Stacked Euler Vector (SERVE): A Gray-Tone Image Feature Based on Bit-Plane Augmentation

Arijit Bishnu and
Bhargab B. Bhattacharya, *Sr. Member*, *IEEE*

Abstract—A new combinatorial feature called Stacked Euler Vector (*SERVE*) is introduced to characterize a gray-tone image. *SERVE* comprises a four-tuple, where each element is an integer representing the Euler number of the partial binary image formed by certain pixel overlap relations among the four most significant bit planes of the gray-tone image. Computation of *SERVE* is simple, fast, and does not involve any floating point operation. *SERVE* can be used to augment other features to improve the performance of image retrieval significantly. Experimental results on the COIL database are reported to demonstrate its performance.

Index Terms—Euler number, bit-plane graph, combinatorial feature, CBIR.

1 Introduction

A compact representation of image features is highly desirable for efficient management of the image database, search, and retrieval. Typical features of an image can be broadly classified into two types: 1) geometric features and 2) luminance signature [12]. With the emergence of the Internet, content-based image retrieval (CBIR) has recently become important. A good survey of the existing CBIR systems has appeared in [13], [14]. The features used by most of the systems comprise low-level features. Determination of a compact set of parameters for a gray-tone image is thus highly needed, which is easy to compute and which remains invariant under various transformations.

In this work, we define a new feature of a gray-tone image called *Stacked Euler vector (SERVE)*, which significantly improves upon a previous work [2]. For a binary image, the *Euler number* (genus) is a well-known geometric feature [3], [11]. See [1] and the references therein for more details. To characterize a gray-tone image using similar topological features, we consider its four most significant binary bit-planes with pixel intensity values (integers) lying in the range [0, 255] for defining *SERVE*. A bit-plane is first modified (augmented) using the information present in the more significant bit-planes. We then derive *SERVE*, which is a four-tuple representing the Euler numbers of these four modified bit-planes. *SERVE* deriving its definition from the Euler number is also topologically invariant. It is shown that *SERVE* enhances retrieval success remarkably, particularly for object or logo type of gray-tone images. In our experiments, the COIL database [7] is used. The objective of this work is not however, to achieve state-of-the-art image classification on COIL, which is available elsewhere [8].

The rest of the paper is organized as follows: Section 2 presents a review on the Euler number and its relation to bit-plane graphs. In Section 3, the concept of *SERVE* is introduced based on bit-plane augmentation. In Section 4, a discussion on *aggregating* and *discriminating* features and experimental results on retrieval using *SERVE* are presented. Discussions and conclusions appear in Section 5.

2 Euler Number and the Bit-Plane Graph

The *Euler number* (genus) of a binary image I_B in 2D is defined as the difference of the number (P) of connected components (objects) and the number (H) of holes [3]. A generalized version of this concept known as *Euler-Poincare Characteristic (EPC)* is widely used in differential topology [5] as an object descriptor. To compute the Euler number of a 2D binary image, we assume the conventional notion of eight-connectivity for objects and four-connectivity for holes [11]. A *run* in a row (column) of the pixel matrix is defined to be a maximal sequence of consecutive 1's in that row (column). Two runs appearing in two adjacent rows (columns) are said to be *neighboring* each, if at least one pixel of a run is in the eight neighborhood of a pixel of the other run. Let $R(i)$ and $O(i)$ denote the number of such runs and neighboring runs between the ith row and the $(i-1)$th row. Then, the Euler number of a binary $(N \times M)$ image can be computed using the relation [11] (see [1] for proof).

$$E(I) = P - H = \sum_{i=1}^{N} R(i) - \sum_{i=2}^{N} O(i). \quad (1)$$

The above two formulations given in [3] and [11] are equivalent in the light of the classical Euler's formula for a connected planar embedded graph, called the bit-plane graph, defined below. This graph can be thought of as an extension and simplification of the graph-based representation of binary images by DiZenzo et al. [15].

For each connected component j of the image I_B, a bit-plane graph $G_j = \{V_j, A_j\}$, $j = 1, \ldots, P$ is constructed as follows: Each node in V_j and each edge in A_j correspond to a run and a neighboring run in component j of G_j, respectively. So, $|V_j| = n_j$ and $|A_j| = o_j$, where n_j and o_j are the number of runs and that of neighboring runs in component j of the graph G_j. For example, the graph corresponding to the binary image (Fig. 1a) is shown in Fig. 1b. Clearly, $V_j \cap V_k = \phi$ and $A_j \cap A_k = \phi$, as any run in component k can have no overlap or intersection with another run in component j $(k \neq j)$. From the definition, it follows that the graph G_j will be planar and bipartite [4]. However, the row-major or column-major construction of the bit-plane graph defined as above may not be isomorphic [4] to each other. See Fig. 1. But, we prove that they will have the same Euler number. The following observation is easy to follow:

Observation 1. *The number of bounded faces [4] (f_j) in the bit-plane graph (G_j) is equal to the number of holes in the image irrespective of whether the graph is constructed in row-major or in column-major fashion.*

Hence, from the Euler's formula for planar graphs [4], we have $n_j - o_j = 1 - f_j$. Since f_j remains same for both the row-major and column-major bit-plane graphs, the difference $n_j - o_j$ is the same in both the cases. Thus, the Euler number E_j of the connected component j, which is $n_j - o_j$, remains invariant for both row-major or column-major bit-plane graphs. The Euler number also satisfies the *local additive property* [3]. Thus, with no intersection between any two maximally connected components, the Euler number $E(I)$ of the entire image can be expressed as $\sum_{j=1}^{P} E_j = \sum_{j=1}^{P} n_j - \sum_{j=1}^{P} o_j = \sum_{i=1}^{N} R(i) - \sum_{i=2}^{N} O_i$ (see (1)). Equation (1) can be used to compute the Euler number without constructing the graph.

3 Stacked Euler Vector for a Gray-Tone Image

3.1 Bit Planes and Stacked Euler Vector

A gray-tone image I_G is usually represented as an $(N \times M)$ matrix, where the intensity $I(x, y)$ of each pixel at (x, y) is an integer lying between [0, 255]. Thus, a eight-bit binary vector, $(b_7, \ldots, b_0)$ represents the intensity value of each pixel. The image may now be considered as an overlay of eight binary bit-planes. The first *four most significant bit planes*, B_7, B_6, B_5, and B_4 (corresponds to (b_7, b_6, b_5, b_4)), are considered as they contain most of the information of the image. However, the defintion can be extended to considering all bit-planes.

The *Euler vector* of a gray-tone image is defined earlier [2] as a four-tuple $\{E_7, E_6, E_5, E_4\}$, where E_j is the Euler number of the

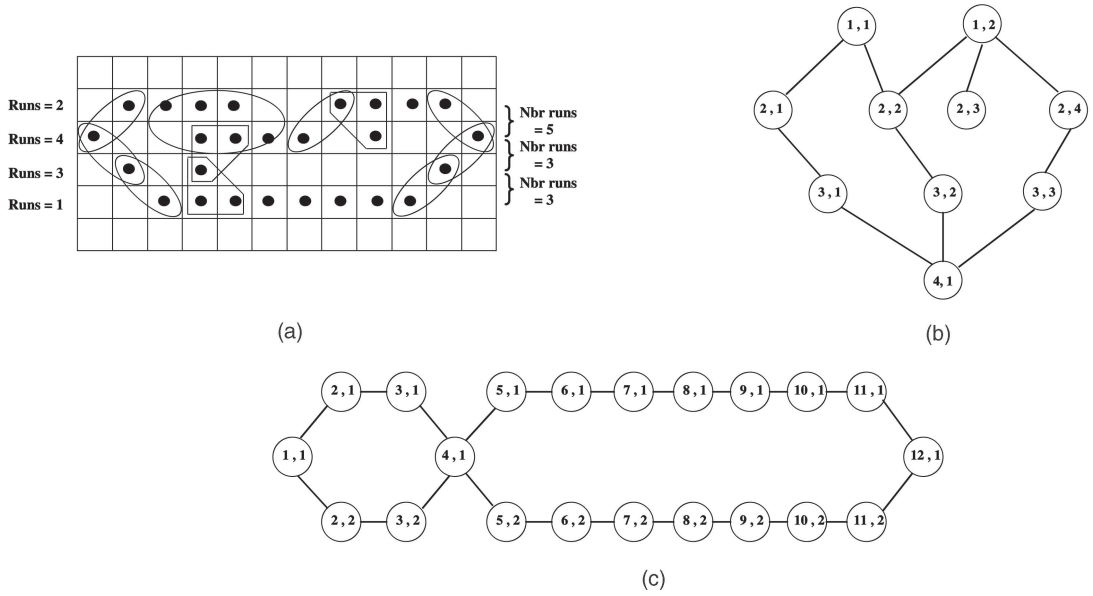


Fig. 1. Illustration of runs and neighboring runs showing that the corresponding graphs are not isomorphic. (a) $\text{Euler number} = \sum \text{runs} - \sum \text{neighboring runs} = -1$. (b) Row-major bit-plane graph; the tuple (α, β) denotes run number β in row α. (c) Column-major bit-plane graph; (γ, β) denotes run number β in column γ.

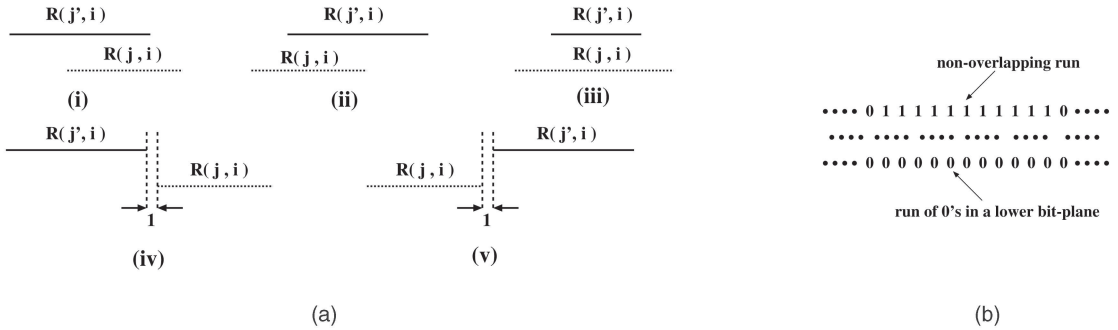


Fig. 2. Overlapping and nonoverlapping runs. (a) Overlapping runs. (b) Nonoverlapping runs.

partial two-tone image formed by the jth bit-plane, $4 \leq j \leq 7$, corresponding to the reflected gray code representation of the intensity values. However, Euler numbers defined merely on isolated bit-planes as in [2] do not capture any interrelationship across the bit-planes present in the actual intensity level structure of the gray-level image. To find out an ideal spatial feature based on bit-planes, we exploit certain "connectivity" that exists across the bit-planes to define the augmentation of bit-planes.

The bit-plane B_7 and its corresponding graph G_7 give an initial structure of the original image. Now, as we go down the bit-planes, each bit-plane and its corresponding graph augment further details. The augmentation is described next.

3.2 Augmentation of Bit-Plane Graphs

Let $R(j,i)$ be a run in the ith row of the jth bit-plane (B_j) starting at column s_{ji} and ending at column e_{ji}, $1 \leq s_{ji} \leq e_{ji} \leq M$.

Run overlap. Let a run $R(j',i)$ start at column $s_{j'i}$ and end at $e_{j'i}$. The run $R(j',i)$ is said to overlap $R(j,i)$ for $(4 \leq j < j' \leq 7)$, if any one of the following conditions hold:

1. $s_{j'i} \leq s_{ji} \leq e_{j'i}$,
2. $s_{j'i} \leq e_{ji} \leq e_{j'i}$,
3. $s_{ji} \leq s_{j'i} \leq e_{j'i} \leq e_{ji}$,
4. $s_{ji} - e_{j'i} = 1$, and
5. $s_{j'i} - e_{ji} = 1$.

Conditions 4 and 5 are for eight-connectivity; for four-connectivity, they are not required. Any run $R(j,i)$ is a *nonoverlapped run* if no such $R(j',i)$ exists for $R(j,i)$. See Figs. 2a and 2b for examples of overlapping and nonoverlapping runs. In Fig. 2a, $R(j',i)$s denote overlapping runs for $R(j,i)$s.

Augmented graphs for an augmented bit-plane. Each bit-plane B_j has a bit-plane graph G_j associated with it. G_j^{aug} represents the augmented graph for an augmented bit-plane B_j^{aug}. Clearly, for B_7, $G_7^{aug} = G_7$. For any other bit-plane $B_j (4 \leq j < 7)$, the augmented graph G_j^{aug} is constructed as follows: For runs $R(j,i)$ in the bit-plane B_j, we determine the overlapping runs $R(j',i) (4 \leq j < j' \leq 7)$. Then, we create new runs (alternately, nodes of G_j^{aug}) of B_j^{aug} using all overlapping runs of $R(j,i)$s. Also, we create new runs (alternately, nodes of G_j^{aug}) of the augmented bit-plane as an union of the corresponding nonoverlapping runs. This finishes the construction of the nodes (alternately, runs) of the augmented graph G_j^{aug}. The edges of G_j^{aug} are formed by the usual definition of neighboring runs between two augmented runs as defined in Section 2. See Figs. 3a, 3b, and 3c for the original bit-planes, augmented bit-planes, and the augmented bit-plane graphs.

Note that, if all bit-planes $B_{j'}$, $(4 \leq j < j' \leq 7)$ are stacked on B_j to create the new bit-plane B_j^{aug}, then we can find the dominating run and neighborhood run relations between $B_{j'}$s and B_j only by looking at the run and neighborhood runs of B_j^{aug}. Thus, B_j^{aug} can be derived by just "OR"-ing the bit-planes $B_{j'}$s with B_j. The augmented graph G_j^{aug} is then constructed corresponding to the bit-plane B_j^{aug}. The above observation, based on which we define *SERVE*, is summarized below.

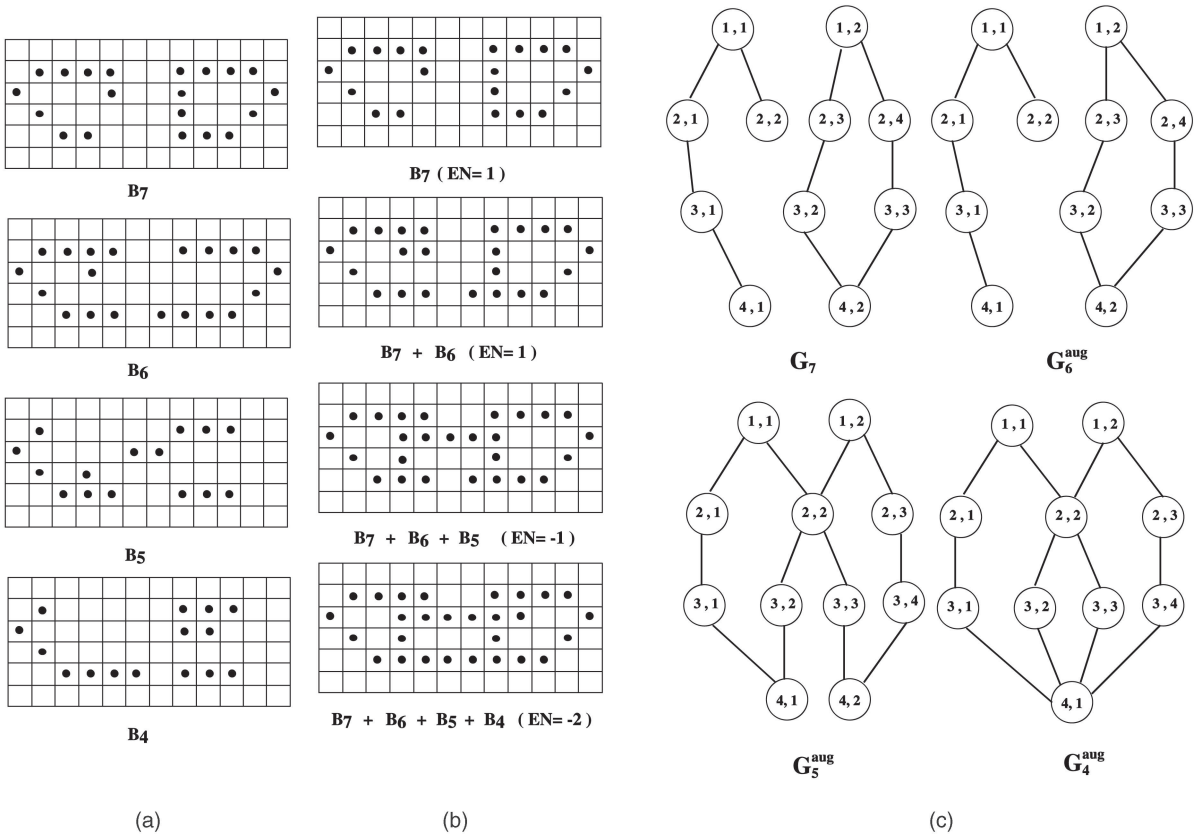


Fig. 3. Normal bit-planes, OR-ed bit-planes with their graphs, and *SERVE*. (a) Original bit-planes. (b) OR-ed bit-planes, $SERVE = \{1, 1, -1, -2\}$. (c) Augmented graphs of the OR-ed bit-planes.

Observation 2. *For any bit-plane B_j, the augmented bit-plane B_j^{aug} is constructed as: $B_j^{aug} = B_j + B_{j+1} + \ldots + B_7$, where $4 \leq j \leq 7$; + denotes bit-wise logical OR operation.*

SERVE (Euler Vector of type-1). The *Stacked Euler Vector (SERVE)* of a gray-tone image is a four-tuple $\{E_7, E_6, E_5, E_4\}$, where E_j is the Euler number of the partial two-tone image represented by the augmented bit-plane B_j^{aug} (see Figs. 3 and 4 for illustrations).

SERVE remains invariant under the transformations in which the Euler number remains invariant. The proof can be deduced as a simple extension of the triangulation independent topology preserving property of the Euler number as shown in [3]. *SERVE*, as a signature of a gray-tone image, uses both geometric and intensity level characterization [12]. An on-chip VLSI implementation of *SERVE* can also be done following the design in [1].

4 Experiments with Stacked Euler Vector (SERVE)

4.1 Aggregating and Discriminating Features

The problem of image feature extraction can be viewed as a function $\mathcal{F}$ that maps a set of images I to a multidimensional feature space $\mathbb{R}^n$ as $\mathcal{F} : I \rightarrow \mathbb{R}^n$. In many CBIR applications, $\mathcal{F}$ being only a bijective mapping is not much of a benefit. Based mostly on human perceptions (which is not mathematically well defined), the set of images I can be classified into a set of equivalence classes $I_1, I_2, \ldots, I_k$. Based on some appropriate distance function, if there exists a relation that divides the multidimensional feature space $\mathbb{R}^n$ into equivalence classes $R_1, R_2, \ldots, R_k$ such that there exists a bijective mapping between equivalence classes $I_i \in I$ and $R_i \in \mathbb{R}^n$, $i = 1, 2, \ldots, k$, then we can say that the feature $\mathcal{F}$ is "*good*." "*Good*" features are both *aggregating* and *discriminating*. Aggregating (discriminating) features are those for which images belonging to the same (different) equivalence classes have feature values "*close*" ("*far*") relative to a distance function in $\mathbb{R}^n$. See [8] and the references therein for efforts in finding such "*good*" features. But, the features used are mostly complex and computationally expensive. On the other hand, most of the existing CBIR systems use low-level, fast-to-compute features [14]. Low-level features suffer from the problem that they are not both aggregating and discriminating. If both are used together, the discriminating feature will obviously have a dominating contribution to the distance measure, making the aggregate feature redundant. A better solution can be obtained by breaking down the retrieval process into two steps. First, retrieve a set of images $I_A \in I$ based on the aggregating feature. Next, use the discriminating feature on I_A to retrieve the final set of images. In Section 3, we show experimentally that *SERVE* is a more (less) aggregating (discriminating) feature than invariant moments and both can be combined in the said two-step process to improve retrieval efficiency. We chose invariant moments as a feature because of its simplicity and wide use to place *SERVE* in its proper perspective. We also provide another experiment with some shape features.

4.2 Image Database

For our experiments, the standard COIL-20 image database [7] of 20 objects is used. Images of all the 20 objects were taken at pose intervals of 5 degrees. This corresponds to a total of 1,440 gray-scale images. Details can be found in [7]. The images shown in Fig. 5 are numbered starting from the top left in a row-major fashion, e.g., the second to last (car) is "obj19," the last image in the second row ("vaseline") is "obj10."

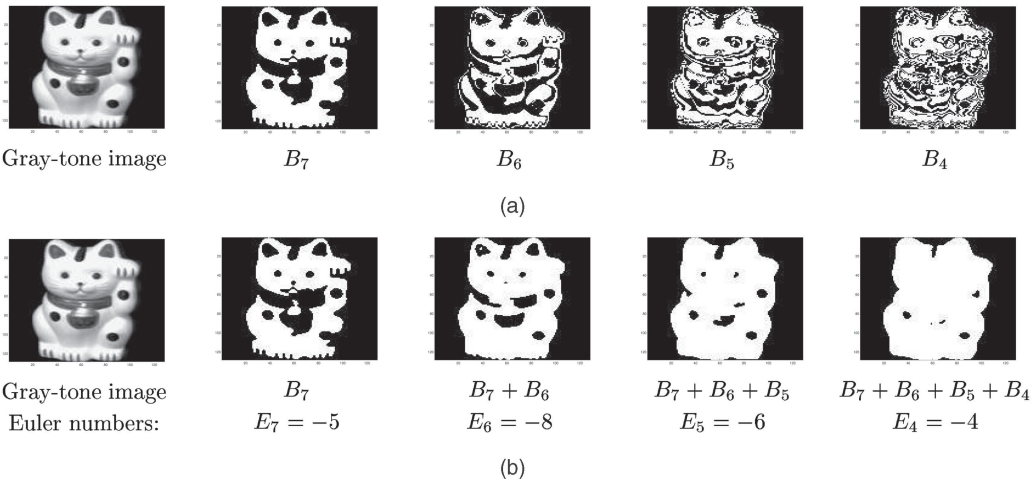


Fig. 4. Illustration of normal and OR-ed bit-planes and *SERVE* for an original image. (a) Original bit-planes B_7, B_6 B_5, and B_4. (b) OR-ed bit-planes and *SERVE*; the *SERVE* here is $\{-5, -8, -6, -4\}$.

4.3 Combining Serve with Invariant Moments for Image Retrieval

Among many image features that are invariant under shifts, changes of scale, and rotation, and to general linear transformations, the set of invariant moments [6], [10] attracted wide attention because of its simplicity of computation. Reiss [10] presented a corrected version of the set of moments invariant to the general linear transformation presented earlier by Hu [6]. In this paper, we show that a simple feature like SERVE can enhance image retrieval efficiency significantly when combined with the set of four invariant moments proposed by Reiss [10].

4.3.1 Measures for Aggregating and Discriminating Features

The 72 images pertaining to each object are partitioned into equivalence classes based on the equality of *SERVE* and moments separately. As seen from Table 1, moments are found to be different for each image of the same object, but *SERVE* for some classes may have the same value for a group of images, e.g., obj15 in Table 1 has only eight different *SERVE* values. Considering either *SERVE* or moments as a feature, each image is a point in a four-dimensional space. With the distances representing the normal Euclidean distances, we calculated the average distance (D_{obj}) ($\binom{72}{2}$ distances) between two images of the same object (intra-object distance). The corresponding coefficients of variances of intra-object distances are also given in Table 1. They indicate that the *SERVE* of different poses of the same object tend to remain close, whereas the moment may discriminate different images. Based on these observations and discussions in Section 1, we devise a series of retrieval experiments by combining SERVE with moments to improve retrieval success.

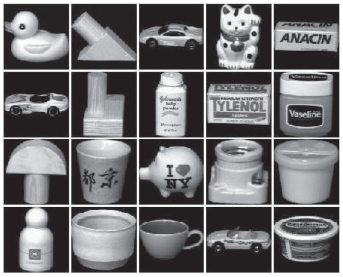


Fig. 5. COIL objects.

4.3.2 Retrieval Experiments

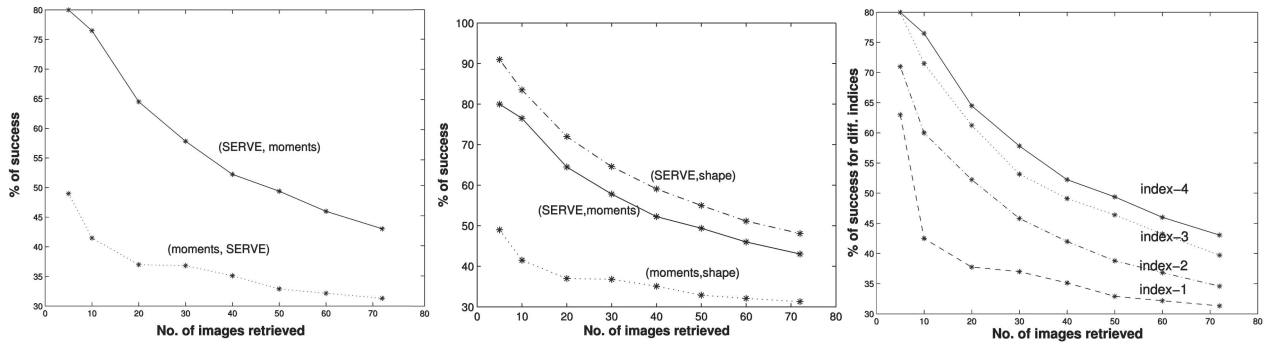
To test the combined feature of *SERVE* and moments, we perform a retrieval experiment by first using *SERVE* followed by moments, and vice versa. Given a query image, all the images (say, m) corresponding to the first n distinct distances from the query image are chosen. These m images are then ranked using their moments. A success results if we retrieve an image of the same object in COIL as the query. We also perform the experiment the other way round, where moments are used first followed by *SERVE*. The results are shown in the graph in Fig. 6a, where n is varied from 5 to 72. The curve marked (*SERVE*, moments) is the one when *SERVE* is used first followed by moments and it clearly shows better performance than the other case. Even when $n = 72$, a retrieval success of 43 percent was reported. This graph best determines the relation of *SERVE* and moments. Further, compared to moments alone, retrieval based on the combined feature of (*SERVE*, moments) shows a marked improvement. We also experimented with an eight-dimensional feature vector comprising the four-tuple *SERVE* and the four invariant moments, instead of the feature augmentation described earlier. It is interesting to observe that the results were almost the same as the case when moments were used first followed by *SERVE*.

SERVE can be used with other possible discriminating features for better retrieval. We devised another set of experiments where the following simple shape features were used instead of moments. A gray-level 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. Again, given a query image, all the images (say, m) corresponding to the first n distinct distances from the query image are chosen. These m images are then ranked using these two features. The retrieval success improves compared to use of *SERVE* followed by moments, as shown in Fig. 6b.

In Section 3, *SERVE* is defined as a 4-tuple $\{E_7, E_6, E_5, E_4\}$. We may call it as index-4 *SERVE*. We have also studied how the success ratio continues to improve with inclusion of each E_i starting from the most significant element $\{E_7\}$. We performed the same retrieval experiment as described earlier considering index-1 feature (i.e., $\{E_7\}$) only, index-2 feature (i.e., $\{E_7, E_6\}$), index-3 (i.e., $\{E_7, E_6, E_5\}$), and, finally, index-4 ($\{E_7, E_6, E_5, E_4\}$), i.e., *SERVE*. The results shown in Fig. 6c demonstrate that the success ratio improves steadily with increasing indices but with diminishing returns. This justifies the choice of 4-tuple *SERVE* as a feature vector. It is also interesting to notice the retrieval results (Fig. 7)

TABLE 1
Comparative Results

COIL objects	Equivalence classes of the objects		Co-efficient of variance of D_{obj}s		Ratio of CPU time for computing moments to *SERVE*	Retrieval results for (*SERVE*, moments) for $n = 10$	
	Moments	*SERVE*	Moments	*SERVE*		No. of images retrieved	No. of images of same obj.
obj1	72	37	79.3	49.0	4.0	10	6
obj2	72	18	187.9	97.9	2.7	10	4
obj3	72	72	252.5	44.5	5.0	10	4
obj4	72	67	89.4	57.1	4.8	10	10
obj5	72	72	148.2	57.2	5.7	10	10
obj6	72	70	330.9	46.0	2.6	10	7
obj7	72	46	72.4	78.7	4.7	10	3
obj8	72	51	130.3	68.9	4.9	10	10
obj9	72	71	163.7	46.4	4.9	10	10
obj10	72	69	147.8	61.3	4.8	10	10
obj11	72	23	81.4	82.9	4.9	10	3
obj12	72	55	77.9	65.9	2.7	10	10
obj13	72	69	91.8	45.4	4.8	10	6
obj14	72	66	93.2	41.4	4.8	10	8
obj15	72	8	75.2	76.9	4.9	10	9
obj16	72	16	70.7	84.0	5.0	10	7
obj17	72	15	80.5	76.1	4.9	10	5
obj18	72	41	135.1	46.6	4.8	10	10
obj19	72	67	241.8	39.7	5.0	10	10
obj20	72	72	82.2	41.0	2.8	10	10



(a) (b) (c)

Fig. 6. Retrieval results when *SERVE* was used followed by moments and vice versa on the COIL database. (SERVE, moments) means *SERVE* was used first followed by moments and vice versa for (moments, SERVE). (a) Retrieval successes of *SERVE* and moment augmentation. (b) Retrieval successes of *SERVE* and shape feature augmentation. (c) Retrieval successes for different indices of *SERVE*.

corresponding to $n = 10$ for some cases (see Table 1) with (*SERVE*, moments) as the feature vector. This corresponds to the case $n = 10$ in the curve marked (*SERVE*, moments) in the graph in Fig. 6a. It may be observed that the failure images of the other retrieved objects show a striking similarity in either shape or luminance to the query object. Concerning the CPU time, Table 1 shows that *SERVE* is faster than moment computation.

4.4 Efficiency of SERVE Based on Bit-Plane Augmentation

To justify empirically the use of bit-plane augmentation as in *SERVE*, we performed some experiments comparing it with few other types of Euler vector as defined below for a gray-tone image.

Euler Vector of type-2. This is defined as a four-tuple $\{E_7, E_6, E_5, E_4\}$, where E_j is the Euler number of the partial two-tone image formed by the jth bit-plane, $4 \leq j \leq 7$, corresponding to the binary code representation of the intensity values.

Euler Vector of type-3. This is a four-tuple $\{E_7, E_6, E_5, E_4\}$, where E_j is the Euler number of the partial two-tone image formed by the jth bit-plane, $4 \leq j \leq 7$, corresponding to the reflected gray code representation of the intensity values. This definition was used earlier as a characteristic feature of a gray-tone image [2].

Euler Vector of type-4. This is a four-tuple $\{E_7, E_6, E_5, E_4\}$, where E_7 is the Euler number of bit-plane B_7 and any other E_j is the Euler number of the partial two-tone image formed by OR-ing the jth bit-plane with the $(j+1)$th bit-plane, $4 \leq j \leq 6$, corresponding to the binary code representation of the intensity values.

We have run our retrieval experiment considering the above three types of Euler vector and compared the results with *SERVE*. From Figs. 8a, 8b, and 8c, it is evident that the retrieval results based on *SERVE* (Euler vector of type-1) outperform significantly those obtained by the other three types of Euler vector including the earlier one reported in [2]. The intuitive justification in favor of *SERVE* lies in its improved capability of capturing connectivity information across the bit-planes of a gray-tone image.

5 Conclusions and Discussions

A new combinatorial signature for a gray-tone image, called the Stacked Euler vector (*SERVE*), is proposed. The definition is derived from the classical Euler number of a binary image and the augmented graphs derived by successive OR-ing of bit-planes. Computation of *SERVE* does not involve any floating point operation and, hence, can be accomplished very fast. Experimental results showing its use as an aggregating feature have been

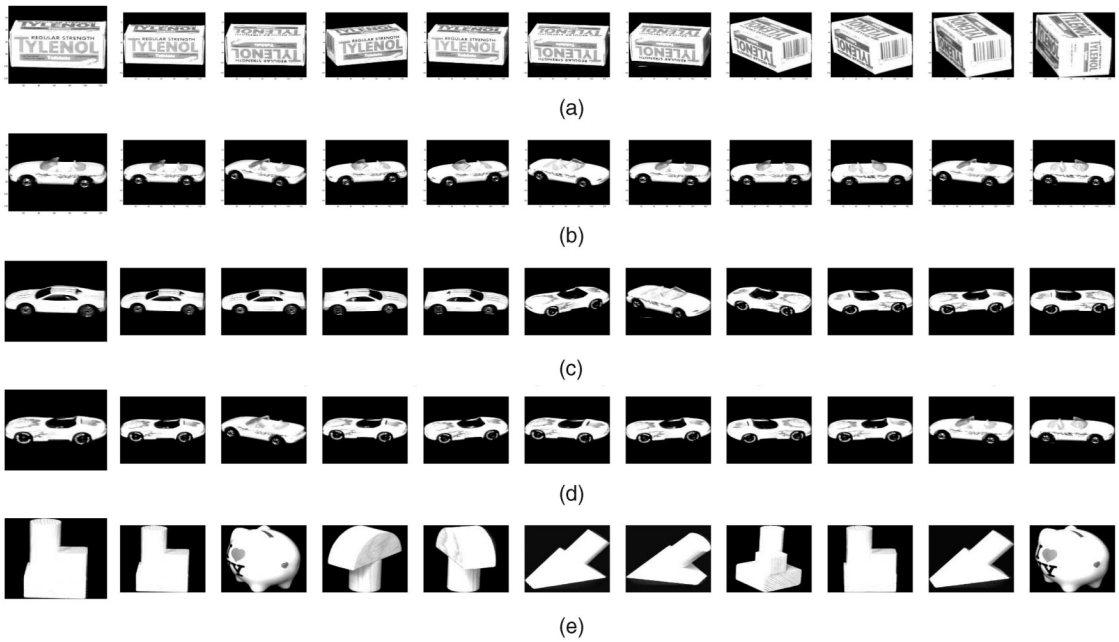


Fig. 7. Retrieval results corresponding to entries "obj9," "obj19," "obj3," "obj6," and "obj7" with the query image at the left and the retrieved images shown from left to right according to descending rank. (a) Retrieval with "obj9" as query where 10 objects are of "obj9." (b) Retrieval with "obj19" as query where 10 objects are of "obj19." (c) Retrieval with "obj3" as query where only four objects are of "obj3;" note the failure objects. (d) Retrieval with "obj6" as query where only seven objects are of "obj6;" note the failure objects. (e) Retrieval with "obj7" as query where only three objects are of "obj7;" note the failure objects.

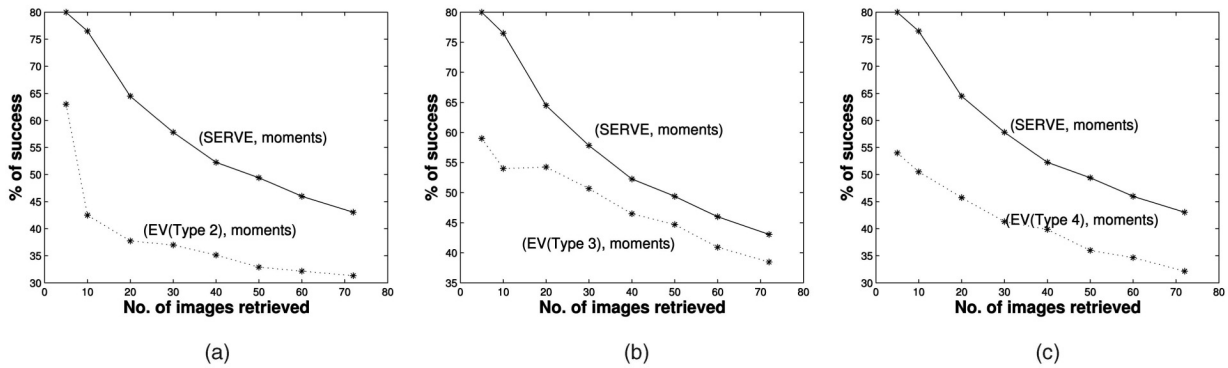


Fig. 8. Retrieval results on COIL database for different types of Euler vector versus *SERVE* both coupled with moments. As in Fig. 6, any legend $(\mathcal{X}, \mathcal{Y})$ means feature $\mathcal{X}$ was used first followed by $\mathcal{Y}$. (a) Euler vector of Type 2 versus *SERVE*. (b) Euler vector of Type 3 versus *SERVE*. (c) Euler vector of Type 4 versus *SERVE*.

reported. The JPEG2000 image compression standard uses a wavelet transform and a bit-plane entropy coder for compression [9]. As *SERVE* uses bit-planes and is topologically invariant, computing *SERVE* in the compressed domain of the image itself would be interesting.

REFERENCES

[1] 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. Systems Architecture,* vol. 51, pp. 470-487, 2005.

[2] A. Bishnu, B.B. Bhattacharya, M.K. Kundu, C.A. Murthy, and T. Acharya, "Euler Vector for Search and Retrieval of Gray-Tone Images," *IEEE Trans. Systems, Man, and Cybernetics, Part B,* vol. 35, pp. 801-812, Aug. 2005.

[3] S.B. Gray, "Local Properties of Binary Images in Two Dimensions," *IEEE Trans. Computers,* no. 5, pp. 551-561, May 1971.

[4] F. Harary, *Graph Theory.* Addison Wesley, 1972.

[5] R.J. Adler, *The Geometry of Random Fields.* John Wiley and Sons, 1981.

[6] M.K. Hu, "Visual Pattern Recognition by Moment Invariants," *IRE Trans. Information Theory 8,* pp. 179-187, Feb. 1962.

[7] S.A. Nene, S.K. Nayar, and H. Murase, "Columbia Object Image Library: COIL-100," Technical Report CUCS-006-96, Dept. of Computer Science, Columbia Univ., Feb. 1996.

[8] S. Obdrzälek and J. Matas, "Object Recognition Using Local Affine Frames on Distinguished Regions," *Proc. British Machine Vision Conf.,* pp. 113-122, Sept. 2002.

[9] M. Rabbani and D.S. Cruz, "The JPEG2000 Still-Image Compression Standard," course given at *Int'l Conf. Image Processing (ICIP),* Oct. 2001.

[10] T.H. Reiss, "The Revised Fundamental Theorem of Moment Invariants," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 13, no. 8, pp. 830-834, Aug. 1991.

[11] A. Rosenfeld and A.C. Kak, *Digital Picture Processing.* Academic Press Inc., 1982.

[12] C. Schmid and R. Mohr, "Local Grayvalue Invariants for Image Retrieval," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 5, pp. 530-535, May 1997.

[13] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Image Retrieval at the End of Early Years," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 12, pp. 1349-1380, Dec. 2000.

[14] R.C. Veltkamp and M. Tanase, "Content-Based Image Retrieval Systems: A Survey," Technical Report, UU-CS-2000-34, Universiteit Utrecht, 2000.

[15] S. DiZenzo, L. Cinque, and S. Levialdi, "Run-Based Algorithms for Binary Image Analysis and Processing," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, no. 1, pp. 83-89, Jan. 1996.

[16] A.K. Jain, *Fundamentals of Digital Image Processing.* Prentice Hall of India, 1990.