Euler Vector for Search and Retrieval of Gray-Tone Images

Arijit Bishnu, Bhargab B. Bhattacharya, Malay K. Kundu, C. A. Murthy, and Tinku Acharya, *Senior Member, IEEE*

***Abstract*—A new combinatorial characterization of a gray-tone image called Euler Vector is proposed. The Euler number of a binary image is a well-known topological feature, which remains invariant under translation, rotation, scaling, and rubber-sheet transformation of the image. The Euler vector comprises a 4-tuple, where each element is an integer representing the Euler number of the partial binary image formed by the gray-code representation of the four most significant bit planes of the gray-tone image. Computation of Euler vector requires only integer and Boolean operations. The Euler vector is experimentally observed to be robust against noise and compression. For efficient image indexing, storage and retrieval from an image database using this vector, a bucket searching technique based on a simple modification of Kd-tree, is employed successfully. The Euler vector can also be used to perform an efficient four-dimensional range query. The set of retrieved images are finally ranked on the basis of Mahalanobis distance measure. Experiments are performed on the COIL database and results are reported. The retrieval success can be improved significantly by augmentiong the Euler vector by a few additional simple shape features. Since Euler vector can be computed very fast, the proposed technique is likely to find many applications to content-based image retrieval.**

***Index Terms*—Content-based image retrieval (CBIR), Euler number, feature extraction, Mahalanobis distance, range query.**

I. Introduction

A COMPACT and easily computable image feature is highly desirable for efficient management of image database, search and retrieval. The characteristic parameters of the image should also preferably remain invariant to various transformations, such as translation, rotation, scaling, rubber-sheet shearing, degradation by noise, compression, etc. Existing methods of feature extraction usually employ either geometric features of an object or luminance signature of an object [35], [41].

A. Geometric Features

Various elementary geometric shapes are used to provide characteristic features such as edge, corner, line, curve, hole, and boundary curvature to define individual features of an image [5], [9]. Several transformations e.g., medial axis transformation, morphological transforms, may be applied for analyzing the structure of the shape patterns. These geometric features can be roughly categorized into three types [9].

1) *Global parameters* are extracted from the geometric and topological parameters of the entire image, like perimeter, area, centroid, curvature, Euler number, contour points, convex hull, etc. Geometric features calculated from moments, like center of mass, orientation, bounding rectangle, etc., are also used. A combination of global features using Euler number, convex hull and its deficiencies has been in use lately [38].
2) *Structural parameters* are the features which are local in nature, each describing a portion of the object. Features like line segment, arc segment with constant curvature, corner specifications, etc., that define pieces of an object's boundary are widely in use.
3) *Relational parameters* represent geometrical relations among local features using graph representations. Distance and relative orientation of substructures and regions of an object are interrelated using mainly graph-based methods.

B. Luminance Features

In this category, features are based on the luminance information represented by the intensity values [16].

1) *Spatial features* are characterized by the gray-levels or colors and their distributions like amplitude and histograms [28], [39].
2) *Transform features* provide the frequency domain information of the image, obtained by zonal filtering in the selected transform space e.g., Fourier descriptor, DCT, with applications to shape analysis [20], [29], [44].
3) *Edges and boundaries* characterizing object boundaries and shape may be extracted using gradient operators. Boundaries are extracted by edge linking techniques like contour following, edge linking, and heuristic graph searching [16].
4) *Invariant moments* [16] are used for shape and scene matching applications [15], [31], [40]. Zernike moments provide features for rotation-invariant image recognition [21].
5) *Texture features* are mostly based on structural, statistical, or spectral properties. There are several methods for texture extraction using gray-level co-occurrence statistics [13], Gabor filters [17], windowed Fourier filters [2], association rules [34].

Manuscript received June 8, 2004; revised November 3, 2004. This work was supported by a grant from Intel Corporation. This paper was presented in part at the International Conference on Information Technology: Coding and Computing, Las Vegas, NV, April 2002. This paper was recommended by Associate Editor S. Sarkar.

A. Bishnu is with the Japan Advanced Institute of Science and Technology, Ishikawa, Nomi-gun 9231292, Japan.

B. B. Bhattacharya, M. K. Kundu, and C. A. Murthy are with the Indian Statistical Institute, Kolkata 700 108, India (e-mail: bhargab@isical.ac.in).

T. Acharya is with Avisere, Inc., Chandler, AZ 85226 USA.

Digital Object Identifier 10.1109/TSMCB.2005.846642

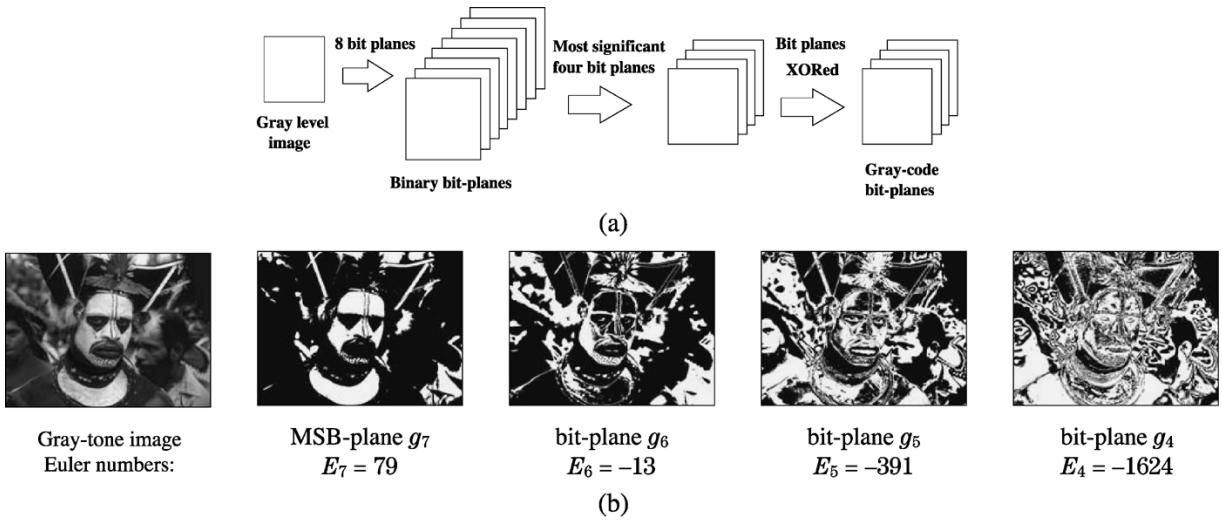


Fig. 1. Gray-code bit plane and Euler vector $= \{79, -13, -391, -1624\}$. (a) Gray-code bit planes. (b) Bit planes and Euler numbers.

With the emergence of the Internet, content-based image retrieval (CBIR) has recently become one of the most important research topics spanning disciplines like image processing, computer vision, information retrieval, multidimensional access methods, databases, etc. A good survey of the technical aspects of the existing CBIR systems has appeared in [37], [42]. Most of the systems use mainly low level features. Determination of a compact set of low-level features for a gray-tone image is thus highly needed for multimedia information transactions across the web. Further, the parameters should be easily computable and suitable for efficient database search. They should also be invariant under various transformations and insensitive against noise, compression, etc.

In this work, we define a new parameter called *Euler vector* of a gray-tone image. For a binary image, the *Euler number* (genus) is defined as the difference of the number of connected components (objects) and the number of holes [11], [33]. The definition of Euler vector is derived from the four most significant binary bit planes corresponding to the gray-code representation of the intensity values of a gray-tone image. We report several experiments using the Euler vector to show that it has a strong discriminatory power and can thus be used to augment other features to facilitate image searching and retrieval. For this purpose, the Euler vector can be used as a signature, and a bucket as well as a four-dimensional (4-D) range query is performed on a modified Kd-tree representing the database. Next, the Mahalanobis distance measure is used to rank the similarity of the retrieved image with respect to the query image. Our experiments on the COIL database [26] show that the Euler vector augmented by two simple shape features can be used to build a very efficient and successful retrieval engine.

The contents of the paper is organized as follows. Section II discusses the computation of the Euler vector and the use of gray codes. Section III introduces the distance measure for the Euler vector and reports experimental results demonstrating its robustness against noise and compression. Section IV deals with the retrieval issues. Finally, Section V presents discussions and concluding remarks.

II. Euler Vector Computation

A. Bit-Planes and Euler Vector

A gray-tone image is assumed to be represented as an $(N \times M)$ matrix, where each element $f(i,j)$ is an integer lying between [0, 255] that denotes the intensity of the corresponding pixel. Thus, a 8-bit binary vector, denoted by $(b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0)$ can represent the intensity value of each pixel, where each b_i is either "0" or "1". The image may now be considered as an overlay of eight bit-planes [Fig. 1(a)]. Each bit-plane can be thought of as a two-tone image and can be represented by a binary matrix of size $(N \times M)$. To characterize a gray-tone image, we now define a 4-tuple called Euler vector that reflects the structural property of the image. The first *four most significant bit planes* (corresponding to (b_7, b_6, b_5, b_4)) are retained as they contain most of the structural information of the image, and the remaining planes are ignored. As we go down the bit-planes, the bit-patterns become more and more random. The lower bit-planes just add on to the brightness values and do not provide any useful structural information. However, each of these 4-bit binary vectors is converted to its corresponding reflected gray code (g_7, g_6, g_5, g_4) [22], where $g_7 = b_7$; $g_6 = b_7 \oplus b_6$; $g_5 = b_6 \oplus b_5$; $g_4 = b_5 \oplus b_4$. Here, $\oplus$ denotes XOR (modulo-2) operation. For any binary vector, the corresponding reflected gray code is unique and vice-versa. Bit planes based on gray codes were also used earlier for other bio-medical applications [23]. The rationale behind using gray codes for defining Euler vector over binary codes is discussed in the next subsection.

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, $4 \leq i \leq 7$, corresponding to the reflected gray-code representation of the intensity values.

Example: For the gray-tone image shown in Fig. 1(b), we compute the Euler number for the four most significant bit-planes by using a run-based algorithm [6] to determine the Euler vector, which is found to be $\{79, -13, -391, -1624\}$.

Euler vector, like the Euler number of a binary image, remains invariant under translation, rotation, scaling, and rubber-sheet transformation of the image. Euler vector thus, as a signature of a gray-tone image, uses both geometric and intensity level characterization [35], [41]. Since Euler number is easily computable and depends on the combinatorial properties of 0–1 runs in the binary pixel matrix [33], the Euler vector may also serve as a quick combinatorial signature of a gray-tone image [7].

B. Use of Gray Code

Reflected gray-code representation of intensity values offers a distinct advantage over standard binary representation because two consecutive numbers have unit hamming distance in gray-code representation, and for most of the cases, a small change in intensity values is not likely to affect all the four bit planes simultaneously. Let z be the amplitude of the noise added to or subtracted from a gray-level value $f(0 \leq f \leq 255)$ to generate two values $f_1 = f + z$ and $f_2 = f - z$. Let $f_b(f_g)$ denote the binary code (gray code) representation of f. Let the binary code (gray code) representation of f_1 be $f_{1b}(f_{1g})$. Similarly, the binary code (gray code) representation of f_2 be $f_{2b}(f_{2g})$. We generate the corresponding values of $f \pm z$ for all values f in the range [0, 255], with z varying in the range [1,120] (if $f+z > 255$, it is truncated to 255; similarly if $f - d < 0$, it is truncated to 0). For all the first *four most significant bits* corresponding to f_{1b} to f_{2b}, the corresponding bits are checked against the first *four most significant bits* of f_b for a particular z with f_b varying in the range [0, 255]. A bit change is said to occur if a bit b_x of f_b flips to $\bar{b_x}$ in the range f_{1b} to f_{2b}. All such bit changes M_b are measured and normalized by z to calculate the average change in bits in the binary code representation. Exactly similar operations are done with the gray-code representations f_{1g}, f_{2g} and f_g to calculate M_g which is again normalized by z. So, for each $z \in [1, 120]$, we get normalized values of M_b and M_g signifying the average change in bits. The plot is shown in Fig. 2. It can be seen that the average change in the number of bits in the binary code is greater than that of the gray-code representation. This justifies the use of gray codes for defining Euler vector.

C. Implementation Details

We linearly rescale the dynamic range of intensity levels of the image. Visually similar images may differ in their dynamic ranges resulting in different bit-plane representations. To circumvent the problem, the images are rescaled such that the dynamic range of the intensity levels is mapped to [0, 255]. The definition of the Euler vector involves bit-planes that are binary images. Since bit-planes are sensitive to changes in the original intensity values, the given image is cleaned first by a 3×3 median filter followed by a similar mean filter to make the vector more robust. The order of filtering is also important. If the mean filter is applied first instead of the median filter, then salt and pepper noise or isolated pixels will be averaged out, and not removed, rendering the successive median filtering ineffective. Other sophisticated filters may also be used.

The Euler vector of a gray-tone image can be computed by determining the Euler number for each of the four binary bit-planes as mentioned above. It is known that the Euler number of a binary image can be computed as the difference of the sum of the number of runs for all rows (or columns), and the sum of the neighboring runs between all consecutive pairs of rows (or columns) [11], [33] and can also be implemented efficiently on-chip [6]. The algorithm for computing the Euler vector of a gray-tone image is described in Fig. 3.

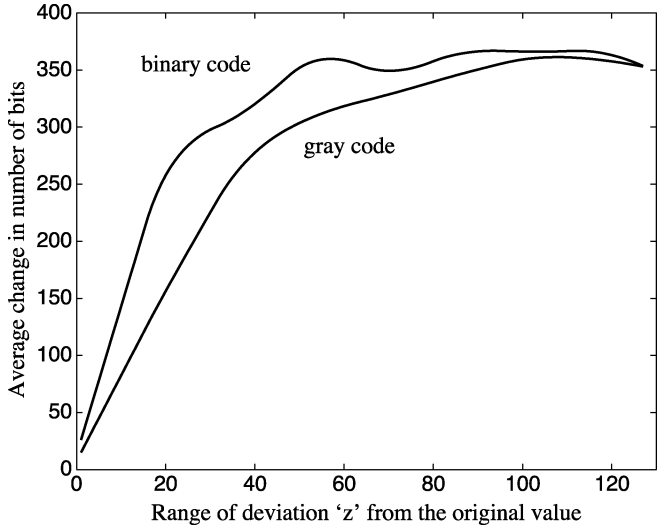


Fig. 2. Bit planes and Euler vector.

Algorithm *Compute_EulerVector*
Input: An $(N \times M)$ gray-tone image I
Output: Euler vector $\{E_7, E_6, E_5, E_4\}$.

1. Apply median filter followed by mean filter on I;
2. Linearly rescale the dynamic range of I to $[0, 255]$;
3. Convert the first 4 MSB binary planes of I to its corresponding reflected gray code;
4. For each gray code bit plane, compute Euler number;
5. **return** the Euler Vector;

Fig. 3. Algorithm for computing the Euler vector.

III. Experimental Results

A. Image Database

The image database used is the Columbia Object Image Library (COIL-20) [26] 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°. 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. In Fig. 4, the frontal pose of each object is shown. We have used the COIL database as it is known as a very standard object image database and it has several images of the same object taken at different poses. Fig. 5 shows the Euler vector values for some images of the COIL database.

B. Mahalanobis Distance as a Measure of Similarity

While characterizing a gray-tone image by its Euler vector, we have observed that the ranges and the variances of various elements of the vector differ widely. The features under consideration are in general correlated. Thus, a desirable distance

Fig. 4. COIL objects.

Fig. 5. Euler vector of some images from the COIL database.

measure should give weights according to the correlation between the concerned variables. A suitable distance measure between two feature vectors $\widetilde{x}' = (x_1, x_2, x_3, \ldots, x_d)$ and $\widetilde{y}' = (y_1, y_2, y_3, \ldots, y_d)$ in $\mathbb{R}^d$ would be given by: $D(\widetilde{x}, \widetilde{y}) = \sqrt{\sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij}(x_i - y_i)(x_j - y_j)}$, where $w_{ij} > 0$. One way of obtaining such w_{ij}'s is to consider a $(d \times d)$ positive definite matrix Γ and taking $D(\widetilde{x}, \widetilde{y})$ as $\sqrt{\{(\widetilde{x} - \widetilde{y})'\Gamma(\widetilde{x} - \widetilde{y})\}}$. In such an environment, the Mahalanobis distance [1], [25] can be adopted to provide a very good measure that captures similarity/dissimilarity properties among the members of a given set of images. The covariance matrix of $\widetilde{x}$, denoted as Λ, is a $(d \times d)$ matrix given by: $\Lambda(i, j) =$ variance of x_i if $i = j$; covariance of x_i and x_j if $i \neq j$. We may use the matrix Λ^{-1} as the weight matrix Γ and compute the distance as follows: $\Delta = \sqrt{(\tilde{x} - \tilde{y})'\Lambda^{-1}(\tilde{x} - \tilde{y})}$. Λ is also known as dispersion matrix. If $\tilde{x}$ and $\tilde{y}$ denote the mean vector of two populations then Δ is called the Mahalanobis distance between those two populations [1], [25]. In a large database of images, the covariance matrix defined by feature vectors of the images may be taken as the population dispersion matrix. If the number of images in the database is large, removal of a few images will not have a significant impact on the entries of the covariance matrix. The covariance matrix can also be incrementally updated when new images are added to the database. Since, Λ is positive definite, it is easy to show that, for a fixed Λ, Δ is indeed a metric. Such a distance measure satisfies the following: 1) a feature with large variance does not contribute more to the distance value and 2) the correlation among variables affects the distance measure. Therefore, it fits very well in the proposed environment of image retrieval based on the Euler vector as the feature.

C. Effects of Noise on the Euler Vector

We tested the robustness of the Euler vector on 1440 images of the COIL database under salt and pepper noise, Gaussian noise, and jpeg compression. The salt and pepper and Gaussian noises are zero-mean random additive noise. For all types of noises, the image intensity level is assumed to be in [0,1] and not in [0,255] in this subsection. In the case of salt and pepper noise, we varied the noise density from 0.01 to 0.10 in gaps of 0.01, i.e. 1% to 10% of the pixels were affected. For Gaussian noise, we varied the variance of the random noise from 0.01 to 0.10 in gaps of 0.01. For jpeg compression, we used the standard parameter of *quality* for compression. Higher quality parameter values indicate less image degradation and hence, larger file size. We used quality parameter values ranging from 5 to 90 for the compression, and the average peak signal-to-noise ratio (PSNR) ($\mathrm{PSNR} = 20\log_{10}(255/\mathrm{RMSE})$, where RMSE is the root mean square error) of all the images in the COIL database pertaining to the said quality values ranged from 26.26 to 43.05. Higher quality values of compression will have lesser image degradation and hence higher PSNR values. For each of the 1440 images in the COIL database, we add noise or compress it as stated above. To find out the effect of noise and compression on Euler vector, we define three indices called Closeness Measures, CM_1, CM_2 and CM_3 corresponding to particular values of noise density and quality factor of compression as follows. We find out the distance of each original image from all the noisy images. Let the distance, as defined in the earlier subsection, of an image $i(1 \leq i \leq 1440)$ from a noisy image $j(1 \leq j \leq 1440)$ be $\Delta_{(i,j)}$. Thus, $\Delta_{(i,i)}$ means the distance of the image i from its corresponding noisy version. The closeness measure $CM_1 = \sum_{i=1}^{1440}(\Delta_{(i,i)}/Ave_i)$, where $Ave_i = \sum_{j=1}^{1440}\Delta_{(i,j)}/1440$. CM_1 measures the average of the ratio of the distance between an image and its noisy version to the average of the distance between the image and all other noisy images. CM_2 is the fraction expressing the number of images out of 1440 that have $\Delta_{(i,i)} \leq Ave_i$. The third closeness measure CM_3 measures the closeness of the noisy image to the original image compared to other noisy images as follows. Let $max_i = max_{j=1}^{1440}(\Delta_{(i,j)})$ and $min_i = min_{j=1}^{1440}(\Delta_{(i,j)})$. Now, for each image, the closeness is calculated as $cm_i = (\Delta_{(i,i)} - min_i)/(max_i - min_i)$. Then, the overall closeness measure is the average calculated as $CM_3 = \sum_{i=1}^{1440} cm_i$. Thus CM_1 gives only the ratio between the distance of the noisy image and its original version to the average distance. The measure CM_3 gives the difference in magnitude compared to the range of the distance values. Lower values of CM_1 and CM_3 indicate that the feature is stable to noise and compression. The value of CM_2 on the other hand, gives the ratio of count of the number of images, where the particular distance value is less than the average value. Clearly, higher values of CM_2 indicate stability to noise. Further, CM_1 and CM_2 will have some sort of inverse relationship. High values of CM_2 would imply small values of CM_1 and vice-versa.

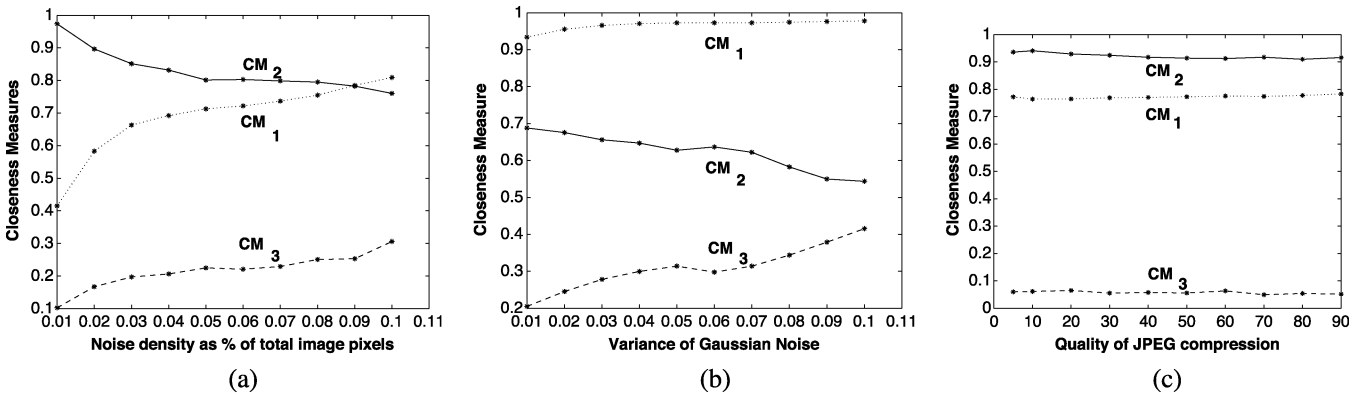


Fig. 6. Effects of noises and compression on the Euler vector. (a) Effect of salt and pepper noise. (b) Effect of Gaussian noise. (c) Effect of jpeg compression.

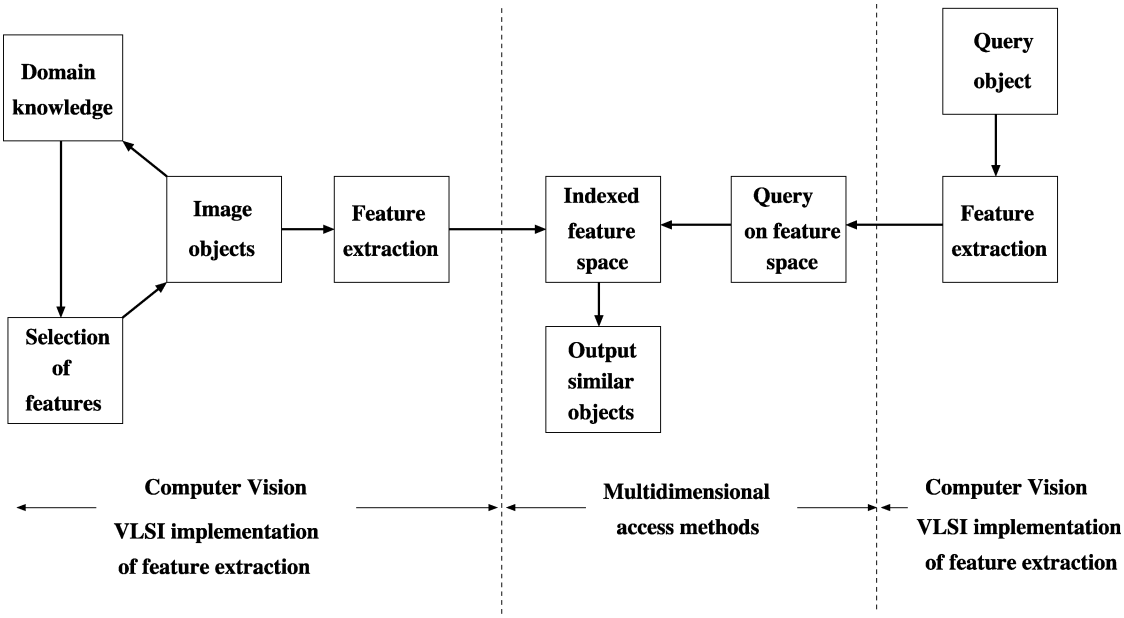


Fig. 7. Simple illustration of basic components of a retrieval system.

Fig. 6 shows the closeness measures CM_1, CM_2 and CM_3 as defined earlier. It can be seen that the Euler vector is most sensitive to Gaussian noise compared to others. It is found to be more or less stable against variations of quality parameter values in jpeg compression for all the three closeness measures. A high CM_2 and low CM_3 value indicate that the Euler vector does not undergo drastic changes under compression. For salt and pepper noise, the stability of the Euler vector degrades as the density of noise increases. However, even in the case when 10% of the pixels (≈ 1638) out of 128×128 pixels were corrupted, the closeness measure values stay within reasonable limits. For all types of noise, the closeness measure CM_3 does have appreciably low values, indicating that the distance between the noisy and the original image is very small compared to the range of the values.

IV. Retrieval Using Euler Vector

A. A Brief Review of Multidimensional Access Methods

Multidimensional search techniques are employed to retrieve a subset of images lying in the neighborhood of the query image in the feature space. Since the Euler vector consists of four elements, a 4-D orthogonal range query can be performed for retrieval. Ranking according to the proposed measure can then be performed on this subset only.

A CBIR system involves a fusion of computer vision with multidimensional access methods [37] (see Fig. 7). The process can be expedited using on-chip implementation of feature extraction (Fig. 7). In a CBIR system, the images are mapped as points in the multidimensional feature space that is indexed with suitable data structures that support similarity-based query of multidimensional feature vectors [43]. Given a query image, the same features are extracted to perform similarity queries that correspond to nearest neighbor queries and range queries [43].

Multidimensional data structures can be classified as 1) *space partitioning* based techniques and 2) *data partitioning* based techniques [8], [10]. The former techniques recursively partition the entire space into mutually disjoint subspaces. Kd-tree [4] and K-D-B tree [32] are examples of such *space partitioning* techniques. In *data partitioning* the data points at a particular level are obtained by clustering them in the sibling nodes using various bounding regions like rectangular hyperbox, sphere, etc.

Depending on the nature of bounding region, a suitable index structure like the R tree [12], R^+ tree [36], R^* tree [3], SS tree [43], and SR tree [19] may be used. Unlike the *data partitioning* index structure, range query on the multidimensional space using a *space partitioning* index structure like Kd-tree, requires only a single check at each node corresponding to the splitting dimension.

Euler vector is a 4-D feature. The queries are limited to point or range queries. In an environment of widely differing dispersions in each dimension as discussed in Section III-B, the partitioning would be better if the subspaces are split guided by the dispersions of their dimensions. The *space partitioning* techniques allow us to choose such splits, as already discussed. Thus, a basic Kd-tree can be tailored to suit our purpose.

B. Proposed Modification to Kd-Tree Construction

To reduce query time for retrieval, the depth of the Kd-tree is to be reduced by assigning to each leaf node a bucket containing a set of merged data points instead of a single data point. The number of data points to be stored in each bucket lies within a fixed range, say $MaxSize$ and $MinSize$ depending on the application and machine platform and its primary and secondary memory size. In a Kd-tree, distances among the data points are not considered while partitioning. The merge, if based on some suitable distance measure, will represent a meaningful ensemble of points. These collections of data points are stored as buckets in the secondary memory. The main memory stores the index structure guiding the search path to the corresponding bucket, which is retrieved from the secondary memory. To address the above issues, we propose a modification of the Kd-tree by performing split and merge operations on its nodes.

1) Split Operations for Partitioning: Let P denote a set of points representing images as feature vectors, where each point p_i is a d-dimensonal vector. Let $k(\leq d)$ be the dimension along which the space P_v represented by the node v is to be split. The value of k is stored in the node v.

Next β_k, the splitting hyperplane along the k^{th} dimension, is calculated as the average of m_{k_v} and m'_{k_v}, where m_{k_v} is the median of the k^{th} coordinates of the points belonging to P_v, and m'_{k_v} is the closest k^{th} dimension coordinate of points in P_v to m_{k_v}. This ensures that the splitting hyperplane does not pass through any point in P. Next, we split P_v into two parts P_{v_1} and P_{v_2} such that the k^{th} dimension coordinates of all points in $P_{v_1}(P_{v_2})$ are less (greater) than β_k. Two new nodes v_1 and v_2 are created and $v_1(v_2)$ is attached as the left (right) child of v. This recursive partitioning scheme is continued till the cardinality of the subspace to be split, represented by a particular node is just less than $MinSize$. Note that, these nodes may not be at the same level. The bounding hyperboxes of the nodes at the same level are nonoverlapping, and each leaf node of the tree gives a bounding hyperbox containing points less than $MinSize$.

2) Merge Operations on the Partitions: In a 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 the hyperboxes that are represented by sibling nodes only. Two sibling leaf nodes v_1 and v_2 with P_{v_1} and P_{v_2} as the corresponding point sets are merged if $|P_{v_1}| + |P_{v_2}| \leq MaxSize$ and P_{v_1} and P_{v_2} are 'close' with respect to a *similarity measure*, i.e., $SimilarityMeasure(P_{v_1}, P_{v_2}) \leq DistThreshold$, where $DistThreshold$ is a threshold on the $SimilarityMeasure$ for merging. The similarity measure should be a measure between two sets. It should also be robust to the outlier points in the sets. We consider the similarity measure between sets P_{v_1} and P_{v_2} as follows: $SimilarityMeasure(P_{v_1}, P_{v_2}) = Min\{Min_{p_1 \in P_{v_1}} Median\{||(p_1, p_2)|| : p_2 \in P_{v_2}\}, Min_{p_2 \in P_{v_2}} Median\{||(p_1, p_2)|| : p_1 \in P_{v_1}\}\}$ where $||(p_1, p_2)|| = MahalanobisDistance(p_1, p_2)$.

The $MahalanobisDistance(p_1, p_2)$ is calculated as discussed in Section III-B. The algorithm of the merge opertation starts from the root and traverses the Kd-tree in post-order fashion. At each internal node (v), if merging is possible between two sibling leaf nodes having buckets, a new bucket with the merged set of points corresponding to the two siblings of v is used to replace the old two buckets in the secondary storage. The siblings of v are deleted from the data structure, and the node v is considered as a leaf node. Finally, the new bucket is attached to v. As the sequence of the merge operations is predefined by the partition of the data, different partitioning techniques can be used for best performance during the merge operation. At the end, the algorithm returns a tree whose leaf nodes correspond to a bucket saved on the disk. Further, the depth of the tree is also reduced.

The entire algorithm for construction of the modified Kd-tree consists of the $SplitOperation$ and $MergeOperation$, and is described in Fig. 8.

3) Space and Time Complexity of Modified Kd-Tree Construction: The modified Kd-tree for a set of n points with each point having d dimensions requires storage $O(dn)$ [27].

The $SplitOperation$ takes time $O(dn \log n)$ [27]. The $MergeOperation$ requires computation of the

Algorithm *BuildModifiedKdtree*(S, $MinSize$, $MaxSize$, $DistThreshold$)
Input: A set of d-dimensional points S and
$MinSize$, the minimum size of a bucket
$MaxSize$, the maximum size of a bucket,
$DistThreshold$, a threshold
on the similarity measure between buckets
Output: v,The root of the Kd-tree,
with bounded buckets at the leaf nodes

1. $v \leftarrow$ *SplitOperation*(S, $MinSize$);
2. $v \leftarrow$ *MergeOperation*(v, $MaxSize$, $DistThreshold$);

Fig. 8. Algorithm for constructing the modified Kd-tree.

TABLE I
THEORETICAL COMPLEXITIES OF THE SEARCH PROCEDURES

n: the number of images in the database, d: the dimension of the data,
k: the number of buckets in the modified Kd-tree,
m: the number of images in the range of the range query.

Type of search	Theoretical complexity
Exhaustive search	$O(dn)$
Bucket search	$O(\log k), k << n$
Range search	$O(n^{1-1/d} + m)$

TABLE II
COMPARISON BETWEEN EXHAUSTIVE AND BUCKET SEARCH WITH ONLY EULER VECTOR ($d = 4$)

r, no. of images to be ranked	Exhaustive search	Bucket search					% savings in CPU time
	Average precision	$Max\ Size$	$Min\ Size$	$Dist\ Threshold$	k, no. of buckets	Average precision	
5	71.00	20	10	0.80	133	65.00	98.12
10	66.00	40	20	0.69	67	63.50	96.37
20	56.25	80	40	0.89	36	52.50	93.57
30	49.67	120	60	1.11	21	43.17	90.84
40	44.75	160	80	0.90	17	39.75	93.04
50	40.90	200	100	1.12	16	35.52	90.84
60	38.33	240	120	1.41	10	34.17	85.80
71	36.14	284	142	1.44	9	29.29	84.61

$SimilarityMeasure$ between two sets of points belonging to the two nodes to be merged. At a depth h from the root of the tree, the maximum number of points that can correspond to a node is $2^{\log n-h}$. Computing $SimilarityMeasure$ between such nodes requires distance computation among all points and requires time of $O(2^{2(\log n-h)})$. The number of pairs of nodes that are the candidates for merge is $O(2^{h-1})$. Therefore, the time required for merging sets of points belonging to nodes at depth h is $O(2^{2(\log n-h)} \cdot 2^{h-1}) \approx O(2^{2\log n-h-1})$. If the merging operation continues for nodes upto k levels from the leaf level (where $k \approx O(\log MaxSize)$), the total time required in the worst case is $\sum_{h=\log n}^{k} 2^{2\log n-h-1} \approx O(2^{2\log n}) \approx O(n^2)$. For d dimensions, the time taken is $O(dn^2)$. Therefore, the construction of the modified Kd-tree requires $O(dn^2 + dn\log n) \approx O(dn^2)$ time in the worst case.

C. Query

We discuss three types of queries for retrieval: 1) exhaustive search; 2) bucket search; and 3) range search.

1) Exhaustive Search: The Mahalanobis distance of the query image to all the other n images in the database are computed and the first r images having the least distances to the query image are reported. The time complexity of this method is $O(dn)$, where d is the dimensionality of the data.

2) Bucket Search: The original Kd-tree stores each point at its leaf and supports range queries. The modification proposed in this paper ensures that the said tree has now a meaningful ensemble of data points stored in buckets in its leaf nodes. The points stored in a bucket defines a d-dimensional hyperbox within which all data points in this bucket lie. The splitting technique used in our construction ensures that there is no intersection between any two d-dimensional hyperboxes corresponding to the buckets stored at the leaf nodes. Hence, a given query image can lie in at most one bucket. The retrieval of a bucket in response to an image query is basically transforming a range query to a point query i.e., a query to find out a range represented by a bucket in which the image point lies. Once the bucket is located, the images stored in the range of the bucket should be similar to the query image corresponding to the particular image feature. Thus, the bucketing technique transforms a range query to a point/bucket query, which involves a comparison at each internal node of a single dimension pertaining to the discriminant dimension of that node. Hence, the time complexity is $O(\log k)$, which is independendent of the dimension and is determined by the height of the modified Kd-tree. The value of k depends on n, $MaxSize$, $MinSize$, and $DistThreshold$, but it is always less than n.

3) Range Search: The bucket searching technique expedites retrieval time as it does not involve the factor of dimension. However, this gain in time is achieved at the cost of efficiency of the success of retrieval. Range searching will improve upon the retrieval success but it will take more time compared to the bucket search.

A range query [27] 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, cannot define a query range. We use a simple statistical heuristic based on the data distribution to find out the size of the query range box, where the user supplies only a fractional value α.

An image is represented as a d-dimensional feature vector $\widetilde{X} = (x_1, x_2, \ldots, x_d)$ 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 $f(x_i) = 1/(b_i - a_i)$; $a_i \leq x_i \leq b_i$. 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 $(b_i - a_i)^2/12$ [18]. Now, we find out the mean, μ_i and

TABLE III
COMPARISON BETWEEN EXHAUSTIVE AND BUCKET SEARCH WITH EULER VECTOR AUGMENTED BY TWO OTHER SHAPE FEATURES ($d = 6$)

r, no. of images to be ranked	Exhaustive search	Bucket search					% savings in CPU time
	Average precision	$Max\ Size$	$Min\ Size$	$Dist\ Threshold$	no. of buckets	Average precision	
5	93.00	20	10	1.26	168	80.00	98.58
10	85.50	40	20	1.41	93	69.00	97.13
20	76.75	80	40	1.43	45	61.00	95.69
30	68.83	120	60	1.09	25	59.67	92.13
40	63.75	160	80	1.09	25	53.12	93.35
50	60.00	200	100	1.75	13	53.20	86.30
60	56.25	240	120	1.75	13	48.83	89.26
71	52.14	284	142	1.75	13	44.57	85.59

TABLE IV
COMPARISON BETWEEN EXHAUSTIVE AND RANGE SEARCH WITH EULER VECTOR ($d = 4$)

r, no. of images to be ranked	Exhaustive search	Range search					
		$\alpha = 0.4$		$\alpha = 0.5$		$\alpha = 0.6$	
	Average precision	Average precision	% savings in CPU time	Average precision	% savings in CPU time	Average precision	% savings in CPU time
5	71.00	70.40	81.5	71.0	78.34	71.0	69.06
10	66.00	66.15	80.9	65.98	78.52	66.5	68.65
20	56.25	54.60	80.89	55.20	78.75	55.73	69.78
30	49.66	48.54	80.38	48.5	78.88	48.57	69.82
40	44.75	45.24	81.48	45.0	78.55	44.50	70.16
50	40.90	42.56	80.11	42.05	78.79	41.10	70.32
60	38.30	40.6	81.44	40.26	78.80	39.45	68.94
71	36.14	38.66	80.83	37.97	78.60	37.4	69.79

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:

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

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, $\widetilde{Q} = (q_1, q_2, \ldots, q_d)$, we need to define ranges δ_i, $i = 1(1)d$, such that the range searching 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.,

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

Therefore, using the values of b_i and a_i, $\delta_i = \sqrt{3}\alpha\sigma_i$. 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.

TABLE V
COMPARISON BETWEEN EXHAUSTIVE AND RANGE SEARCH WITH EULER VECTOR AUGMENTED BY TWO OTHER SHAPE FEATURES ($d = 6$)

r, no. of images to be ranked	Exhaustive search	Range search					
		$\alpha = 0.4$		$\alpha = 0.5$		$\alpha = 0.6$	
	Average precision	Average precision	% savings in CPU time	Average precision	% savings in CPU time	Average precision	% savings in CPU time
5	93.00	93.68	88.58	93.00	86.41	93.00	85.25
10	85.50	87.13	89.59	86.77	87.80	85.42	84.36
20	76.75	79.46	87.88	79.04	85.70	77.95	86.43
30	68.83	75.00	87.19	70.19	86.70	70.90	83.02
40	63.75	73.45	87.48	65.34	88.50	65.92	83.46
50	60.00	72.48	88.30	63.30	88.28	61.46	84.80
60	56.25	70.94	89.00	63.18	88.60	58.09	84.81
71	52.14	69.79	86.70	61.70	84.5	54.85	85.10

If the user wants to specify the query ranges, the retrieval will take $O(n^{1-1/d} + m)$ time, where m is the number of images overlapping with the range of query [27]. It may be noted that in high dimensional space the data will tend skewing together, and a range query will result in searching most of the leaf nodes of the kd-tree to ensure accuracy. For this reason, sophisticated techniques are to be used for range searching in higher dimension [14].

D. Experimental Results

The image database used in our experiment is COIL as discussed in Section III-A (Fig. 4). Two sets of experiments for retrieval were performed. First, Euler vector is used alone as a 4-D feature; second, it is further augmented with two other shape features (i.e., a 6-D feature). The two additional features are chosen as follows. An image is thresholded at a value corresponding to the fourth bit-plane (a value of 16). The features are [16] 1) the ratio of area and the square of perimeter and 2) the convex hull deficiency ratio defined as the ratio between the original area and the area formed by the convex hull of the edge points. The experiments were then conducted for exhaustive search, bucket search, and range search.

For the exhaustive search, given a query image and a value r of the number of images to be ranked, let r_1 be the number of images of the same object as the query image. The average precision is then defined as the ratio of r_1 and r.

For the bucket search, the construction of the modified Kd-tree requires specification of the three parameters $MaxSize$, $MinSize$, and $DistThreshold$ for the merge operation corresponding to the $SimilarityMeasure$. Fixing a value of these parameters is a design problem based on values of parameters like main memory size, cache size, disk page size, etc. However, in our experiments henceforth, to show the performance of the bucket search in comparison to the exhaustive search, we fix the parameters as follows: 1) $MaxSize = 4 \times r$; 2) $MinSize = 2 \times r$; and 3) $DistThreshold$ for the merge operation is fixed as the average of the $SimilarityMeasure$ values just after the splitting operation is finished and before the merge operation starts. For retrieval using bucket search, given a query image, the bucket corresponding to the query image is located from the modified Kd-tree; the Mahalanobis distances of the query image from all the images in that bucket are then computed and the first r images are reported. The average precision is defined as in the exhaustive case.

For the range search, given a query image and a value of the parameter α, the query ranges of axis-parallel hyperboxes are formed as discussed in Section IV-C.3. Now, using a Kd-tree all the images lying within the said range is retrieved. Let the number of retrieved images be r_{rq}. Note that r_{rq} may be greater than or even less than r, the number of images to be ranked, as r_{rq} depends on the value of the ranges. Let r_2 be the number of images of the same object as the query image. The average precision is then defined as the ratio of r_2 to $maximum\{r_{rq}, r\}$.

Table I lists the theoretical complexities of the three different types of searches. It may be noted that the bucket search on the modified Kd-tree proposed here is independent of the factor of dimension. Table II gives the comparison between exhaustive and bucket searches with the Euler vector as the only feature. Each row of the table corresponds to a value of r, the number of images to be ranked. The corresponding values of $MaxSize$, $MinSize$, $DistThreshold$, and k, the number of buckets for the modified Kd-tree are also shown. The number of buckets k depends on the parameters $MaxSize$, $MinSize$, and $DistThreshold$. The last column shows the savings in CPU

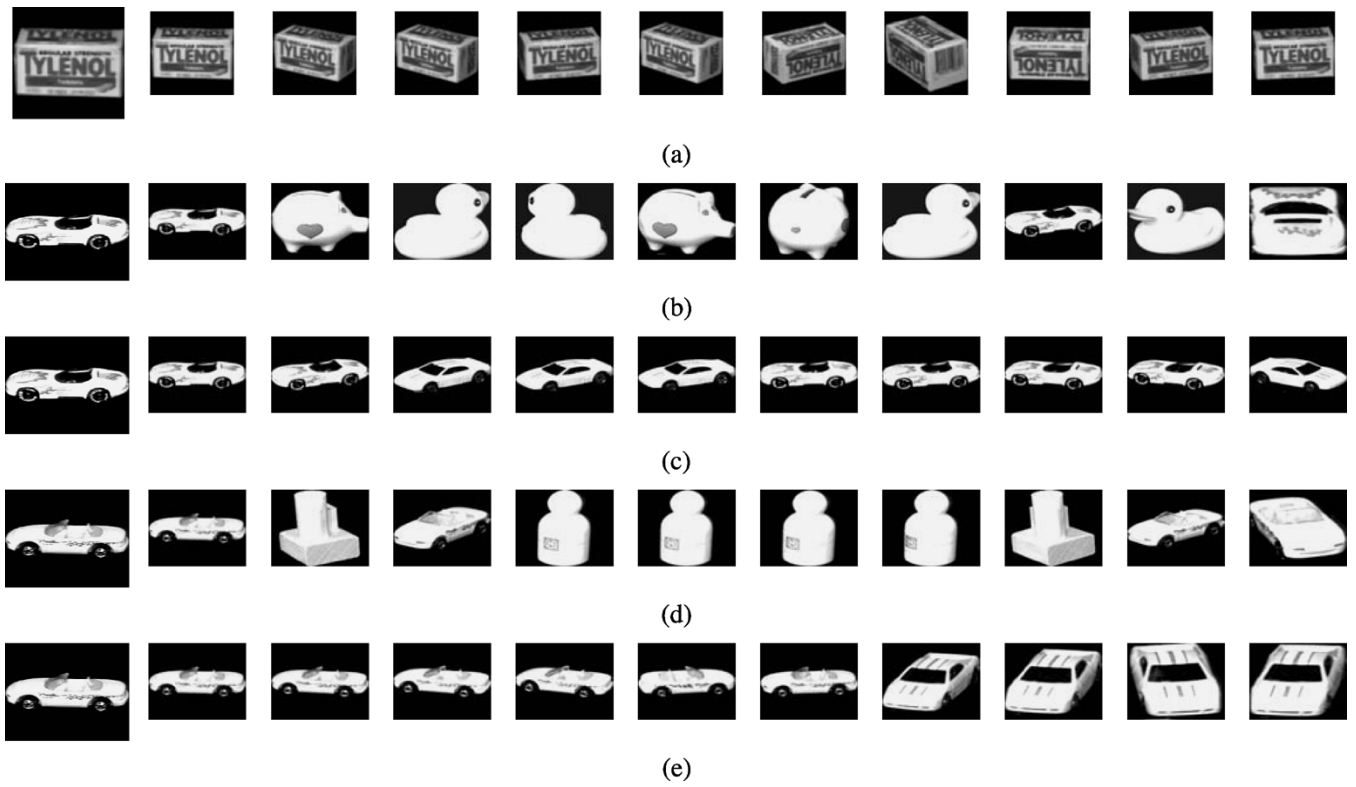


Fig. 9. Sample results with the query image at the left and the retrieved images shown from left to right according to descending rank when $r = 10$ for bucket search in Tables II and III. (a) Sample result showing a good retrieval with only the Euler vector. (b) Sample result with only Euler vector as the feature. (c) Sample result for the same query as in (b) with the Euler vector augmented by the shape features. (d) Sample result with only the Euler vector as the feature. (e) Sample result for the same query as in (d) with the Euler vector augmented by the shape features.

time computed as follows. If t_{ex} is the time taken for the exhaustive search and t_{kd} is the time taken for the bucket search, then the percentage savings in time is computed as $((t_{ex} - t_{kd})/t_{ex}) \times 100$. Table III reports the same results but with the two other shape features augmenting the Euler vector. Table IV and Table V give the comparisons of exhaustive search with range search in the cases where only the Euler vector is used as the feature and where the Euler vector is augmented by two shape features, respectively. It is observed that the average precision goes down with increasing α. This happens because with the increase in α the axis-parallel hyperboxes will increase in size and include more and more images of other objects. Lee and Street [24] used a 36-dimensional feature based on the centroid-radii model for a shape based query system on the same COIL database we used. Their query system uses an incremental feature weight learning method based on both the clustered database and relevance feedback. They report an initial success ratio of 56.7% when querying with a single cluster based on the 36-dimensional feature and when 71 images were ranked. The success ratio was then improved by including more clusters and using relevance feedback. Although our line of study in this paper differs from that proposed in [24], it can be seen that we achieve comparable results in terms of retrieval success. Our results for percentage of success with 71 images being ranked are 36.14% for exhaustive search, 29.29% using bucket search, and 38.66% for range search when the Euler vector is used alone. With two additional shape features, the success rises to 52.14% and 44.57% for the exhaustive and bucket searches respectively. We show that range search can be used to improve the success ratio further, *viz.* 69.79% ($\alpha = 0.4$), at the cost of a small increase in retrieval time. The average precision or success ratio in the case of range search increases because fewer number of irrelevant images are picked up. Fig. 9 shows the retrieved images after ranking for some queries when $r = 10$ for bucket search. Fig. 9(a) shows an example of good retrieval success when only Euler vector was used as the feature. Fig. 9(b) shows an example of a poor retrieval when only Euler vector was used. Only three objects (1st, 8th, 10th) out of ten were of the same object as the query object. The last object, though visually looking different, is of the same object as the query image, but with a different pose. With the shape feature augmentation, six objects (1st, 2nd, 6th, 7th, 8th, 9th) out of ten were of the same object as the query object as shown in Fig. 9(c). But the other failure objects retrieved bear a striking similarity with the query object. Similarly, Fig. 9(d) shows an example of a poor retrieval when only Euler vector was used. Only four objects (1st, 3rd, 9th, 10th) out of ten were of the same object as the query object. With the shape feature augmentation, six objects (1st, 2nd, 3rd, 4th, 5th, 6th) out of ten were of the same object as the query object as shown in Fig. 9(e). The other failure objects retrieved are however, very similar to the query object.

V. Conclusions and Discussions

A new combinatorial signature of a gray-tone image called the Euler vector is proposed. It may be used along with other

standard features in order to provide better characterization of an image. Since the Euler vector is topologically invariant, easily computable, and has strong discriminatory power, this feature may find many applications to CBIR. Experimental results demonstrating its robustness and efficacy of retrieval using this new feature have been presented. A modified Kd-tree can be employed to support easy and efficient retrieval using the Euler vector. It may be noted that the JPEG2000 image compression standard uses a wavelet transform and a bit-plane entropy coder to provide state-of-the-art compression [30]. Thus, it supports multiresolution in terms of scale and embedded quantization by bit-plane coding. Since the Euler vector is linked with the bit-plane representation and is topologically invariant, it can be tailored to support multiresolution in terms of scale. Deriving the Euler vector of the original image from the Euler vectors of the different wavelet subband images would be an interesting future problem. Another important problem would be to determine query ranges on the modified Kd-tree for retrieval, using the information on statistical distribution of features. Determination of the optimal values of $Minsize$, $Maxsize$, and $DistThreshold$ based on the system specification also requires further investigation.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their constructive suggestions and critical comments.

REFERENCES

[1] T. W. Anderson, *An Introduction to Multivariate Statistical Analysis*. New York: Wiley, 2003.

[2] R. Azencott, J. Wang, and L. Younes, "Texture classification using windowed Fourier filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 2, pp. 148–153, 1997.

[3] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R^*-tree: an efficient and robust access method for points and rectangles," in *Proc. ACM SIGMOD Int. Conf. Management of Data*, Atlantic City, NJ, 1990, pp. 322–331.

[4] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[5] P. J. Besl and R. C. Jain, "Three-dimensional object recognition," *ACM Comput. Surv.*, vol. 17, no. 1, pp. 75–145, 1985.

[6] A. Bishnu, B. B. Bhattacharya, M. K. Kundu, C. A. Murthy, and T. Acharya, "A pipeline architecture for computing the Euler number of a binary image," *J. Syst. Architect.*, 2005, to be published.

[7] ——, "Euler vector: a combinatorial signature for gray-tone images," in *Proc. Int. Conf. Information Technology: Coding and Computing*, Las Vegas, NV, 2002, pp. 121–126.

[8] K. Chakrabarti and S. Mehrotra, "The hybrid tree: an index structure for high dimensional feature space," in *Proc. 15th Int. Conf. Data Engineering*, 1999, pp. 440–447.

[9] R. T. Chin and C. R. Dyer, "Model-based recognition in robot vision," *ACM Comput. Surv.*, vol. 18, no. 1, pp. 67–108, 1986.

[10] V. Gaede and O. Günther, "Multidimensional access methods," *ACM Comput. Surv.*, vol. 30, no. 2, pp. 170–231, Jun. 1998.

[11] S. B. Gray, "Local properties of binary images in two dimensions," *IEEE Trans. Comput.*, vol. C-20, no. 5, pp. 551–561, May 1971.

[12] A. Guttman, "R-tree: a dynamic index structure for spatial searching," in *Proc. ACM SIGMOD Int. Conf. Management of Data*, Boston, MA, 1984, pp. 47–54.

[13] R. M. Haralick, "Statistical and structural approaches to texture," *Proc. IEEE*, vol. 67, no. 5, pp. 786–804, May 1979.

[14] G. R. Hjaltason and H. Samet, "Incremental Similarity Search in Multimedia Databases," Dept. Comput. Sci., Univ. Maryland, College Park, Tech. Rep. 4199, Nov. 2000.

[15] M. K. Hu, "Visual pattern recognition by moment invariants," *IRE Trans. Inform. Theory 8*, pp. 179–187, Feb. 1962.

[16] A. K. Jain, *Fundamentals of Digital Image Processing*. New Delhi: Prentice Hall of India, 1989.

[17] A. K. Jain and K. Karu, "Learning texture discrimination masks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 2, pp. 195–205, Feb. 1996.

[18] N. L. Johnson and S. Kotz, *Distributions in Statistics: Continuous Distributions–2*. New York: Wiley, 1970.

[19] N. Katayama and S. Satoh, "The SR-tree: an index structure for higher-dimensional nearest neighbor queries," in *Proc. Int. Conf. Management of Data, ACM SIGMOD*, May 1997, pp. 13–15.

[20] M. Khalil and M. M. Bayoumi, "Affine invariants for object recognition using the wavelet transform," *Pattern Recognit. Lett.*, vol. 23, no. 1–3, pp. 57–72, 2002.

[21] A. Khotanzad and Y. H. Hong, "Rotation invariant image recognition using features selected by a systematic method," *Pattern Recognit.*, vol. 23, no. 10, pp. 1089–1101, 1990.

[22] Z. Kohavi, *Switching and Finite Automata Theory*, 2nd ed. New York: McGraw-Hill, 1978.

[23] M. Kunt, "Source coding of x-ray pictures," *IEEE Trans. Biomed. Eng.*, vol. BME-25, no. 2, pp. 121–138, Mar. 1978.

[24] K.-M. Lee and W. N. Street, "Incremental feature weight learning and its application to a shape-based query system," *Pattern Recognit. Lett.*, vol. 23, no. 7, pp. 865–874, 2002.

[25] P. C. Mahalanobis, "On tests and measures of group divergences," *J. and Proc. Asiatic Soc. Bengal*, pp. 541–548, 1930.

[26] S. A. Nene, S. K. Nayar, and H. Murase, "Columbia Object Image Library: COIL-100," Dept. Comput. Sci., Columbia Univ., New York, Tech. Rep. CUCS-006-96, Feb. 1996.

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

[28] G. Pass and R. Zabith, "Comparing images using joint histograms," *Multimedia Syst.*, vol. 7, pp. 234–240, 1999.

[29] E. Persoon and K. S. Fu, "Shape discrimination using Fourier descriptors," *IEEE Trans. Syst. Man. Cybern.*, vol. SMC-7, pp. 170–179, 1977.

[30] M. Rabbani and D. S. Cruz, "The JPEG2000 still-image compression standard," in *Int. Conf. on Image Processing (ICIP)*, Thessaloniki, Greece, Oct. 2001, [Online] Available: http://jj2000.epfl.ch/jj_publications/index.html.

[31] T. H. Reiss, "Recognizing planar objects using invariant image features," in *Lecture Notes in Computer Science*. Berlin, Germany: Springer Verlag, 1993, vol. 676.

[32] J. T. Robinson, "The K-D-B-tree: a search structure for large multidimensional dynamic indexes," in *Proc. ACM SIGMOD Int. Conf. Management of Data*, Ann Arbor, MI, 1981, pp. 10–18.

[33] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*. New York: Academic, 1982.

[34] J. A. Rushing, H. S. Ranganath, T. H. Hinke, and S. J. Graves, "Using association rules as texture features," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 8, pp. 845–858, Aug. 2001.

[35] C. Schmid and R. Mohr, "Local gray-value invariants for image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 5, pp. 530–535, May 1997.

[36] T. Sellis, N. Roussopoulos, and C. Faloutsos, "The R^+-tree: a dynamic index for multidimensional objects," in *Proc. Int. Conf. Very Large Databases*, 1987, pp. 507–518.

[37] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Image retrieval at the end of early years," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 12, pp. 1349–1380, Dec. 2000.

[38] A. Stavrianopoulou and V. Anastassopoulos, "The Euler feature vector," in *Proc. 15th Int. Conf. Pattern Recognition (ICPR)*, vol. 3, 2000, pp. 7034–7036.

[39] M. J. Swain and B. H. Ballard, "Color indexing," *Int. J. Comput. Vis.*, vol. 7, no. 1, pp. 11–32, 1991.

[40] C. H. Teh and R. T. Chin, "On image analysis by the method of moments," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, no. 4, pp. 496–513, Jul. 1988.

[41] R. C. Veltkamp, H. Burkhardt, and H.-P. Kriegel, *State-of-the-Art in Content-Based Image and Video Retrieval*. Dordrecht, The Netherlands: Kluwer, 2001.

[42] R. C. Veltkamp and M. Tanase, "Content-Based Image Retrieval Systems: A Survey," Univ. Utrecht, Utrecht, The Netherlands, Tech. Rep. UU-CS-2000-34. [Online] Available: http://ftp.cs.ruu.nl/research/techreps/aut/remcov.html.

[43] D. A. White and R. Jain, "Similarity indexing with the SS-tree," in *Proc. 12th Int. Conf. Data Engineering*, New Orleans, LA, Feb. 1996, pp. 516–523.

[44] C. T. Zahn and R. Z. Roskies, "Fourier descriptors for plane closed curves," *IEEE Trans. Comput.*, vol. C-21, pp. 269–281, 1972.

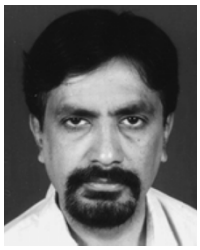
Arijit Bishnu received the B.E. degree in electrical engineering from Burdwan University, India, in 1995, the M. Tech. degree in computer science and the Ph.D. degree, both from the Indian Statistical Institute, Kolkata, India, in 1998 and 2003, respectively.

Currently he is an Associate in the School of Information Sciences, Japan Advanced Institute of Science and Technology, Ishikawa. His research interests include image processing, digital geometry, and combinatorial optimization.

Bhargab B. Bhattacharya received the B.Sc. degree in physics from the Presidency College, Calcutta, India, the B.Tech. and M.Tech. degrees in radiophysics and electronics, and the Ph.D. degree in computer science, all from the University of Calcutta, India.

Since 1982, he has been on the faculty of the Indian Statistical Institute, Calcutta, where he is a Full Professor. He held visiting positions at the University of Nebraska-Lincoln during 1985–1987 and 2001–2002, at the Institute of Informatics, the University of Potsdam, Potsdam, Germany during 1998–2000, and at the Indian Institute of Technology, Kharagpur, in 2005. His research interests include logic synthesis and testing of VLSI circuits, physical design, graph and geometric algorithms, and image processing architecture. He has published more than 150 papers in archival journals and refereed conference proceedings, and holds four U.S. patents.

Dr. Bhattacharya is a Fellow of the Indian National Academy of Engineering. He is on the Editorial Board of the *Journal of Circuits, Systems, and Computers* (World Scientific, Singapore) and the *Journal of Electronic Testing Theory and Applications* (JETTA).



Malay K. Kundu received the B. Tech., M. Tech., and Ph.D. (Tech.) degrees, all in radiophysics and electronics, from the University of Calcutta, Calcutta, India.

In 1982, he joined the Indian Statistical Institute (ISI), Calcutta, as a Faculty Member, where he is currently a Full Professor of the Machine Intelligence Unit. His research interests include image processing and analysis, image compression, digital watermarking, wavelets, fractals, VLSI design for digital imaging, and soft computing. He has contributed about 80 research papers to well-known and prestigious archival journals, international refereed conferences, and as chapters in monographs and edited volumes. He holds four U.S. patents and is co-author of *Soft Computing for Image Processing* (Heidelberg, Germany: Physica-Verlag, 2000).

Dr. Kundu received the Sir J. C. Bose memorial award in 1986 and the prestigious VASVIK award in 1999 for industrial research in electronic sciences and technology. He is a Fellow of the National Academy of Sciences, India, and a Fellow of the Institute of Electronics and Telecommunication Engineers, India.

C. A. Murthy received the B. Stat (Hons), M. Stat., and Ph.D. degrees from the Indian Statistical Institute (ISI), Calcutta.

He was a Visiting Professor at Michigan State University, East Lansing, in 1991–1992, and at Pennsylvania State University, University Park, in 1996–1997. Currently, he is a Professor at the Machine Intelligence Unit of ISI. His fields of research interest include pattern recognition, image processing, machine learning, neural networks, fractals, genetic algorithms, wavelets, and data mining.

Dr. Murthy received the Best Paper Award in computer science in 1996 from the Institute of Engineers, India. He received the Vasvik Award for industrial research in electronic sciences and technology in 1999, along with his two colleagues. He is a Fellow of the Indian National Academy of Engineering.

Tinku Acharya (M'96–SM'01) received the B.Sc. (Hons.) degree in physics and the B.Tech. and M.Tech. degrees in computer science from the University of Calcutta, Calcutta, India, and the Ph.D. degree in computer science from the University of Central Florida, Orlando.

He is currently Senior Executive Vice President and Chief Science Officer of Avisere Inc., Tucson, AZ. Since 1997, he has also been an Adjunct Professor in the Department of Electrical Engineering, Arizona State University, Tempe. He was a Principal Engineer at Intel Corporation (1996–2002), a Consulting Engineer at AT&T Bell Laboratories (1995–1996), a Research Faculty Member at the Institute of Systems Research, University of Maryland at College Park (1994–1995), and held Visiting Faculty Positions at the Indian Institute of Technology (IIT), Kharagpur. He is an inventor of 80 U.S. patents and 15 European patents in diverse areas of biometrics, data compression, multimedia computing, electronic imaging, VLSI architectures, and has more than 50 patents pending. He has contributed to over 70 refereed technical papers. He is the author of *Data Mining: Multimedia, Soft Computing and Bioinformatics* (New York: Wiley, 2003) and *JPEG2000 Standard for Image Compression: Concepts, Algorithms, and VLSI Architectures* (New York: Wiley, 2004).

Dr. Acharya served on the U.S. National Body of the JPEG2000 Standards Committee (1998–2002). He was nominated the "Most Prolific Inventor" by Intel Corporation Worldwide in 1999 and "Most Prolific Inventor" by Intel Corporation's Arizona site for five consecutive years (1997–2001). He is a Life Fellow of the Institution of Electronics and Telecommunication Engineers.