View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by SZTAKI Publication Repository

Search in WikiImages using mobile phone

László Havasi, Mihály Szabó, Máté Pataki, Domonkos Varga, 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, mihaly.szabo, mate.pataki, domonkos.varga, tamas.sziranyi, laszlo.kovacs}@sztaki.hu

***Abstract*— Demonstration will focus on the content based retrieval of Wikipedia images (Hungarian version). A mobile application for iOS will be used to gather images and send directly to the crossmodal processing framework. Searching is implemented in a high performance hybrid index tree with total ~500k entries. The hit list is converted to wikipages and ordered by the content based score.**

Keywords*— *content based retrieval, mobile applications, hierarchical tree

I. 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.
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.

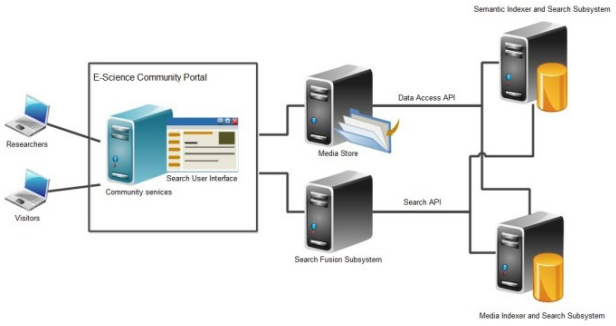

Fig. 1. Architecture of CrossMedia e-Science platform

Three demo applications for image processing were built using CrossMedia. The first is an image search application which, after searching for similar images by using the image descriptors enriches the results using the semantic text annotations.

The second application was written for iPhone. The user can take a photo with the built-in camera and similar images and, if it is found, the original source of the image are displayed. The database used for this application is the Hungarian Wikipedia, which has around 204500 images.

This application can be used to search a big database for the source of a given image, but the same mechanism can also be used to display additional information to an image, a billboard, advertisement or even a building. With the widespread use of digital photography, the amount of images stored even in one person's computer is extremely large. This application can also be used to search for images containing an object or motif, thus finding the date and context of a previously taken photo.

The third application enables us to use the CrossMedia platform for image plagiarism search. In the demo version one can upload documents to the system, from which the images are extracted and compared against the Wikipedia database. If

978-1-4799-0956-8/13/$31.00 ©2013 IEEE

the image was not changed significantly the system can tell with very high confidence where it was copied from.

MTA SZTAKI operates a plagiarism search service, KOPI, which is able to detect textual similarities between documents. Extending this service with image search makes this service more versatile, more valuable. Image plagiarising is a big problem for publishers, mainly in technical fields, as many articles, theses and even books contain non licenced copyrighted material. On the other hand, the web is full with plagiarised copyrighted contents from newspapers, photographers, painters and other visual artists, one can find them on blogs, personal homepages but sometimes also on company websites and even smaller newspapers. These images could be easily detected with this algorithm and the content owner can be noticed to take appropriate actions.

II. APPLIED MOBILE TECHNOLOGY

A mobile frontend application for iOS is prepared for CrossMedia to find images in the Hungarian Wikipedia. It is capable of taking photos or choosing existing ones from an album on the device and sending them to the CrossMedia backend server. After this, it accepts the reply and shows the result set to the user. The application preprocesses the selected photo; first it resizes the photo to 1080 pixels by p pixels, where the longer side will be resized to 1080 pixels and the shorter one will be resized accordingly (p). A JPEG compression and a Base64 encoding are applied before sending the image to the CrossMedia backend.

The user can fine tune some query parameters in the application, which helps her in testing the query performance. These parameters are also sent to the backend with the image:

- Index (3-5): tells the backend which index tree it should use for the query,
- DescCount (10-300): scanning resolution,
- MaxDist (0-100): sets the maximum particle distance,
- NNCount (5-50): maximum size of the result set,
- CompressionRatio (100-50): JPEG compression ratio.

The application and the backend communicate via REST API using JSON. After the application initiated the query by sending the image and additional data to the server, it replies with a status JSON structure. This structure contains a URL where the actual status of the given query is available, a version number, and the status itself with “wait” and “finished” values. The “wait” value tells the application the queue position of the request and the “finished” value tells the application that the result is ready to dispatch. When the status is changed to “finished”, the JSON also contains a result set with the title and URL of the Wikipedia page where the relevant image was found and the score of the match. This result set is displayed to the user as a list of the titles with stars according to the score level. The golden star means excellent relevance, the half golden, half silver star means good relevance, the silver star means medium relevance and the results with no stars have negligible relevance. When the user taps on a title, the referred Wikipedia page opens up.

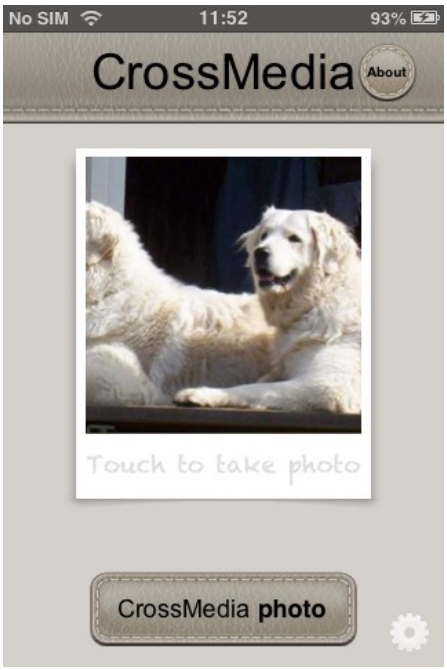

Fig. 2. iPhone application screenshot

III. 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.

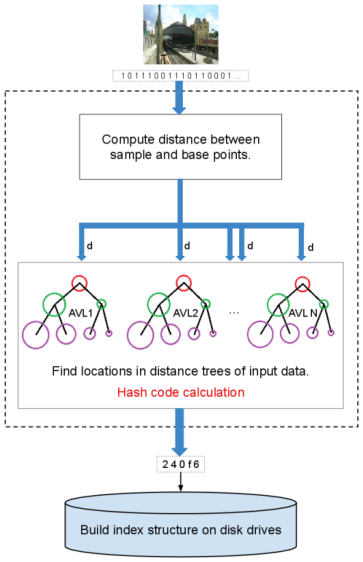

Fig. 3. Flow-chart of LHI-tree building algorithm. The embedded AVL trees give a fast way to prepare hash code from input data. Next, the hash code is translated into directory structure on disks.

In contrast to permutation-based scheme, LHI-tree uses base points to compute reference distances and to calculate hash codes 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.

The applied region based descriptor consists of four parts:

1. The edge histogram: We do the calculation of the gradient vectors on the original picture. We calculate the magnitude and the orientation of the gradient vectors in each pixel. We divide the range between minus 180 degrees and plus180 degrees into twelve equal parts. We take the gradient vectors one after the other and we examine the orientation. We increase the suitable element of the block with the absolute value of the gradient vector.
2. The entropy histogram: At first we create the entropy map of the preprocessed image. We take a square from the preprocessed image. We calculate the Renyi-entropy of this square. Then we subsitute this value to the entropy map to the place of the center pixel of the square. The entropy histogram is calculated in the following manner: the circle of the region is into eight parts divided. We determine the median value of the entropy map in each regions. This gives a 8-bin histogram.
3. The pattern histogram: Local binary pattern is a type feature used for classification. The LBP feature vector is created in the following manner: for each pixel in a cell, compare the pixel to each of its eight neighbors. Follow the pixels along a circle clockwise. Where the center pixel's value is greater than the neighbor's value write „1". Otherwise, write „0". This gives an 8-digit binary number. Compute the histogram, over the cell, of the frequency of each „number" occurring. Concatenate normalized histograms of all cells.
4. The dominant component of the color content: After transforming the picture into HSV color space, we divide the region into four parts. Then we calculate the median of the hue values on all sectors.

In summary the total number of dimension is 32. Based on our experiments the Euclidean distance is a good choice for similarity measure.

IV. EXPERIMENTAL RESULTS

The problem with the Wikipedia database is that the images are very different in topic, quality and size. Topics are very diverse, but each topic is represented with a few images only. The same search algorithm worked more reliably with a database containing few topics with hundreds of images each.

The system with the current image descriptors is capable of finding the most similar couple of images to a given example. In its current state it is perfect for finding the source of an image, for example finding the source of an image used in an article, or finding the original of a family photo on a disc by taking a snapshot of a printed version.

Based on the validated images the built tree locates 71 GBytes in 1.420.277 files on the SSD (Solid-State Drive) for media indices. The memory usage is about 1.5 GByte for the tree structure. Index building procedure finished in 8.2 hours.

An additional database was built for translating media identifiers to wikipage identifiers. This translation is necessary to determine the valid URLs and titles for the result list (table contains 1996158 records for the Hungarian pages).

Following table summarizes the performance of retrieval engine. Computational times depend on the number of region based descriptors (marked with Count column in the table) to be sent to the search engine.

TABLE I. COMPUTATION TIMES IN RETRIEVAL PROCESS

Count	**Retrieval process**		
	Feature extraction	***Similarity search***	***Ranking and translating IDs***
80	1400 msec	434 msec	600 msec
250	1400 msec	1425 msec	1700 msec
450	1400 msec	3150 msec	1880 msec

V. CONCLUSIONS

We have introduced the mobile extension of CrossMedia Portal to search similar images in Hungarian Wikipedia content. Building an index for this image dataset is challenging, because of the limited number of similar images related to the same topic. Our experiment validates that the CrossMedia retrieval engine is useful for such indexing problems as well.

Our future work contains a better optimized image processing module (for descriptor extraction) and a reduced size index structure.

Currently we are working on the index for the whole image set of Wikipedia.

REFERENCES

[1] P. Geetha , Vasumathi Narayanan: A Survey of Content-Based Video Retrieval. Journal of Computer Science, (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.