

## CrossMedia: supporting collaborative research of media retrieval

László Havasi, Péter Mátételki, Márton Gergő, András Micsik,  
Tamás Szirányi, László Kovács

MTA SZTAKI, Computer and Automation Research Institute of the Hungarian Academy of Sciences,  
H-1111 Budapest XI. Lágymányosi u. 11. Hungary  
{laszlo.havasi, peter.matetelki, marton.gergo, andras.micsik, tamas.sziranyi, laszlo.kovacs}@sztaki.hu

**Abstract.** The goal of our new e-science platform is to support collaborative research communities by providing a simple solution to jointly develop semantic- and media search algorithms on common and challenging datasets processed by novel feature extractors. Querying of nearest neighbor (NN) elements on large data collections is an important task for several information or content retrieval tasks. In the paper a flexible framework for research purposes is introduced for testing features, metrics, distances and indexing structures. The core part of the content based retrieval system is the LHI-tree, a disk-based index scheme for fast retrieval of multimodal features. Additionally, we compare LHI-tree to FLANN, an effective implementation of ANN search and show that LHI-tree gives similar list of retrieved images.

**Keywords:** multimedia indexing, hierarchical data structure, information retrieval, portal

### 1 Introduction

Multimedia information systems are becoming increasingly important with the advent of multi-sensor networks, mobile phone data capture and increasing number of multimedia databases. Since visual, auditory media and the adherent information requires large amounts of memory and computing power for storage and processing, there is a need to efficiently index, store, and retrieve the visual information from multimedia/cross-media databases [1].

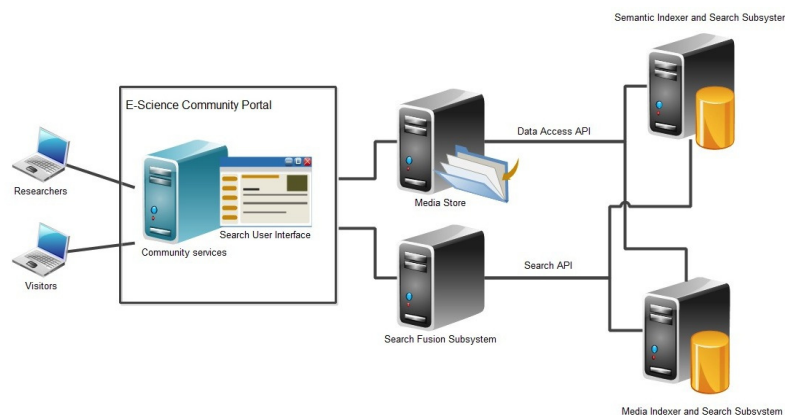
Experiments based on the above system are limited by several real life scenarios:

- How to process the extremely increased information quantity? The sophisticated processing and index building methods need significantly more time than real-time. The retrieval system will not be able to follow the incoming data flow.
- How to insert novel features? Several features based on heuristics, thus there is no fixed dimensional (vectorial) form.

- How to update database with data from different modalities? Real life sensor networks or multimedia contents built for several modalities need various feature extraction and indexing methods.

CrossMedia portal presents an all-in-one solution for such problems: storage and processing capacity, flexible interfaces, built in index structure and innovative user interfaces.

Users form communities on the CrossMedia portal where they can jointly create, build and share search algorithms and media datasets in an iterative way. The system automatically builds indexes and also provides a testing facility as indexes are instantly included in the portal's search interface to be tested with any desired media-based, semantic, or combined multimodal input. The generated indexes can be shared among research communities for reviewing or with the public for demonstration purposes.



**Fig. 1.** Architecture of CrossMedia e-Science platform

## 2 CROSSMEDIA E-SCIENCE PLATFORM

The CrossMedia system (see Figure 1) addresses many diverse tasks, all which are made accessible for the users through the portal that serves as an access point for all available services. Behind the portal we set up a distributed system consisting of multiple processing units organized in a loosely coupled service-oriented architecture. The system consists of the following subsystems:

- The Media Store (MS) responsible for safekeeping all searchable multimedia elements.
- The Media Indexer and Search Subsystem (MISS) responsible for generating index trees for a specific algorithm on a specific media source in the MS and

capable of executing similarity-based search on the generated index trees for any input media.

- The Semantic Indexer and Search Subsystem (SISS) responsible for creating semantic databases and indexes by processing media annotations of the MS entities and capable of executing semantic search on the built dataset.
- The Search Fusion Subsystem (SFS) responsible for combining the results of the MISS and SISS in case of multi-input multimodal search expressions.
- The Search User Interface (SUI) enabling users to easily create complex multimodal search expressions and to see and evaluate search results.
- The E-Science Community Portal (ECP) responsible for integrating and providing all the above functionalities through a stylish modern web interface for enabling users to create and jointly work in research communities and support the collaborative work.

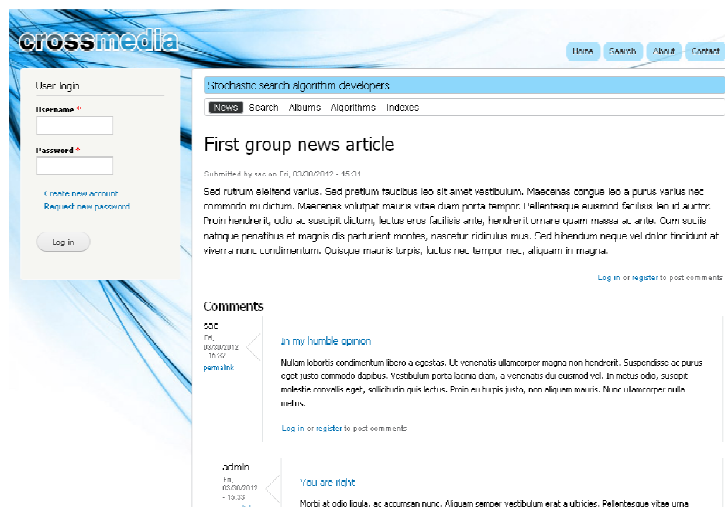


Fig. 2. Community page in the portal

## 2.1 Managing datasets

In this area typically large sample datasets are needed for testing of search algorithms – for example we used a database containing about 5 million entities when testing our custom developed image search algorithms. As the platform aims to enable its users to collaboratively build and maintain these large datasets we had to find a sustainable solution. After evaluating many currently available solutions we realized that the shared data and permission management is far from trivial and we are in need of defining and developing a custom architecture to suit our special needs.

Our system architecture separates the portal's community management and the data management into detached, loosely coupled components: the E-Science Community Portal is in charge of all group- and community management tasks, and the Media Store is a simple but very responsive data storage, manager and retrieval component. This separation keeps the community-generated large datasets and the community management independent thus enabling us to flexibly redesign the frontend system component if necessary. The solution requires considerably less resources on the portal side as representing the large datasets only needs a low quantity of data to be synchronized, so the portal remains scalable.

Apart from simple data storage the Media Store component is the one connecting community generated (portal) content to the media processing components: the Semantic- and Media Indexer and Search Subsystems. The communication is achieved using REST API.

## **2.2 Protocol for instant multimodal search**

The CrossMedia Search User Interface component can handle search inputs of different (semantic and content-based) modalities even in a single query expression as we use a uniform query input representation consisting of the search index (either semantic or content-based), the search target (text data or binary media) and a weight parameter. This generic search-input definition not only enables to easily expand the search modality coverage in the future but also supports the composition of multi-input and multimodal search queries. The user-generated complex multimodal search requests are handled by the Search Fusion Subsystem. It evaluates the search expressions and divides them into single terms, forwards the individual terms to the appropriate search subsystem and aggregates the separate results into one unified result list that is sent to the Search User Interface as the final response. The protocol and also the SFS are able to generate partial (not final) results, so the user gets an instant search experience: when adding search inputs the preliminary result list is immediately displayed. As soon as the user composes a more complex search expression or refines an existing expression the result list immediately reflects the current state of the search parameters.

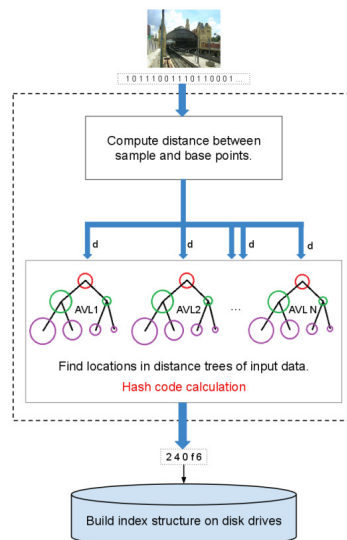
## **3 DISK BASED INDEXING**

When the dimension of feature set increasing dramatically, the methods used for low-dimensional indexing are not applicable more. The classical kd-tree algorithm [2] is efficient in low dimensions, but its performance degrades by higher.

[3] uses a multiple randomized kd-tree, where it splits the data for a randomly chosen set of dimensions (e.g. 5) instead of that of the greatest variance. In [4] this random kd tree is applied by using multiple trees, priority search on hierarchical k-means trees, and automatic parameter setting. They have demonstrated that this configuration of algorithms can speed the matching of high-dimensional vectors by up to several orders of magnitude compared to linear search.

Recent method proposed in [5] the Nearest Vector Tree which is designed for approximate nearest neighbor search in very large, high-dimensional databases. It transforms the high dimensionality search task into an efficient one dimensional space based on the combination of projections of data points to lines and the partitioning of the projected space. 150.000 images have been indexed by the NV-Tree.

The LHI-tree is similar to M-index where base points (so called pivots) are chosen randomly to reduce the high-dimensional feature vectors. A modification of this random selection is applied, where a quasi orthogonality criteria is forced during random point selection. Beyond the point selection we estimate basic statistical properties of input space from the representative sample set.



**Fig. 3.** Flow-chart of LHI-tree building algorithm. The embedded AVL trees give a fast way to prepare hash code from input data. Next, the hash code is translated into directory structure on disks.

In contrast to permutation-based scheme, LHI-tree uses base points to compute reference distances and calculate has code for every input vector from the quantized distances. It is carried out by using AVL-trees inside the LHI-tree, connected to every base point. Input of AVL-trees are the distances of the input image to base points, while the outputs are the number of bin in which the quantized distance fell. Visually, LHI-tree contains base points as hyper-sphere centers in feature space. The indices from AVL-trees are the shells of such spheres with different radius. To assign a disk partition to a part of the feature space, we have used hashing function of quantized distances. This hash function guarantees that the near vectors are placed into the same disk partition (file). Figure 3 demonstrates the building process step-by-step.

### 3.1 Experimental results

During evaluation process two datasets were used:

- CoPhIR: color structure, edge histogram and homogenous texture descriptors were selected. The applied distance definitions are the Euclidean [4] and pattern difference [6].
- Caltech 101: SIFT vectors. The applied distance definition is the Euclidean.

Table 1 summarizes the performance parameters of some building and retrieval trials.


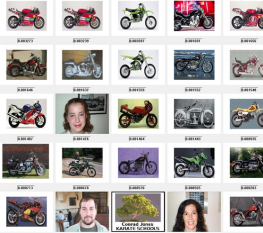
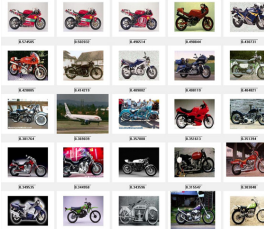

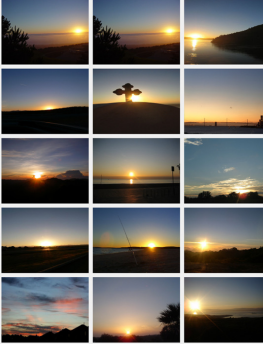
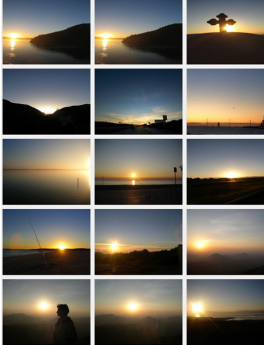
**Table 1.** Run-time parameters of the LHI-tree index. Number of base spheres/number of input points/dimension of input vector.

	3sphs/300k 64dims	5sphs/600k 64dims	5sphs/829k 80dims	5sphs/63 2k 128dims
Build time (sec)	456	12194	17619	14901
RAM storage (MB)	0.912	1.248	1.432	0.945
HDD storage (MB)	80200	161000	215000	329210
Search time (msec)	67	45	110	160
RAM usage during search (MB)	1.2	0.97	1.2	2.2

The data is stored in uncompressed format. Search times contain all additional operation such as extraction, db operations etc.

## 4 USING THE PORTAL

The CrossMedia E-Science Portal aims to satisfy the domain-specific needs of researchers primarily in the image processing scientific field. Registered users can be engaged in different research groups by creating a new society or by joining the desired groups. Available functionality in a group not only satisfies the specific needs of researchers but naturally offers other community-based collaboration facilities such as discussion forums, commenting and activity monitoring.

SIFT features without dimension reduction, pattern difference for LHI-Tree and Euclidean metric for FLANN	
	Specific object category without background content. Total 630 descriptors in the image.
FLANN	LHI-tree
	
Color structure descriptor, pattern difference for LHI-tree and Euclidean metric for FLANN	
	Descriptor length is 64 dims.
FLANN	LHI-tree
	

**Fig. 4.** Qualitative results of LHI-tree compared to FLANN.

Every group has its own separate private space in the portal where group members share and collaboratively work with group content. Media collections (so-called albums) can be created to store sample media for the index trees to be built. Category-based and free text annotations can be attached to any media item in the albums thus enabling the Semantic Indexer and Search Subsystem to perform semantic search on

these media items. Groups can also upload their developed media indexer algorithms to the portal that can later be used to create indexes. Image indexes logically consist of exactly one algorithm and may include multiple media collections. Defining an index on the portal interface launches a series of automated operations that produce the desired index structure in the backend and finally results in making the generated index available for building search requests on the Search User Interface.

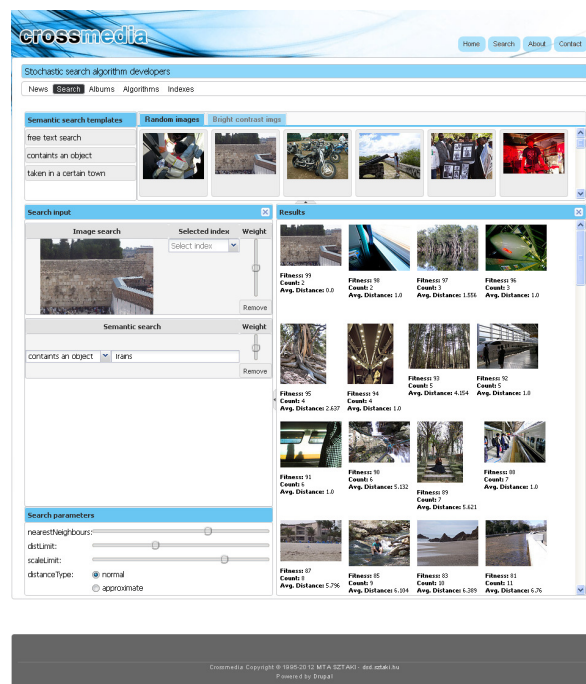


Fig. 5. Search User Interface.

Index generation tasks execute when creating a new index for an algorithm on a collection, when adding a new collection to an existing index or when modifying collections that are already indexed. In either case similar backend operations occur: the Media Store receives the index modification request and creates the atomic jobs that basically consist of an algorithm and a target media item to be processed by that algorithm. These jobs can be polled from the Media Store job queue and processed asynchronously by the indexer subsystems. Especially for large collections this may be a weary and relatively slow task so its results are fed back to the portal regularly to keep users updated of the current index processing state at all times. This mechanism keeps indexes up-to-date with the corresponding media collections so researchers or even anonymous users are provided with the most recent versions at all times.



Every piece of group content, including media collections, algorithms and indexes can be (separately) shared at two different levels: either with other group(s) or with the public. This creates a wide variety of possibilities for inter-group collaboration and for demonstration purposes as well, e.g. setting public visibility on an index tree makes it available for every visitor of the site. This could be used to show off the capabilities of a research group as the index (including its description that explains how the corresponding search algorithm works) becomes available for any user and can be queried with a user-selected media item. A group can also share collections with other groups so they can test their (private) algorithms and indexes on a common dataset. A trickier scenario can also be imagined in case a group shares one of its indexes but keeps the corresponding collections and algorithms private. This enables other groups to use their shared index to search in their index tree resulting in media items that originate from the private album but in the meantime this setup will not let other groups to use either the corresponding algorithm to index their own collections or the corresponding collection to be indexed with other algorithms.

For testing image descriptor and semantic indexes the portal provides an intuitive Search User Interface (see Figure 5) for every group and also one public SUI for collections and indexes that are published site-wide. Group SUIs allow group members to assemble complex, multipart search queries. Every part of a query consists of an input media item chosen from the accessible collections, an index and a weight parameter. An index can either be image content-based or semantic. Users have the possibility to create multi-input queries using just one (content-based or semantic) index type or they even can compose multi-input and multimodal queries where both content-based and semantic indexes are involved. The Search Fusion Subsystem generates the unified result lists using internal weighting mechanisms. Users have the ability to tune the weighting by providing positive or negative feedback on a result item regarding their opinion on moving that item forward or backward in the result list. The SUI is implemented as a platform- and browser independent web-application using the Sencha platform based on JavaScript and AJAX technologies.

## 5 CONCLUSIONS

We have introduced CrossMedia e-Science Community Portal where different kind of features and distance definitions can be integrated into the disk-based indexing scheme (LHI-tree). LHI-tree is the initial indexing structure of the portal. The goal of the portal is to provide a flexible framework for research purposes on multimodal database mining and retrieval tasks. We showed that the proposed retrieval engine could achieve acceptable speed for searching with very low memory and CPU loads. The test results were validated with subjective comparison to results of FLANN.

By using the CrossMedia e-Science Community Portal researchers gain the possibility to jointly work in groups as well as collaborating with other research communities. The domain-specific functionality of the portal enables them to easily produce testable results in the field of image processing and to compare or reuse each other's results.

## References

- 1 P. Geetha , Vasumathi Narayanan: A Survey of Content-Based Video Retrieval (2008).
- 2 J.H. Friedman, J.L. Bentley, and R.A. Finkel. An algorithm for finding best matches in logarithmic expected time. ACM Transactions on Mathematical Software, 3(3):209-226, September 1977.
- 3 C. Silpa-Anan, R. Hartley, Optimised KD-trees for fast image descriptor matching, in CVPR 2008.
- 4 M. Muja and D. G. Lowe, Fast approximate nearest neighbors with automatic algorithm configuration, in VISAPP 2009.
- 5 Lejsek, H.; Asmundsson, F.H.; Jonsson, B.T.; Amsaleg, L. : NV-Tree: An Efficient Disk-Based Index for Approximate Search in Very Large High-Dimensional Collections, Pattern Analysis and Machine Intelligence, IEEE Transactions on 31, (5), May 2009, 869 - 883.
- 6 Horst Eidenberger, Distance measures for MPEG-7-based retrieval, in MIR 2003