WEB-SCALE SYSTEM FOR IMAGE SIMILARITY SEARCH: WHEN THE DREAMS ARE COMING TRUE

David Novak, Michal Batko and Pavel Zezula

Masaryk University, Brno, Czech Republic {david.novak|batko|zezula}@fi.muni.cz

ABSTRACT

Digital images have become a commodity which is searched on the Web as ordinarily as web pages. However, current large-scale engines search the images only on the basis of their annotations, while the content-based similarity systems do not seem to be ready for such scales. In this paper, we open the way to Web-scale image similarity search. We present a flexible system based on the metric space model and on the peer-to-peer paradigm. It uses M-Chord and M-Tree structures as its fundamental components and measures the image similarity by a combination of five MPEG-7 features. The system has been implemented including a graphical interface for online demonstrations and it currently indexes 10 million images crawled from the Web. We propose a novel strategy for approximate evaluation of similarity queries and we test its performance by a series of experiments. The results show that the system provides high-quality answers with response times around 0.5 second.

1. INTRODUCTION

The massive expansion of information technologies and the spontaneous growth of the World Wide Web have led to mass production and publishing of digital images. Currently, the images on the Web can be retrieved on the basis of their textual annotations via several large-scale search engines. Although the annotation-based searching is straightforward and suitable for many scenarios, there is a natural requirement for searching the images by their content. To the best of our knowledge, there is no technology for Web-scale content-based search which would allow a user to say: "Give me images from the whole Web which are similar to this photo."

In this paper, we introduce a very general, fully functional system for content-based similarity search on digital images which currently indexes 10 million pictures from the Web. Its distributed architecture adopts the principles of peer-to-peer (P2P) data structures which proved to be scalable and very flexible. Unlike majority of other approaches, our system is based purely on the metric space model [1]. Using this model is a two-edged sword: (1) It provides us with a unique generality and almost absolute freedom in defining and combin-

ing the similarity measures. (2) On the other hand, indexing techniques have to abandon classic schemas and the metric searching can be less efficient, especially on simple data domains. For complex data, like the features extracted from digital images, the classic techniques may loose their performance because of high dimensionality and may have problems with efficient indexing of very large collections. The relative efficiency of the metric-based approach grows with the volume and complexity of data and it seems to be a good solution for the application considered in this work.

The presented system uses the M-Chord [2] and the M-Tree [3], two efficient metric-based indexing and searching techniques born from a decade of intensive research. Even with their support, the precise evaluation of similarity queries may be expensive. In this paper, we seek efficient approximate strategies which would cut high fraction of the costs while keeping most of the quality of the answer. As our system is P2P based, we do the approximation on two levels – the query is navigated only to the most relevant peers and an approximate search is used on the peers' local data. The experiments prove that our approximate strategies are valuable.

Let us summarize the purpose and contribution of the paper and its further organization:

- In Section 2, we describe several related image-search systems and approaches.
- Section 3 mentions fundamentals of the presented system: indexing based on metric space and P2P paradigm.
- In Section 4, we describe particular components of a scalable system for image similarity search, namely, the P2P structure M-Chord, the dynamic metric index M-Tree, and a specific combination of MPEG-7 features. Section 4.3 comments on the prototype realization of the system.
- Section 5 provides a set of performance experiments which focus on the search approximation; the results prove the efficiency of our algorithms and their on-line response times. The paper is concluded in Section 6.

2. RELATED WORK

Large-scale searching and indexing methods for digital images have been the center of attention of many academic and commercial research groups. In the case of a retrieval based on annotations or other attributes associated with the images, large data collections can be managed efficiently because traditional well-established indexing and searching techniques can be applied. A number of systems use this approach to index Web-scale image collections, e.g. Google Images, Yahoo!, ExaLead¹, or PicSearch². Because these systems may suffer from a lack of trustworthy image annotations, there are attempts to increase the quality of image labeling via interactive public-domain games, e.g. The ESP Game³ or Google Image Labeler⁴. Another example of an attribute-based search application is a photo-searching by the geographic locations where the images were taken. This functionality is provided, e.g. by Flickr system⁵, which allows users to query about 30 million images in this way.

The content-based image retrieval is a an orthogonal approach where the images are searched according to their visual content similarity. Such techniques typically extract a characterization (signature) of the image content, which is then used for indexing and searching. Recent comprehensive surveys [4, 5] describe a number of approaches and their applications. These systems are usually specialized and tuned for a specific application domain and/or they manage relatively small data collections. The Tiltomo project⁶ indexes about 140,000 images from Flicker and searches them by color and texture characteristics; it also allows to combine it with "subject" searching. The ImBrowse⁷ system allows to search a collection of about 750,000 images by color, texture, shapes and combinations of these (employing five different search engines).

Another direction of research is an automatic image annotating based on the analysis of image content. System ALIPR [6] uses a training set of images which is separated into a number of concept categories; these categories are annotated. The indexed images are categorized and annotated by means of extracting visual features from them and by applying statistical methods. Searching is then based purely on the assigned annotations. The ALIPR has a public demo⁸; its internal functionality is provided by an older system SIMPLIcity [7].

A great research effort is still put into this area, let us mention at least activities integrated in the Chorus⁹ initiative. To the best of our knowledge, there is no technology for general image similarity search which would scale to data volumes comparable to the number of images covered by the Web (tens of millions to billions).

3. SYSTEM FUNDAMENTALS

Let us describe the fundamental ideas of our system: (1) indexing and searching based on the metric space abstraction and (2) distributed architecture based on the P2P paradigm.

3.1. Metric Space Approach

The metric space [1] is considered to be the most general data model for similarity search which can still be indexed and searched efficiently. The model treats the data as unstructured objects together with a function which measures proximity of object pairs. Formally, *metric space* \mathcal{M} is a pair $\mathcal{M} = (\mathcal{D}, d)$, where \mathcal{D} is the *domain* of objects and *d* is the total *distance function* $d : \mathcal{D} \times \mathcal{D} \longrightarrow \mathbb{R}$ satisfying the following postulates for all objects $x, y, z \in \mathcal{D}$:

$d(x,y) \ge 0$	(non-negativity),	
d(x,y) = 0 iff x = y	(identity),	
d(x,y) = d(y,x)	(symmetry),	
$d(x,z) \le d(x,y) + d(y,z)$	(triangle inequality).	

The semantics of this concept is: The smaller the distance between two objects, the more similar they are. The metric space is typically searched by queries which follow the query-by-example paradigm. A query is formed by an *object* $q \in D$ and some *constraint* on the data to be retrieved from the indexed dataset $I \subseteq D$. There are two basic types of these queries: (1) the *range query* R(q, r), which retrieves all objects $o \in I$ within the range r from q (i.e. $\{o|d(q, o) \leq r\}$), and (2) the *nearest neighbors query* kNN(q, k), which returns the k objects from I with the smallest distances to q.

Generality of Metric Space

A great advantage of the metric space model is its generality – there are no constraints on the data types, only the distance function is restricted by the metric postulates, which are quite natural. In an *n*-dimensional vector space, the basic family of metric distances are the *Minkowski Distances* (or L_p metrics); for an integer parameter *p*, they are defined as follows:

$$L_p[(x_1,...,x_n),(y_1,...,y_n)] = \left(\sum_{i=1}^n |x_i - y_i|^p\right)^{\frac{1}{p}}.$$

The L_2 metric is the classic Euclidean distance. There is a number of other metric functions for vectors, for instance, *Quadratic Form* [1] distance or the *Earth Mover's Distance* [8]. These measures express more complicated relationships and are important for, e.g., color histogram comparison. We can also define metric functions on non-vector data. A typical

¹http://www.exalead.com/

²http://www.picsearch.com/

³http://www.espgame.org/

⁴http://images.google.com/imagelabeler/

⁵http://www.flickr.com/map/

⁶http://www.tiltomo.com/

⁷http://media-vibrance.itn.liu.se/vinnova/cse.php

⁸http://www.alipr.com/

⁹http://www.ist-chorus.org/

way of measuring proximity of two strings is by *Edit distance* (Levenshtein distance). The *Jaccard's Coefficient* or *Hausdorff Distance* can be used to measure the dissimilarity of two sets. For more examples of metric functions see the recent books [1, 9].

Indexing and Searching

In general, the metric-searching problem can be defined as follows: Given a finite dataset $I \subseteq D$, preprocess or structure I so that similarity queries are answered efficiently. The indexing and searching techniques can only view the data as a metric space – the objects have no visible structure and the distance measure d is a "black-box" function.

The general purpose of any index or data structure is to *partition* the data space into segments so that not all of them have to be searched at query time. In the metric space, the partitioning can only be defined with the aid of some designated objects (pivots) from the domain \mathcal{D} . See specialized literature [1] for details on metric-based partitioning, filtering and searching mechanisms.

3.2. Structured Peer-to-Peer Networks

In the current era of digital explosion, the scalability of searching techniques is very important. In practice, it means that they should adapt to increasing data volumes and number of users without significant performance degradation. Since any centralized architecture is limited by the hardware infrastructure which cannot be boosted boundlessly, a natural solution is to shift towards distributed computing.

The *peer-to-peer (P2P) paradigm* was established in order to define fully decentralized, highly scalable and virtually limitless environment. The P2P network consists of *peers* – nodes participating in the network – which provide their resources to the network in exchange for the access to the network services. Every peer can contact any other peer directly, provided it knows the destination peers' identification.

To build a distributed index, a variant of the P2P paradigm called *structured peer-to-peer networks* can be used. Each peer manages a part of the overall dataset and maintains a navigation information which allows to route a query from any peer to the one with relevant data. In the case of metric similarity indices, there are usually several peers that hold the data which satisfies a query. Therefore, a query is routed to multiple peers which search their local data and compute partial answers. The originating peer then gathers all the partial answers and merges them into the total result of the query.

4. SYSTEM ARCHITECTURE AND SETTINGS

In this section, we describe the architecture and settings of the presented system for image similarity search. The data and the similarity measure are described in Section 4.1 and they form a metric space. We use the above-described P2P approach to build a very flexible distributed data structure; namely we use a system called M-Chord [2] to distribute the data among individual peers and to evaluate similarity queries. Each peer organizes the data locally in a metric-based index structure M-Tree [3]. Details about these structures are provided in Section 4.2. Section 4.3 says a few words about the system implementation and current configuration.

4.1. Measuring Image Similarity

In our system, the similarity of images is measured by virtue of several of MPEG-7 features [10] extracted from the images. As the individual features with their distance functions [11] form metric spaces, we can combine them into a single metric function by a weighted sum of the individual feature distances. The following table summarizes the MPEG-7 features we use, their respective distance measures, and their weights in the aggregate metric function.

MPEG-7 Feature	Metric	Weight
Scalable Color	L_1 metric	2
Color Structure	L_1 metric	3
Color Layout	sum of L_2 metrics	2
Edge Histogram	special [11]	4
Homogeneous Texture	special [11]	0.5

The weights have been determined experimentally trying to reflect human notion of similarity for general images. In this scenario, the weights are fixed but our technology also enables user-specified weights and thus tuning of the similarity measure for various image types [12]. The computation of the aggregate distance function takes approximately 0.01 ms on a standard CPU and the five-features image representation requires about 1 kB of memory (the features together form a vector of more than 280 dimensions).

4.2. Scalable Similarity Indexing

This section is devoted to the M-Chord and the M-Tree structures – the fundamental components of the introduced system.

4.2.1. M-Chord

The M-Chord [2] is a scalable distributed data structure for indexing and similarity searching in metric data. It is based on the concept of *structured P2P networks* introduced in Section 3.2. The system has been inspired by a centralized vector-based method called *iDistance* [13] – they both map the data space into a one-dimensional domain in a similar way. The M-Chord then applies one-dimensional P2P protocol, for instance Chord [14] or Skip Graphs [15], in order to divide the data among the peers and to provide navigation within the system. The M-Chord search algorithms navigate the similarity queries to the relevant peers.



Fig. 1. M-Chord mapping and search principles.

The M-Chord mapping, schematically depicted in Figure 1a, works basically as follows: Several reference points are selected from the sample dataset – we call these objects *pivots* and are denoted as p_i . The data space is partitioned in a Voronoi-like manner into *clusters* (C_i). Following the iDistance idea, the one-dimensional mapping of the data objects is defined according to their distances from the cluster's pivot. Having a separation constant c, the M-Chord key for an object $x \in C_i$ is calculated as

$$mchord(x) = d(p_i, x) + i \cdot c.$$

To evaluate a range query R(q, r), the data space to be searched is specified by *mchord* domain intervals in clusters which intersect the query region – see an example in Figure 1b.

Having the data space mapped into the *mchord* domain, every active node of the system takes over responsibility for an interval of keys. The range queries are routed to peers overlapping with the relevant intervals as seen in Figure 1b. Processing of a nearest neighbors query kNN(q, k) consists of two phases: (1) The query is evaluated locally on "the most promising peer(s)"; in this way, we obtain an upper bound r_k on the distance to the k^{th} nearest object from q. (2) Range query $R(q, r_k)$ is executed and the k nearest objects from the result are returned. There is a tradeoff between the costs of the first phase and the precision of the r_k estimation and, thus, cost of the second phase. We use an experimentally tuned settings which visits several peers in the first phase.

The experiments [2, 16] prove the *M*-Chord efficiency and scalability but the precise similarity searching is inherently expensive. The largest part of the costs is paid for the preciseness of the answer, which is not necessarily required in real applications. Therefore, we propose an approximate strategy of kNN queries evaluation. The algorithm is adjustable and the basic idea is to explore only the "most promising" parts of the "most promising" clusters. The distance between the query object q and the cluster pivot $d(q, p_i)$ is taken as a heuristic of the quality of the cluster, so the algorithm visits only a specified number of clusters in the order of this distance. Within a cluster C_i , the query is first navigated to a peer responsible for key $d(p_i, q) + i \cdot c$, which is the most promising spot on the *mchord* domain for cluster C_i . This peer searches its local data, returns the partial answer to the originator, and the query is sent to some number of its adjacent peers within the cluster. The efficiency of this algorithm and its variants is studied in Section 5.

4.2.2. M-Tree

The M-Tree [3] is a popular dynamic disk-oriented structure for metric data indexing. Similarly to B-Trees and R-Trees, it is a balanced tree built in a bottom-up fashion by splitting overfilled nodes. Each entry of the M-Tree internal node contains a pivot and a *covering radius* which specify a sphere-like region of the space covering by the entry and its sub-tree. The leaf nodes store data objects together with their distances to the pivot in the parent node. The internal nodes keep distances to the parent node's pivot as well. All these values are utilized in order to achieve pruning effect for the search algorithms [3].

The M-Tree can evaluate standard similarity queries – the range query and the kNN query. The latter can be processed also in an approximate fashion. The approximate algorithm follows the generic scheme [1]: It maintains a dynamic queue of M-Tree nodes sorted according to a heuristic which ensures that the "most promising" entries are processed first. If the currently processed entry is an internal node, its child nodes are re-inserted into the queue; the leaf nodes are processed according to standard kNN algorithm. The processing can be stopped at any time according to a predefined condition; in our implementation, the approximate search is stopped when a certain portion of data has been searched.

A number of M-Tree extensions have been published [1]. We implement the Pivoting M-Tree (PM-Tree) extension [17], which employs additional filtering and pruning by means of precomputed distances between the indexed objects and a fixed set of pivots. We use the same set of pivots as the M-Chord and thus the object-pivot distances are computed only once during the insert operation. The M-Tree is used as a local index at each peer of the M-Chord network. The overall system is schematically depicted in Figure 2.

4.3. Implementation and Settings

The presented system has a fully functional prototype implementation. We have exploited the Metric Similarity Search Implementation Framework (MESSIF) [18], which supports the implementation and integration of individual components of the system. The whole system is written in Java.

The dataset is composed of 10 million images (photos and other graphics) downloaded from a photo-sharing system Flickr¹⁰. The data is distributed among 500 M-Chord peers (each managing approximately 20,000 images in its local PM-Tree storage). The M-Chord uses twenty pivots and

¹⁰http://www.flickr.com/



Fig. 2. Schema of the distributed system.

shares these pivots with the PM-Trees. Although the PM-Tree is a disk-oriented structure, we currently try to keep all the data in main memory – all the peers require around 50 GB of memory including the Java overhead.

In the current configuration, the peers share a hardware infrastructure with sixteen CPUs and 64 GB RAM in total. Because each of the peers is a self-standing Java process and peers communicate via standard protocols from the IP family, migration of the system to practically any infrastructure is a straightforward process. The distributed structure indexes the extracted features together with a unique identifier of the corresponding image. The similarity queries then return these identifiers, so the actual images can be retrieved from an external source and displayed by the graphical user interface. An example of the interface is shown in Figure 3. The query objects are either taken from the system image collection or can be external images downloaded from the Web – in the latter case the features have to be extracted before the query can be processed.

5. PERFORMANCE EVALUATION

This section is devoted to the performance analysis of our system. We focus on the aspects important for the practical use of the system: the quality and costs of the approximate kNN queries. The experiments do not refer to the query throughput of the system but it directly depends on the number of CPUs the system uses.

5.1. Experiments Description

In the experiments, we focus on various settings of the M-Chord approximate algorithm for kNN queries. Let us recall that it has two free variables: number of clusters to visit and number of nodes to access within each cluster. We first vary



Fig. 3. Example of a result of kNN(q, 10) query.

these values and search for an optimal setting. Then we propose an adaptive algorithm to determine the number of clusters and we compare it with the optimal fixed setting deduced from the first two experiments.

We measure the quality of approximate results by *recall* and *error on position*. The recall is the fraction of the precise kNN answer which is retrieved by the approximate query. In this case, the recall is equal to the *precision*. The error on position [1] assesses the accuracy of the approximate search by measuring the discrepancy between the ordered list of answer objects returned by the approximate and the precise evaluations. Having an approximate kNN(q, k) answer and the set I of all objects indexed by the system, the relative error on position is measured as:

$$EP = \frac{\sum_{i=1}^{k} (I(o_i) - i)}{k \cdot |I|}$$

where $I(o_i)$ is the position of the *i*-th object from the approximate answer in the list of all objects form I ordered by their distance from object q.

The costs of the search are measured by the query *response time*, the *messages sent* during the query processing, the *number of peers* contacted by a particular query, and by the *number of computations* of the distance function d. If not specified otherwise, the results are for k = 30 and are taken as an average over 50 query objects selected from the indexed dataset.

5.2. Fixed Approximation: Number of Peers in Clusters

Data from every M-Chord cluster is spread among a certain number of adjacent peers. In the first experiment, we vary the number of peers involved in query processing within each cluster and try to find a suitable setting. The queries visit all clusters, that means 20, in our setting. As the clusters are not all of the same size, we specify the peers visited per cluster relatively as a "percentage of peers in a particular cluster". This requires that we estimate the number of peers in the cluster. Recall that the M-Chord pivots are selected from a given



Fig. 4. Approximate search: varying percentage of visited peers per cluster for all 20 clusters.



Fig. 5. Approximate search: varying number of visited clusters; number of peers visited per cluster fixed at 40 %.

sample set (before the system is built). In this phase, we can calculate the percentage distribution of the data in individual clusters. Having this knowledge and having the total number of peers in the system, each peer can estimate the number of peers in its cluster. The current number of peers in the system can be sent along with each query.

Figure 4 shows the results of this experiment. As each peer organizes its local data in an PM-Tree, which provides an approximate search strategy, we present all results for precise local search at every peer and also for approximate search with a 30% threshold (experimentally estimated to have the best performance/costs ratio). We can see that the recall, presented in graph (a), grows very fast up to some 40 % where it is about 90%. The strategy with local approximation exhibits the same trend and does not degrade the recall significantly. This is confirmed by the *error on position* in graph (b) which very soon gets near zero. Note that values presented in this graph are not divided by the size for the indexed dataset (10^7) . Let us find a word-interpretation of the error on position, e.g. for 40 % which is about value of 2: The approximated result misses some objects so that each object in the result is, on average, "shifted by two positions higher".

The average query response times are shown in graph (c). We can see that the values increase linearly for both variants of the local search, however, the 30% local approximation is significantly faster. The growth is caused by the fact that the 500-peers system shares 16 CPUs – note that visiting all clusters and 100% peers per cluster means that all peers actually

search their local data. The conclusion from this experiment is twofold: (1) The local approximation spares a significant portion of the costs while preserving high quality of the answer; (2) visiting about 40% of nodes in the cluster is a reasonable tradeoff between the gain on the quality and the costs.

5.3. Fixed Approximation: Number of Clusters

In the second experiment, we fix the percentage of peers accessed in each cluster at 40 % and we vary the number of clusters visited by a query. The results are shown in Figure 5 again for both the full and the approximated local search. We can see that visiting one cluster gives results with quite a low recall around 65 % and a rather high error on position. However, visiting only one additional cluster improves the results significantly. The quality does not further improve for more than five clusters. In correspondence with the previous experiment, the recall cannot get over 90 % because every query visits only 40 % of peers in each cluster.

The response times exhibit an expected behavior: They increase linearly with the number of clusters visited, since more peers need to search their local data and they compete for the 16 CPUs. We can see that visiting five clusters with the full local search takes around 2 seconds. This can be considered an on-line response with a very good accuracy: Almost 90% results of the precise answer shifted by no more than three positions are retrieved. By visiting five clusters with a local approximation, we reach only slightly worse quality with response times of less than one second.



Fig. 6. Fixed vs. adaptive approximate kNN(q, k) search for variable k.

5.4. Adaptive Approximation

In the previous experiments, we have searched for an average optimal setting for the approximation algorithm – number of clusters and peers accessed in each cluster. Now we define a heuristic that determines the number of clusters individually for every query and we compare this approach with the optimal fixed setting. The heuristic takes the clusters in the order of distances between their pivots and the actual query point $d(p_i, q)$. Cluster C_0 is always visited and then, in a loop, the next cluster C_{i+1} is marked to be visited if $d(p_{i+1}, q) \leq b \cdot d(p_i, q)$, where b is an experimentally-tuned constant. Value of b is, naturally, greater than one and it is gradually decreased as i grows. The value was determined so that queries visit about 1–7 clusters.

In the last experiment, we compare this adaptive algorithm with the algorithm which always visits five clusters. Both algorithms access 40% of peers in each cluster. We vary k, the number of nearest neighbors to be retrieved by kNN(q, k). The results for both precise and 30%-approximate local search are presented in Figure 6. If we compare the precise and the approximate variants separately, we can see that the recall and the error on position are almost identical for smaller k. The differences slightly grow up to 3% (in recall) and $5 \cdot 10^{-7}$ (in the relative error on position) for k = 100. This proves that the objects missed by the approximation are not within the most similar objects, which is important for the user. Even if we compare the fixed algorithm with full local search and the adaptive algorithm with approximate local search, the answer qualities are very close.

The average response times of the adaptive algorithm are roughly one half, compared to the fixed algorithm. This is due to the fact that the adaptive algorithm visited only 2.5 clusters on average (in fact, half of the queries hit only one cluster). We also report on the other – more detailed – cost measures. The number of messages interchanged between peers indicate the load of the underlying network. The pure CPU costs can be estimated by the number of computations of the distance function d, because this is by far the most expensive operation. The parallel distance computations then represent the maximal costs at individual peers. Table 1 summarizes these values (again, averaged over 50 queries) for the adaptive and the fixed approximation algorithms with k = 30 and with 30% local approximation. To establish a baseline, we show the costs of the precise kNN.

kNN	Adaptive	Fixed	Precise
Contacted peers	29.6	57.4	253.1
Total dist. comp.	166,885	326,418	1,731,418
Parallel dist. comp.	5,993	6,277	17,744
Messages sent	73	141	589
Response time (ms)	562	985	3,373

Table 1. Adaptive and fixed approximation and precise *kNN*.

We can see that the adaptive algorithm contacts practically half the number of peers compared with the fixed approximation. On this account, the adaptive algorithm accesses less data and the total number of distance computation is lower. The parallel costs are practically equal. Since the accessed peers compete for the shared pool of CPUs, the response times are nearly two times higher for the fixed approximation. Also the network is twice as much loaded. Note that at the navigation always needs at least two messages per peer whose local index is consulted (one to contact it and one to return the partial answer). In particular, the adaptive algorithm needs about 13 messages ($73 - 2 \cdot 29.6$) for additional query routing.

We can conclude that the adaptive algorithm with the local approximation exhibits superior response times of about 0.5 s and beats the fixed strategy also in all other cost measures. The quality of its answers is slightly worse but the degradation is tolerable and it is negligible for smaller values of k.

6. CONCLUSIONS AND FUTURE WORK

In this work, we have presented a general distributed system for content-based similarity search in large image collections. The system is based purely on the metric space model and it uses the P2P structure M-Chord and the dynamic metric index M-Tree. The similarity of the images is measured by a metric which combines five MPEG-7 features. The system has been implemented including a graphical interface for demonstrations and it currently indexes 10 million digital images downloaded from the Web. We have conducted a set of experiments to measure the performance of the system. We focused on the response times and the quality of our approximate search. The results prove that, comparing to the precise similarity search, our approach cuts major part of the costs while preserving a high quality of the answer. The search response of the system is online. The architecture of the system is very flexible and migrating to another hardware infrastructure is straightforward.

The scalability of the architecture was confirmed when we shifted from 1 million images to 10 million practically without any performance degradation. In the future, we plan to scale up to 100 million objects; at the moment we have preliminary experience with 50 million images. Other directions of our current and future work cover setting some theoretical guarantees on the approximation and improving the performance of both the precise and the approximation algorithms.

7. ACKNOWLEDGEMENTS

This research was supported by EU IST FP6 project 45128 (SAPIR), by national research project 1ET100300419, and by Czech Science Foundation projects No. 201/06/1338 and 201/08/P507. The images were crawled and the features extracted by ISTI-CNR¹¹, Pisa, Italy – a SAPIR project partner. We are also grateful to project MetaCenter¹² for providing the hardware infrastructure.

8. REFERENCES

- Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko, *Similarity Search: The Metric Space Approach*, vol. 32 of *Advances in Database Systems*, Springer-Verlag, 2006.
- [2] David Novak and Pavel Zezula, "M-Chord: A scalable distributed similarity search structure," in *Proceedings* of INFOSCALE '06, Hong Kong. 2006, ACM Press.
- [3] Paolo Ciaccia, Marco Patella, and Pavel Zezula, "M-Tree: An efficient access method for similarity search in metric spaces.," in *Proceedings of VLDB'97, August* 25–29, 1997, Athens, Greece, 1997, pp. 426–435.
- [4] Remco C. Veltkamp and Mirela Tanase, "Content-based image retrieval systems: A survey," Tech. Rep. UU-CS-2000-34, Department of CS, Utrecht University, 2002.
- [5] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z. Wang, "Image retrieval: Ideas, influences, and trends of the new age," ACM Computing Surveys, 2008, To appear.
- [6] Jia Li and James Z. Wang, "Real-time computerized annotation of pictures," in *Proceedings of MULTIMEDIA* '06, New York, 2006, pp. 911–920, ACM Press.

- [7] James Z. Wang, Jia Li, and Gio Wiederhold, "SIMPLIcity: Semantics-sensitive integrated matching for picture libraries," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 9, pp. 947–963, 2001.
- [8] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas, "A metric for distributions with applications to image databases," in *Proceedings of ICCV '98*, Washington, DC, 1998, pp. 59–67, IEEE Computer Society.
- [9] Hanan Samet, Foundations of Multidimensional and Metric Data Structures, Computer Graphics and Geometric Modeling. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.
- [10] MPEG-7, "Multimedia content description interfaces. Part 3: Visual," ISO/IEC 15938-3:2002, 2002.
- [11] B.S. Manjunath, Phillipe Salembier, and Thomas Sikora, Eds., Introduction to MPEG-7: Multimedia Content Description Interface, John Wiley & Sons, Inc., New York, NY, USA, 2002.
- [12] Michal Batko, Petra Kohoutková, and Pavel Zezula, "Combining metric features in large collections," in *Proceedings of SISAP '08*, 2008, To appear.
- [13] 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 TODS 2005, vol. 30, no. 2, pp. 364–397, 2005.
- [14] Ion Stoica, Robert Morris, David R. Karger, Frans M. Kaashoek, and Hari Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of SIGCOMM 2001, San Diego, CA, August 27–31, 2001.* 2001, pp. 149–160, ACM Press.
- [15] James Aspnes and Gauri Shah, "Skip graphs," in Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2003, pp. 384–393.
- [16] Michal Batko, David Novak, Fabrizio Falchi, and Pavel Zezula, "On scalability of the similarity search in the world of peers," in *Proceedings of INFOSCALE 2006*, *Hong Kong*, New York, 2006, pp. 1–12, ACM Press.
- [17] Tomás Skopal, Jaroslav Pokorný, and Václav Snášel, "PM-tree: Pivoting metric tree for similarity search in multimedia databases," in *Proceedings of ADBIS, Budapest, Hungary*, 2004.
- [18] Michal Batko, David Novak, and Pavel Zezula, First International DELOS Conference, Pisa, Italy, February 13-14, 2007, Revised Selected Papers, vol. 4877 of LNCS, chapter MESSIF: Metric Similarity Search Implementation Framework, pp. 1–10, Springer, 2007.

¹¹ http://www.isti.cnr.it/

¹²http://meta.cesnet.cz/