

Affinity Learning on a Tensor Product Graph with Applications to Shape and Image Retrieval

Xingwei Yang and Longin Jan Latecki

Department of Computer and Information Sciences

Temple University, Philadelphia, USA

xingwei@temple.edu, latecki@temple.edu

Abstract

*As observed in several recent publications, improved retrieval performance is achieved when pairwise similarities between the query and the database objects are replaced with more global affinities that also consider the relation among the database objects. This is commonly achieved by propagating the similarity information in a weighted graph representing the database and query objects. Instead of propagating the similarity information on the original graph, we propose to utilize the tensor product graph (TPG) obtained by the tensor product of the original graph with itself. By virtue of this construction, not only local but also long range similarities among graph nodes are explicitly represented as higher order relations, making it possible to better reveal the intrinsic structure of the data manifold. In addition, we improve the local neighborhood structure of the original graph in a preprocessing stage. We illustrate the benefits of the proposed approach on shape and image ranking and retrieval tasks. We are able to achieve the bull's eye retrieval score of **99.99%** on MPEG-7 shape dataset, which is much higher than the state-of-the-art algorithms.*











1. Introduction











Image and shape retrieval belong to central topics in computer vision. Similar to other ranking/retrieval tasks, once a query object is given, the goal is to retrieve the most similar objects in the database. Traditionally, the performance of the retrieval is decided by the similarity/dissimilarity measure, which separately compares the query to each database object. However, these pairwise comparisons ignore the structure of the data manifold determined by similarities between the database objects. As shown by a sequence of recent papers, [30, 27, 9, 2], considering the data manifold structure significantly improves the performance of ranking/retrieval. The basic idea is inspired by the success of google PageRank ranking. The data manifold is represented as a graph with edge weights determined by the initial pairwise similarity values. Then the pairwise











similarities between the query and each database object are reevaluated in the context of other database objects, where the context of each object is a set of other objects most similar to it and the reevaluation is obtained by propagating the similarity information following structure of the weighted edge links in the graph. The reevaluation is closely related to random walks on the graph, e.g., [22, 28].

Compared to the existing methods, our approach differs in two main aspects. First, instead of propagating the similarity information on the original graph, we propose to utilize the tensor product graph (TPG) obtained by the tensor product of the original graph with itself. Since TPG takes into account a higher order information compared to the original methods, it comes at no surprise that we obtain better retrieval performance. Higher order information has been utilized in many applications before, e.g., [6, 29], but it comes at the price of higher order computational complexity and storage requirement. The key feature of the proposed approach is that the information propagation can be computed with the same computational complexity and the same amount of storage as the propagation on the original graph. We utilize a graph diffusion process to propagate the similarity information on TPG, but we never compute it directly on TPG. Instead, we derive a novel iterative algorithm to compute it directly on the original graph, which is guaranteed to converge. After its convergence we obtain new edge weights that can be interpreted as new, learnt similarities. They are then used for final retrieval ranking.

Fig. 1 compares the retrieval results of the learnt similarities to those of original similarities. The queries are shown in the first column. The first row shows retrieval results of an original image similarity measure on a subset of Caltech 101 dataset. The second row shows the retrieval results after learning the similarities. The third row shows retrieval results of an original shape similarity measure [15] on the MPEG-7 dataset. The results after learning the similarities with the proposed method are shown in the fourth row. As can be seen the proposed similarity learning is able to correct wrong retrieval results of the original similarities.

Query	1st	2nd	3rd	4th	5th	6th	7th	8th	9th
									

Query	1st	2nd	3rd	4th	5th	6th	7th	8th	9th
									

Query	1st	2nd	3rd	4th	5th	6th	7th	8th	9th
									





















Query	1st	2nd	3rd	4th	5th	6th	7th	8th	9th
									

Figure 1. First row: the query and the retrieval results with an original image similarity measure on subset of Caltech 101 dataset. Second row: the same query and the retrieval results after the proposed similarity learning. Third row: the query and the retrieval results with a shape similarity measure on the MPEG-7 shape dataset. Fourth row: the same query and the retrieval results with learned similarities.

Second, it has been noticed that if the pairwise similarities are not accurate, the full graph contains too much noise, which hurts the affinity propagation [10, 22]. Thus, it is natural to constrain the relation from each element to its neighbors [22]. This significantly reduces the amount of noisy pairwise similarities, since the pairwise similarities are more accurate for close neighbors. A common practice to achieve this is to keep only the edge weights of k nearest neighbors (kNN) of each object and zero out the other edge weights, i.e., remove the corresponding edges. However, the selection of kNN is also easily influenced by errors in the pairwise similarities and a "good" number of nearest neighbors k may be different for different objects. To better define the neighbors of a point, we propose a novel way to construct the neighborhood structure, which is called Dominant Neighborhood (DN). Like a dominant set defined in [19], DN considers the affinities among the neighbors to determine the best neighborhood structure, which makes it more robust to errors and outliers in pairwise similarities. Another advantage of DN is that it automatically determines the optimal number of neighbors. This solves one of the serious problems of kNN . If k is too large for a given point, its kNN includes points that are not its true neighbors. This fact is illustrated for binary shapes in Fig. 2. The fact that kNN for $k = 9$ of the dog contains two horses (first row), may cause any affinity learning algorithm to lose the distinction between dogs and horses. The DN of the dog in the second row correctly contains only dogs, making it easier to learn the distinction between dogs and horses. To illustrate the problem of kNN from the point of view of a data manifold, we show a toy example in Fig. 3(a). The point marked with a triangle incorrectly contains points of two classes in its kNN for $k = 50$. As shown in Fig. 3(b), the domi-

Query	1st	2nd	3rd	4th	5th	6th	7th	8th	9th
									









Query	1st	2nd	3rd	4th	5th	6th	7th
							

Figure 2. First row: a classic kNN for $k = 9$ of a dog. It contains two horses making it harder for any affinity learning algorithm to discriminate dogs from horses. Second row: the proposed dominant neighborhood (DN) obtained from kNN in the first row.

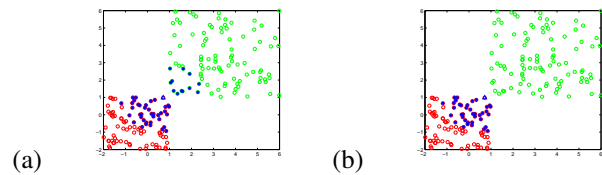


Figure 3. (a) The blue stars show a classical kNN of the point marked with the blue triangle for $k = 50$. (b) The proposed, dominant neighborhood (DN) of the same point. It is obtained as a dominant subset from the kNN in (a).

nant neighborhood of this point only contains points from its class.

In §2 we discuss the difference between our method to several related works. In §3, the distance learning algorithm on TPG is introduced in details. The construction of DN is described in §6. The experimental results are given in §7.

2. Related Work

Zhou et al. [30] proposed to improve ranking of retrieved objects by utilizing the data manifold structure. In other words, the context information of similar database objects is

used to improve ranking results. Recently, Bai et al. [2] proposed to utilize the context information for shape retrieval with Label Propagation, which was originally designed for semi-supervised learning. Kontschieder et al. [9] proposed a different way to utilize the context information to improve the performance. A graph diffusion process is utilized for retrieval in Yang et al. [27], where a Locally Constraint Diffusion Process (LCDP) is proposed. LCDP has been used in [15] and [23] to improve their results. Graph diffusion has also been utilized for manifold embedding [3]. However, the distances relation among data after embedding are not suitable for retrieval, which is demonstrated in our experimental results. Furthermore, Bai et al. [1] introduce a novel retrieval methods, co-transduction, motivated by co-training. It fuses different similarity/dissimilarity measures to better determine the relation between objects.

Content based image retrieval is getting more attention recently. The classic methods [18, 20] use the bag of feature image representation for image retrieval. Jegou et al. [8] refine the affinity among the objects by enforcing the neighborhood relation to be symmetric. To reduce the complexity of current algorithms, Jegou et al. [7] proposes a novel features extraction approach. Different from the current methods, we improve the image retrieval results by utilizing the higher order information of the TPG.

3. Affinity Learning

In this section we describe a novel context-sensitive affinity learning algorithm. It is introduced as a diffusion process on a Tensor Product Graph (TPG). However, the size of TPG is quadratic as compared to the original graph, which makes the diffusion on the TPG impractical on large datasets due to both high computation time and high memory requirement. To solve this problem, we propose a novel iterative algorithm on the original graph (Section 5), and prove that it is equivalent to the diffusion process on TPG. Consequently, both time complexity and memory requirements of the iterative algorithm are comparable to other affinity learning methods like diffusion on the original graph [22] or LGC[30, 28].

In the paper, the data is represented as an edge-weighted graph $G = (V, A)$, where $V = \{v_1, \dots, v_n\}$ is the set of vertices representing the data points and A is the graph adjacency matrix $A(i, j) = (a_{ij})$ for $i, j = 1, \dots, n$, where a_{ij} presents the edge weight from v_i to v_j . We assume that A is nonnegative and the sum of each row is smaller than one. A matrix A that satisfies these requirements can be easily created from a stochastic matrix (see Section 6).

It is well known that a graph diffusion process is able to reveal the intrinsic relation between objects [3, 22]. Probably the simplest realization of a diffusion process on a graph is by computing powers of the graph matrix, i.e., the edge weights at time t are given by A^t . Usually, the time is dis-

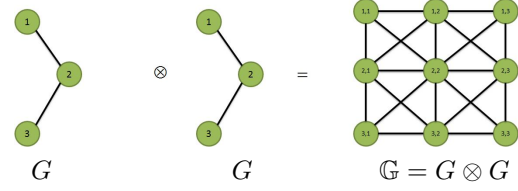


Figure 4. An example of a tensor product graph. We do not show the self connections (loops), but each node has a loop in \mathbb{G} .

crete and t corresponds to the iteration number. However, this process is sensitive to the number of iterations [10]. For example, if the sum of each row of A is smaller than one, as we assumed, then it converges to zero matrix, in which case determining a right stopping time t is critical. In order to make the graph diffusion process independent from the number of iteration, accumulation between different numbers of iterations is widely used [10]. Following this strategy, we consider the graph diffusion process defined as

$$A^{(t)} = \sum_{i=0}^t A^i \quad (1)$$

Our assumption that the sum of each row of $A < 1$ is equivalent to the fact that the the maximum of the row-wise sums of matrix $A < 1$. It is known that the maximum of the absolute values of the eigenvalues is bounded by the the maximum of the row-wise sums. Therefore, we obtain that the maximum of the absolute values of the eigenvalues of A is smaller than one. Consequently, (1) converges to a fixed and nontrivial solution given by $\lim_{t \rightarrow \infty} A^{(t)} = (I - A)^{-1}$, where I is the identify matrix.

4. Diffusion Process on Tensor Product Graph

The Tensor Product Graph (TPG) $\mathbb{G} = G \otimes G$ is defined as $\mathbb{G} = (V \times V, \mathbb{A})$. Thus, each vertex of \mathbb{G} is a pair of vertices in G , and consequently, it is indexed with a pair of indices. The adjacency matrix of \mathbb{G} is defined as $\mathbb{A} = A \otimes A$, where \otimes is the Kronecker product [11, 24]. In particular, for $\alpha, \beta, i, j = 1 \dots, n$ we have

$$\mathbb{A}(\alpha, \beta, i, j) = A(\alpha, \beta) \cdot A(i, j) = a_{\alpha, \beta} \cdot a_{i, j}.$$

Thus, if $A \in \mathbb{R}^{n \times n}$, then $\mathbb{A} = A \otimes A \in \mathbb{R}^{nn \times nn}$. An example is shown in Fig. 4.

We define the **diffusion process on TPG** as

$$\mathbb{A}^{(t)} = \sum_{i=1}^t \mathbb{A}^i. \quad (2)$$

Since the edge weights of TPG relate 4 tuples of original vertices, \mathbb{G} contains high order information than the input graph. The higher order information is helpful for revealing the intrinsic relation between objects, which is obtained by the diffusion process on TPG.

As is the case for (1), the process (2) also converges to a fixed and nontrivial solution

$$\lim_{t \rightarrow \infty} \mathbb{A}^{(t)} = \lim_{t \rightarrow \infty} \sum_{i=1}^t \mathbb{A}^i = (I - \mathbb{A})^{-1}. \quad (3)$$

To show this, we only need to show that the sum of each row of \mathbb{A} is smaller than 1, i.e., $\sum_{\beta,j} \mathbb{A}(\alpha\beta, ij) < 1$, where β, j both range from 1 to n . This holds, since

$$\sum_{\beta,j} \mathbb{A}(\alpha\beta, ij) = \sum_{\beta,j} a_{\alpha\beta} a_{ij} = \sum_{\beta} a_{\alpha\beta} \sum_j a_{ij} < 1. \quad (4)$$

Consequently, (3) provides a closed form solution for the diffusion process on TPG. However, our goal was to utilize TPG to learn new affinities on the original graph G . i.e., to obtain a new affinity matrix A^* of size $n \times n$. The matrix A^* containing the **learned affinities** is defined as

$$A^* = \text{vec}^{-1}((I - \mathbb{A})^{-1} \text{vec}(I)), \quad (5)$$

where I is an $n \times n$ identity matrix and vec is an operator that stacks the columns of a matrix one after the next into a column vector. Formally, for a given $m \times n$ matrix B

$$\text{vec}(B) = (b_{11}, \dots, b_{m1}, b_{12}, \dots, b_{m2}, \dots, b_{1n}, \dots, b_{mn})^T.$$

Since $\text{vec} : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{mn}$ is an isomorphism, its inverse exists, and we denote it with vec^{-1} .

To summarize, for the input affinity matrix A , the new learned affinities are given by matrix A^* defined in (5). However, the affinity learning with the proposed diffusion process on TPG (2) is impractical for large graphs due to high storage and computing cost. The diffusion on the original graph G requires $O(n^2)$ storage (number of the matrix elements) and its computation cost is determined by the cost of matrix inversion, which is $O(n^3)$ for Gauss-Jordan elimination or about $O(n^{2.4})$ for the Coppersmith-Winograd algorithm. In contrast the diffusion on TPG requires $O(n^4)$ storage and its computation cost is $O(n^6)$ for Gauss-Jordan elimination or about $O(n^{4.8})$ for the Coppersmith-Winograd algorithm. Therefore, we propose a novel iterative algorithm in Section 5 to compute (2). Its storage and computation cost is comparable to the diffusion on the original graph, since it is executed on the original graph.

5. Iterative Algorithm for Diffusion on TPG

We define $Q^{(1)} = A$ and

$$Q^{(t+1)} = A Q^{(t)} A^T + I, \quad (6)$$

where I is the identity matrix. We iterate (6) until convergence. Let us denote the limit matrix by $Q^* = \lim_{t \rightarrow \infty} Q^{(t)}$. The proof of the convergence of (6) and a

closed form expression for Q^* both follow from the following key equation

$$\lim_{t \rightarrow \infty} Q^{(t)} = Q^* = A^* = \text{vec}^{-1}((I - \mathbb{A})^{-1} \text{vec}(I)). \quad (7)$$

The remainder of this section is devoted to prove this equation. Since $Q^* = A^*$, we obtain that the iterative algorithm on the original graph G defined by (6) yields the same affinities as the TPG diffusion process on \mathbb{G} .

In order to prove (7), we first transform (6) to

$$\begin{aligned} Q^{(t+1)} &= A Q^{(t)} A^T + I = A(A Q^{(t-1)} A^T + I)A^T + I \\ &= A^2 Q^{(t-1)} (A^T)^2 + A I A^T + I = \dots \\ &= A^t A (A^T)^t + A^{t-1} I (A^T)^{t-1} + \dots + I \\ &= A^t A (A^T)^t + \sum_{i=1}^{t-1} A^i I (A^T)^i \end{aligned} \quad (8)$$

Since (by our assumption) sum of each row of $A < 1$, we have $\lim_{t \rightarrow \infty} A^t A (A^T)^t = 0$, and consequently,

$$Q^* = \lim_{t \rightarrow \infty} Q^{(t+1)} = \lim_{t \rightarrow \infty} \sum_{i=1}^{t-1} A^i I (A^T)^i \quad (9)$$

We observe that the following identity holds

$$\text{vec}(A S A^T) = (A \otimes A) \text{vec}(S) = \mathbb{A} \text{vec}(S), \quad (10)$$

where we recall that \otimes is the Kronecker product. As a consequence we obtain for every $i = 1, 2, \dots$

$$\text{vec}(A^i I (A^T)^i) = \mathbb{A}^i \text{vec}(I). \quad (11)$$

Our proof of (11) is by induction. Suppose

$$\text{vec}(A^k I (A^T)^k) = \mathbb{A}^k \text{vec}(I)$$

holds for $i = k$, then for $i = k + 1$ we have

$$\begin{aligned} \text{vec}(A^{k+1} I (A^T)^{k+1}) &= \text{vec}(A (A^k I (A^T)^k) A^T) \\ &= \mathbb{A} \text{vec}(A^k I (A^T)^k) = \mathbb{A} \mathbb{A}^k \text{vec}(I) = \mathbb{A}^{k+1} \text{vec}(I) \end{aligned}$$

From (11) and from the fact that vec of a sum of matrices is sum of their vec 's, we obtain

$$\text{vec}\left(\sum_{i=1}^{t-1} A^i I (A^T)^i\right) = \sum_{i=1}^{t-1} \mathbb{A}^i \text{vec}(I). \quad (12)$$

Finally from (9) and (12), we derive

$$\begin{aligned} \text{vec}(Q^*) &= \lim_{t \rightarrow \infty} \text{vec}\left(\sum_{i=1}^{t-1} A^i I (A^T)^i\right) = \lim_{t \rightarrow \infty} \sum_{i=1}^{t-1} (\mathbb{A}^i \text{vec}(I)) \\ &= \left(\lim_{t \rightarrow \infty} \sum_{i=1}^{t-1} \mathbb{A}^i\right) \text{vec}(I) = (I - \mathbb{A})^{-1} \text{vec}(I) \end{aligned} \quad (13)$$

This proves our key equation (7). Hence the iterative algorithm (6) on G yields the same affinities as the TPG diffusion process on \mathbb{G} .

Since our iterative algorithm works on the original graph G , both its storage and computational cost requirements are significantly lower than those of the TPG diffusion process. It requires $O(n^2)$ storage and its computation cost is determined by the cost of matrix multiplication, which is $O(n^3)$ for direct implementation or about $O(n^{2.4})$ for the Coppersmith-Winograd algorithm. Consequently, if the number of iterations is $t = T$, then its computational cost is $O(Tn^3)$ or $O(Tn^{2.4})$, correspondingly.

Graph G in Fig. 4 provides a simple example to illustrate the fact that the diffusion on the TPG considers the information from more edge weights than the diffusion on the original graph. For simplicity we compare only the second iteration, i.e., we compare $A^{(2)}$ to $Q^{(2)}$ and focus on the edge weight between 1 and 3. Since there is no edges between 1 and 3 in G , we have $a_{13} = a_{31} = 0$. Therefore, in $A^{(2)}$ we have $a_{13}^{(2)} = a_{12} \cdot a_{23}$. The corresponding weight of the edge between 1 and 3 in $Q^{(2)}$ is given by

$$q_{13}^{(2)} = a_{12} \cdot a_{23} \cdot (a_{11} + a_{22}) + a_{12} \cdot a_{33} \cdot a_{33}.$$

While $a_{13}^{(2)}$ only depends on the edge weights a_{12} and a_{23} , $q_{13}^{(2)}$ also depends on the self similarities a_{11}, a_{22}, a_{33} . In particular, we can have $a_{13}^{(2)} < q_{13}^{(2)}$ if $a_{11} + a_{22} > 1$, but we can also have $a_{13}^{(2)} > q_{13}^{(2)}$. Thus, TPG diffusion utilizes more information to determine the strength of the connection between 1 and 3 than just the connections a_{12} and a_{23} considered by the diffusion on the original graph. The difference in the number of connections considered is even more dramatic for $t > 2$. TPG diffusion also utilizes the self-reinforcement in that the strength of the connections depends on the ratio between the similarity of each database object to itself and the sum of its similarities to other objects.

6. Dominant Neighbors

The derivations in the previous section depend on the assumption that the affinity matrix A of graph G is nonnegative and the sum of each row is smaller than one. However, the original affinity matrix of graph G , let us call it W , usually does not satisfy these assumptions. In this section we propose a particular way to transform W to a matrix A that satisfies them.

In the case of retrieval and ranking, W contains pairwise similarities between the database objects and between the query and the database objects. Therefore, we can assume that all entries in W are positive. It is also natural to assume that for each object i the self similarity of i to itself is the largest, i.e., $\forall i \forall j \neq i (w_{ii} > w_{ij})$. We also can assume

that W is symmetric, since if this is not the case we can replace $W = \frac{1}{2}W + W^T$.

As we observed in the introduction, the pairwise similarities are not accurate, and consequently, the graph contains many noisy similarities. Since the pairwise similarities are more accurate for close neighbors, the amount of noisy similarities is significantly reduced if we set to zero all edge weights except the k nearest neighbors (kNN) of each object.

However, the selection of kNN is also easily influenced by errors in the pairwise similarities and the number of suitable nearest neighbors k may be different for different objects. Therefore, to better define the neighbors of a point, we propose a novel way to construct the neighborhood structure, which is called Dominant Neighborhood (DN). The main idea is that the dominant neighborhood $DN(i)$ of a vertex i should correspond to a maximal clique that satisfies $DN(i) \subseteq kNN(i)$. As stated in [19], a maximal clique in a weighted graph, which is called a dominant set, is a subset of V with maximal average affinity between all pairs of its vertices, which is equivalent to the fact that the overall similarity among internal elements is the highest in that adding any new element would lower it.

As is the case for $kNN(i)$, we do not want to include vertex i in its $DN(i)$, therefore, we set the diagonal entries of W to zero and obtain a matrix W_0 . In order to select vertices that belong to a dominant set, we introduce an indicator vector $\mathbf{x} = (x_1, x_2, \dots, x_n)$ over the vertices V of G . A vertex $j \in V$ is selected as belonging to a maximal clique if and only if $x_j > 0$. As shown in [19], each dominant set can be obtained as a local maximizer of the following quadratic program

$$\text{maximize } f(\mathbf{x}) = (\mathbf{x})^T W_0 \mathbf{x} \quad (14)$$

$$\text{subject to } \mathbf{x} \in \Delta = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} \geq 0, \sum_{j=1}^n x_j = 1\}.$$

Pavan and Pelillo [19] also provide an iterative method to compute local maximizers of (14). Given an initialization $\mathbf{x}(1)$, the corresponding local solution \mathbf{x}^* of (14) can be obtained by the replicator equation [26]:

$$x_j(t+1) = x_j(t) \frac{(W_0 \mathbf{x}(t))_j}{\mathbf{x}(t)^T W_0 \mathbf{x}(t)} \quad j = 1, \dots, n \quad (15)$$

It is easy to see that $\mathbf{x}(t) \in \Delta$ with increasing t , which means that every trajectory starting in Δ will remain in Δ . Moreover, since W_0 is symmetric, the target function $f(\mathbf{x}) = (\mathbf{x})^T W_0 \mathbf{x}$ is strictly increasing for a given initial vector $\mathbf{x}(1)$ and is guaranteed to converge.

In order to obtain a dominant neighborhood $DN(i)$ of vertex i , we initialize (15) with the classical $kNN(i)$. More precisely, we set $x_j(1) = \frac{1}{k}$ if $j \in kNN(i)$ and $x_j(1) = 0$

otherwise. After (15) converged to the corresponding local solution \mathbf{x}^* , a vertex $j \in DN(i)$ if and only if $x_j^* > 0$.

As discussed above, the dominant neighborhood of $DN(i)$ is determined not only by the pairwise relation of i to other objects, but also the relation between the other objects, which makes $DN(i)$ more robust to noisy pairwise similarities than $kNN(i)$. Thus, we use it to first refine the matrix W to W^* so that the neighbors of each data is robust to noise and outliers. The matrix W^* is obtained from W by setting $w_{ij}^* = w_{ij}$ if $j \in DN(i)$ and $w_{ij}^* = 0$ otherwise. Then, W^* is transformed into a stochastic matrix. A is derived from W^* , where $a_{ij} = w_{ij}^*$ if $j \in KNN(i)$ and $a_{ij} = 0$ otherwise.

7. Experimental Results

To demonstrate the advantages of our approach, we test our algorithm on both shape and image retrieval tasks. On all test datasets, the proposed method achieves excellent results, which are better than the state-of-art methods. Since our iterative algorithm to compute the TPD diffusion is guaranteed to converge, we only need to ensure that the number of iterations is not too small. It is set to 200 for all test datasets.

If pairwise distances are provided for a given dataset, we transform the distances to similarities with the method introduced in [25]. Once we obtain a similarity matrix W , we first use DN to obtain the matrix A . Then, we run the proposed, iterative algorithm to compute the TPD diffusion. It returns the new affinity matrix A^* representing the learned similarities, which are then used for ranking, i.e., if vertex i represents the query objects, the most similar objects to it are obtained by sorting in descendent order the row i of matrix A^* .

7.1. MPEG-7 Dataset

The proposed framework is tested for shape classification on a commonly used MPEG7 CE-Shape-1 part B database [12]. The dataset contains 1400 silhouette images from 70 classes, where each class has 20 different shapes (some shapes are shown in Figs. 1 and 2). The retrieval rate is measured by the bull's eye score: every shape in the database is submitted as a query and the number of shapes from the same class in the top 40 is counted. The bull's eye score is then defined as the ratio of the number of correct hits to the best possible number of hits (which is 20×1400).

As shown in Table 1 the proposed affinity learning method can successfully improve on the state-of-the-art methods. We selected two different shape similarity methods: Aspect Shape Context (ASC) [15] and Articulated Invariant Representation (AIR) [5] as the input pairwise distance measure. kNN with $k = 10$ was used to initialize (15). We observe that the affinities learned by our method improve the original retrieval score of ASC by over 8%.

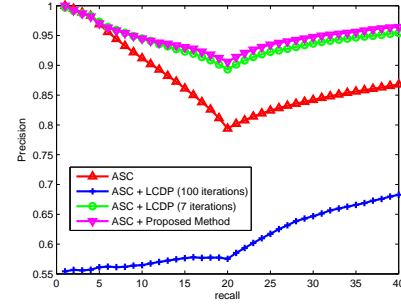


Figure 5. Precision/Recall curves on MPEG-7 shape dataset.

reach nearly perfect bull's eye score 99.99% on MPEG7 Dataset by using AIR for shape similarity. This is the best ever reported score on this popular shape dataset.

In order to visualize the gain in retrieval rates (precision) by our method, we plot the percentage of correct results among the first k most similar shapes for $k = 1, \dots, 40$ in Fig. 5, where we use ASC for shape similarity. We observe that not only does the proposed method increase the bulls eye score, but also consistently achieves the best retrieval rates. Recall that each class has 20 shapes, which is the reason for the precision curves to increase for $k > 20$. In order to illustrate the problem with the stopping time of the graph classical diffusion computed by matrix power, we show two curves for LCDP [27], one when it is stopped after 7 iterations and the second one when it is stopped after 100 iterations, which clearly illustrates the problem of diffusing relevant information. In contrast, the proposed algorithm is robust to the number of iterations.

7.2. Nister and Stewenius (N-S) dataset

In this section, we demonstrate the performance of the proposed approach on image retrieval. We compare it to other diffusion based methods and to a recently proposed method, Contextual Dissimilarity Measure (CDM) [8], which can significantly improve the similarity computed by bag-of-features. CDM learns affinities following a different principles than the proposed method. CDM is motivated by an observation that a good ranking is usually not symmetrical in image search. CDM makes two images similar when they both obtain a good ranking position when using each other as a query.

We selected the Nister and Stewenius (N-S) dataset [21] composed of 10,200 images. A few example images from N-S dataset are shown in Fig. 6. The N-S dataset consists of 2,550 objects or scenes, each of which is imaged from 4 different viewpoints. Hence there is only 4 images in each class and total of 2,550 image classes, which makes this dataset very challenging for any manifold learning approach, and in particular, for any diffusion based approach.

To obtain the pairwise distance relation between images for our algorithm, we implemented a baseline method de-

Table 1. Retrieval rates (bull’s eye) of different context shape retrieval methods on the MPEG-7 shape dataset.

IDSC [14]	IDSC +LP [2]	IDSC +Mutual graph [9]	Perc. R. [23]	Perc. R. + LCDP [23]	ASC [15]	ASC + LCDP [15]	ASC + DN + TPG Diffusion	AIR [5]	AIR + DN + TPG Diffusion
85.40%	91.61%	93.40%	88.39%	95.60%	88.30%	95.96%	96.47%	93.67%	99.99%

Table 2. Retrieval results on N-S Dataset. The highest possible score is 4.

Baseline [8]	Classic Diffusion with $t = 2$	Classic Diffusion with $t = 5$	Diffusion Maps [3]	CDM [8]	TPG Diffusion with Classic kNN	TPG Diffusion with DN
3.22	3.42	0.245	1.01	3.57	3.58	3.61



Figure 6. Some images from Nister and Stewenius (N-S) dataset.



Figure 7. Sample images from our subset of Caltech 101 dataset.

scribed in [8]. The image descriptor is a combination of Hessian-Affine region detector [17] and SIFT descriptor [16]. A visual vocabulary is obtained using the k-means algorithm on the sub-sampled image descriptors.

The results are shown in Table 2. The retrieval rate is measured by the average number of correct images among the four first images returned. Thus, the maximum value is 4 and the higher the value the better is the result. Each image has been submitted as a query. The fact that our method can significantly improve the retrieval result of the baseline method (from 3.22 to 3.61) clearly shows the benefits of utilizing higher order relations by the TPG diffusion. We also observe that the result of our method is better than CDM. Finally, the usage of DN improves on the result obtained with classic kNN . We did not have much choice to set the neighborhood size k for this dataset. kNN with $k = 3$ was used to initialize (15).

Since each image class has only 4 images, it is very difficult to correctly propagate the similarity relations. Therefore, the classic diffusion [3] can only improve the baseline result for a very small number of iterations. The best retrieval rate of the classic diffusion is for $t = 2$, i.e., when the original similarity matrix is raised to power $t = 2$. Already for $t = 5$, the retrieval rate is much lower than the rate of the baseline. We also report the retrieval result obtained after embedding the data by Diffusion Maps [3], which are significantly lower than the rate of the baseline. This justifies our observation that although Diffusion Maps are excellent for embedding into Euclidean spaces, the distances

obtained after the embedding cannot be used for retrieval tasks.

7.3. Caltech 101 dataset

Besides N-S dataset, we also test our algorithm on a well known Caltech 101 dataset [4]. The Caltech-101 dataset contains 101 classes (including animals, vehicles, flowers, etc.) with high shape variability. The number of images per category varies from 31 to 800. Most images are medium resolution, i.e. about 300×200 pixels. We selected 12 classes from Caltech-101, which contain total 2788 images. Example images are shown in Fig. 7. Different from experiments on N-S dataset, we just use pure SIFT descriptor [16] to calculate the distance between images. The SIFT features are extracted from 16×16 pixel patches densely sampled from each image on a grid with step size of 8 pixels. To get the codebook, we use standard K-means clustering and fix the codebook size to 2048. Each image is represented by multiple assignment [8] and Spatial Pyramid Matching [13]. The distance between two images is obtained by the χ^2 distance between the two vectors.

Table 3. Retrieval rates on 12 image classes from Caltech-101. The best possible rate is 1.

Baseline	Classic Dif. with $t = 5$	Classic Dif. with $t = 50$	Dif. Maps [3]	TPG Dif. with DN
0.801	0.859	0.267	0.534	0.903

The results are shown in Table 3. It is clear that with adjusted number of iterations according to the ground truth, which is $t = 5$, the classic diffusion process is able to reveal the relation between images. However, as discussed above, it is very sensitive to number of iterations, which we illustrate with its retrieval rate for $t = 50$. Besides, the results of Diffusion Maps [3] demonstrates that the relation between objects after embedding by Diffusion Maps [3] is not suitable for retrieval. In our algorithm, to initialize (15), kNN with $k = 400$ was used. Again TPG diffusion is able

to significantly improve the retrieval rate of the input pairwise distance measure. In particular, this demonstrates that TPG diffusion is robust to large variance in the number of images in each class.

8. Conclusion

The proposed framework for shape and image retrieval can be applied whenever original pairwise distances/similarities cannot perfectly rank the database objects. The key advantage of the proposed Tensor Product Graph diffusion is the utilization of higher order similarity relations, which are both local and long range. Usually higher order relations lead to a substantially higher computation cost. However, we are able to introduce an iterative algorithm to compute TPG diffusion that has the same space and time complexity as the classical diffusion on the original graph. We also provide a formal proof that the iterative algorithm and the TPG diffusion converge to the same solution. Hence the proposed TPG diffusion explores the benefits of higher order relations without the price of higher computational cost.

Acknowledgments

The work was supported by the NSF under Grants IIS-0812118, BCS-0924164, OIA-1027897, the AFOSR Grant FA9550-09-1-0207, and the DOE Award 71498-001-09. The idea of the dominant neighborhood was inspired by our discussions with HaiRong Liu.

References

- [1] X. Bai, B. Wang, X. Wang, W. Liu, and Z. Tu. Co-transduction for shape retrieval. In *ECCV*, 2010. 2371
- [2] X. Bai, X. Yang, L. J. Latecki, W. Liu, and Z. Tu. Learning context sensitive shape similarity by graph transduction. *IEEE Trans. on PAMI*, 2010. 2369, 2371, 2375
- [3] R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21:5–30, 2006. 2371, 2375
- [4] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE PAMI*, 28:594–611, 2006. 2375
- [5] R. Gopalan, P. Turaga, and R. Chellappa. Articulation-invariant representation of non-planar shapes. In *ECCV*, 2010. 2374, 2375
- [6] Y. Huang, Q. Liu, and D. Metaxas. Video object segmentation by hypergraph cut. In *CVPR*, 2009. 2369
- [7] H. Jegou, M. Douze, and C. Schmid. Improving bag-of-features for large scale image search. *IJCV*, 87:191–212, 2010. 2371
- [8] H. Jegou, C. Schmid, H. Harzallah, and J. Verbeek. Accurate image search using the contextual dissimilarity measure. *IEEE PAMI*, 2010. 2371, 2374, 2375
- [9] P. Kotschieder, M. Donoser, and H. Bischof. Beyond pairwise shape similarity analysis. In *ACCV*, 2009. 2369, 2371, 2375
- [10] S. Lafon and A. B. Lee. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction graph partitioning, and data set parameterization. *IEEE PAMI*, 28:1393–1403, 2006. 2370, 2371
- [11] P. Lancaster and L. Rodman. *Algebraic Riccati Equations*. Clarendon Press, Oxford, 1995. 2371
- [12] L. J. Latecki and R. Lakämper. Shape similarity measure based on correspondence of visual parts. *IEEE Trans. PAMI*, 22(10):1185–1190, 2000. 2374
- [13] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 2375
- [14] H. Ling and D. Jacobs. Shape classification using the inner-distance. *IEEE Trans. PAMI*, 29:286–299, 2007. 2375
- [15] H. Ling, X. Yang, and L. J. Latecki. Balancing deformability and discriminability for shape matching. In *ECCV*, 2010. 2369, 2371, 2374, 2375
- [16] D. Lowe. Distinctive image features from scale-invariant key points. *IJCV*, 60:91–110, 2004. 2375
- [17] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *IJCV*, 60(1):63–86, 2004. 2375
- [18] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006. 2371
- [19] M. Pavan and M. Pelillo. Dominant sets and pairwise clustering. *IEEE Trans. PAMI*, 2007. 2370, 2373
- [20] J. Sivi and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, 2003. 2371
- [21] H. Stewenius and D. Nister. Object recognition benchmark. <http://vis.uky.edu/stewe/ukbench/>. 2374
- [22] M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. In *NIPS*, 2001. 2369, 2370, 2371
- [23] A. Temlyakov, B. C. Munsell, J. W. Waggoner1, and S. Wang. Two perceptually motivated strategies for shape classification. In *CVPR*, 2010. 2371, 2375
- [24] C. van Loan. The ubiquitous kronecker product. *J. of Computational and Applied Math.*, 123:85–100, 2000. 2371
- [25] J. Wang, S.-F. Chang, X. Zhou, and T. C. S. Wong. Active microscopic cellular image annotation by superposable graph transduction with imbalanced labels. In *CVPR*, 2008. 2374
- [26] J. Weibull. *Evolutionary game theory*. MIT Press, 1997. 2373
- [27] X. Yang, S. Köknar-Tezel, and L. J. Latecki. Locally constrained diffusion process on locally densified distance spaces with applications to shape retrieval. In *CVPR*, 2009. 2369, 2371, 2374
- [28] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Scholkopf. Learning with local and global consistency. In *NIPS*, 2003. 2369, 2371
- [29] D. Zhou, J. Huang, and B. Scholkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *NIPS*, 2007. 2369
- [30] D. Zhou, J. Weston, A. Gretton, Q. Bousquet, and B. Scholkopf. Ranking on data manifolds. In *NIPS*, 2003. 2369, 2370, 2371