CONTENT BASED IMAGE RETRIEVAL: RELATED ISSUES USING EULER VECTOR†

Arijit Bishnu , Piyush K. Bhunre, Bhargab B. Bhattacharya‡ , Malay K. Kundu, C. A. Murthy
{bishnu_t, piyush_t, bhargab, malay, murthy}@isical.ac.in
Indian Statistical Institute, 203 B. T. Road, Kolkata - 700108
and
Tinku Acharya
tinku.acharya@intel.com
Intel Corporation, Chandler, AZ - 85226, USA

ABSTRACT

A combinatorial characterization of a gray-tone image called Euler vector is discussed. Euler vector comprises a 4-tuple, where each element is an integer representing the Euler number of the partial binary image formed by the four most significant bit planes of the gray-tone image. The vector is topologically invariant and can be used for image indexing and retrieval. The Euler vector for all the images in the database can be arranged using any multi-dimensional data structure. For retrieval, a query range is to be defined around the query image vector. We use a simple statistical technique to specify the query range. Next, we propose a modification in the Kd-Tree construction to build a simple hybrid tree that supports efficient adaptive clustering and indexing. The same data structure is used for clustering and indexing.

1. INTRODUCTION

Defining a good numerical characterization of an image is a fundamental problem in image processing. Determination of a compact set of parameters for a gray-tone image which is easy to compute, suitable for efficient database search, and admits robustness against transformations and noise, is now highly needed in the emerging domain of the Internet technology. Earlier approaches to image characterization include, i) spatial features like amplitude and histogram descriptors; ii) transform features like Fourier descriptor, DCT, iii) shape based features like area, Euler number, center of mass, moments, eccentricity, etc., iv) syntactic features based on structural peculiarities, v) statistical and structural texture features[1, 2]. In this work, we discuss a new parameter called Euler vector[3] of a gray-tone image. For a binary image, Euler number (genus) is defined as the difference between the number of connected components (objects) and the number of holes[1]. Efficient techniques of computing Euler number of a binary image and its VLSI implementation are also well known[4].

Features are extracted from images to map the image as a point in a multi-dimensional feature space which is indexed using a multi-dimensional data structure[5, 6]. We use a simple statistical technique to define an isothetic query box around a query point for feature-based similarity searching. In the section, thereafter, we use a modified construction of a Kd-Tree[6] to cluster the images in the multi-dimensional feature space. The same data structure is used for indexing as well. A brief introduction is given in that section itself. We give results using Euler vector as a 4-dimensional feature vector.

2. EULER VECTOR COMPUTATION

In a gray-tone image, if each pixel has an integer lying between $[0, 255]$ as its intensity value, then an 8-bit binary vector, $\{b_7, b_6, b_5, b_4, \ldots, b_0\}$ can represent it. Each b_i is either '0' or '1'. The given image comprises 8-bit planes of the same size as the image. We retain the first *4 most significant bit planes* (corresponding to (b_7, b_6, b_5, b_4)) as they contain most of the information of the image, and ignore the remaining planes. Each of these 4-bit binary vectors is converted to its corresponding reflected gray code (g_7, g_6, g_5, g_4)[7], which is defined as: $g_7 = b_7; g_6 = b_7 \oplus b_6; g_5 = b_6 \oplus b_5; g_4 = b_5 \oplus b_4$, where, $\oplus$ denotes XOR (modulo-2) operation. For any binary vector, the corresponding reflected gray code is unique and vice-versa. Each gray-code bit-plane represents a 2-tone image.

Definition: The Euler vector of a gray-tone image is a 4-tuple E_7, E_6, E_5, E_4 where E_i is the Euler number of the partial two-tone image formed by the i^{th} bit-plane, $7 \leq i \leq 4$, corresponding to the reflected gray code representation of intensity values.

Two consecutive numbers have unit hamming distance in gray-code representation, and a small change in intensity value is not likely to affect all the 4-bit planes simultane-

† This work is funded by a grant from Intel Corp., USA (PO #CAC042717000).

‡ Author for correspondence.

ously. Euler vector is found to be more insensitive to noise and other changes, if the gray-code is used. To make the Euler vector more robust, the given image is cleaned successively by median and mean filters. The images are also rescaled such that the dynamic range of the intensity levels is mapped to $[0, 255]$. Since, Euler number is easily computable[4], Euler-vector of a gray-tone image also provides a quick combinatorial signature. The Euler vector of a image is found to remain near invariant under inclusion of salt and pepper or gaussian noise followed by filtering, and also under JPEG compression. It has a strong discriminatory power and can thus be used to augment other features to facilitate image searching and retrieval. The upcoming JPEG2000 image compression standard [8] supports multiresolution in terms of scale and embedded quantization by bit-plane coding. Euler vector being topologically invariant can support multiresolution scale and the bit-plane representation is inherent to its definition.

3. RANGE QUERY AND RETRIEVAL

3.1. Range Determination

For feature-based similarity searching, images are mapped as points in a multi-dimensional (say, d) feature space which is indexed using a multi-dimensional data structure[5, 11, 13]. Given a query image, the same 'd' features are extracted and the query image is a point in the same space. Now, a range query[6] asks to report all images whose features lie within a d-dimensional axis-parallel box centered around the query image point. The user is oblivious to the features used for indexing the images and hence, can't define a query range. We use a simple statistical technique based on the data distribution to find out the size of the query range box.

An image is represented as an d-dimensional feature vector $\vec{X} = (x_1, x_2, \ldots, x_d)$ where x_i's, $i = 1(1)d$ are the d features of the image. We assume that the x_i's are independent with each x_i being uniformly distributed as $f(x_i) = \frac{1}{b_i - a_i}$; $a_i \leq x_i \leq b_i$. We estimate b_i's and a_i's from the mean and the variance of the uniform distribution of $f(x_i)$. Expectation of such an uniformly distributed x_i is $(b_i + a_i)/2$ and the variance is $(b_i - a_i)^2/12$. Now, we find out the mean, μ_i and variances, σ_i^2 from the values of the features x_i in the image database, and estimate b_i's and a_i's as follows.

$$\frac{b_i + a_i}{2} = \mu_i \text{ and } \frac{(b_i - a_i)^2}{12} = \sigma_i^2$$

Solving the above two equations for getting the estimate of b_i's and a_i's, we have $b_i = \mu_i + \sqrt{3}\sigma_i$ and $a_i = \mu_i - \sqrt{3}\sigma_i$. Given, now a query image with feature vector, $\vec{Q} = (q_1, q_2, \ldots, q_d)$, we need to define ranges $\delta_i, i = 1(1)d$, such that the range searching in the multi-dimensional data structure takes place for ranges $((q_1 - \delta_1, q_1 + \delta_1), (q_2 - \delta_2, q_2 + \delta_2), \ldots, (q_d - \delta_d, q_d + \delta_d))$. The δ_i's are determined in such a way that within the said range, a fraction α(provided by the user) of the entire images lie. For a uniform distribution, $f(x_i)$ as shown above, we got to determine ranges in such a way so that α fraction lies within the said range i.e.

$$\int_{q_i - \delta_i}^{q_i + \delta_i} f(x_i) dx_i = \alpha, \text{ or } \delta_i = \frac{\alpha(b_i - a_i)}{2}.$$

Therefore, using the values of b_i and a_i, $\delta_i = 2\sqrt{3}\alpha\sigma_i$. Having found out δ_i's, $i = 1(1)d$, we can easily define the axis-parallel range box. A better estimate for the query range can be obtained by finding a statistical distribution that fits with the data.

The results are shown in $Table1$ for the five images in the last row of $Fig.1$ for $\alpha = 0.1$ e.g. the Euler vector for Object-16 is $(2, -1, -14, -87)$, and with $\alpha = 0.1$, the corresponding ranges are $([0, 3], [-2, 0], [-21, -6], [-103, -70])$. If the retrieved image in the said range is an image of the same object as given in the query, we call it a match.

Table 1.

Object	Euler vector	No. of images retrieved	No. of images matched	Percent
obj16	(2, -1, -14, -87)	51	41	80.39
obj17	(-1, -3, -18, -40)	33	24	72.73
obj18	(3, -8, -32, -71)	10	9	90.00
obj19	(0, 0, -5, -110)	16	9	56.25
obj20	(6, -11, -42, -71)	3	3	100.00

3.2. Image Database Used

The image database used is the Columbia Object Image Library(COIL-20) [12] of 20 objects. Each object was placed on a motorized turntable which was rotated through 360 degrees with respect to a fixed camera. Images of the objects were taken at pose intervals of 5 degrees. This corresponds to 72 images per object making a total of 1440 gray-scale images. The objects have a wide variety of complex geometric and reflectance characteristics. The images are size normalized.

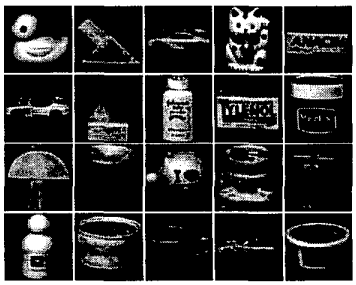

Fig. 1. COIL objects

4. CLUSTERING AND INDEXING

Similarity searching corresponds to a range search on a multidimensional data structure which indexes multidimensional

feature space[5]. Traditional multidimensional data structures can be classified into space partitioning (SP) techniques (e.g. variants of Kd-Tree [6]) and data partitioning (DP) techniques (e.g. R-Tree [9], SS-Tree [5], SR-Tree [10] etc.). SP techniques use a single dimension to split the space into subspaces, whereas, DP techniques use all the dimensions to create a containment hierarchy of bounding rectangles. Chakrabarti et al. [11] proposed a hybrid tree for indexing higher dimensional feature spaces. The hybrid tree combines positive aspects of the SP and DP techniques to achieve better search performance. The hybrid tree uses a single dimension to partition the space into two subspaces when a node splits like any SP technique (e.g. Kd-Tree). Here DP techniques are used by adding a second split position field in addition to the split position along the same splitting dimension required by the Kd-Tree which represents overlap free splits.

Index structures supporting similarity queries perform badly if the index structure doesn't fit in the main memory. A portion of the index structure needs to be fetched from the disk, making the search algorithm inefficient. Further, neighboring subregions can grow exponentially in terms of the dimensionality of the data and may require near exponential I/O access. To circumvent this problem, Chang et al.[14] uses i) a clustering technique to cluster similar data on disk to minimize disk latency for retrieving similar objects and ii) a hashing technique to index clusters. The cluster centroids can be indexed using any multidimensional index structure. The clustering techniques may fragment the existing natural clusters and cluster boundaries might overlap in the multidimensional feature space. Note that, if the clustering and indexing are done separately, the inherent drawbacks of the clustering algorithm is reflected in the indexing and query procedure.

In our work, we use a modification of a Kd-Tree to build a 'simple' hybrid tree that supports efficient clustering and indexing. We use SP techniques to divide the data set into several small non overlapping isothetic hyperboxes. This division is stopped under certain criteria. After the division stops, we start merging two hyperboxes at a time sharing a common face (called a separating plane) if certain similarity criterion is satisfied. After the process ends, we get some non overlapping hyperboxes, indexed by the same Kd-Tree, containing data points and each of them gives a cluster. If the optimality condition and the parameter values are chosen correctly, we get adaptive clusters that have been indexed as well, with each leaf node containing the cluster.

4.1. Partition of Data and Tree Construction

Let S be a d-dimensional data set comprising n points. We shall denotes the coordinate axes as $X^{(1)}, X^{(2)}, \cdots, X^{(d)}$. The data set is partitioned using the SP techniques as follows: find the median of the first dimension (component) of all the point in S. Split S into two parts S_1^1 and S_1^2 so that S_1^1 contains all the point of S whose first component is less than or equal to the median and the remaining points are put into S_1^2. The *splitting point*, a point on $X^{(1)}$, is the average of the maximum and minimum of the first components of all points in S_1^1 and S_1^2 respectively. S_1^1 and S_1^2 are readjusted so that all points in S_1^1 and S_1^2 lie respectively to the negative and positive sides of the splitting point in $X^{(1)}$. Next, find the median of the second component of all the data points in S_1^1 and split S_1^1 according to the same procedure. Similar operation is performed on S_1^2. This process is continued and at each *level* the component to be used for partition is chosen as *level(mod d)* $+1$. If size of the data set under consideration is less than *MinSize*, a threshold value, then partitioning terminates. The hyperboxes and the corresponding data points are organized into a multidimensional tree which is similar to a Kd-tree [6]. We shall refer this tree as *hybrid tree*. The root corresponds to the whole data set S. A *global bounding box* of the data set is stored in the root. When splitting of S is done, bounding boxes of S_1^1 and S_1^2 are computed. Left subtree and right subtree of the root are built on these two subsets respectively. This process is continued till further split is possible. So, each leaf node of the tree gives a bounding hyperbox containing a small set of data points. Note that, bounding boxes of the nodes at same level are non overlapping.

4.2. Clustering by Merge

The number of nodes in the said tree increases exponentially with the increase of the data size. We would try minimizing the depth of the tree and find optimal clusters as well. For clustering, we shall perform a merge operation between two neighboring hyperboxes, represented by sibling nodes, according to their data size and the distance based on the spatial information. Each of the leaf node contains one hyperbox containing some of the data points and any two pair of boxes are non overlapping. In d-dimensional space any hyperbox has $2d$ neighboring hyperboxes sharing *(d-1)* dimensions with it. For merging operation, we shall consider only these neighboring hyperboxes. As the time of finding all such hyperboxes would depend on the dimensionality of the data, we choose only hyperboxes which are represented by sibling nodes. Note that, they also share in between them $d-1$ dimensions. Let the chosen hyperboxes for merging be A and B; let *MinSize* and *MaxSize* be the minimum and maximum allowable data size respectively of the leaf node. If the distance between A and B is less than a distance threshold, *DistThreshold*, hyperboxes A and B are candidates for merging. For measuring the distance between two sets (here hyperboxes) A and B, we have used a similarity measure defined as follows: $SimilarityMeasure(A,B)$

$$= Min\{Min_{a \in A} Median\{||(a,b)|| : b \in B\},$$
$$Min_{b \in B} Median\{||(a,b)|| : a \in A\}\}$$

where $||(a,b)|| = EuclideanDistance(a,b)$. Some other

Table 2.

Min size	*Max size*	*Dist Threshold*	No. of Clusters	Percentage of acceptance		Avg. search time ratio (Exhaustive/Kd-Tree)
				Search by Kd-Tree	Exhaustive search	
20	30	300	100	59	59	14.5
30	50	300	44	58	59	7.66
25	50	300	45	60	59	8.30
40	100	300	26	58	59	14
40	80	300	29	56	59	24

similarity measures can be used. The algorithm of merge operation starts from the root and follows the paths towards two subtrees. For each internal node, it checks for merging between two children (sibling) (which are sharing one common separating plane); if merging is possible, we shall delete both the subtrees and the internal node under consideration will be a leaf node and gives a cluster. This cluster is written onto secondary storage. If the merging is not possible, then we shall call merge operation on the subtrees of that internal node recursively. So, the sequence of the merge operations is basically predefined by the partition of the data. To get better clustering, different partitioning techniques can be used for the best merging sequence during the merge operation. At the end, this algorithm returns a tree whose leaf nodes correspond to a cluster saved on the disk. Note that the depth of the tree has also reduced.

4.3. Search for a Query Point

Each cluster corresponds to a leaf node. Each leaf node keeps a pointer to disk block where the cluster is stored and the internal nodes guide the search path to the required leaf node or cluster. For an input data point, the search algorithm returns a leaf node whose bounding box contains the query point and corresponding to that leaf node there is one cluster which can be read from the disk. If the query point is outside the global bounding box no images are returned. So, by one I/O operation the corresponding cluster can be read and a linear or sublinear search can be performed on it in the main memory for exact match.

4.4. Results

We did the experiment with different values of $Minsize$, $Maxsize$ and $DistThreshold$ as shown in $Table2$ and for each of their combinations with different query images pertaining to all objects of the database. We found out the cluster in which the query point lies from the modified Kd-Tree; found its distance from all the images in the found out cluster and reported the first 10 having minimum distance from the query point. The acceptance percentage is calculated as the ratio of the number of images belonging to the same COIL object image by 10.

5. REFERENCES

[1] Jain, A. K., *Fundamentals of Digital Image Processing*, Prentice Hall of India, 1997.

[2] Teh, C. H., and Chin, R. T., "On Image Analysis by the Methods of Moments", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.10, No. 4, July, 1998.

[3] Bishnu, A., Bhattacharya, B. B., Kundu, M. K., Murthy, C. A., Acharya, T. "Euler Vector: A Combinatorial Signature for Gray-tone Images", *Accepted for publication in ITCC, 8-10, April, 2002, Las Vegas.*

[4] Bishnu, A., Bhattacharya, B. B., Kundu, M. K., Murthy, C. A., and Acharya, T., "On-chip computation of Euler number of a binary image for efficient database search", *Proc. Intl. Conf. on Image Processing(ICIP)*, Vol. III, pp. 310-313, Greece, Oct. 2001.

[5] David, A. W. and Ramesh, J., "Similarity Indexing with the SS-tree", *Proc. of the* 12^{th} *Int. Conf. on Data Engineering*, New Orleans, USA, pp.516-523, Feb. 1996.

[6] Overmars, M., Berg, M. D., Kreveld, M. V., Schwarzkopf, O., *Computational Geometry: Algorithms and Applications*, Springer-Verlag, 1997.

[7] Kohavi, Z., *Switching And Finite Automata Theory*, McGraw Hill, New York, 1977.

[8] Rabbani, M. and Cruz, D. S., "The JPEG2000 Still-Image Compression Standard", *http://jj2000.epfl.ch/jj-publications/*

[9] Guttman, A., "R-tree: a Dynamic Index Structure for Spatial Searching", *Proc. ACM SIGMOD*, Boston, USA, pp.47-57, Jun. 1984.

[10] Katayama, N. and Satoh, S., "The SR-tree: An Index structure for Higher-Dimensional Nearest Neighbor Queries", *Proc. of the International Conference on Management of Data, ACM SIGMOD*, pp.13-15, May 1997.

[11] Chakrabarti, K., Mehrotra, S., "The Hybrid Tree: An Index Structure for Higher Dimensional Feature Space", *Proceedings of the* 15^{th} *International Conference on Data Engineering*, 1999.

[12] Nene, S.A., Nayar, S.K. and Murase H. "Columbia Object Image Library: COIL-100", *Technical Report CUCS-006-96*, Department of Computer Science, Columbia University, February 1996.

[13] Gaede, V. and Günther, O. "Multidimensional Access Methods", *ACM Computing Surveys*, pp. 170-231, Vol. 30, No. 2, June 1998.

[14] Chang, E., Li, C., Wang, J. Z., Mork, P., and Wiederhold, G, "Searching Near-Replicas of Images via Clustering", *Proceedings of SPIE Symposium of Voice, Video, and Data Communications*, pp.281-92, Boston, September 1999.