A String Matching Approach for Visual Retrieval and Classification

Mei-Chen Yeh
Dept. of Electrical and Computer Engineering,
University of California, Santa Barbara, USA
meichen@umail.ucsb.edu

Kwang-Ting Cheng
Dept. of Electrical and Computer Engineering,
University of California, Santa Barbara, USA
timcheng@ece.ucsb.edu

ABSTRACT

We present an approach to measuring similarities between visual data based on approximate string matching. In this approach, an image is represented by an ordered list of feature descriptors. We show the extraction of local features sequences from two types of 2-D signals – scene and shape images. The similarity of these two images is then measured by 1) solving a correspondence problem between two ordered sets of features and 2) calculating similarities between matched features and dissimilarities between unmatched features. Our experimental study shows that such a *globally ordered and locally unordered* representation is more discriminative than a bag-of-features representation and the similarity measure based on string matching is effective. We illustrate the application of the proposed approach to scene classification and shape retrieval, and demonstrate superior performance to existing solutions.

Categories and Subject Descriptors

I.4.7 [**Image Processing and Computer Vision**]: Feature Measurement—*feature representation*; I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis—*object recognition*; I.4.9 [**Image Processing and Computer Vision**]: Applications

General Terms

Algorithms, Experimentation

Keywords

Image classification, shape retrieval, string matching

1. INTRODUCTION

Two fundamental problems in visual retrieval and classification are the design of good data representations and the definition of a quantitative metric that measures the similarities or dissimilarities between each pair of visual data. The characteristics of a good representation include a high sensitivity to data that represent different concepts and invariability to data that are perceptually alike or belong to the same class. A good similarity measure appropriately quantifies the similarity and is robust to imperfect features, such as the presence of noise. For example, two images that contain the same object should have a small distance measurement even though the object is viewed at different angles or is posed in different backgrounds. Both tasks are challenging because of the well known sensory and semantic gap [23].

With the recent development of robust local features [15, 1], representations that consist of parts that described by local descriptors have demonstrated impressive performance in various domains. For example, bag-of-features methods, which represent an image as a distribution of visual words' frequency, have shown promising results for recognizing natural scene categories and for texture classification [25]. Another successful example is the use of shape context descriptors for contour matching [1]. In these methods, the set of multi-dimensional features is unordered, and does not have a fixed cardinality (i.e., the number of features is not fixed). For such representations, partial matching, which optimally determines corresponding points of two feature sets, is widely used to measure the similarity [1, 2, 9, 10].

However, the *order* of parts carries useful information as it preserves some spatial layout information of the features. In fact, many real-world objects might be better represented by a sequence of features. For example, a contour image can be described by ordered local descriptors gathered from the contour. A person's face could be represented by patches, which are data sets grouped around different facial features such as nose, ears, and mouth. If we order the data from the top to the bottom, the eye patches should appear before the mouth patch. In this paper, we show that the descriptive ability of representations based on sets of features can be improved if their order is considered.

Once we represent visual data as sequences of features, the partial matching method has several drawbacks. First, the matching of corresponding features derived by partial matching might not preserve the ordering. Consider the face example again; the outer brow might be matched with the upper lip because of the shape similarity. By enforcing the ordering constraint, the occurrence of such mismatches will be minimized and the accuracy and geometric fidelity of the matching should be increased. Second, the similarity is measured only based on a fraction of the feature set. Given two sets of features of different sizes, partial matching maps features in the smaller set to a subset of those in the larger

MIR'08, October 30–31, 2008, Vancouver, British Columbia, Canada.

set such that the aggregated similarity between the matched features is maximal. The resulting measurement reflects the similarity between those corresponding features; however, it does not take into account their dissimilarities, which are contributed by those unmatched features in the larger set. Although partial matching allows for outliers and irregularities, our experimental study indicates that a metric for data retrieval and classification that takes into account both the similarities and dissimilarities achieves better performance than one that considers similarities only.

In this paper we propose a representation for visual data based on ordered features and a method to measure the similarity based on such a representation. To compare two sequences of features, we explore the use of *approximate string matching* – a method that addresses the problem of string matching and that can tolerate errors. The key characteristic of this method is that if one of the two strings-undermatching is likely an erroneous variant of the other, the distance between them would be small. The types of defined errors are application dependent. In this paper we explore the *Levenshtein distance* (the so-called *edit distance*), which allows automatic deletion, insertion, and substitution of features or descriptors in both input sequences for matching. Therefore, the descriptor sequences need not be of equal length. In computing the distance between two sequences of descriptors, based on the *ground distance* defined between two descriptors, each of these operations (deletion, insertion, and substitution) could be assigned a different cost function. With this formulation, the distance reflects not only the similarity of the corresponding features (identified by the substitution operation), but also the dissimilarity of unmatched features (identified by the insertion and deletion operations).

In the rest of the paper, we first review previous work in this area. We then describe our formulation of the representation and the matching method in section 3. Section 4 demonstrates two case studies – scene classification and shape retrieval. Finally, we conclude the paper in section 5.

2. PREVIOUS WORK

In this section, we briefly review prior work on representations and similarity measures for comparing two images. For a more detailed survey, please refer to [23, 19]. We start with the global representations and then describe two recent methods built upon local features – pyramid matching [9] and spatial pyramid matching [12].

Global features, which are extracted from the image as a whole and have a fixed number of features, are widely used for image representation. Examples include color histograms and raw pixels. Many feature-by-feature metrics, such as the Euclidean distance, can be applied to measure the similarity/dissimilarity. Such methods are known to be sensitive to various conditions, such as changes in illumination, pose, or the presence of noises or occlusions in images. Local features, on the other hand, describe a local region of an image. An image is represented by a number of local features, and its number is not necessarily fixed. Such representations are known to be more robust because local features are invariant to certain image transformations. In the following, we discuss two recent methods that place the local features into a pyramid of histograms, on which the matching is performed.

Grauman and Darrell proposed a *pyramid match kernel* [9] to find an approximate correspondence between two sets of features. This method places increasingly coarser grids over the feature space and computes a weighted sum of matches that occur at each level of resolution. A match is made if two features fall into the same cell of a grid at a particular resolution. Matches at a finer resolution contribute a heavier weight to the final similarity measurement. This method approximates the optimal partial matching method and requires a computational time that is linear to the number of features. For this method, features are quantized into histograms and those locations from which the local features were extracted are discarded.

Lazebnik *et al.* extended the pyramid matching framework and introduced a spatial pyramid representation [12]. Instead of performing quantization over the feature space, this method constructs several levels of resolution in the two-dimensional image space. For each level i of resolution, the image is spatially partitioned into 4^i regions of equal sizes, each of which is represented by a bag of features. Thus, the image is represented by 4^i bags of features at level i. At each level, the representation can be viewed as an ordered set of bags of features – more bags for a larger i. The overall similarity between two images is then computed as a weighted sum of a histogram intersection across all levels. According to the reported results on scene and object recognition, spatial partitioning of the image does improve the performance. Combining multiple levels of resolution offers another 1-2% of performance improvement in comparison with using a single level alone. They showed a good empirical performance for level-2 partitioning, which corresponds to 16 bags of features. Note that bags in each level are ordered. For comparing two ordered sets of 4^i bags, the histogram intersection metric is used to match these 4^i bag-pairs.

Our goal is to find a representation combining the best global and local features. A representation that is "locally unordered and globally ordered" would likely retain the holistic structure of an image while being invariant to various conditions. Based on this idea, we describe a more specific formulation and our matching approach.

3. MATCHING APPROACH

We start from describing our formulation of matching two ordered sets of features, along with an approach for solving the problem. As long as the feature descriptors are arranged as an ordered set, the formulation is independent of the specifics of the features. This formulation can be used for different applications based on different feature descriptors. We will explain the extraction of representations for two applications, scene classification and shape retrieval, in section 4.

3.1 Problem formulation

Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m]$ and $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_n]$ denote two sequences of feature vectors with sizes m and n respectively. The distance between $\mathbf{X}$ and $\mathbf{Y}$ is defined as the minimal cost of a sequence of *operations* that transforms $\mathbf{X}$ into $\mathbf{Y}$.

$$d(\mathbf{X}, \mathbf{Y}) = \min \sum_i (c_i), \tag{1}$$

where c_i is the cost of an operation, denoted in the form of

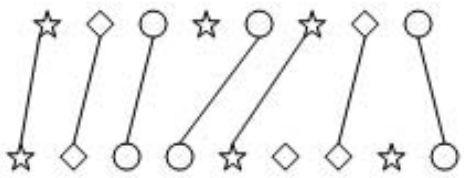

Figure 1: A simple example of approximate string matching. There are three types of feature vectors, which are indicated by stars, diamonds and circles. The best match occurs by deleting the 4^{th} feature in the top sequence and the 6^{th} and 8^{th} features in the bottom sequence. Equivalently, the top string can be transformed to the bottom string by deleting the 4^{th} feature and adding a diamond between the 6^{th} and 7^{th} features, and a star between the 7^{th} and 8^{th} features.

$\delta(\cdot,\cdot)$, which transforms a feature of $\mathbf{X}$ so that the resulting feature set is closer to $\mathbf{Y}$, and the total cost is the sum of the costs of all operations that complete the transformation from $\mathbf{X}$ to $\mathbf{Y}$.

We use the Levenshtein distance, in which the set of operations is restricted to (1) insertion $\delta(\varepsilon, \mathbf{a})$, i.e. inserting a feature vector $\mathbf{a}$ to $\mathbf{X}$, (2) deletion $delta(\mathbf{a}, \varepsilon)$, i.e. deleting a feature vector $\mathbf{a}$ from $\mathbf{X}$, and (3) substitution $\delta(\mathbf{a}, \mathbf{b})$, i.e. substituting $\mathbf{a}$ in $\mathbf{X}$ with $\mathbf{b}$. This formulation can then be rephrased as "the minimal cost of insertions, deletions and substitutions to make two sequences of feature vectors, $\mathbf{X}$ and $\mathbf{Y}$, equal." Fig. 1 shows a simple example of matching two sequences that consist of three feature types.

The determination of each operation's cost is application dependent; however, a logical solution would be to integrate a ground distance function into this framework. The ground distance function $f(\mathbf{a}, \mathbf{b})$ returns a non-negative real number that defines the dissimilarity between two feature vectors. For example, one may define the substitution cost as the ground distance function between the two features and assign a constant cost for the deletion and insertion operations. Note that $d(\mathbf{X}, \mathbf{Y})$ is symmetric if each operation cost is symmetric. That is, $\delta(\mathbf{a}, \varepsilon) = \delta(\varepsilon, \mathbf{a})$ and $\delta(\mathbf{a}, \mathbf{b}) = \delta(\mathbf{b}, \mathbf{a})$.

The Levenshtein distance can be easily computed using dynamic programming. While it may not be very efficient, the approach is flexible and adaptable to different cost operations. For more efficient computations, automata-based approaches might be applied [16]. We construct a matrix $D_{[0,...,m;0,...,n]}$. The matrix D is computed using the recurrent equation:

$$D(i,j) = \min\{ \quad D(i-1, j-1) + \delta(\mathbf{x_i}, \mathbf{y_j}), \\ D(i-1, j) + \delta(\mathbf{x_i}, \varepsilon), \\ D(i, j-1) + \delta(\varepsilon, \mathbf{y_j})\}, \qquad (2)$$

where $D_{i,j}$ is the minimum cost of operations needed to match $\mathbf{x}_1, ..., \mathbf{x}_i$ to $\mathbf{y}_1, ..., \mathbf{y}_j$ and $D_{0,0} = 0$. We return $D_{m,n}$ as the Levenshtein distance. Fig. 2 shows a directed graph for an example with $m = 2$ and $n = 3$. The algorithm finds the minimum cost path, from the upper left corner to the lower right corner. In the graph, vertical and horizontal edges are assigned the cost of inserting a feature vector,

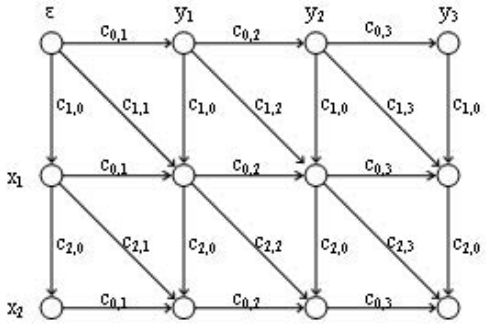

Figure 2: A dynamic programming graph. Each node records the total cost $D_{i,j}$ for comparing two feature subsets, and each edge represents the cost of an operation. Here $c_{i,j}$ is the output from the function $\delta(\cdot,\cdot)$ where both horizontal and vertical edges are costs for insertion and deletion, and diagonal edges are costs for substitution.

while diagonal edges are assigned the substitution cost. Note that an insertion in one feature sequence is equivalent to a deletion in the other. Moreover, multiple paths might exist to achieve the minimal cost. The computational complexity is $O(mn)$ and the space required is $O(\min(m, n))$ since we can choose either row-wise or column-wise processing based on their sizes. The sequences of operations performed to transform $\mathbf{X}$ into $\mathbf{Y}$ can be easily recovered from the matrix; but if we choose to do so, we need to store the entire matrix.

We can extend this algorithm to search for an alignment between two feature sequences. That is, we want to search for a short sequence $\mathbf{X}$ in a long sequence $\mathbf{Y}$. This is useful for comparing objects that are represented by a cyclic sequence of local descriptors. The algorithm is basically the same, but the difference is that we allow any position in $\mathbf{Y}$ to be the potential starting point of a match, which is achieved by setting $D_{0,j}$ to zero for all $j \in 0..n$. That is, the empty set matches with zero errors at any position in $\mathbf{Y}$. Also, we return the minimum of $D_{m,j}$ for $j \in 1..n$, which is the cost of the best alignment found by the algorithm, as the Levenshtein distance.

4. CASE STUDIES

4.1 Scene classification

We first demonstrate the application of our approach to scene classification. We used a dataset that consists of 15 scene categories[1] that were provided by Lazebnik *et al.* [12]. Each category has 200 to 400 images. The major sources of the images include the COREL dataset, personal photos, and Google image search. Fig.3 shows example images from the scene dataset.

4.1.1 Representations

We represented a scene image by the spatial pyramid representation [12]. First, as suggested in [25], we applied the Harris-Hessian-Laplace detector followed by Scale-invariant

[1]A subset of categories was collected by other groups. See [12] for details.

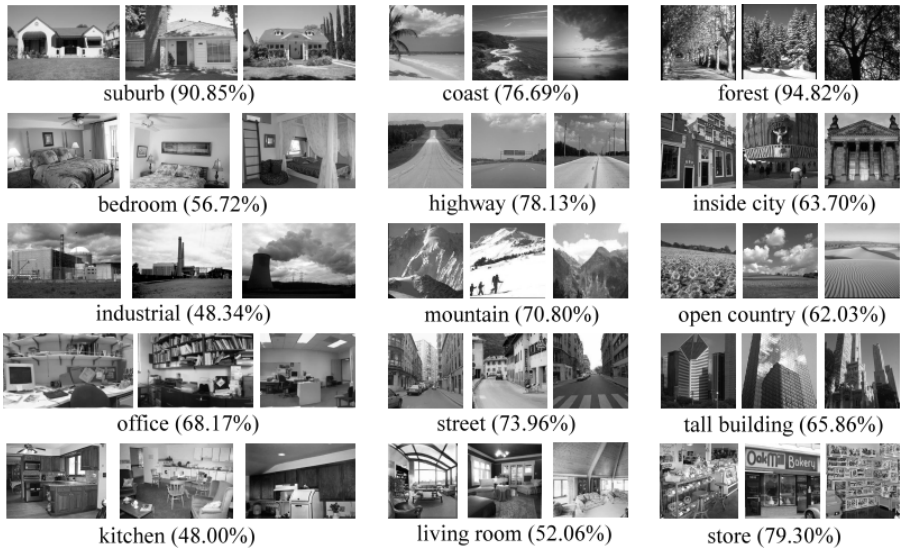

Figure 3: Exemplar images in the scene dataset. The average category-level classification rates over 10 trials of our method are also shown.

Feature Transform (SIFT) [15] for extracting a set of local features for each image. We used the implementation of [25] for feature extraction. Next, we followed the setup proposed in [12] and performed K-means clustering on a random subset of local features from the training set to form a visual vocabulary with 200 visual words. Each local feature is then quantized to a visual word and an image is represented by a distribution of frequencies of visual words. This gives us a standard bag-of-features representation.

We next obtained the spatial pyramid representation by repeatedly subdividing an image and computing the bag-of-features representation over the resulting sub-regions, as described in section 2. For each level i of resolution, the image is spatially partitioned into 4^i regions of equal sizes, each of which is represented by a bag of features. Thus, the image is represented by 4^i bags of features at level i. In [12, 3], the authors suggest that a level-2 partitioning (L2), which corresponds to 16 bags of features, achieves the best performance among all levels. Following their suggestion, we used the L2 representation in our implementation. That is, each image is represented by 16 ordered bags of features, each of which is 200-dimensional, following the raster-scan ordering.

To compare two bags, which are basically two histograms, we again used χ^2 as the ground distance and a constant value of 0.5 as the insertion and deletion cost. Since representations in this dataset do not have the rotational alignment problem, we used our basic algorithm for matching.

4.1.2 Results

We followed the same setup proposed in [12] to evaluate our method. One hundred images per class were used for training; the rest were for testing. Multi-class classification is executed using a Support Vector Machine (SVM) [5], which is trained by specifying the matrix of kernel values between all pairs of training examples. Note that the edit distance-based kernel methods have shown some success for handwriting and fingerprint recongition [18]. The experiment was repeated 10 times with randomly selected training images. The mean and the standard deviation of the classification accuracy are shown in Table 1. We compare our method with our implementation of [12] using two distance metrics – histogram intersection and χ^2, and a bipartite graph matching that will be explained in details later.

First, we observed that our implementation of [12] (the "intersect" column in Table 2) achieved a moderate performance in comparison with their implementation. The accuracies of L0, L1, L2, and all levels (Pyramid) using the so-called weak features reported in [12] are 45.3%, 53.6%, 61.7%, and 64.7%, respectively. If the SIFT features are used, the accuracies are 72.2%, 77.9%, 79.4%, and 81.1%, respectively. The performance disparity between our implementation and that in [12] may be caused by our use of interest point detectors instead of a dense regular grid. However, both implementations have a consistent trend – the amount of improvement across the levels is similar in both implementations. Also, we observed a slightly better accuracy by using χ^2 rather than the histogram intersect while using the L1 or the L2 alone. This, again, is consistent with the results reported in [3]; however, the claim does not hold if all levels of histograms are used.

We compare our method with several others. We implemented a bipartite graph matching method for direct comparison. Bipartite graph matching treats those bags as unordered and minimizes the cost of matching, but subject to the only constraint that the matching is one-to-one. As shown in Table 1, our approach achieves the best performance. For the case of L2 alone, our method achieves a greater accuracy (2.79% and 5.91%, respectively) in comparison with [12] and bipartite graph matching. Our method is also better than the spatial pyramid kernel, which uses all levels. Note that all methods are based on the same ground distance (χ^2). One merit of our approach is that we allow matching across adjacent bags. Consider the dynamic programming graph shown in Fig. 2 again. The kernel value derived in [12] at level i is equal to summing up all diagonal elements in the graph. This assumes that bag-by-bag matching was performed. That is, a bag in the first image is only compared to the corresponding bag in the second im-

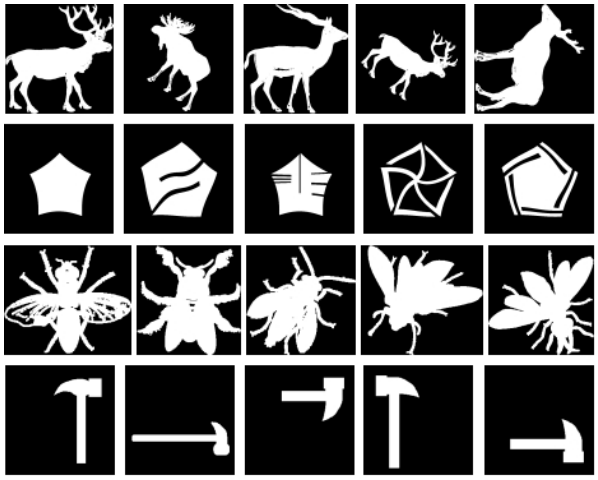

Figure 4: Exemplar shapes in the MPEG-7 shape database for four different categories. Variations caused by scale, rotation, change of viewpoints, appearance of noise, and non-rigid motion are present.

age that is at the same location. However, the diagonal axis might not be the optimal path for minimizing the matching cost. Please note that we can interpret the method in [12] as a special case of our method in which only the substitution operation is allowed. The bipartite graph matching method does not consider the locations of bags and performs the worst of all the methods tested. Considering the previously mentioned factors, it should not be surprising that our approach could achieve the best performance among these methods.

4.2 Shape retrieval

Now we demonstrate our approach on shape retrieval. The edit distance based similarity measurement has been proposed for comparing shape images [4, 17, 6, 21, 24] and for handwritten text recognition [22]. However, those methods compute the edit distance from shock graphs or curvature graphs extracted from the boundaries. Here we design to edit lists of local descriptors. We conducted experiments on the MPEG-7 shape database, namely the Core Experiment CE-Shape-1 part B, which measures the performance of similarity-based shape retrieval [11]. The database consists of 1,400 shapes in 70 categories, each of which consists of 20 shapes. Fig. 4 shows some examples in the dataset, where the images in the same row belong to the same class.

4.2.1 Representations

We uniformly sampled 100 points from the contour of each shape. Starting from the top point of the contour, we traversed the contour clock-wise and extracted the shape context descriptor [1] at each sampled point. The shape context descriptor at a point is a 60-dimensional feature vector, resulting from combinations of five distance bins and twelve orientation bins (as proposed in [1]) which encode the relative coordinates of remaining points using the log-polar space. As a result, a shape is represented by 100 ordered, 60-dimensional shape context descriptors.

As illustrated by the examples shown in Fig. 4, two contours, $\mathbf{X}$ and Y, may not be rotationally aligned at the top point. To address this issue, we used rotationally invariant shape context descriptors that were computed based on the method proposed in [1]. We also duplicated one of the sequences, say $\mathbf{Y}$, to form a longer sequence, $\mathbf{Y}$-$\mathbf{Y}$, resulting in a size 200. Applying our extended algorithm to match $\mathbf{X}$ and $\mathbf{Y}$-$\mathbf{Y}$ would automatically find an optimal alignment fairly independent of the starting points of the ordered sequences [4].

We used χ^2 as the ground distance function to compare two shape descriptors, and a constant value of 0.5 as the cost of insertion and deletion operations. Note that 0.5 is the χ^2 distance between a normalized shape context descriptor and a zero vector.

4.2.2 Results

The performance is measured based on the so-called bull's eye test, in which each shape is compared to every other shape in the database, and the number of correct matches in the top 40 retrieved shapes is counted. The retrieval rate is calculated by the total number of correct matches divided by the total number of possible hits (20 x 1,400 = 28,000).

Table 2 shows the reported results from several state-of-the-art algorithms. We implemented the Fourier Descriptors [8] to represent global features. We have experimented with the city block distance, the Euclidean distance, and the cosine measures, among which the best performance was obtained using the city block distance. For local features, the shape context descriptor (SC) was used in [1, 20], and an extension of the shape context descriptor, called the inner-dist shape contexts descriptor (IDSC), was proposed in [13]. In [1], bipartite graph matching is used to find point correspondence. However, the shape similarity in [1] is measured in conjunction with metrics based on both appearance and deformation energy. That is, given the points derived from two shapes, the point correspondences are first found through bipartite graph matching. Then, the thin-plate-spline model (TPS) was used to estimate the transformation between them. After that, the similarity is measured as a weighted sum of three parts: the appearance difference, the shape context distance, and the bending energy. In our experiment, we implemented a bipartite graph matching method with shape context descriptors (the shape context distance) for direct comparison.

To better understanding the strengths and weaknesses of the proposed method, we compare its performance with several other methods. In comparison with SC+TPS [1], our method is better and simpler since it automatically finds the alignment during matching and does not require a transformation model to align two shapes. Transformations based on point correspondences found by bipartite graph matching in [1] might result in a bad alignment since spatial information about the points is not considered. In [20, 13], dynamic programming is also used for matching to preserve the order of local descriptors. As shown in Table 2, IDSC+DP [13] achieves 3% greater accuracy than SC+DP [20] because of their elaborate features design. The same improvement (3%) in a setting where the only difference is the descriptors used in the experiment was also reported in [13]. By replacing SC with IDSC in our method, it seems reasonable to expect our approach would yield even better results. To address the rotational alignment problem, authors in [13] proposed to search for a good alignment by trying a number of points as the starting point, and then select the one that offers the

Table 1: Classification results for the scene dataset.

level	intersect [12]	χ^2	Bipartite	Ours
L0 (1x1)	59.16±0.77	58.84±0.78		
L1 (2x2)	64.33±0.78	64.48±0.67		
L2 (4x4)	66.05±0.60	66.42±0.62	63.30±0.42	69.21±0.66
Pyramid [12]	68.28±0.58	67.95±0.50		

Table 2: The bull's eye score of different methods for the MPEG-7 CE-Shape-1.

Algorithm	Fourier Descriptors	SC+TPS [1]	SC+Bipartite
Score	73.51%	76.51%	71.81%
Algorithm	COPAP(SC+DP) [20]	IDSC+DP [13]	Ours(SC+ASM)
Score	82.46%	85.40%	84.29%

best performance. This strategy would likely lead to a sub-optimal solution. Furthermore, in [20, 13] a small fraction of points is allowed to be unmatched and a small penalty (0.3) is set for a point with no matching. The results of our experiments indicate that assessing a greater penalty (say, 0.5) would be a better choice. Matching of dissimilar shapes could be achieved by using a set of insertion/deletion operations. Thus, appropriately penalizing the insertion/deletion operations and reflecting these factors in the total cost is important for accurately measuring similarity.

Recently two robust methods were proposed and achieved great accuracy on the MPEG-7 shape database. Felzenszwalb *et al.* [7] proposed a hierarchical matching method and achieved 87.7% recognition rate. In [14], the authors used the Earth Mover's Distance as the ground distance for comparing a pair of local descriptors, and they achieved 86.56% accuracy. Comparing with those methods, our approach is much simpler and computational cheaper. For example, given two sequences of size m and n, the algorithm proposed in [7] runs in $O(mn^3)$ while ours runs in $O(mn)$.

5. CONCLUSIONS

In this paper we present a "globally ordered and locally unordered" representation for visual data, and a method to measure their dissimilarity based on such a representation. Our experimental study has shown that representations based on a set of local descriptors would become more discriminative if the descriptors' order is also considered. We have demonstrated the application of our approach to scene classification and shape retrieval, and obtained a better performance in comparison with some previous methods.

One future research direction we are pursuing is the use of correspondences found by our method for image registration. Our method produces geometrically faithful matching since the order is constrained. Thus, it should provide more robust matching than other assignment methods typically used. Another research direction is the exploration of this approach for video retrieval. A video can be naturally represented as a sequence of features in which a feature represents a frame in the video. Comparing the similarities between video clips fits well into the formulation that the proposed method addresses.

6. REFERENCES

[1] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:509–522, 2002.

[2] A. Berg, T. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 26–33, 2005.

[3] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *ACM International Conference on Image and Video Retrieval*, pages 401–408, 2007.

[4] H. Bunke and U. Buhler. Applications of approximate string matching to 2d shape recognition. *Pattern Recognition*, 26(12):1797–1812, December 1993.

[5] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001.

[6] M. R. Daliri and V. Torre. Shape recognition and retrieval using string of symbols. In *International Conference on Machine Learning and Applications*, pages 101–108, December 2006.

[7] P. F. Felzenszwalb and J. D. Schwartz. Hierarchical matching of deformable shapes. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2007.

[8] R. C. Gonzalez and R. E. Woods. *Digital Image Processing (2nd Edition)*. Prentice Hall, 2002.

[9] K. Grauman and T. Darrel. The pyramid match kernel: Discriminative classification with sets of image features. In *IEEE International Conference on Computer Vision*, pages 1458–1465, 2005.

[10] K. Grauman and T. Darrell. Fast contour matching using approximate earth mover's distance. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 220–227, 2004.

[11] L. J. Latecki, R. Lakamper, and U. Eckhardt. Shape descriptors for non-rigid shapes with a single closed contour. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 424–429, 2000.

[12] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing

natural scene categories. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 2169–2178, 2006.

[13] H. Ling and D. W. Jacobs. Using the inner-distance for classification of articulated shapes. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 719–726, 2005.

[14] H. Ling and K. Okada. EMD-L1: An efficient and robust algorithm for comparing histogram-based descriptors. In *European Conference on Computer Vision*, pages 330–343, 2006.

[15] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[16] G. Navapro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33:31–88, 2001.

[17] M. Neuhaus and H. Bunke. An error-tolerant approximate matching algorithm for attributed planar graphs and its application to fingerprint classification. In *International Workshop on Structural and Syntactic Pattern*, pages 180–189, October 2004.

[18] M. Neuhaus and H. Bunke. Edit distance-based kernel functions for structural pattern classification. *Pattern Recognition*, 39(10):1851–1863, October 2006.

[19] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40:99–121, 2000.

[20] C. Scott and R. Nowak. Robust contour matching via the order-preserving assignment problem. *IEEE Transactions on Image Processing*, 15(7):1831–1838, July 2006.

[21] T. Sebastian, P. Klein, and B. Kimia. Recognition of shapes by editing their shock graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):550–571, May 2004.

[22] G. Seni, V. Kripasundar, and R. Srihari. Generalizing edit distance to incorporate domain information: Handwritten text recognition as a case study. *Pattern Recognition*, 29(3):405–414, March 1996.

[23] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1349–1380, 2000.

[24] H. Zaboli, M. Rahmati, and A. Mirzaei. Shape recognition and retrieval: A structural approach using velocity function. In *International Conference on Computer Analysis of Images and Patterns*, pages 734–741, 2007.

[25] J. Zhang, M. Marsalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *International Journal of Computer Vision*, 73:213–238, 2007.