LHI-TREE: AN EFFICIENT DISK-BASED IMAGE SEARCH APPLICATION

László Havasi, Domonkos Varga, Tamás Szirányi

MTA SZTAKI, Computer and Automation Research Institute of the Hungarian Academy of Sciences

ABSTRACT

Querying of nearest neighbour (NN) elements on large data collections is an important task for several information or content retrieval tasks. In the paper Local Hash-indexing tree (LHI-tree) is introduced, which is a disk-based index scheme that uses RAM for quick space partition localization and hard disks for the hash indexing. When large collections are considered, such hybrid data structure should be used to implement an effective indexing service. The proposed structure can produce list of approximate nearest neighbours. We compare LHI-tree to FLANN (Fast Library for Approximate Nearest Neighbors), an effective implementation of ANN search. We show that they produce similar lists of retrieved images (although FLANN works only on RAM). In case of huge multimedia database a disk based indexing and retrieval method has a significant advantage against a vector based system running in RAM data. For the visual content indexing we built an image descriptor composed of four different information representations: edge histogram, entropy histogram, pattern histogram and dominant component colour characteristics. The paper will mention the content based retrieval of Hungarian Wikipedia images.

***Index Terms*—** multimedia indexing, content based retrieval, high dimensional data searching, disk-based tree, hierarchical tree, image descriptor

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].

Today image retrieval even with 25.000.000 pictures based on image and textual features is a very difficult task, and retrieval times above a few seconds are not acceptable. In the video retrieval area, current retrieval challenges are raised with e. g. given 50GB/200 hours of video data. As to other multimedia retrieval problems, e. g. 3D object content based retrieval is presumably a new field of investigation with interesting future applications. On this kind of information the 3D surface representation, local and global similarity measures, effective indexing are challenging problems.

The general approach to build such multimedia databases comprises the following, or similar steps. The first step is feature selection and extraction. Then comes the preprocessing with dimension reduction and/or sophisticated clustering of feature space. After that some tree-like or hashing-based data structure is selected and built for storage. At last we connect relevance feedback to query interface. Further limitation of semantics based retrieval systems in the missing capability of generalization. These systems need to rebuild feature clusters and indexes when new features, information streams or objects are added.

2. STATE OF THE ART

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

[3] uses a multiple randomized kd-tree, where it splits the data for a randomly chosen set of dimensions (e. g. 5) instead 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 up the matching of high-dimensional vectors by up to several orders of magnitude compared to linear search.

Many graph mining problems revolve around the computation of the most similar k-nodes to a given query node. The similarity measures based on random walks on graphs may result in low speed and memory allocation problems. The obvious solution [5] for the problem of computing similarities at query time when the graph is too large to be memory resident is to split the graph into clusters of nodes and store each cluster on a disk page; ideally random walks will rarely cross cluster boundaries and cause page-faults. Another speedup can be achieved by using only sequential sweeps over data files. However, in this solution the vertices have the same feature dimension and the similarity between the query node and others is examined by local neighbourhoods around the query node instead of a distance, relying on dynamic programming or power iteration like techniques which involve updating the neighbourhood.

The clustering approach is applied in [6] as well. It examines a cluster CBIR system using self-organizing tree map approach, with relevance feedback using radial-basis function network. Computational cost is reduced by adopting clustering CBIR over the conventional centralized CBIR approach. The proposed relevance feedback technique resulted in a higher retrieval precision for the clustered CBIR systems.

Recent method proposed in [7] is the Nearest Vector Tree which is designed for approximate nearest neighbour 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.

In our previous demo work [8] we have demonstrated that LHI-tree performs well in image search applications. A mobile frontend application for iOS was prepared for CrossMedia to find images in the Hungarian Wikipedia. We compare the results in this article with other methods.

3. DISK BASED INDEXING

The LHI-tree is similar to M-index [9] 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.

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 the base points, while the outputs are the number of bin in which the quantized distance fell. The use of AVL-tree for such quantization task is favourable because it is balanced and very fast. Visually, LHI-tree contains base points as hyper-sphere centers in the feature space. The indices from AVL-trees are the shells of such spheres with different radius.

Thus, parameters of the LHI-tree are:

- Number of spheres: It defines the number of base points, namely the number of AVL-trees. This parameter controls the non-linear reduction, assigns new dimensionality to the high dimension feature space.

- Number of shells: It defines the bin count for distance quantization. A non-linear quantization was selected. This parameter controls the number of intersections through the complexity of the generated hash code.

Table 1. Run-time parameters of the proposed indexing method. Number of base spheres/number of input points/dimension of input vector.

	3sphs/300k 64 dims	5sphs/600k 64 dims	5sphs/829k 64 dims	5sphs/632k 64 dims
Build time	456 sec	12194 sec	17619 sec	14901 sec
RAM storage	0.912 MB	1.248 MB	1.432 MB	0.945 MB
HDD storage	80200 MB	161000 MB	215000 MB	329210 MB
Search time	67 msec	45 msec	110 msec	160 msec
RAM usage during search	1.2 MB	0.97 MB	1.2 MB	2.2 MB

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 close vectors are placed into the same disk partition (file). Figure 1 demonstrates the building process step-by-step.

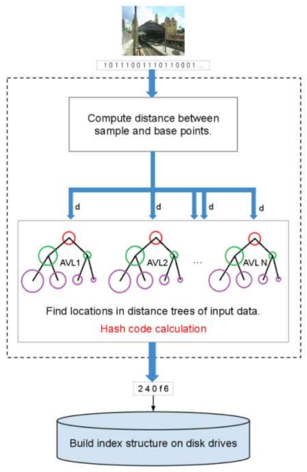

1011110011101110001...
Compute distance between sample and base points.
d
d
d
d
AVL1
AVL2
...
AVL N
Find locations in distance trees of input data.
Hash code calculation
240f6
Build index structure on disk drives

Fig. 1. Flow-chart of LHI-tree building algorithm. The embedded AVL trees give a fast solution to prepare hash code from input data. Next the hash code is translated into directory structure on disks.

During evaluation process two datasets were used:

- CoPhIR (Content-based Photo Image Retrieval): colour structure, edge histogram and homogeneous texture descriptors were selected.

- Caltech101 [10]: SIFT vectors.

CoPhIR dataset contains MPEG7 descriptors in XML format, while we extracted SIFT region descriptors on images from CALTECH. For building indices and searching a desktop Intel Core7 computer was used with standard speed HDD storage.

4. THE REGION BASED DESCRIPTORS

The goal of the preprocessing is to preserve the substantive content of the image, first of all the dominant edges and the contours. After this the selecting of the regions will be possible. From among the simply image transformations the median filter is a practical choice. The localization of the descriptor serves the determination of two basic parameters. The first is the center of the operative region of the descriptor, the second is the radius of the descriptor. These two parameters determine together the image content used in the computation of the descriptor. In contrast to the previously used descriptors, the goal is a description of the parts of the shapes. Therefore the goal is not the determination of the corner points and the coverage of their environment, but the finding the contour defined, nearly homogeneous areas of the image. Also the location of the descriptors can be divided into two phases:

1. The determination of the position: is done based on the localization of the edges. Nearly the sides of the descriptors the same direction, arranged edges must be rewarded. In the middle of the descriptors the presence of the edges is also punished. The degree of the disordering of the edges is determined by Renyi entropy.

2. The determination of the radius: It is determined with the help of the information content by the radius bounded area.

Our applied region based descriptor vectors will each consist of four individual feature vectors representing the edges, the entropy, the texture and the image colour:

1. The edge histogram: The magnitude and the orientation of the gradient vectors is calculated in each pixel. The range between minus 180 degrees and plus 180 degrees is divided into twelve equal bins. We take the gradient vectors one after the other, examine the orientation and the bin of the suitable element will be increased by the absolute value of the gradient vector. It is invariant to translation and rotation of the images and normalizing the histogram leads to scale invariance.

2. The entropy histogram: This type of descriptor is used for the analysis of the grey level inhomogeneity. First the entropy map of the gray-level preprocessed image is created in the following way: we take a square from the preprocessed image and the entropy of this square will be calculated. Then we substitute this value into the entropy map to the place of the center pixel of the square. The entropy histogram is calculated in the following way: the circle of the region is divided into eight equal parts. We determine the median value of the entropy map in each region. This gives us an 8-bin histogram.

3. The pattern histogram: Local Binary Pattern (LBP) feature performs very well in various applications, including texture classification and segmentation [11], surface analysis [12] and face recognition [13]. The original LBP operator labels the pixels of an image by thresholding the 3-by-3 neighbourhood of each pixel with central pixel value and the result is taken as a binary number [14]. The LBP operator was later extended to $LBP_{R,P}$ [15], which probes P circularly symmetric neighbouring samples on a circle of radius R, instead of a local 3-by-3 neighbourhood. With larger sampling radius and more neighbouring samples considered, the operator is able to capture information from a larger area. The differences the value of the center pixel and its neighbouring values are encoded into a binary number. Formally,

$$LBP_{R,P} = \sum_{p=0}^{P-1} u(g_p - g_c) \cdot 2^p, \ u(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{1}$$

where g_c is the grey value of the center pixel, g_p is the grey value of the pth neighbour of the center pixel, P is the number neighbours located on a circle of radius R. For our pattern histogram we use $LBP_{R,P}$ operator with the following parameters, $R = 1$ and $P = 8$. On this circle, if a neighbour position does not exactly fit with the center of the pixel, its grey value will be calculated by bilinear interpolation. The pattern histogram is computed in the following way: we determine to all pixels over the cell the binary numbers with the help of $LBP_{R,P}$ operator. These binary numbers are taken one after the other and on the same place located digits are added together. This process gives us an 8-bin histogram then it will be normalized.

4. The dominant color characteristics: There are many different methods to obtain colour descriptors [16, 17, 18]. Our colour descriptor is calculated in the following way: after transforming the picture into HSV colour space, we divide the region into four parts. Then we calculate the median of the hue values in all sectors. Our colour descriptor is scale-invariant and shift-invariant with respect to light intensity.

Therefore the whole the descriptor is made up of four parts:

$$D_i = [d_{i0} \mid d_{i1} \mid d_{i2} \mid d_{i3}]. \tag{2}$$

1. Vector d_{i0} is the edge descriptor in 30 degrees resolution that means 12 dimensions.

2. Vector d_{i1} is the entropy descriptor of the masked image item, the number of the dimension is 8.

3. Vector d_{i2} is the pattern histogram, 8 dimensions.

4. Vector d_{i3} is the colour descriptor, 4 dimensions.

In summary the total number of dimension is 32. Previous experience has shown that complex metrics are not useful in comparing the histograms. Based on our experiments the Euclidean distance is a good choice for similarity measure.

5. EXPERIMENTAL RESULTS

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).

To summarize our experiments, the LHI-tree retrieves NN list with acceptable response time. Because of its RAM-based nature the FLANN is slightly faster with the pure RAM accesses. The subjective evaluations of the results validate the usability of the index structure. In most cases the list of retrieved images is similar to the output of FLANN [19]. We demonstrate the promising results shown on Figure 2. Our precision and recall values are little less, but the first three or four hits are usually the same. Among the first twenty hit in the 93-98 % of the cases there are relevant images. We measured that the precision of the system is between 90-95 %. On the other hand the average ratio of the irrelevant hits among the first twenty hit is between 70 and 80 %. Consequently, the recall of the system is between 20-30 %.

Table 2 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. We compared the times of the retrievel process with the values of FLANN. In Table 2 the times of FLANN can be seen in parenthesis.

We compared the results with another systems and we experienced that our retrieval and run times were shorter, our precision and recall values were slightly less [7].

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.

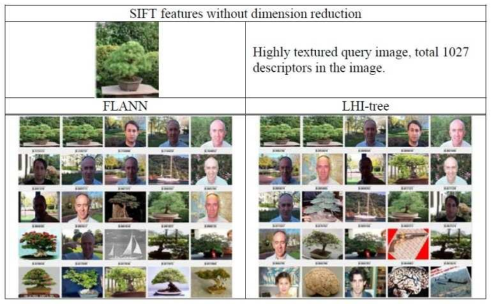
6. CONCLUSIONS AND FUTURE WORK

In this paper we proposed the LHI-tree, which is a disk-based data structure for retrieving high-dimensional image descriptors on large datasets. We introduced the region based image descriptors in detail. Main goals of this indexing scheme are: disk-based operation to reduce the RAM usage and neglect vector data representation and metric norms to achieve approximate NN search.

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 to FLANN.

The main advantage of our index is that the building speed is not (or only slightly) dependent on the number of previously added elements and its querying time is not dependent on the index size. However, its prerequisite is the availability of some representative samples during initialization stage.

Our future work contains a better optimized image processing module and a reduced size index structure.


SIFT features without dimension reduction
Highly textured query image, total 1027 descriptors in the image.
FLANN
LHI-tree

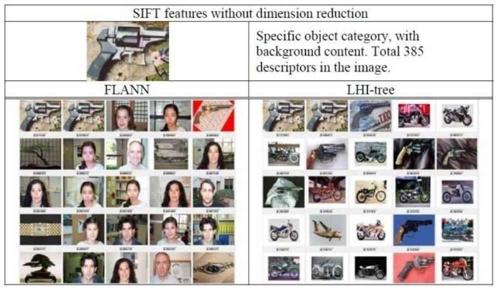

SIFT features without dimension reduction
Specific object category, with background content. Total 385 descriptors in the image.
FLANN
LHI-tree

Fig. 2. We compared the LHI-tree to FLANN. In most cases the lists of images are similar.

Table 2. Computation times in retrieval process. The retrieval process consists of three phases: feature extraction, similarity search, ranking and translating IDs. In parenthesis there is the value of FLANN.

Count	Feature extraction	Similarity search	Ranking and translating IDs
80	250 msec	434 msec (420 msec)	600 msec (590 msec)
250	780 msec	1425 msec (1400 msec)	1700 msec (1650 msec)
450	1400 msec	3150 msec (3025 msec)	1880 msec (1870 msec)

7. REFERENCES

[1] P. Geetha and V. Narayanan, “A Survey of Content-Based Video Retrieval”, *Journal of Computer Science*, vol. 4, pp. 474-486, 2008

[2] J.H. Friedman, J.L. Bentley and R.A. Finkel, “An algorithm for finding best matches in logarithmic time”, *ACM Transactions on Mathematical Software*, vol. 3, pp. 209-226, 1977

[3] C. Silpa-Anan and R. Hartley, “Optimised KD-trees for fast image descriptors matching”, *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-8, 2008

[4] M. Muja and D.G. Lowe, “Fast approximate nearest neighbors with automatic algorithm configuration”, *In VISAPP International Conference on Computer Vision Theory and Applications*, pp. 331-340, 2009

[5] P. Sarker and A.W. Moore, “Fast nearest-neighbor search in disk-resident graphs”, *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 513-522, 2010

[6] I. Lee, “Feedback for Distributed Content Based Image Retrieval”, *Proceedings of International Symposium on Computer Network and Multimedia Technology*, pp. 1-4, 2009

[7] H. Lejsek, F.H. Asmundsson, B.T. Jonsson and L. Amsaleg, “NV-Tree: An Efficient Disk-Based Index for Approximate Search in Very Large High-Dimensional Collections”, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 31, pp. 869-883, 2009

[8] L. Havasi, M. Szabo, M. Pataki, D. Varga, T. Sziranyi and L. Kovacs, “Search in WikiImages using mobile phone”, *Proceedings of 11th International Workshop on Content-based Multimedia Indexing*, pp. 219-222, 2013

[9] G. Amato and P. Savino, “Approximate similarity search in metric spaces using inverted files”, *Proceedings of the 3rd International Conference on Scalable Information Systems*, pp. 1-10, 2008

[10] L. Fei-Fei, R. Fergus and P. Perona, “Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories”, *Computer Vision and Image Understanding*, vol. 106, pp. 59-70, 2007

[11] M. Topi, O. Timo, P. Matti and S. Maricor, “Robust texture classification by subsets of local binary patterns”, *Proceedings of the 15th International Conference on Pattern Recognition*, pp. 935-938, 2000

[12] M. Pietikainen, “Image Analysis with Local Binary Patterns”, *Proceedings of the 14th Scandinavian Conference*, pp. 115-118, 2005

[13] T. Ahonen, A. Hadid and M. Pietikainen, “Face Recognition with Local Binary Patterns”, *Proceedings of the 8th European Conference on Computer Vision*, pp. 469-481, 2004

[14] T. Ojala, M. Pietikainen and T. Maenpaa, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns”, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, pp. 971-987, 2002

[15] M. Heikkila, M. Pietikainen and J. Heikkila, “A Texture-based Method for Detecting Moving Objects”, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, pp. 657-662, 2006

[16] B.S. Manjunath, J.-R. Ohm, V.V. Vasudevan and A. Yamada, “Color and Texture Descriptors”, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 11, pp. 703-715, 2001

[17] K.E.A. van de Sande, T. Gevers and C.G.M. Snoek, “Evaluating Color Descriptors for Object and Scene Recognition”, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 32, pp. 1582-1596, 2010

[18] O.A.B. Penatti and R. da Silva Torres, “Color Descriptors for Web Image Retrieval: A Comparative Study”, *SIBGRAPI ’08. XXI Brazilian Symposium on Computer Graphics and Image Processing*, pp. 163-170, 2008

[19] M. Muja and D. G. Lowe, “Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration”, *In VISAPP International Conference on Computer Vision Theory and Applications*, pp. 331-340, 2009