone Santini and Ramesh Jain. Beyond Query by Example. In *Proceedings of the Sixth ACM International Conference on Multimedia*, MULTIMEDIA '98, pages 345–350, New York, NY, USA, 1998. ACM.

[21] Arnold W. M. Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. Content-Based Image Retrieval at the End of the Early Years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, December 2000.

[22] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 40(2):5:1–5:60, May 2008.

[23] Daniel Heesch. A survey of browsing models for content based image retrieval. *Multimedia Tools and Applications*, 40(2):261–284, November 2008.

[24] Christian Beecks, T. Skopal, K. Schöffmann, and Thomas Seidl. Towards Large-Scale Multimedia Exploration. In *Fifth International Workshop on Ranking in Databases (DBRank 2011), Seattle, WA, USA*, August.

[25] BusinessDictionary.com. Definition of word *browsing*, `http://www.businessdictionary.com/definition/browsing.html`, 2015.

[26] Giang P. Nguyen and Marcel Worring. Interactive Access to Large Image Collections Using Similarity-based Visualization. *Journal of Visual Languages & Computing*, 19(2):203–224, April 2008.

[27] Christos Faloutsos and King-Ip Lin. FastMap: A Fast Algorithm for Indexing, Data-mining and Visualization of Traditional and Multimedia Datasets. *SIGMOD Records*, 24(2):163–174, May 1995.

[28] Karl Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901.

[29] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, 1933.

[30] Joseph B. Kruskal and Myron Wish. *Multidimensional Scaling*. Number 11 in 07. SAGE Publications, 1978.

[31] Wojciech Basalaj. Incremental Multidimensional Scaling Method for Database Visualization. In *In Proceedings of the Visual Data Exploration and Analysis VI, SPIE*, volume 3643, pages 149–158, January 1999.

[32] John W. Sammon. A Nonlinear Mapping for Data Structure Analysis. *IEEE Transactions on Computers*, C-18(5):401–409, May 1969.

[33] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, 1982.

[34] Peter Eades. A Heuristic for Graph Drawing. *Congressus Numerantium*, 42:149–160, 1984.

[35] Thomas M. J. Fruchterman and Edward M. Reingold. Graph Drawing by Force-directed Placement. *Software Practice and Experience*, 21(11):1129–1164, November 1991.

[36] Ron Davidson and David Harel. Drawing Graphs Nicely Using Simulated Annealing. *ACM Transactions on Graphics*, 15(4):301–331, October 1996.

[37] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science (New York, N.Y.)*, 290(5500):2319–2323, December 2000.

[38] Sam T. Roweis and Lawrence K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science (New York, N.Y.)*, 290(5500):2323–2326, December 2000.

[39] Geoffrey Hinton and Sam Roweis. Stochastic Neighbor Embedding. *Advances in Neural Information Processing Systems*, 15:833–840, 2003.

[40] Zoran Pečenović, MinhN. Do, Martin Vetterli, and Pearl Pu. Integrated Browsing and Searching of Large Image Collections. In Robert Laurini, editor, *Advances in Visual Information Systems*, volume 1929 of *LNCS*, pages 279–289. Springer Berlin Heidelberg, 2000.

[41] Kerry Rodden, Wojciech Basalaj, David Sinclair, and Kenneth Wood. Evaluating a Visualization of Image Similarity as a Tool for Image Browsing. In *Proceedings of the 1999 IEEE Symposium on Information Visualization*, INFOVIS '99, pages 36–43, Washington, DC, USA, 1999. IEEE Computer Society.

[42] Jakub Lokoč, Tomáš Grošup, and Tomáš Skopal. SIR: The Smart Image Retrieval Engine. In Gonzalo Navarro and Vladimir Pestov, editors, *Similarity Search and Applications*, volume 7404 of *Lecture Notes in Computer Science*, pages 240–241, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[43] Jakub Lokoč, Tomáš Grošup, and Tomáš Skopal. Image exploration using online feature extraction and reranking. In *Proceedings of the 2Nd ACM International Conference on Multimedia Retrieval*, ICMR '12, pages 66:1–66:2, New York, NY, USA, 2012. ACM.

[44] Jakub Lokoč, Tomáš Grošup, Přemysl Čech, and Tomáš Skopal. Towards efficient multimedia exploration using the metric space approach. In *Content-Based Multimedia Indexing (CBMI), 2014 12th International Workshop on*, pages 1–4, June 2014.

[45] Tomáš Grošup, Přemysl Čech, Jakub Lokoč, and Tomáš Skopal. A Web Portal for Effective Multi-model Exploration. In Xiangjian He, Suhuai Luo, Dacheng Tao, Changsheng Xu, Jie Yang, and MuhammadAbul Hasan, editors, *MultiMedia Modeling*, volume 8936 of *Lecture Notes in Computer Science*, pages 315–318. Springer International Publishing, 2015.

[46] Daniel Heesch and Stefan Rüger. NN$^{\mathrm{k}}$ Networks for Content-Based Image Retrieval. In Sharon McDonald and John Tait, editors, *Advances in Information Retrieval*, volume 2997 of *Lecture Notes in Computer Science*, pages 253–266. Springer Berlin Heidelberg, 2004.

[47] Stijn van Dongen. A Cluster Algorithm for Graphs. Technical report, Amsterdam, The Netherlands, The Netherlands, 2000.

[48] Hao Liu, Xing Xie, Xiaoou Tang, Zhi-Wei Li, and Wei-Ying Ma. Effective Browsing of Web Image Search Results. In *Proceedings of the 6th ACM SIGMM International Workshop on Multimedia Information Retrieval*, MIR '04, pages 84–90, New York, NY, USA, 2004. ACM.

[49] Christian Beecks, Sascha Wiedenfeld, and Thomas Seidl. Improving the Efficiency of Content-Based Multimedia Exploration. In *20th International Conference on Pattern Recognition (ICPR), 2010*, pages 3163–3166, August 2010.

[50] Christian Beecks, Philip Driessen, and Thomas Seidl. Index Support for Content-based Multimedia Exploration. In *Proceedings of the International Conference on Multimedia*, MM '10, pages 999–1002, New York, NY, USA, 2010. ACM.

[51] C. Beecks, M.S. Uysal, P. Driessen, and T. Seidl. Content-based exploration of multimedia databases. In *Content-Based Multimedia Indexing (CBMI), 2013 11th International Workshop on*, pages 59–64, June 2013.

[52] Paolo Ciaccia, Marco Patella, and Pavel Zezula. M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, VLDB '97, pages 426–435, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[53] Gerald Schaefer. A next generation browsing environment for large image repositories. *Multimedia Tools and Applications*, 47(1):105–120, March 2010.

[54] Gerald Schaefer. Interactive Browsing of Image Repositories. In Leonard Bolc, Ryszard Tadeusiewicz, LeszekJ. Chmielewski, and Konrad Wojciechowski, editors, *Computer Vision and Graphics*, volume 7594 of *Lecture Notes in Computer Science*, pages 236–244, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[55] Ilaria Bartolini, Paolo Ciaccia, and Marco Patella. The PIBE Personalizable Image Browsing Engine. In *Proceedings of the 1st International Workshop on Computer Vision Meets Databases*, CVDB '04, pages 43–50, New York, NY, USA, 2004. ACM.

[56] Jau-Yuen Chen, Charles A. Bouman, and John C. Dalton. Hierarchical browsing and search of large image databases. *IEEE Transactions on Image Processing*, 9(3):442–455, March 2000.

[57] Raphael A. Finkel and Jon L. Bentley. Quad trees a data structure for retrieval on composite keys. *Acta Informatica*, 4(1):1–9, 1974.

[58] Kai Uwe Barthel, Nico Hezel, and Radek Mackowiak. ImageMap - Visually Browsing Millions of Images. In Xiangjian He, Suhuai Luo, Dacheng Tao,

Changsheng Xu, Jie Yang, and MuhammadAbul Hasan, editors, *MultiMedia Modeling*, volume 8936 of *Lecture Notes in Computer Science*, pages 287–290. Springer International Publishing, 2015.

[59] Radek Mackowiak, Nico Hezel, and Kai Uwe Barthel. Picsbuffet, `https://picsbuffet.com`, 2015.

[60] KaiUwe Barthel, Nico Hezel, and Radek Mackowiak. Graph-Based Browsing for Large Video Collections. In Xiangjian He, Suhuai Luo, Dacheng Tao, Changsheng Xu, Jie Yang, and MuhammadAbul Hasan, editors, *MultiMedia Modeling*, volume 8936 of *Lecture Notes in Computer Science*, pages 237–242. Springer International Publishing, 2015.

[61] Grant Strong and Minglun Gong. Browsing a Large Collection of Community Photos Based on Similarity on GPU. In George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Paolo Remagnino, Fatih Porikli, Jörg Peters, James Klosowski, Laura Arns, YuKa Chun, Theresa-Marie Rhyne, and Laura Monroe, editors, *Advances in Visual Computing*, volume 5359 of *Lecture Notes in Computer Science*, pages 390–399, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[62] William Plant and Gerald Schaefer. Navigation and Browsing of Image Databases. In *Soft Computing and Pattern Recognition, 2009. SOCPAR '09. International Conference of*, pages 750–755, December 2009.

[63] Klaus Schoeffmann, David Ahlström, and Laszlo Böszörmenyi. A User Study of Visual Search Performance with Interactive 2D and 3D Storyboards. In Marcin Detyniecki, Ana García-Serrano, Andreas Nürnberger, and Sebastian Stober, editors, *Adaptive Multimedia Retrieval. Large-Scale Multimedia Retrieval and Evaluation*, volume 7836 of *Lecture Notes in Computer Science*, pages 18–32. Springer Berlin Heidelberg, 2013.

[64] Klaus Schoeffmann, David Ahlstrom, and Christian Beecks. 3D Image Browsing on Mobile Devices. In *IEEE International Symposium on Multimedia (ISM), 2011*, pages 335–336, Washington, DC, USA, December 2011. IEEE Computer Society.

[65] Klaus Schoeffmann, Marco A. Hudelist, Manfred del Fabro, and Gerald Schaefer. Mobile Image Browsing on a 3D Globe: Demo Paper. In *Proceedings of the 2Nd ACM International Conference on Multimedia Retrieval*, ICMR '12, pages 61:1–61:2, New York, NY, USA, 2012. ACM.

[66] Manojit Sarkar and Marc H. Brown. Graphical Fisheye Views. *Communications of the ACM*, 37(12):73–83, December 1994.

[67] Cees G. M. Snoek, Marcel Worring, Dennis C. Koelma, and Arnold W. M. Smeulders. A Learned Lexicon-Driven Paradigm for Interactive Video Retrieval. *IEEE Transactions on Multimedia*, 9(2):280–292, February 2007.

[68] Ork de Rooij, Cees G. M. Snoek, and Marcel Worring. MediaMill: Semantic Video Search Using the RotorBrowser. In Nicu Sebe and Marcel Worring, editors, *Proceedings of the 6th ACM International Conference on Image*

and Video Retrieval, CIVR '07, pages 649–649, New York, NY, USA, 2007. ACM.

[69] Chaomei Chen, George Gagaudakis, and Paul L. Rosin. Similarity-Based Image Browsing, 2000.

[70] Chaomei Chen, George Gagaudakis, and Paul L. Rosin. Content-Based Image Visualization. In *IEEE International Conference on Information Visualization, 2000. Proceedings.*, pages 13–18, 2000.

[71] R.W. Schvaneveldt, F T Durso, and D.W. Dearholt. *Network structures in proximity data*, pages 249–284. 24 edition, 1989.

[72] Patrick Chiu, Andreas Girgensohn, Surapong Lertsithichai, Wolf Polak, and Frank Shipman. MediaMetro: Browsing Multimedia Document Collections with a 3D City Metaphor. In *Proceedings of the 13th Annual ACM International Conference on Multimedia*, MULTIMEDIA '05, pages 213–214, New York, NY, USA, 2005. ACM.

[73] Gonzalo Navarro. Analyzing Metric Space Indexes: What For? In *Similarity Search and Applications, 2009. SISAP '09. Second International Workshop on*, pages 3–10, August 2009.

[74] Gonzalo Navarro. Searching in metric spaces by spatial approximation. *The VLDB Journal*, 11(1):28–46, August 2002.

[75] María Luisa Micó, José Oncina, and Enrique Vidal. A New Version of the Nearest-neighbour Approximating and Eliminating Search Algorithm (AESA) with Linear Preprocessing Time and Memory Requirements. *Pattern Recognition Letters*, 15(1):9–17, 1994.

[76] Magnus L. Hetland. Ptolemaic Indexing. *CoRR*, abs/0911.4384, 2009.

[77] Jakub Lokoč, Magnus Lie Hetland, Tomáš Skopal, and Christian Beecks. Ptolemaic indexing of the signature quadratic form distance. In *Proceedings of the Fourth International Conference on SImilarity Search and APplications*, SISAP '11, pages 9–16, New York, NY, USA, 2011. ACM.

[78] Tomáš Skopal, Jaroslav Pokorný, and Václav Snášel. Nearest Neighbours Search Using the PM-Tree. In Lizhu Zhou, BengChin Ooi, and Xiaofeng Meng, editors, *Database Systems for Advanced Applications*, volume 3453 of *Lecture Notes in Computer Science*, pages 803–815. Springer Berlin Heidelberg, 2005.

[79] Tomáš Skopal, Jaroslav Pokorný, and Václav Snášel. PM-tree: Pivoting Metric Tree for Similarity Search in Multimedia Databases. In *ADBIS (Local Proceedings)*, 2004.

[80] Jakub Lokoč and Tomáš Skopal. On Reinsertions in M-tree. In *Proceedings of the First International Workshop on Similarity Search and Applications (Sisap 2008)*, SISAP '08, pages 121–128, Washington, DC, USA, 2008. IEEE Computer Society.

[81] Tomáš Skopal, Jaroslav Pokorný, Michal Krátký, and Václav Snášel. Revisiting M-Tree Building Principles. In Leonid Kalinichenko, Rainer Manthey, Bernhard Thalheim, and Uwe Wloka, editors, *Advances in Databases and Information Systems*, volume 2798 of *Lecture Notes in Computer Science*, pages 148–162. Springer Berlin Heidelberg, 2003.

[82] Tomáš Skopal and Jakub Lokoč. New dynamic construction techniques for M-tree. *Journal of Discrete Algorithms*, 7(1):62–77, 2009.

[83] R. Bayer and E.M. McCreight. Organization and maintenance of large ordered indexes. *Acta Informatica*, 1(3):173–189, 1972.

[84] Edgar Chávez, Karina Figueroa, and Gonzalo Navarro. Effective Proximity Retrieval by Ordering Permutations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(9):1647–1658, September 2008.

[85] David Novák and Michal Batko. Metric Index: An Efficient and Scalable Solution for Similarity Search. In *SISAP '09*, pages 65–73, Washington, DC, USA, 2009. IEEE Computer Society.

[86] David Novák, Michal Batko, and Pavel Zezula. Metric Index: An Efficient and Scalable Solution for Precise and Approximate Similarity Search. *Information Systems*, 36(4):721–733, June 2011.

[87] David Novák, Michal Batko, and Pavel Zezula. Large-scale similarity data management with distributed Metric Index. *Information Processing & Management*, 48(5):855–872, 2012. Large-Scale and Distributed Systems for Information Retrieval.

[88] H. V. Jagadish, Beng Chin Ooi, Kian-Lee Tan, Cui Yu, and Rui Zhang. iDistance: An Adaptive B+-tree Based Indexing Method for Nearest Neighbor Search. *ACM Transactions on Database Systems*, 30(2):364–397, June 2005.

[89] Tomáš Skopal and Jakub Lokoč. NM-Tree: Flexible Approximate Similarity Search in Metric and Non-metric Spaces. In Sourav S. Bhowmick, Josef Küng, and Roland Wagner, editors, *Database and Expert Systems Applications*, volume 5181 of *Lecture Notes in Computer Science*, pages 312–325. Springer Berlin Heidelberg, 2008.

[90] Tomáš Bartoš, Tomáš Skopal, and Juraj Moško. Towards Efficient Indexing of Arbitrary Similarity: Vision Paper. *SIGMOD Records*, 42(2):5–10, July 2013.

[91] Tomáš Bartoš, Tomáš Skopal, and Juraj Moško. Efficient Indexing of Similarity Models with Inequality Symbolic Regression. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, GECCO '13, pages 901–908, New York, NY, USA, 2013. ACM.

[92] Hakan Ferhatosmanoglu, Ertem Tuncel, Divyakant Agrawal, and Amr El Abbadi. Approximate nearest neighbor searching in multimedia databases. In *17th International Conference on Data Engineering (ICDE)*, pages 503–511, 2001.

[93] Jeffrey K. Uhlmann. Implementing Metric Trees to Satisfy General Proximity/Similarity Queries, 1991. Manuscript.

[94] Ben Shneiderman. Response Time and Display Rate in Human Performance with Computers. *ACM Computuer Surveys*, 16(3):265–285, September 1984.

[95] Simon Thorpe, Denis Fize, and Catherine Marlot. Speed of processing in the human visual system. *Nature*, 381:520–522, 1996.

[96] HumanBenchmark.com. Human benchmark on human reaction time, `http://www.humanbenchmark.com/tests/reactiontime`, 2015. visited on 2015-10-27.

[97] Clifford W. Mercer. An Introduction to Real-Time Operating Systems: Scheduling Theory, 1992. Unpublished manuscript.

[98] Simone Santini and Ramesh Jain. Integrated Browsing and Querying for Image Databases. *IEEE Multimedia*, 7(3):26–39, July 2000.

[99] Susan B. Davidson and Aaron Watters. Partial Computation in Real-Time Database Systems. In *Proc. Workshop Real-Time Software and Operating Systems*, 1988.

[100] Jan-Mark Geusebroek, Gertjan J. Burghouts, and Arnold W. M. Smeulders. The Amsterdam Library of Object Images. *International Journal of Computer Vision*, 61(1):103–112, January 2005.

[101] Paolo Bolettieri, Andrea Esuli, Fabrizio Falchi, Claudio Lucchese, Raffaele Perego, Tommaso Piccioli, and Fausto Rabitti. CoPhIR: a Test Collection for Content-Based Image Retrieval. *Computing Research Repository*, abs/0905.4627v2, 2009.

[102] B.S. Manjunath, J.-R. Ohm, V.V. Vasudevan, and A. Yamada. Color and texture descriptors. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6):703–715, Jun 2001.

[103] Tomáš Skopal. Where Are You Heading, Metric Access Methods?: A Provocative Survey. In *Proceedings of the Third International Conference on SImilarity Search and APplications*, SISAP '10, pages 13–21, New York, NY, USA, 2010. ACM.

[104] Tomáš Grošup, Juraj Moško, and Přemysl Čech. Continuous hierarchical exploration of multimedia collections. In *Content-Based Multimedia Indexing (CBMI), 2015 13th International Workshop on*, pages 1–4, June 2015.

[105] Tomáš Grošup, Juraj Moško, and Přemysl Čech. Continuous Hierarchical Exploration of Multimedia Collections, SIRET Research Group, `http://www.siret.cz/HierarchicalExploration`, 2015.

[106] Juraj Moško, Jakub Lokoč, and Tomáš Skopal. Clustered Pivot Tables for I/O-optimized Similarity Search. In *Proceedings of the Fourth International Conference on SImilarity Search and APplications*, SISAP '11, pages 17–24, New York, NY, USA, 2011. ACM.

[107] Jakub Lokoč, Juraj Moško, Přemysl Čech, and Tomáš Skopal. On indexing metric spaces using cut-regions. *Information Systems*, 43:1–19, 2014.

[108] Edgar Chávez and Gonzalo Navarro. A compact space decomposition for effective metric indexing. *Pattern Recognition Letters*, 26(9):1363 – 1376, 2005.

[109] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[110] Meelis Kull and Jaak Vilo. Fast approximate hierarchical clustering using similarity heuristics. *BioData Mining*, 1(1), 2008.

[111] Christian Böhm, Bernhard Braunmüller, Florian Krebs, and Hans-Peter Kriegel. Epsilon Grid Order: An Algorithm for the Similarity Join on Massive High-dimensional Data. *SIGMOD Records*, 30(2):379–388, May 2001.

[112] Chi-Wing Fu, Wooi-Boon Goh, and Junxiang Allen Ng. Multi-touch Techniques for Exploring Large-scale 3D Astrophysical Simulations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 2213–2222, New York, NY, USA, 2010. ACM.

[113] Nitin Bhatia and Vandana. Survey of Nearest Neighbor Techniques. *CoRR*, abs/1007.0085, 2010.

[114] Mohammad Reza Abbasifard, Bijan Ghahremani, and Hassan Naderi. A Survey on Nearest Neighbor Search Methods. *International Journal of Computer Applications*, 95(25):39–52, June 2014.

[115] Juraj Moško, Jakub Lokoč, Tomáš Grošup, Přemysl Čech, Tomáš Skopal, and Jan Lánský. Evaluating Multilayer Multimedia Exploration. In Giuseppe Amato, Richard Connor, Fabrizio Falchi, and Claudio Gennaro, editors, *Similarity Search and Applications*, volume 9371 of *Lecture Notes in Computer Science*, pages 162–169. Springer International Publishing, October 2015.

[116] Tomáš Grošup, Přemysl Čech, Jakub Lokoč, and Tomáš Skopal. Multimedia Exploration Using Metric Indexes, SIRET Research Group, `http://www.siret.cz/MAMExploration`, 2014.

[117] Tomáš Grošup, Přemysl Čech, Jakub Lokoč, Juraj Moško, and Tomáš Skopal. Multimedia Exploration Using Metric Indexes, SIRET Research Group, `http://www.siret.cz/ExplorationService`, 2015.

[118] Petra Budikova, Michal Batko, and Pavel Zezula. Evaluation Platform for Content-based Image Retrieval Systems. In Stefan Gradmann, Francesca Borri, Carlo Meghini, and Heiko Schuldt, editors, *Research and Advanced Technology for Digital Libraries*, volume 6966 of *Lecture Notes in Computer Science*, pages 130–142, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[119] Yossi Rubner and Carlo Tomasi. *Perceptual Metrics for Image Database Navigation.* Kluwer Academic Publishers, Norwell, MA, USA, 2001.

[120] Mark J. Huiskes, Bart Thomee, and Michael S. Lew. New Trends and Ideas in Visual Concept Detection: The MIR Flickr Retrieval Evaluation Initiative. In *Proceedings of the International Conference on Multimedia Information Retrieval*, MIR '10, pages 527–536, New York, NY, USA, 2010. ACM.

Appendix

A.1 Architecture of the Demo Application

As you can see in Figure A.1, the demo application is based on a client-server architecture. The client is a single-page application and is responsible for handling the user actions, for preserving the context between the individual exploration queries and for the visualization of the result data. The server is an ASP.NET web application which handles HTTP requests, creates and manages the collection of metric indexes and prepares data for the client part in form of a similarity graph.

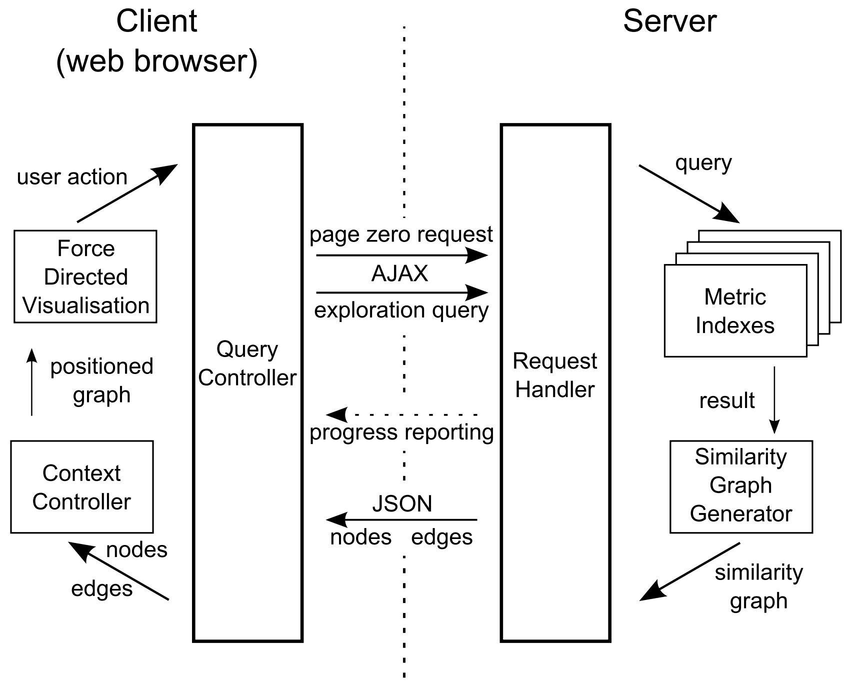


Figure A.1: The architecture of the demo application.

When the user opens the demo application, the *Query Controller* sends the request to the server for the *page zero* via an AJAX call. This request is handled by the *Request Handler*, which retrieves result objects from the page zero. The result objects are the collection of multimedia objects containing the identifier of the object, the specific URL address where the multimedia object itself can be retrieved from and the index-specific metadata, which are preserved between the individual exploration steps. After the objects are retrieved, the result collection is passed to the *Similarity Graph Generator*, which is responsible for the computation of distances between the objects in the result and for saving them in the similarity matrix. The matrix is subsequently transformed into the similarity graph by keeping only those edges (the relations between individual the objects from the matrix) whose similarity value is higher than the system-specific threshold. The similarity graph is returned back to the client in the form of a JSON structure containing the nodes (objects from the matrix) and those kept edges.

On the client side, the received similarity graph is passed to the *Context Controller*, which compares the new result with the previous one. The objects that are not present in the new result are removed from the screen, the rest of objects stay at the same position where they had been located before the query started. After that, each remaining object of the new result is placed next to its most similar object already added in the visualized space (see Section 4.2.1). This can be computed very quickly, since the result set already contains the edges with their similarity weights. The graph with the x, y coordinates adjusted by the *Context Controller* is passed to the *Force Directed Visualization* component. Its responsibility is to tune the position of the visualized objects by using the particle physics model (i.e., force-directed placement) until stable distribution of the images on the screen is achieved.

In addition to the visualization, the client provides the user interface for the exploration of the visualized collection by selecting the objects of the user interest. After one or more objects are selected, the *Query Controller* generates the exploration query from them and sends the query request to the server. Since each of the visualized objects contains the index-specific metadata, the server can retrieve the current context from the query request – the state in which the exploration currently is – and then, according to the context, the sever determines which layer (the page zero, the middle page or the whole database) of the underlying index should be queried.

A.2 Reachability of the MLES Operations

In this section, we introduce a qualitative criterion for the MLES (see Section 4.1), where we focus on the number of objects which can be reached using the defined exploration operations. Ideally, all objects should be reachable, however, the *reachability* depends on several factors including the number of layers, the number of objects in the layers (i.e., on the selection function ϕ, see Definition 4.1), the layer selection techniques, the utilized exploration operations and the number of displayed objects. Whereas the number of displayed objects is given by the size of the concrete screen and the advanced (not random) layer selection techniques can be too expensive for huge datasets, we focus mainly on the number of objects in the layers to guarantee the user-given reachability for the specific exploration operations (we consider just the Zoom-In and the Pan), and on the number of layers. Let us note, the number of layers affects the number of similarity indexes to maintain and also the number of user clicks to reach the searched items, therefore, the number of layers should not be too high.

In order to define the reachability of the MLES operations, we utilize the exploration closure which describes the set of objects that are accessible from the initial layer L_0 using a set of exploration operations. For simplicity, we start with the closure according to the operation Zoom-In, which represents the natural top-down manner of accessing information.

Definition A.1. *Let $E = MLES(\mathbb{S}, m, k, \phi)$ be a multilayer exploration structure, $l \in \mathbb{N}, l \leq m$ and $\mathbb{Z}_l \subseteq \mathbb{S}$ such that $\mathbb{L}_0 \subseteq \mathbb{Z}_l$ and also $\forall i \in \mathbb{N}, 0 \leq i < l : \forall q \in \mathbb{Z}_i : Zoom-In(q, i) \subseteq \mathbb{Z}_{i+1}$, then $\mathbb{Z}_l$ is called the exploration closure according to the operation Zoom-In and the layer l.*

Using the exploration closure $\mathbb{Z}_l$ according to the operation Zoom-In, we can simply define the Zoom-In reachability ZR_l in $E = MLES(\mathbb{S}, m, k, \phi)$ and the layer l as the fraction:

$$ZR_l = |\mathbb{Z}_l|/|\mathbb{L}_l|.$$

If we utilize also the Pan operation defined for all layers, the number of reachable objects could be significantly higher because more objects from the specific layer can be accessed by their neighbors. In some cases, the Pan operation can even help users to access the desired objects more quickly. On the other side, the Pan operation has also its limits, for example, according to our experience the Pan operations can cycle in lower layers showing still the same objects (see Section 4.1.1). However, the Pan operation is a significant extension of the Zoom-In/Out exploration and can be used to extend the definition of the exploration closure according to the operation Zoom-In as follows:

Definition A.2. *Let $E = MLES(\mathbb{S}, m, k, \phi)$ be a multilayer exploration structure, $l \in \mathbb{N}, l \leq m$ and $\mathbb{ZP}_l \subseteq \mathbb{S}$ such that $\mathbb{L}_0 \subseteq \mathbb{ZP}_l$ and also $\forall i \in \mathbb{N}, 0 \leq i < l :$ $\forall q \in \mathbb{ZP}_i : Zoom - In(q, i) \subseteq \mathbb{ZP}_{i+1}$ and $\forall q \in \mathbb{ZP}_{i+1} : Pan(q, i + 1) \subseteq \mathbb{ZP}_{i+1}$, then $\mathbb{ZP}_l$ is called the exploration closure according to the operations Zoom-In and Pan and the layer l.*

Using the exploration closure $\mathbb{ZP}_l$ according to the operations Zoom-In and Pan, we can simply define the Zoom-In and the Pan reachability ZPR_l in $E = MLES(\mathbb{S}, m, k, \phi)$ and the layer l as the fraction:

$$ZPR_l = |\mathbb{ZP}_l|/|\mathbb{L}_l|.$$

Having defined the Zoom-In (and the Pan) reachability, we can empirically evaluate the effect of the number of layers and the selection function ϕ that should be transparent to the users. However, to decide which exploration operations are allowed is more complicated problem, because it is hard to decide whether just the Zoom-In/ZoomOut operations can lead to the desired result faster then using these operations also with the Pan operation. The suitability of the used strategy depends on many factors like, for example, on the user preferences and behavior, on the GUI, on the dataset or on the search intents, and many user studies have to be evaluated to make some conclusions. Furthermore, employing the Pan operation leads to the high computational complexity of the empirical evaluation. Therefore, for this preliminary study on the reachability of the MLES operations, we utilize just the closures with the limited number of the Pan operations. For example, for the 3-layer exploration structure and two allowed Pan operations, only the $\{Zoom{-}In, Zoom{-}In, Pan, Pan\}$, $\{Zoom{-}In, Pan, Zoom{-}In, Pan\}$ and $\{Zoom - In, Pan, Pan, Zoom - In\}$ user action sequences[1] are investigated.

[1]The user action sequence represents performing all exploration operations in order of the sequence, while starting from the exploration starting point – the visualization of the layer L_0.

Configuration	Layer						Sum
	0	1	2	3	4	5	
CoPhIR(k(20), pow(0.5))	0.01	38	54	66	76	85	319
CoPhIR(k(20), pow(2))	0.01	3	13	31	55	85	187

Table A.1: The memory complexity of the multilayer exploration structures (in MBs)

A.2.1 Experimental Settings and Results

Having introduced the theoretical definition of the MLES (see Chapter 4), we experimentally demonstrate its assets, especially we focus on the reachability properties. We start with the short description of the experimental environment, then we provide the preliminary results of the Zoom-In and the Pan reachability and then we end this section with some discussion on the matter.

All following experiments were performed on two datasets based on the MPEG-7 descriptors [102], both consisted of 100,000 images. The first used dataset is a subset of the MIRFLICKR [120] where the utilized descriptor vectors comprises two of the MPEG-7 descriptor, the Homogeneous Texture and the Edge Histogram. The second dataset is a subset of the CoPhIR [101], where the descriptor vectors are represented by the MPEG-7 descriptors the Scalable Color and the Color Structure. For a distance function, we used the Euclidean distance for the CoPhIR, while for the MIRFLICKR we used the combination of the weighted Manhattan distance, as it is suggested by the authors of the dataset and the MPEG-7 standard.

To study the reachability properties thoroughly, we have used various configurations of the MLES. The very first parameter k determines the number of objects that can be displayed all at once on the screen (see the definition of the page zero view in Section 4.1). We used the configuration with $k = 20$ to simulate the exploration on the devices with the small screen, e.g., smartphones, and $k = 50$ for the larger screens. This parameter also defines the number of objects in the zero layer of the structure as well as the number of objects retrieved by the $k\mathcal{NN}$ queries[2].

The next parameter we used is the number of layers – m that varies from 2 to 6, the first layer contains k objects and the last layer contains all objects of the dataset as follows from the definition of the MLES in Section 4.1.

For the generation of the layer capacity, we employed the selection functions $\phi(l) = (l/m-1)^{pow}$, where m stays for the total number of layers and the exponent pow in our tests varies from 0.5 to 6. For better imagination of used selection functions, we refer to Figure A.2, where we can observe how the number of objects in the particular layer is directly affected by the selection function. In order to process similarity queries efficiently, we have create the PM-Tree index for each layer except the zero layer. As we can see in Table A.1, the choice of the selection function is reflected also in the size of the PM-Trees. Hence, if the memory complexity is one of the criteria for choosing the right configuration for the multimedia exploration, the index sizes should be also taken into an account.

[2]The latter states only for these experiments, not for the MLES operations in general.

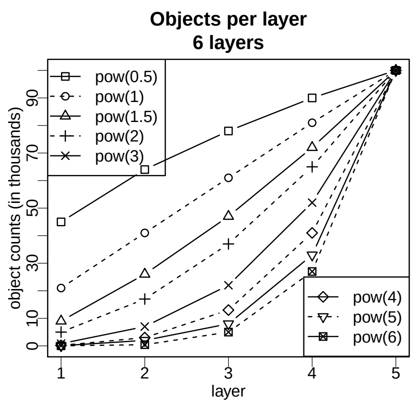
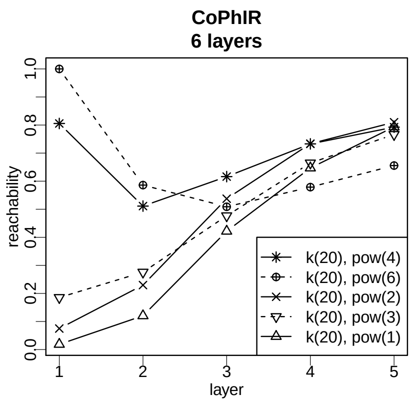
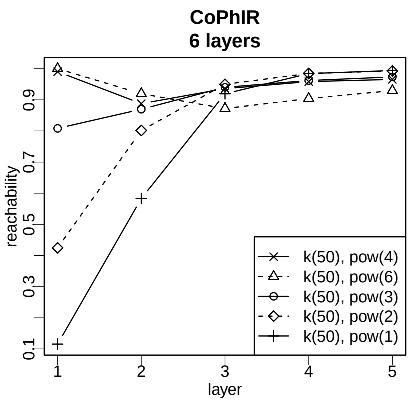


Figure A.2: The influence of the selection functions used in the experiments on object counts.

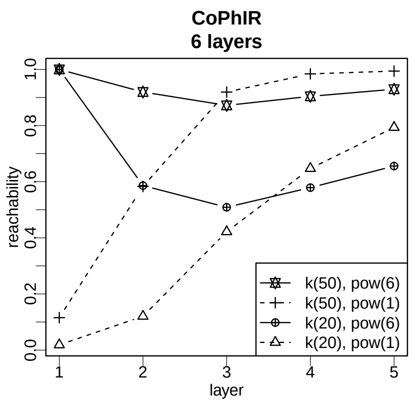
A.2.2 Zoom-In Reachability



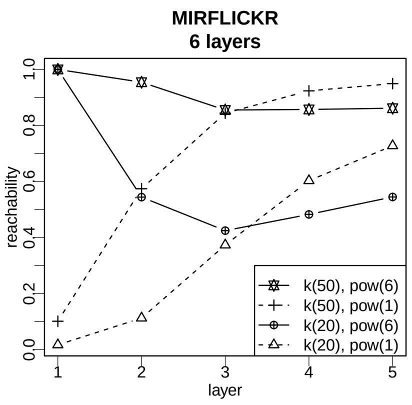
(a)



(b)



(c)

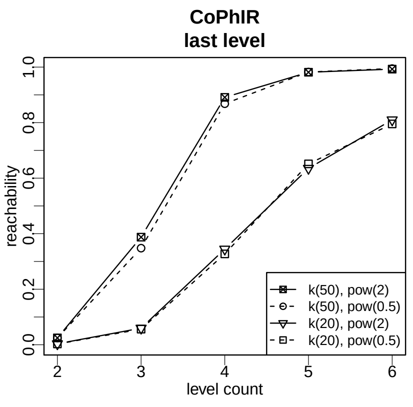


(d)

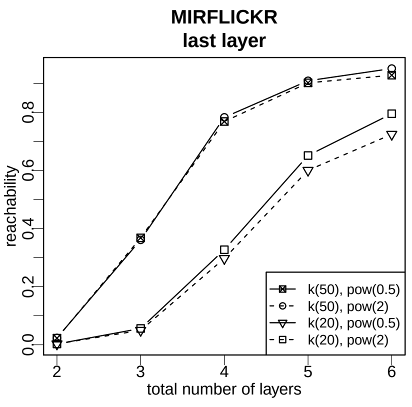
Figure A.3: The Zoom-In reachability per layer of the 6-layer structure – the comparison of different kernel functions: **a)** 20 displayed objects **b)** 50 displayed objects, and the comparison of a different number of displayed objects: **c)** CoPhIR **d)** MIRFLICKR

In the first experiment, we studied the Zoom-In reachability ZR_l on the MLES with six layers. Let us note that one point in Figure A.3 represents the Zoom-In reachability (y-axis) for the individual layer l (x-axis) of total six layers with $k = 20$ or $k = 50$ displayed objects and the kernel function power parameter *pow*. From the results, we can observe the influence of the kernel function on the individual layers. For low values of the *pow* parameter, the reachability in top sparse layers is small due to big amounts of the indexed data in the layer L_1, while in bottom denser layers the kernel functions with the lower *pow* become competitive. We can also observe that for the variants with 20 displayed objects, the values of the reachability in bottom layers (up to 80%) do not reach the results of the variants with 50 displayed objects (almost 100%). Hence, for 20 displayed objects and the random selection of objects in layers it is reasonable to use the MLES with more than 6 layers. In the bottom part of Figure A.3, we provide the direct comparison of the MLESes with the different number of displayed objects in the single graph, where we may observe how significantly a the higher number of displayed objects affects the reachability. We may also observe that all experiments on both datasets result in the reachability values similar to each other when using the same parameters.

While the previous experiments were performed on the MLESes with a fixed number of layers, the next evaluation focuses on the comparison of the MLESes with the varying number of layers where the Zoom-In reachability was always examined just for the last bottom layer (see Figure A.4). One point in the figure in this scenario represents, for example, the Zoom-In reachability (y-axis) for the layer $l = 2$ in the 3-layer (x-axis) structure with $k = 20$ displayed objects and the kernel function parameter $pow = 2$. From the results, we can observe that the 2-layer or the 3-layer MLESes are not suitable for indexing 100,000 objects because the Zoom-In reachability is too low (below 50%). Furthermore, for $k = 20$ displayed objects even the 4-layer structure is not sufficient. On the contrary, for $k = 50$ displayed objects the improvement ensured by the additional layer of the 6-layer structure in comparison to the 5-layer structure is not significant.



(a)

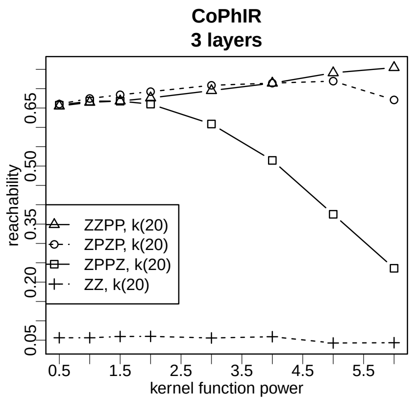


(b)

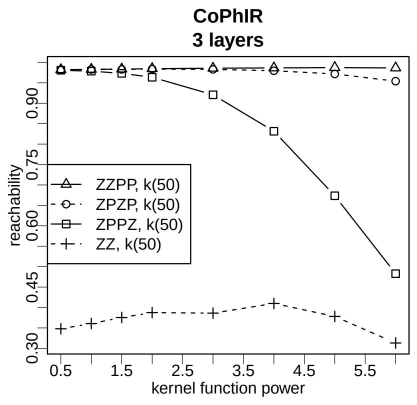
Figure A.4: The Zoom-In reachability for the last layer, the comparison of the multilayer structures with the varying number of layers

A.2.3 Zoom-In and Pan Reachability

In the second set of experiments, we studied the Zoom-In and the Pan reachability ZPR_l in the 3-layer MLESes. Additionally to two Zoom-In operations, we allowed just two Pan operations, one performed in the layerL_1 and one in the layer L_2 (denoted as $ZPZP$), or both in the layer L_2 ($ZPPZ$) or both in the layer L_2 ($ZZPP$). In Figure A.5), we can observe the increased values of the reachability when including also the Pan operations. The figure shows the results of the experiments with the varying power of the kernel function, where we measured the reachability right after the last operation of the sequence (e.g., the Pan in sequence $ZPZP$) was performed. For example, one point in the left part of the figure means the Zoom-In and the Pan reachability for the layer $l = 2$ in the 3-layer MLES with $k = 20$ displayed objects, the kernel function parameter *power* $= 1$ (x-axis) and the sequence of operations $ZPZP$. We may notice that the results are acceptable if the Pan operation is performed also on the last layer ($ZPZP$, $ZZPP$). But, if the last operation is the Zoom-In ($ZPPZ$), the results are quite poor. For comparison, we also added the sequence containing just the Zoom-In operations (ZZ), which yields the worst performance.



(a)



(b)

Figure A.5: The Zoom-In and Pan reachability on the 3-layer MLES with the varying power of the kernel function: **a)** 20 displayed objects **b)** 50 displayed objects

A.2.4 Discussion

In the experiments, we experimentally demonstrated that the exploration performed on the MLES can result in the sufficiently high reachability (under the specific configurations). Someone could object that 100,000 of objects and the maximum of six layers is just not enough to prove the scalability of the MLES and we partly agree. But as the evaluation of the reachability properties on bottom layers was computationally too expensive and as we focused on the basic behavior of the reachability properties and not on the absolute values, the suggested experimental settings is merely enough for this study.

List of Figures

List of Tables

List of Abbreviations

DC	distance computation
CBMR	Content-based multimedia retrieval
GPU	Graphics processing unit
ISOMAP	Isometric mapping
$k\mathcal{N}\mathcal{N}$	k-nearest neighbors
LLE	Local linear embedding
MAM	Metric access method
MDS	Multidimensional scaling
MLES	Multi-layer Exploration Structure
PCA	Principal Component Analysis
SNE	Stochastic neighbor embedding
SOM	Self-organizing map
SQFD	Signature Quadratic Form Distance
WWW	World Wide Web