Incremental Nonlinear Dimensionality Reduction By Manifold Learning

Martin H. C. Law, *Student Member, IEEE* Anil K. Jain *Fellow, IEEE*

***Abstract*— Understanding the structure of multidimensional patterns, especially in unsupervised case, is of fundamental importance in data mining, pattern recognition and machine learning. Several algorithms have been proposed to analyze the structure of high dimensional data based on the notion of *manifold learning*. These algorithms have been used to extract the intrinsic characteristics of different types of high dimensional data by performing nonlinear dimensionality reduction. Most of these algorithms operate in a "batch" mode and cannot be efficiently applied when data are collected sequentially. In this paper, we describe an incremental version of ISOMAP, one of the key manifold learning algorithms. Our experiments on synthetic data as well as real world images demonstrate that our modified algorithm can maintain an accurate low-dimensional representation of the data in an efficient manner.**

***Index Terms*— Incremental learning, dimensionality reduction, ISOMAP, manifold learning, unsupervised learning.**

I. Introduction

The purpose of dimensionality reduction is to transform a high dimensional data set into a low dimensional space, while retaining most of the underlying structure in the data. This is important for several reasons, with the most important being to circumvent the curse of dimensionality; many classifiers perform poorly in a high dimensional space given a small number of training samples. Dimensionality reduction can also be used to visualize the data by transforming the data into two or three dimensions, thereby giving additional insight into the problem at hand. Feature selection, feature extraction and feature weighting are the three common approaches for dimensionality reduction.

Most dimensionality reduction methods are linear, meaning that the extracted features are linear functions of the input features. Examples include principal component analysis (PCA) and linear discriminant analysis (LDA) [18]. Linear methods are easy to understand and are very simple to implement, but the linearity assumption does not lead to good results in many real world scenarios. Images of handwritten digits do not conform to the linearity assumption [25]; rotation, shearing and variation of stroke widths can at best be approximated by linear functions only in a *small* neighborhood (as in the use of tangent distance [18]). A transformation as simple as translating an object on a uniform background cannot be represented as a linear function of the pixels. This has motivated the design of nonlinear mapping methods in a general setting.

The history of nonlinear mapping traces back to Sammon's mapping published in 1969 [41]. Over time, different nonlinear mapping techniques have been proposed, such as self-organizing maps (SOM) [29], principal curve and its extensions [24], [28], [45], [47], auto-encoder neural networks [1], [15], generative topographic maps (GTM) [6] and kernel principal component analysis [42]. A comparison of some of these methods can be found in [34].

A new line of nonlinear mapping algorithms has been proposed recently based on the assumption that the data lie on a (Riemannian) manifold. Often, high dimensional data observed in the real world are the consequences of a small number of factors. In appearance-based vision, for example, the image of an object is represented as a high dimensional vector of pixel intensities. The observed image is often controlled by a small number of factors like the view angle and the lighting direction. Such relationship, even though nonlinear globally, is often smooth and approximately linear in a local region. In this case, it is reasonable to assume the high dimensional data lie approximately on a (Riemannian) manifold. Dimensionality reduction can be achieved by constructing a mapping that respects certain properties of the manifold. Isometric feature mapping (ISOMAP) [46], for example, tries to preserve the geodesic distances. Locally linear embedding (LLE) embeds data points in a low dimensional space by finding the optimal linear reconstruction in a small neighborhood. Laplacian eigenmap [2] restates the nonlinear mapping problem as an embedding problem for the vertices in a graph and uses the graph Laplacian to derive a smooth mapping. Semidefinite embedding [49] first "unrolls" the manifold to a flat hyperplane before applying PCA. Charting [7] and co-ordination-based ideas [40], [48] combine multiple local co-ordinate systems in the manifold in a consistent manner. The utility of manifold learning has been demonstrated in different applications, such as face pose detection [23], [32], face recognition [51], [53], analysis of facial expressions [11], [20], human motion data interpretation [26], gait analysis [19], [20], visualization of fiber traces [8] and wood texture analysis [38].

Most of these nonlinear mapping algorithms operate in a batch mode[1], meaning that all data points need to be available during training. In applications like surveillance, where (image) data are collected sequentially, batch method is computationally demanding: repeatedly running the "batch" version whenever new data points become available takes a long time. It is wasteful to discard previous computation results. Data accumulation is particularly beneficial to manifold learning algorithms due to their non-parametric nature. Another reason for developing incremental (non-batch) methods is that the gradual changes in the data manifold can be visualized. As more and more data points are obtained, the evolution of the

[1]Sammon's mapping can be implemented by a feed-forward neural network [34] and hence can be made online if an online training rule is used.

data manifold can reveal interesting properties of the data stream. An incremental algorithm can be easily modified to be adaptive by incorporating "forgetting", i.e., the old data points gradually lose their significance. The algorithm can then adjust the manifold in the presence of the drifting of data characteristics. Incremental learning is also useful when there is an unbounded stream of possible data to learn from. This situation can arise when an invariance transformation controlled by a continuous parameter is applied to the training data in order to reflect pattern invariance [43]. Since the parameter is continuous, there is no limit on the amount of "virtual" training data that can be generated.

In this paper, we have modified the ISOMAP algorithm so that it can update the low dimensional representation of data points gradually as more and more samples are collected. Both the original ISOMAP [46] and the landmark points version [14] are considered. We are interested in ISOMAP because it is intuitive, well understood and produces good mapping results [26], [51]. Furthermore, there are theoretical studies supporting the use of ISOMAP, such as its convergence proof [4] and the conditions for successful recovery of co-ordinates [17]. There is also a continuum extension of ISOMAP [52] as well as a spatio-temporal extension [26]. However, the motivation of our work is applicable to other mapping algorithms as well.

The main contributions of this study include

1) An algorithm that efficiently updates the solution of the all-pairs shortest path problems. This contrasts with previous work like [36], where different shortest path trees are updated independently.
2) More accurate mappings for new points by a superior estimate of the inner products.
3) An incremental eigen-decomposition problem with increasing matrix size is solved by subspace iteration with Ritz acceleration. This differs from the previous work [50] where the matrix size is assumed to be constant.
4) A vertex contraction procedure that improves the geodesic distance estimate without additional memory.

The rest of this paper is organized as follows. After a recap of ISOMAP in section II, the proposed incremental methods are described in section III. Experimental results are presented in section IV, followed by discussions in section V. Finally, in section VI we conclude and describe some topics for future work. The incremental algorithm for the standard ISOMAP in section III-A was earlier presented by us in [30].

II. ISOMAP

Given a set of data points $\mathbf{y}_1, \ldots, \mathbf{y}_n$ in a D-dimensional space $\mathbb{R}^D$, ISOMAP assumes that the data lie on a (Riemannian) manifold and maps $\mathbf{y}_i$ to its d-dimensional representation $\mathbf{x}_i$ in such a way that the *geodesic distance* between $\mathbf{y}_i$ and $\mathbf{y}_j$ is as close to the Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ in $\mathbb{R}^d$ as possible. The geodesic between two points is defined as the shortest curve[2] on the manifold connecting the two points. The length of the geodesic is the geodesic distance. The power of ISOMAP can be demonstrated by the three-dimensional "Swiss roll" data set in Figure 1(a), where points are colored according to their locations on the manifold. When PCA is used to reduce the dimension to two (Figure 1(b)), points with different colors are mixed together. So, disconnected regions on the manifold are mapped to similar locations. When ISOMAP is used (Figure 1(c)), the color of the points change gradually, indicating that the representation discovered by ISOMAP faithfully corresponds to the structure of the curved manifold.

ISOMAP algorithm has three stages. First, a neighborhood graph is constructed. Let Δ_{ij} be the (Euclidean) distance between $\mathbf{y}_i$ and $\mathbf{y}_j$. ISOMAP requires the user to specify a definition of the "neighborhood". It can either be ϵ-neighborhood, where $\mathbf{y}_i$ and $\mathbf{y}_j$ are neighbors if Δ_{ij} is less than a parameter ϵ, or $k$nn-neighborhood, where $\mathbf{y}_i$ and $\mathbf{y}_j$ are neighbors if $\mathbf{y}_i$ ($\mathbf{y}_j$) is one of the k nearest neighbors ($k$nn) of $\mathbf{y}_j$ ($\mathbf{y}_i$). The value of k is specified by the user. A weighted undirected neighborhood graph $\mathcal{G} = (V, E)$ with the vertex $v_i \in V$ corresponding to $\mathbf{y}_i$ is constructed. An edge $e(i, j)$ between v_i and v_j exists iff $\mathbf{y}_i$ is a neighbor of $\mathbf{y}_j$. The weight of $e(i, j)$, denoted by w_{ij}, is set to Δ_{ij}. The set of indices of the vertices adjacent to v_i in $\mathcal{G}$ is denoted by $adj(i)$.

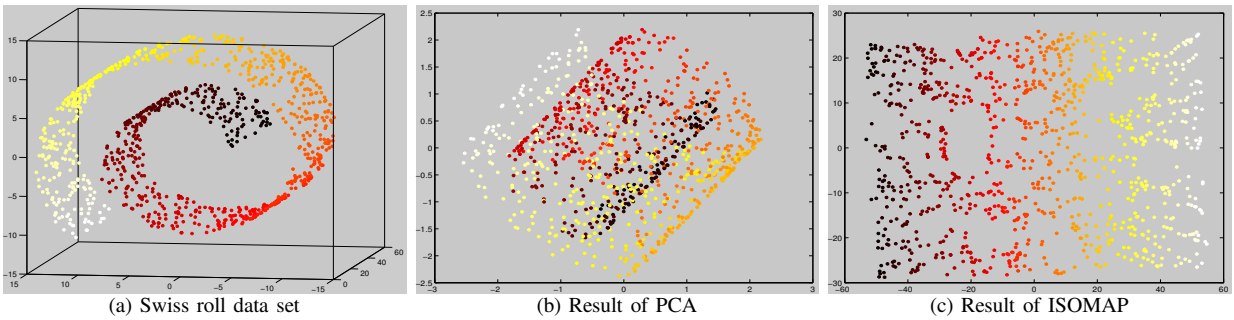
ISOMAP proceeds with the estimation of geodesic distances. The key assumption is that the geodesic between two points on the manifold can be approximated by the shortest path between the corresponding vertices in the neighborhood graph. Let g_{ij} denote the length of the shortest path $sp(i, j)$ between v_i and v_j. The shortest paths can be found by the Floyd-Warshall algorithm or the Dijkstra's algorithm with different source vertices. The shortest paths can be stored efficiently by the predecessor matrix π_{ij}, where $\pi_{ij} = k$ if v_k is immediately before v_j in $sp(i, j)$. Conceptually, however, it is useful to imagine a shortest path tree $\mathcal{T}(i)$, where the root node is v_i and $sp(i, j)$ consists of the tree edges from v_i to v_j. The subtree of $\mathcal{T}(i)$ rooted at v_a is denoted by $\mathcal{T}(i; a)$. Examples of $\mathcal{T}(i)$ can be found in Figures 2 and 3. Since g_{ij} is the approximate geodesic distance between $\mathbf{y}_i$ and $\mathbf{y}_j$, we shall call g_{ij} the "geodesic distance". Note that $\mathbf{G} = \{g_{ij}\}$ is a symmetric matrix.

Finally, ISOMAP recovers $\mathbf{x}_i$ by using the classical scaling [13] on the geodesic distances. Define $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$. Let $\mathbf{B}$ be the target inner product matrix, i.e., the matrix of the target inner products between different $\mathbf{x}_i$. By restricting $\sum_i \mathbf{x}_i = 0$, $\mathbf{B}$ is recovered as $\mathbf{B} = -\mathbf{H}\tilde{\mathbf{G}}\mathbf{H}/2$, where $\mathbf{H} = \{h_{ij}\}$, $h_{ij} = \delta_{ij} - 1/n$ and δ_{ij} is the Kronecker delta. The entries $\tilde{g}_{ij}$ of $\tilde{\mathbf{G}}$ are simply g_{ij}^2. Computing $\mathbf{H}\tilde{\mathbf{G}}\mathbf{H}$ is effectively a centering operation on $\tilde{\mathbf{G}}$. We seek $\mathbf{X}^T\mathbf{X}$ to be as close to $\mathbf{B}$ as possible in the least square sense. This is achieved by $\mathbf{X} = (\sqrt{\lambda_1}\mathbf{v}_1 \ \ldots \sqrt{\lambda_d}\mathbf{v}_d)^T$, where $\lambda_1, \ldots, \lambda_d$ are the d largest eigenvalues of $\mathbf{B}$, with corresponding eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_d$.

III. Incremental Version of ISOMAP

The key computation in ISOMAP involves solving an all-pairs shortest path problem and an eigen-decomposition problem. As new data arrive, these quantities usually do not

[2]Strictly speaking, geodesics are curves with zero covariant derivatives of their velocity vectors along the curve. A shortest curve must be a geodesic, whereas a geodesic might not be a shortest curve.



(a) Swiss roll data set (b) Result of PCA (c) Result of ISOMAP

Fig. 1. ISOMAP on "Swiss roll' with 1000 points, using $k$nn neighborhood with $k = 6$. (a) Points are colored according to their positions on the manifold. (b) Points with different colors are mixed together when they are plotted by the two PCA co-ordinates. (c) When ISOMAP co-ordinates are used, a clear trend of the color is observed, indicating that the structure of the manifold is recovered.

change much: a new vertex in the graph often changes the shortest paths among only a subset of the vertices, and the simple eigenvectors and eigenvalues of a slightly perturbed real symmetric matrix stay close to their original values. This justifies the reuse of the current geodesic distance and co-ordinate estimates for update. We restrict our attention to $k$nn neighborhood, since ϵ-neighborhood is awkward for incremental learning: the neighborhood size should be constantly decreasing as additional data points become available.

The problem of incremental ISOMAP can be stated as follows. Assume that the low dimensional co-ordinates $\mathbf{x}_i$ of $\mathbf{y}_i$ for the first n points are given. The new sample $\mathbf{y}_{n+1}$ is observed. How should we update the existing set of $\mathbf{x}_i$ and find $\mathbf{x}_{n+1}$? Our solution consists of three stages. The geodesic distances g_{ij} are first updated in view of the change of neighborhood graph due to v_{n+1}. The geodesic distances of the new point to the existing points are then used to estimate $\mathbf{x}_{n+1}$. Finally, all $\mathbf{x}_i$ are updated in view of the change in g_{ij}.

In section III-A, we shall describe the modification of the original ISOMAP for incremental updates. A variant of ISOMAP that utilizes the geodesic distances from a fixed set of points (landmark points) [14] is modified to become incremental in section III-B. Because ISOMAP is non-parametric, the data points themselves need to be stored. A vertex contraction procedure is described in section III-C, which improves the geodesic distance estimate with the arrival of new data, without storing the new data. This procedure can be applied to both the variants of ISOMAP. Throughout this section we assume d (dimensionality of the projected space) is fixed. This can be estimated by analyzing either the spectrum of the target inner product matrix or the residue of the low rank approximation as in [46], or by other methods to estimate the intrinsic dimensionality of a manifold [9], [10], [12], [27], [31], [39].

A. Incremental ISOMAP: Basic Version

We shall modify the original ISOMAP algorithm (summarized in section II) to become incremental. The correctness of the algorithm as well as an analysis of its time complexity is given in [30]. Throughout this section, the shortest paths are represented by the more economical predecessor matrix, instead of multiple shortest path trees $\mathcal{T}(i)$.

1) Updating the Neighborhood Graph: Let $\mathcal{A}$ and $\mathcal{D}$ denote the set of edges to be added and deleted after inserting v_{n+1} to the neighborhood graph, respectively. Let τ_i be the index of the k-th nearest neighbor of v_i. So, v_j is in the $k$nn neighborhood of v_i if $\Delta_{ij} \leq \Delta_{i,\tau_i}$. When $\Delta_{i,\tau_i} > \Delta_{i,n+1}$, v_{n+1} replaces v_{τ_i} in the $k$nn neighborhood of v_i. It is easy to see that

$$\begin{aligned}\mathcal{A} = \{&e(i, n+1) : v_i \text{ is in the } k\text{nn neighborhood} \\ &\text{of } v_{n+1} \text{ or } \Delta_{i,\tau_i} > \Delta_{i,n+1}\} \\ \mathcal{D} = \{&e(i, \tau_i) : \Delta_{i,\tau_i} > \Delta_{i,n+1} \text{ and } \Delta_{\tau_i,i} > \Delta_{\tau_i,\iota_i}\},\end{aligned} \tag{III.1}$$

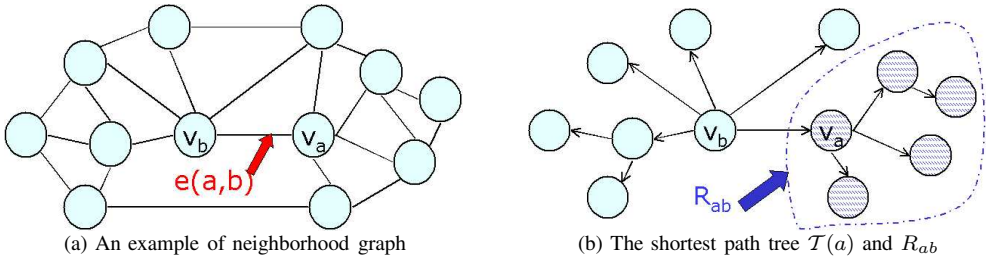
where ι_i is the index of the k-th nearest neighbor of v_{τ_i} after inserting v_{n+1} in the graph.

2) Updating the Geodesic Distances: The deleted edges can break existing shortest paths, while the added edges can create improved shortest paths. This is much more involved than it appears, because the change of a single edge can modify the shortest paths among multiple vertices.

Consider $e(a, b) \in \mathcal{D}$. If $sp(a, b)$ is not simply $e(a, b)$, deletion of $e(a, b)$ has no effect on the geodesic distances. Hence we shall suppose that $sp(a, b)$ consists of the single edge $e(a, b)$. We propagate the effect of the removal of $e(a, b)$

Input: $e(a,b)$, the edge to be removed; $\{g_{ij}\}$; $\{\pi_{ij}\}$
Output: $F_{(a,b)}$, set of "affected" vertex pairs
$R_{ab} := \emptyset$; Q.enqueue(a);
while Q.notEmpty **do**
  $t := Q$.pop; $R_{ab} = R_{ab} \cup \{t\}$;
  for all $u \in adj(t)$ **do**
    If $\pi_{ub} = a$, enqueue u to Q;
  end for
end while{Construction of R_{ab} finishes when the loop ends.}
$F_{(a,b)} := \emptyset$;
Initialize $\mathcal{T}'$, the expanded part of $\mathcal{T}(a; b)$, to contain v_b only;
for all $u \in R_{ab}$ **do**
  Q.enqueue(b)
  while Q.notEmpty **do**
    $t := Q$.pop;
    if $\pi_{at} = \pi_{ut}$ **then**
      $F_{(a,b)} = F_{(a,b)} \cup \{(u, t)\}$;
      if v_t is a leaf node in $\mathcal{T}'$ **then**
        If $s \in adj(t)$ and $\pi_{as} = t$, insert v_s as a child of v_t in $\mathcal{T}'$
      end if
      Insert all the children of v_t in $\mathcal{T}'$ to the queue Q;
    end if
  end while
end for{$\forall\ u \in R_{ab}, \forall\ s \in \mathcal{T}(u; b)$, $sp(u, s)$ uses $e(a, b)$.}

Alg. 1: *ConstructFab*: $F_{(a,b)}$, the set of vertex pairs whose shortest paths are invalidated when $e(a, b)$ is deleted, is constructed.



(a) An example of neighborhood graph (b) The shortest path tree $\mathcal{T}(a)$ and R_{ab}

Fig. 2. The edge $e(a,b)$ is to be deleted from the neighborhood graph shown in (a). The shortest path tree $\mathcal{T}(a)$ is shown as directed arrows in (b). R_{ab} (c.f. Algorithm 1) consists of all the vertices v_u such that $sp(b,u)$ contains $e(a,b)$, i.e., $\pi_{ub} = a$.

Input: u; $C(u)$; $\{g_{ij}\}$; $\{w_{ij}\}$
Output: the updated geodesic distances $\{g_{uv}\}$
for all $j \in C(u)$ **do**
  $H := adj(j) \cap \big(V/C(u)\big)$;
  $\delta(j) = \min_{k \in H}(g_{uk} + w_{kj})$, or ∞ if $H = \emptyset$;
  Insert $\delta(j)$ to a heap with index j;
end for
while the heap is not empty **do**
  $k :=$ the index of the entry by "Extract Min" on the heap;
  $C(u) := C(u)/\{k\}$; $g_{uk} := \delta(k)$; $g_{ku} := \delta(k)$;
  for all $j \in adj(k) \cap C(u)$ **do**
    $dist := g_{uk} + w_{kj}$;
    If $g_{uk} + w_{kj} < \delta(j)$, perform "Decrease Key" on $\delta(j)$ to become $dist$;
  end for
end while

Alg. 2: *ModifiedDijkstra*: The geodesic distances from the source vertex u to the set of vertices $C(u)$ are updated.

Input: Auxiliary graph $\mathcal{B}$
Output: None. The geodesic distances are updated as a side-effect
$l[i] :=$ an empty linked list, for $i = 1, \ldots, n$;
for all $v_u \in \mathcal{B}$ **do**
  $f :=$ degree of v_u in $\mathcal{B}$. Insert v_u to $l[f]$;
end for
$pos := 1$;
for $i := 1$ to n **do**
  If $l[pos]$ is empty, increment pos one by one and until $l[pos]$ is not empty;
  Remove v_u, a vertex in $l[pos]$, from the graph $\mathcal{B}$;
  Call *ModifiedDijkstra*(u, $adj(u)$ in $\mathcal{B}$);
  for all v_j that is a neighbor of v_u in $\mathcal{B}$ **do**
    Find f such that $v_j \in l[f]$ by an indexing array;
    Remove v_j from $l[f]$ if $f = 1$, and move v_j from $l[f]$ to $l[f-1]$ otherwise;
    $pos = \min(pos, f-1)$;
  end for
end for

Alg. 3: *OptimalOrder*: a greedy algorithm to remove the vertex with the smallest degree in the auxiliary graph $\mathcal{B}$. The removal of v_u corresponds to the execution of *ModifiedDijsktra* (Algorithm 2) with u as the source vertex.

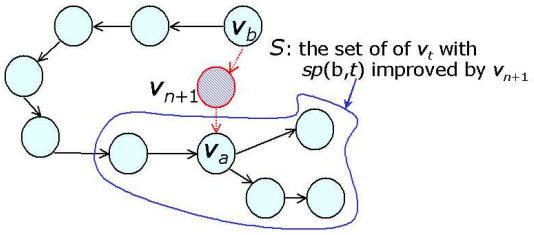


Fig. 3. Effect of edge insertion. $\mathcal{T}(a)$ before the insertion of v_{n+1} is represented by the arrows between vertices. The introduction of v_{n+1} creates a better path between v_a and v_b. S denotes the set of vertices such that $t \in S$ iff $sp(b,t)$ is improved by v_{n+1}. Note that v_t must be in $\mathcal{T}(n+1;a)$. For each $u \in S$, *UpdateInsert* (Algorithm 4) finds t such that $sp(u,t)$ is improved by v_{n+1}, starting with $t = b$.

Input: a; b; $\{g_{ij}\}$; $\{w_{ij}\}$
Output: $\{g_{ij}\}$ are updated because of the new shortest path $v_a \rightarrow v_{n+1} \rightarrow v_b$.
$S := \emptyset$; Q.enqueue(a);
while Q.notEmpty **do**
  $t := Q$.pop; $S := S \cup \{t\}$;
  for all v_u that are children of v_t in $\mathcal{T}(n+1)$ **do**
    u is enqueued into Q if $g_{u,n+1} + w_{n+1,b} < g_{u,b}$;
  end for
end while{S has been constructed.}
for all $u \in S$ **do**
  Q.enqueue(b);
  while Q.notEmpty **do**
    $t := Q$.pop; $g_{ut} := g_{tu} := g_{u,n+1} + g_{n+1,t}$;
    for all v_s that are children of v_t in $\mathcal{T}(n+1)$ **do**
      s is enqueued into Q if $g_{s,n+1} + w_{n+1,a} < g_{s,a}$;
    end for
  end while
end for{$\forall\ u \in S$, update $sp(u,t)$ if v_{n+1} helps.}

Alg. 4: *UpdateInsert*: given that $v_a \rightarrow v_{n+1} \rightarrow v_b$ is a better shortest path between v_a and v_b after the insertion of v_{n+1}, its effect is propagated to other vertices.

to the set of vertices R_{ab} (Figure 2). R_{ab} is used in turn to construct $F_{(a,b)}$, the set of all (i,j) pairs with $e(a,b)$ in $sp(i,j)$. This is done by *ConstructFab* (Algorithm 1), which finds all the vertices v_t under $\mathcal{T}(a;b)$ such that $sp(u,t)$ contains v_b, where $u \in R_{ab}$. The set of vertex pairs whose shortest paths are invalidated due to the removal of edges in $\mathcal{D}$ is thus $F = \cup_{e(a,b) \in D} F_{(a,b)}$. The shortest path distances between these vertex pairs are updated by *ModifiedDijkstra* (Algorithm 2) with source vertex v_u and destination vertices $C(u)$. It is similar to the Dijkstra's algorithm, except that only the geodesic distances from v_u to $C(u)$ (instead of all the vertices) are unknown.

The order of the source vertex in invoking *ModifiedDijkstra* can impact the run time significantly. An approximately optimal order is found by interpreting F as an auxiliary graph $\mathcal{B}$ (the undirected edge $e(i,j)$ is in $\mathcal{B}$ iff $(i,j) \in F$), and removing the vertices in $\mathcal{B}$ with the smallest degree in a greedy manner (*OptimalOrder*, Algorithm 3). When v_u is removed from $\mathcal{B}$, *ModifiedDijkstra* is called with source vertex v_u and $C(u)$ as the neighbors of v_u in $\mathcal{B}$.

The next stage of the algorithm finds the geodesic distances

between v_{n+1} and the other vertices. Since all the edges in $\mathcal{A}$ (edges to be inserted) are incident on v_{n+1}, we have

$$g_{n+1,i} = g_{i,n+1} = \min_{\substack{j \text{ such that} \\ e(n+1,j)\in A}} \big(g_{ij} + w_{j,n+1}\big) \quad \forall i. \quad \text{(III.2)}$$

Finally, we consider how $\mathcal{A}$ can shorten other geodesic distances. This is done by first locating all the vertex pairs (v_a, v_b), both adjacent to v_{n+1}, such that $v_b \to v_{n+1} \to v_a$ is a better shortest path between v_a and v_b. Starting from v_a and v_b, *UpdateInsert* (Algorithm 4) searches for all the vertex pairs that can use the new edge for a better shortest path, based on the updated graph.

For all the priority queues in this section, binary heap is used instead of the asymptotically faster Fibonacci's heap. Since the size of our heap is typically small, binary heap, with a smaller time constant, is likely to be more efficient.

3) Finding the Co-ordinates of the New Sample: The co-ordinate $\mathbf{x}_{n+1}$ is found by matching its inner product with $\mathbf{x}_i$ to the values derived from the geodesic distances. This approach is in the same spirit as the classical scaling [13] used in ISOMAP. Define $\tilde{\gamma}_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||^2 = ||\mathbf{x}_i||^2 + ||\mathbf{x}_j||^2 - 2\mathbf{x}_i^T\mathbf{x}_j$. Since $\sum_{i=1}^n \mathbf{x}_i = \mathbf{0}$, summation over j and then over i for $\tilde{\gamma}_{ij}$ leads to

$$||\mathbf{x}_i||^2 = \frac{1}{n}\big(\sum_j \tilde{\gamma}_{ij} - \sum_j ||\mathbf{x}_j||^2\big),$$
$$\sum_j ||\mathbf{x}_j||^2 = \frac{1}{2n}\sum_{ij} \tilde{\gamma}_{ij}.$$

Similarly, if we define $\gamma_i = ||\mathbf{x}_i - \mathbf{x}_{n+1}||^2$, we have

$$||\mathbf{x}_{n+1}||^2 = \frac{1}{n}\big(\sum_{i=1}^n \gamma_i - \sum_{i=1}^n ||\mathbf{x}_i||^2\big),$$
$$\mathbf{x}_{n+1}^T\mathbf{x}_i = -\frac{1}{2}\big(\gamma_i - ||\mathbf{x}_{n+1}||^2 - ||\mathbf{x}_i||^2\big) \quad \forall i.$$

If we approximate $\tilde{\gamma}_{ij}$ by g_{ij}^2 and γ_i by $g_{i,n+1}^2$, the target inner product f_i between $\mathbf{x}_{n+1}$ and $\mathbf{x}_i$ can be estimated by

$$2f_i \approx \frac{\sum_j g_{ij}^2}{n} - \frac{\sum_{lj} g_{lj}^2}{n^2} + \frac{\sum_l g_{l,n+1}^2}{n} - g_{i,n+1}^2. \quad \text{(III.3)}$$

$\mathbf{x}_{n+1}$ is obtained by solving $\mathbf{X}^T\mathbf{x}_{n+1} = \mathbf{f}$ in the least-square sense, where $\mathbf{f} = (f_1, \ldots, f_n)^T$. One way to interpret the least square solution is by noting that $\mathbf{X} = (\sqrt{\lambda_1}\mathbf{v}_1 \ \ldots \sqrt{\lambda_d}\mathbf{v}_d)^T$, where $(\lambda_i, \mathbf{v}_i)$ is an eigenpair of the target inner product matrix. The least square solution can be written as

$$\mathbf{x}_{n+1} = (\frac{1}{\sqrt{\lambda_1}}\mathbf{v}_1^T\mathbf{f}, \ldots, \frac{1}{\sqrt{\lambda_d}}\mathbf{v}_d^T\mathbf{f})^T. \quad \text{(III.4)}$$

The same estimate is obtained if Nyström approximation [21] is used.

A similar procedure is used to compute the out-of-sample extension of ISOMAP in [3], [14]. However, there is an important difference: in these studies, the inner product between the new sample and the existing points is estimated by

$$2\tilde{f}_i = \sum_{j=1}^n \frac{g_{ij}^2}{n} - g_{i,n+1}^2. \quad \text{(III.5)}$$

It is unclear how this estimate is derived. This estimate is different from that in Equation (III.3) because $\sum_l g_{l,n+1}^2/n - \sum_{ij} g_{ij}^2/n^2$ does not vanish in general; in fact, most of the time this is a large number. Empirical comparisons indicate that our inner product estimate given in Equation (III.3) is much more accurate than the one in Equation (III.5).

Finally, the new mean is subtracted from $x_i, i = 1, \ldots, (n+1)$, to ensure $\sum_{i=1}^{n+1} \mathbf{x}_i = 0$, in order to conform to the convention in the standard ISOMAP.

4) Updating the Co-ordinates: The co-ordinates $\mathbf{x}_i$ should be updated in view of the modified geodesic distance matrix $\mathbf{G}_{\text{new}}$. This can be viewed as an incremental eigenvalue problem, as $\mathbf{x}_i$ can be obtained by eigen-decomposition. However, since the size of the geodesic distance matrix is increasing, traditional methods (such as [50]) cannot be applied directly. An iterative scheme is used to update $\mathbf{X}$ by finding the eigenvalues and eigenvectors of $\mathbf{B}_{\text{new}}$. A good initial guess for the subspace of dominant eigenvectors of $\mathbf{B}_{\text{new}}$ is the column space of $\mathbf{X}^T$. Subspace iteration together with Rayleigh-Ritz acceleration [22] is used to find a better eigen-space:

1) Compute $\mathbf{Z} = \mathbf{B}_{\text{new}}\mathbf{X}^T$ and perform QR decomposition on $\mathbf{Z}$, i.e., we write $\mathbf{Z} = \mathbf{QR}$ and let $\mathbf{V} = \mathbf{Q}$.
2) Form $\mathbf{Z}^* = \mathbf{V}^T\mathbf{B}_{\text{new}}\mathbf{V}$ and perform eigen-decomposition of the d by d matrix $\mathbf{Z}^*$. Let λ_i and $\mathbf{u}_i$ be the i-th eigenvalue and the corresponding eigenvector.
3) $\mathbf{V}_{\text{new}} = \mathbf{V}[\mathbf{u}_1 \ldots \mathbf{u}_d]$ is the improved set of eigenvectors of $\mathbf{B}_{\text{new}}$.

Since d is small, the time for eigen-decomposition of $\mathbf{Z}^*$ is negligible. We do not use any variant of inverse iteration because $\mathbf{B}_{\text{new}}$ is not sparse and its inversion takes $O(n^3)$ time.

5) Complexity: In [30], we showed that the average-case complexity of the overall geodesic distance procedure is dominated by $O\left(q(|F| + |H|)\right)$, under a conjecture of random graph. Here, F (H) is the set of vertex pairs whose geodesic distances are lengthened (shortened) because of v_{n+1}, and q is the maximum degree of the vertices in the graph. We also want to note that the proposed algorithm is reasonably efficient; its complexity to solve the all-pairs shortest path by updating all geodesic distances is $O(n^2 \log n + n^2 q)$. This is the same as the complexity of the best known algorithm for the all-pairs shortest path problem of a sparse graph, which involves running Dijkstra's algorithm multiple times with different source vertices. For the update of co-ordinates, subspace iteration takes $O(n^2)$ time because of the matrix multiplication.

B. ISOMAP With Landmark Points

One drawback of the original ISOMAP is its quadratic memory requirement: the geodesic distance matrix is dense and is of size $O(n^2)$, making ISOMAP infeasible for large data sets. Landmark ISOMAP was proposed in [14] to reduce the memory requirement while lowering the computation cost. Instead of all the pairwise geodesic distances, landmark ISOMAP finds a mapping that preserves the geodesic distances originating from a small set of "landmark points". Without loss of generality, let the first m points, i.e., $\mathbf{y}_1, \ldots, \mathbf{y}_m$,

be the landmark points. After constructing the neighborhood graph as in the original ISOMAP, landmark ISOMAP uses Dijkstra's algorithm to compute the $m \times n$ landmark geodesic distance matrix $\mathbf{G} = \{g_{ij}\}$, where g_{ij} is the length of the shortest path between v_i (a landmark point) and v_j. In [14] the authors suggested that $\mathbf{X}$ can be found by first embedding the landmark points and then embedding the remaining points with respect to the landmark points. However, our preliminary experiments indicate that this is not very robust, particularly when the number of landmark points is small. Instead, we follow the implementation of landmark ISOMAP[3] and decompose $\mathbf{B} = \mathbf{H}_m\tilde{\mathbf{G}}\mathbf{H}_n$ by singular value decomposition, $\mathbf{B} = \mathbf{USV}^T = (\mathbf{U}(\mathbf{S})^{1/2})(\mathbf{V}(\mathbf{S})^{1/2})^T$, where $\mathbf{U}^T\mathbf{U}$ and $\mathbf{V}^T\mathbf{V}$ are identity matrices of corresponding sizes, and $\mathbf{S}$ is a diagonal matrix of singular values. The vectors corresponding to the largest d singular values are used to construct a low-rank approximation, $\mathbf{B} \approx \mathbf{Q}^T\mathbf{X}$.

1) Incremental Landmark ISOMAP: After updating the neighborhood graph, the incremental version for landmark ISOMAP proceeds with the update of geodesic distances. Since only the shortest paths from a small number of source vertices are maintained, the computation that can be shared among different shortest path trees is limited. Therefore, a shortest path tree $\mathcal{T}(i)$ for each landmark point is maintained, instead of the predecessor matrix as in section III-A.2. The shortest path trees are updated independently by adopting the algorithm I presented in [36]. First, InitializeEdgeWeightIncrease (Algorithm 5) is called to initialize the edge weight increase, which includes edge deletion as a special case. RebuildSPT (Algorithm 7) is then executed to rebuild the shortest path tree. InitializeEdgeWeightDecrease (Algorithm 6) is then called to initialize the edge weight decrease, which includes edge insertion as a special case. RebuildSPT is again called to rebuild the tree. Deletion of edges is done before the addition of edges because this is more efficient in practice. The readers can refer to [36] for details of these algorithms.

Input: the four tuples $(r_i, s_i, w_i^{\text{old}}, w_i^{\text{new}})$; $\mathcal{T}(a)$; $\{g_{ij}\}$
Output: The priority queue Q of "affected" vertices
$\mathcal{I} := \emptyset$;
for all $(r_i, s_i, w_i^{\text{old}}, w_i^{\text{new}})$ in the input **do**
  Swap r_i and s_i if v_{r_i} is a child of v_{s_i} in $\mathcal{T}(a)$;
  if v_{s_i} is a child of v_{r_i} in $\mathcal{T}(a)$ **then**
    $\mathcal{J} := \{v_{s_i}\} \cup$ descendent of v_{s_i} in $\mathcal{T}(a)$;
    $g_{aj} = g_{aj} + w_i^{\text{new}} - w_i^{\text{old}} \quad \forall j \in \mathcal{J}$;
    $\mathcal{I} = \mathcal{I} \cup \mathcal{J}$;
  end if
end for
for all $j \in \mathcal{J}$ **do**
  $b := \min_{k \in adj(j)} g_{ak} + w_{kj}$; {Find a new path to v_j}
  Q.enqueue(j, $\arg\min_{k \in adj(j)} g_{ak} + w_{kj}$, b) if $b < g_aj$
end for

Alg. 5: InitializeEdgeWeightIncrease for the shortest path tree $\mathcal{T}(a)$. The weight of $e(r_i, r_j)$ should increase from w_i^{old} to w_i^{new}.

The co-ordinate of the new point $\mathbf{x}_{n+1}$ is determined by solving a least-square problem similar to that in section III-A.3. The difference is that the columns of $\mathbf{Q}$, instead of $\mathbf{X}$, are used. So, $\mathbf{Q}^T\mathbf{x}_{n+1} = \mathbf{f}$ is solved in the least-square sense. Finally, we use subspace iteration together with Ritz acceleration [44] to improve singular vector estimates. The steps are

1) Perform SVD on the matrix $\mathbf{BX}$, $\mathbf{U}_1\mathbf{S}_1\mathbf{V}_1^T = \mathbf{BX}$
2) Perform SVD on the matrix $\mathbf{B}^T\mathbf{U}_1$, $\mathbf{U}_2\mathbf{S}_2\mathbf{V}_2^T = \mathbf{B}^T\mathbf{U}_1$
3) Set $\mathbf{X}_{\text{new}} = \mathbf{U}_2(\mathbf{S}_2)^{1/2}$ and $\mathbf{Q}_{\text{new}} = \mathbf{U}_1(\mathbf{S}_2)^{1/2}$

As far as time complexity is concerned, the time to update one shortest path tree is $O(\delta_d \log \delta_d + q\delta_d)$, where δ_d is the minimum number of nodes that must change their distance or parent attributes or both [36], and q is the maximum degree of vertices in the neighborhood graph. The complexity of updating the singular vectors is $O(nm)$, which is linear in n, because the number of landmark points m is fixed.

[3]We are referring to the "official" implementation by the authors of ISOMAP in `http://isomap.stanford.edu`.

Input: the four tuples $(r_i, s_i, w_i^{\text{old}}, w_i^{\text{new}})$; $\mathcal{T}(a)$; $\{g_{ij}\}$
Output: The priority queue Q of "affected" vertices
$\mathcal{I} := \emptyset$;
for all $(r_i, s_i, w_i^{\text{old}}, w_i^{\text{new}})$ in the input **do**
  Swap r_i and s_i if $g_{a,r_i} > g_{a,s_i}$;
  $diff := g_{a,r_i} + w_i^{\text{new}} - g_{a,s_i}$;
  if $diff < 0$ **then**
    Move v_{s_i} to be a child of v_{r_i} in $\mathcal{T}(a)$;
    $\mathcal{J} := \{v_{s_i}\} \cup$ descendent of v_{s_i} in $\mathcal{T}(a)$;
    $g_{aj} = g_{aj} + diff \quad \forall j \in \mathcal{J}$;
    $\mathcal{I} = \mathcal{I} \cup \mathcal{J}$;
  end if
end for
for all $j \in \mathcal{J}$ **do**
  for all $k \in adj(j)$ **do**
    Q.enqueue($k, j, g_{aj} + w_{jk}$) if $g_{aj} + w_{jk} < g_{ak}$
  end for
end for

Alg. 6: InitializeEdgeWeightDecrease for the shortest path tree $\mathcal{T}(a)$. The weight of $e(r_i, r_j)$ should decrease from w_i^{old} to w_i^{new}.

Input: The priority Q; $\mathcal{T}(a)$
Output: Updated $\mathcal{T}(a)$
while Q.notEmpty **do**
  $(i, j, d) :=$ 'Extract Min" on Q;
  $diff = d - g_{ai}$;
  if $diff < 0$ **then**
    Move v_i to be a child of v_j in $\mathcal{T}(a)$;
    $g_{ai} = d$;
    for all $k \in adj(i)$ **do**
      $newd = g_{ai} + w_{ik}$;
      Q.enqueue(k,i,$newd$) if $newd < g_{ak}$;
    end for
  end if
end while

Alg. 7: RebuildSPT: The part of $\mathcal{T}(a)$ for the vertices in Q is to be rebuilt.

C. Vertex Contraction

Owing to the non-parametric nature of ISOMAP, the data points collected need to be stored in the memory in order to refine the estimation of the geodesic distances g_{ij} and the co-ordinates $\mathbf{x}_i$. This can be undesirable if we have an arbitrarily large data stream.

One simple solution is to discard the oldest data point when a pre-determined number of data points have been accumulated. This has the additional advantage of making

the algorithm adaptive to drifting in data characteristics. The deletion should take place after all the updates due to the new point have been completed. Deleting the vertex v_i is easy: the edge deletion procedure is used to delete all the edges incident on v_i for both ISOMAP and landmark ISOMAP.

We can do better than deletion, however. A vertex contraction heuristic can be used to record the improvement in geodesic distance estimate without storing additional points. Most of the information the vertex v_j contains about the geodesic distance estimate is represented by the shortest paths passing through v_j. Suppose $sp(a, b)$ can be written as $v_a \rightsquigarrow v_i \rightarrow v_j \rightarrow v_b$. The geodesic distance between v_a and v_b can be preserved by introducing a new edge $e(i, b)$ with weight $(w_{i,j} + w_{j,b})$, even though v_j is deleted. Both the shortest path tree $\mathcal{T}(a)$ and the graph are updated in view of this new edge. This heuristic increases the density of the edges in the graph, however.

Which vertex should be contracted? A simple choice is to contract either the oldest or the newest vertex, after adjusting the geodesic distances. In our experiment, an alternative strategy is adopted, which deletes the vertices that are most "crowded", so that the points are spread more evenly along the manifold. This is done heuristically by contracting the non-landmark point whose nearest neighbor is the closest to itself.

IV. Experiments

We have implemented our main algorithm in Matlab, with the graph theoretic parts written in C++. The running time is measured on a Pentium IV 3.2 GHz PC with 512MB memory running Windows XP, using the profiler of Matlab with the java virtual machine turned off.

A. Incremental ISOMAP: Basic Version

The accuracy and the efficiency of the basic incremental algorithm is evaluated by comparing it with the batch version on several data sets. The first experiment is on the Swiss roll data set. Initialization is done by finding the co-ordinate estimate $\mathbf{x}_i$ for 100 randomly selected points using the "batch" ISOMAP, with a $k$nn neighborhood of size 6. Random points from the Swiss roll data set are added one by one, until 1500 points are accumulated. The incremental algorithm described in section III-A is used to update the co-ordinates. The first two dimensions of $\mathbf{x}_i$ correspond to the true structure of the manifold. The gap between the second and the third eigenvalues is fairly significant and it is not difficult to determine that the intrinsic dimensionality of this data set is two. Figure 4 shows several snapshots of the algorithm[4]. The dots ($\hat{\mathbf{x}}_i^{(n)}$) and the circles ($\mathbf{x}_i^{(n)}$) correspond to the co-ordinates estimated by the incremental and the batch version of ISOMAP, respectively. The circles and the dots match very well, indicating that the co-ordinates updated by the incremental ISOMAP follow closely with the co-ordinates estimated by the batch version for different numbers of data points.

To quantify the accuracy of the co-ordinate update of the incremental algorithm, we consider an error measure, defined as the square root of the mean square error between $\hat{\mathbf{x}}_i^{(n)}$ and $\mathbf{x}_i^{(n)}$, normalized by the total sample variance:

$$\mathcal{E}_n = \sqrt{\frac{1}{n}\sum_{i=1}^{n} ||\mathbf{x}_i^{(n)} - \hat{\mathbf{x}}_i^{(n)}||^2 \Big/ \sum_{i=1}^{n} ||\mathbf{x}_i^{(n)}||^2}. \qquad \text{(IV.1)}$$

Figure 5(a) displays $\mathcal{E}_n$ against the number of data points n for Swiss roll. We can see that the proposed updating method is fairly accurate, with an average error of 0.05%. The "spikes" in the graph correspond to the instances where many geodesic distances change dramatically because of the creation or deletion of "short-cuts" in the neighborhood graph. These large errors fade very quickly, however, as evident from the graph.

Table I shows the computation time decomposed into different tasks. Our incremental approach has significant savings in all three aspects of ISOMAP: graph update, geodesic distance update and co-ordinate update. The time to search for the k nearest neighbors is included in "graph update". Note that both the batch and incremental versions perform the same number of distance computations. We have also run the batch algorithm once for different number of data points (n). Table II shows the measured time averaged over 5 identical trials, after excluding the time for distance computation. The time for computing the distances for all the n points, together with the time to run the incremental algorithm once to update when the n-th point arrives, is also included in the table. See Section V for further discussion of the result.

A similar experimental procedure was applied to other data sets. The "S-curve" data set, a standard benchmark for manifold learning, contains points in a 3D space with an effective dimensionality of two. The "rendered face" data set[5] contains 698 face images[6] with size 64 by 64 rendered at different illumination and pose conditions. Some examples are shown in Figure 6. "MNIST digit 2" is a 576-dimensional data set derived from the digit images "2" from MNIST[7], and contains 28 by 28 digit images. Several typical images are shown in Figure 7. The rendered face data set and the MNIST digit 2 data sets were used in the original ISOMAP paper [46]. Our last data set, `ethn`, contains the face images used in [33]. The task is to classify a 64 by 64 face image as Asian or non-Asian. This database contains 1320 images for Asian class and 1310 images for non-Asian class, and is composed of several face image databases, including the PF01 database[8], the Yale database[9] the AR database [35], as well as the non-public NLPR database[10]. Some example face images are shown in Figure 8. The neighborhood size for MNIST digit 2 and `ethn` is set to 10 in order to demonstrate that the proposed approach is efficient and accurate irrespective of the neighborhood used. With the exception of "rendered face", 1500 points have been used during the experiment. The

[4]The avi files can be found at `http://www.cse.msu.edu/prip/ResearchProjects/iisomap/`.

[5]`http://isomap.stanford.edu`

[6]The 30 principal components in that data set are used in the experiment.

[7]`http://yann.lecun.com/exdb/mnist/`.

[8]`http://nova.postech.ac.kr/archives/imdb.html`.

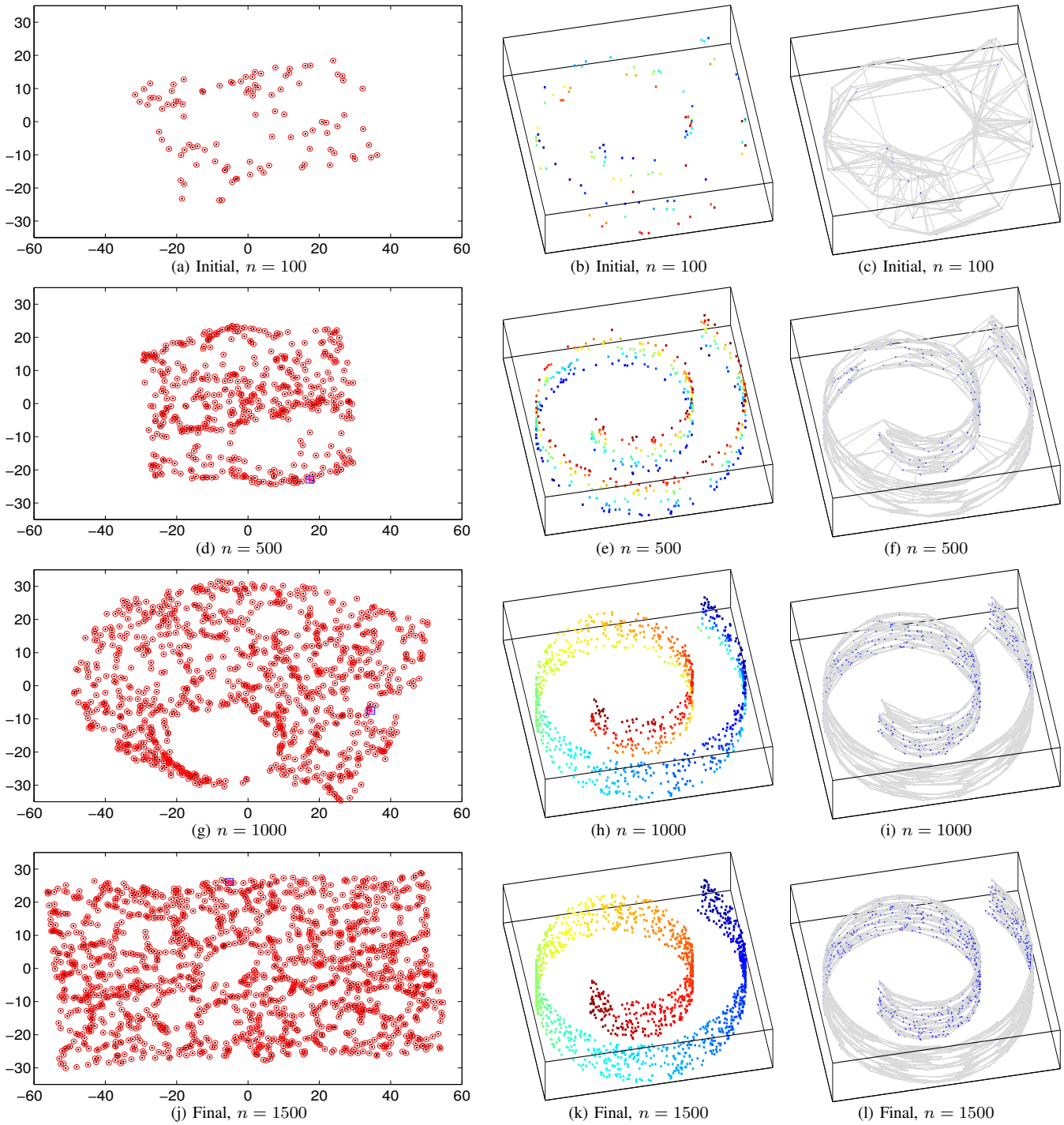
[9]`http://cvc.yale.edu/projects/yalefaces/yalefaces.html`.

[10]It is provided by Dr. Yunhong Wang.

	Swiss roll (3→2)		S-curve (3→2)		Rendered face (30→3)		MNIST 2 (576→6)		ethn (4096→6)	
	Batch	Incr.	Batch	Incr.	Batch	Incr.	Batch	Incr.	Batch	Incr.
Neighborhood graph	230.0	1.4	229.9	1.2	20.8	0.4	239.7	1.8	239.3	1.5
Geodesic distance	1618.5	53.2	1632.8	56.2	157.8	5.8	1683.9	41.0	1752.9	39.6
Computing $\mathbf{x}_{n+1}$	804.6	0.9	760.0	0.8	85.1	0.3	671.3	0.8	645.2	1.0
Updating $\mathbf{x}_i$		49.5		49.4		5.5		51.5		48.9

TABLE I

RUN TIME (SECONDS) FOR BATCH AND INCREMENTAL ISOMAP. THE DIMENSIONS FOR THE INPUT AND THE OUTPUT OF ISOMAP ARE ALSO LISTED. FOR BATCH ISOMAP, COMPUTATION OF $\mathbf{x}_{n+1}$ AND UPDATING OF $\mathbf{x}_i$ ARE PERFORMED TOGETHER. HENCE THERE IS ONLY ONE COMBINED RUN TIME.



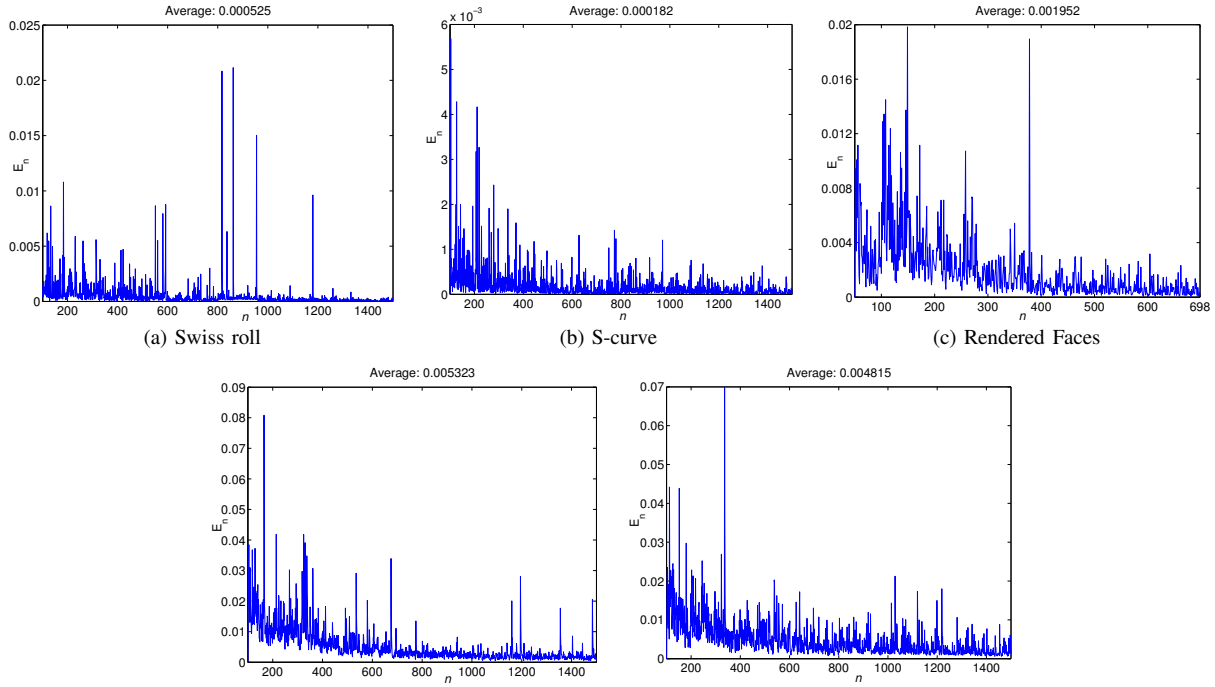
(a) Initial, $n = 100$ (b) Initial, $n = 100$ (c) Initial, $n = 100$

(d) $n = 500$ (e) $n = 500$ (f) $n = 500$

(g) $n = 1000$ (h) $n = 1000$ (i) $n = 1000$

(j) Final, $n = 1500$ (k) Final, $n = 1500$ (l) Final, $n = 1500$

Fig. 4. Snapshots of "Swiss Roll" for incremental ISOMAP. In the first column, the circles and dots in the figures represent the co-ordinates estimated by the batch and the incremental version, respectively. The second column contains scatter plots, where the color of the points correspond to the x co-ordinate of the first column. The third column illustrates the neighborhood graphs, from which the geodesic distances are estimated.



(a) Swiss roll (b) S-curve (c) Rendered Faces

(d) MNIST digit 2 (e) ethn

Fig. 5. Approximation error ($\mathcal{E}_n$) between the co-ordinates estimated by the basic incremental ISOMAP and the basic batch ISOMAP for different numbers of data points (n).

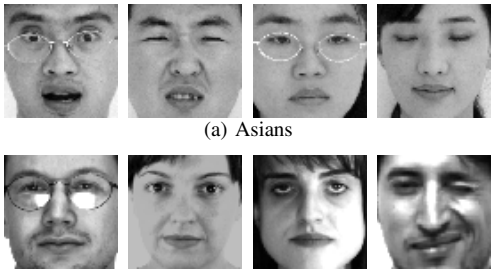
	Swiss roll			S-curve			Rendered face			MNIST 2			ethn		
n	Dist.	Batch	Incr.	Dist.	Batch	Incr.	Dist.	Batch	Incr.	Dist.	Batch	Incr.	Dist.	Batch	Incr.
500	0.09	0.66	0.04	0.09	0.64	0.04	0.16	0.60	0.05	0.28	0.62	0.04	1.19	0.61	0.04
1000	0.38	2.62	0.14	0.38	2.47	0.11	N/A	N/A	N/A	1.09	2.34	0.07	4.52	2.45	0.08
1500	0.84	5.72	0.17	0.84	5.65	0.25	N/A	N/A	N/A	2.42	5.41	0.18	10.06	5.65	0.15

TABLE II

RUN TIME (SECONDS) FOR EXECUTING BATCH AND INCREMENTAL ISOMAP ONCE FOR DIFFERENT NUMBER OF POINTS (n). "DIST" CORRESPONDS TO THE TIME FOR DISTANCE COMPUTATION FOR ALL THE n POINTS.

Fig. 6. Example images from the rendered face image data set.

Fig. 7. Example "2" digits from the MNIST database.



(a) Asians

(b) Non-Asians

Fig. 8. Example face images from ethn database.

approximation error and the computation time for these data sets are shown in Figure 5 and Table I. We can see that the incremental ISOMAP is accurate and efficient for updating the co-ordinates in all these data sets.

Since the ethn data set is from a supervised classification problem, we also want to investigate the quality of the ISOMAP mapping with respect to classification. This is done quantitatively by computing the leave-one-out nearest neighbor (with respect to L_2 distance) error rate using different dimensions of the co-ordinates estimated by incremental ISOMAP with 1500 points. For comparison, we project the data linearly to the best hyperplane by PCA and also evaluate the corresponding leave-one-out error rate. Figure 9 shows the result. The representation recovered by ISOMAP leads to a smaller error rate than PCA. Note that the performance of PCA can be improved by rescaling each feature so that all of them have equal variance, though the rescaling is essentially

	Swiss roll (3→2)		S-curve (3→2)		Rendered face (30→3)		MNIST 2 (576→6)		ethn (4096→6)	
	Batch	Incr.	Batch	Incr.	Batch	Incr.	Batch	Incr.	Batch	Incr.
Neighborhood graph	8745.7	6.6	8692.7	7.1	20.2	0.5	8742.9	7.6	1296.0	3.3
Geodesic distance	824.3	38.9	879.9	39.2	6.5	0.5	913.3	16.9	217.3	6.0
Computing $\mathbf{x}_{n+1}$	199.1	0.5	200.8	0.9	6.6	0.1	210.2	0.9	69.5	0.5
Updating $\mathbf{x}_i$		61.4		62.5		1.2		43.0		17.1

TABLE III

RUN TIME (SECONDS) FOR BATCH AND INCREMENTAL LANDMARK ISOMAP. THE DIMENSIONS FOR THE INPUT AND THE OUTPUT OF ISOMAP ARE ALSO LISTED. FOR BATCH ISOMAP, COMPUTATION OF $\mathbf{x}_{n+1}$ AND UPDATING OF $\mathbf{x}_i$ ARE PERFORMED TOGETHER. HENCE THERE IS ONLY ONE COMBINED RUN TIME.

	Swiss roll			S-curve			Rendered face			MNIST 2			ethn		
	Dist.	Batch	Incr.	Dist.	Batch	Incr.	Dist.	Batch	Incr.	Dist.	Batch	Incr.	Dist.	Batch	Incr.
500	0.09	0.14	0.01	0.09	0.21	0.03	0.16	0.22	0.02	0.28	0.15	0.02	1.19	0.16	0.02
2000	1.53	1.56	0.02	1.50	1.54	0.03	N/A	N/A	N/A	4.25	1.55	0.01	17.70	1.57	0.02
3500	4.61	11.90	0.03	4.64	12.91	0.06	N/A	N/A	N/A	21.72	7.77	0.04	N/A	N/A	N/A
5000	208.52	168.01	0.05	338.12	174.63	0.06	N/A	N/A	N/A	501.81	226.25	0.05	N/A	N/A	N/A

TABLE IV

RUN TIME (SECONDS) FOR BATCH AND INCREMENTAL LANDMARK ISOMAP.

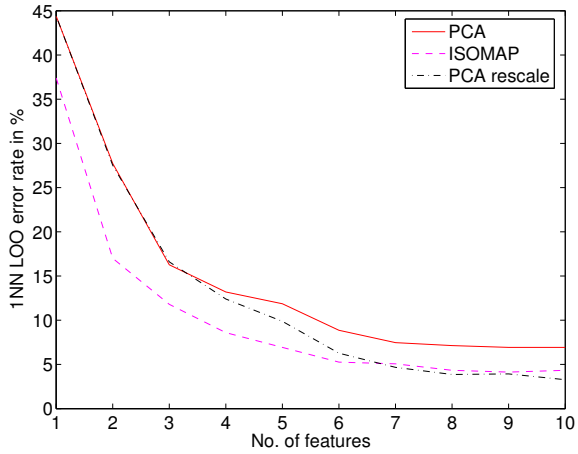


Fig. 9. Classification performance on ethn database for basic ISOMAP.

a post-processing step, not required by ISOMAP.

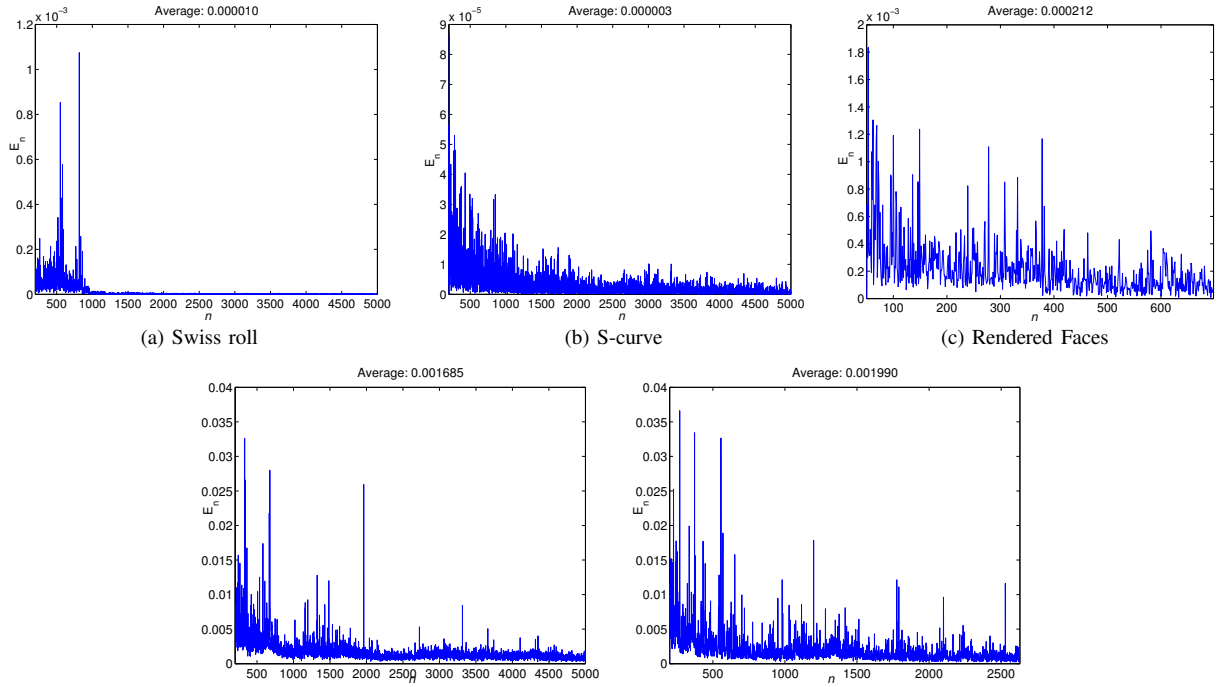
B. Experiments on Landmark ISOMAP

A similar experimental procedure is applied to the incremental landmark ISOMAP described in section III-B for Swiss roll, S-curve, rendered face, MNIST digit 2 and ethn data sets. Starting with 200 randomly selected points from the data set, random points are added until 5000 points[11] are accumulated. Forty points from the initial 200 points are chosen randomly to be the landmark points. The snapshots for incremental landmark ISOMAP are fairly similar to those for incremental ISOMAP in Figure 4, and are omitted due to the lack of space. The approximation error and the computation time for the batch and incremental version of landmark ISOMAP are shown in Figure 10 and Table III, respectively. The time to run the batch version only once is listed in Table IV. Once again, the co-ordinates estimated by the incremental version is accurate with respect to the batch version, and the computation time is much less. We have also investigated the classification accuracy using landmark ISOMAP on all the 2630 images in the ethn data set. The result is similar to that for the standard ISOMAP (Figure 9), and is omitted due to the lack of space.

[11] When the data set has less than 5000 points, the experiment stops after all the points have been used.

C. Vertex Contraction

The utility of vertex contraction is illustrated in the following experiment. Consider a manifold of a 3-dimensional unit hemisphere embedded in a 10-dimensional space. The geodesic on this manifold is simply the great circle, and the geodesic distance between $\mathbf{x}_1$ and $\mathbf{x}_2$ on the manifold is given by $\cos^{-1}(\mathbf{x}_1^T\mathbf{x}_2)$. Data points lying on this manifold are randomly generated. With $K = 6$, 40 landmark points and 1000 points in memory, the vertex that is most "crowded" is contracted, until 10000 points are examined. The geodesic distances between the landmark points $\mathcal{X}^L$ and the points in memory $\mathcal{X}^M$ are compared with the ground-truth, and the discrepancy is shown by the solid line in Figure 11. As more points are encountered, the error decreases, indicating that vertex contraction indeed improves the geodesic distance estimate. There is, however, a lower limit (around 0.03) on the achievable accuracy, because of the finite size of samples retained in the memory. For comparison, the same procedure is repeated, but with all the points stored in the memory instead of being contracted. The quality of the geodesic distance estimate between the same $\mathcal{X}^L$ and $\mathcal{X}^M$ is evaluated in the same manner. As we can see in the dash-dot line in Figure 11, the improvement of geodesic distance estimate is significantly slower. Vertex contraction improves the geodesic distance estimate, partly because it spreads the data points more evenly, and partly because more points are included in the neighborhood effectively.



(a) Swiss roll (b) S-curve (c) Rendered Faces

(d) MNIST digit 2 (e) ethn

Fig. 10. Approximation error ($\mathcal{E}_n$) between the co-ordinates estimated by the incremental landmark ISOMAP and the batch landmark ISOMAP for different numbers of data points (n).

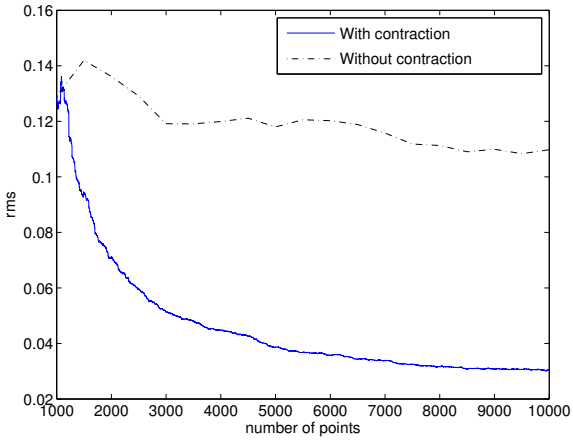


Fig. 11. Utility of vertex contraction. Solid line: the root-mean-square error (when compared with the ground truth) of the geodesic distance estimate for points currently held in memory when vertex contraction is used. Dash-dot line: the corresponding root-mean-square error when the new points are stored in the memory instead of being contracted.

D. Incorporating Variance By Incremental Learning

One interesting use of incremental learning is to incorporate invariance by "hallucinating" training data. Given a training sample $\mathbf{y}_i$, additional training data $\mathbf{y}_i^{(1)}, \mathbf{y}_i^{(2)}, \ldots$ can be created by applying different invariance transformations on $\mathbf{y}_i$. This idea has been exploited in [43] for improving the accuracy in digit classification. Given a digit image, simple distortions like translation, rotation and skewing are applied to create additional training data for improving the invariance property of a neural network. In this setting, the number of training data can be infinite, because the number of possible invariance transformations is infinite. This unboundedness calls for an incremental algorithm which can accumulate the effect of the data generated. An incremental algorithm has the additional benefit of maintaining a constant memory usage, as well as monitoring the progress of the learning process.

We tested a similar idea using the proposed incremental ISOMAP. The training data are generated by first randomly selecting an image from 500 digit "2" images in the MNIST training set. The image is then rotated randomly by θ degree, where θ is uniformly distributed in $[-30, 30]$. The image is used as the input for the incremental landmark ISOMAP with 40 landmarks and a memory size of 10000, with vertex contraction enabled. The training is stopped when 60000 training images are generated. We want to investigate how well the rotation angle is recovered by the nonlinear mapping. This is done by using an independent set of digit "2" images from the MNIST testing set, which is of size 1032. For each image $\tilde{\mathbf{y}}^{(i)}$, it is rotated in 15 different angles: $30j/7$ for $j = -7, \ldots, 7$. The mappings of these 15 images, $\tilde{\mathbf{x}}_{-7}^{(i)}, \ldots, \tilde{\mathbf{x}}_{7}^{(i)}$, are found using the out-of-sample extension of ISOMAP. If ISOMAP can discover the rotation angle, there should exist a linear projection direction $\mathbf{h}$ such that $\mathbf{h}^T \tilde{\mathbf{x}}_l^{(i)} \approx c_i + l$ for all i and l, where c_i is a constant specific to $\tilde{\mathbf{y}}^{(i)}$. This is equivalent to

$$\mathbf{h}^T \left(\tilde{\mathbf{x}}_l^{(i)} - \tilde{\mathbf{x}}_0^{(i)} \right) \approx l, \tag{IV.2}$$

which is an over-determined linear system. The goodness of the mapping $\tilde{\mathbf{x}}_l^{(i)}$ in terms of how well the rotation angle is recovered can thus be quantified by the residue of the above equation. For comparison, a similar procedure is applied for PCA using the first 10000 generated images. Figure 12 shows the result. We can see that the residue for ISOMAP is smaller

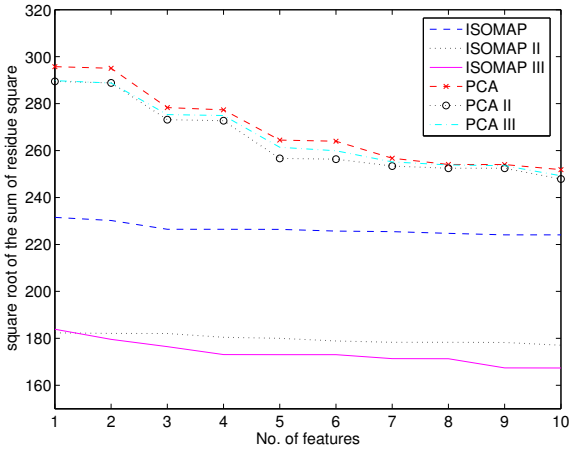


Fig. 12. Sum of residue square for 1032 images in 15 rotation angles. The larger the residue, the worst the representation. "PCA" and "ISOMAP" correspond to the nonlinear mapping obtained by PCA and ISOMAP when 10000 generated images are used for training, respectively. "ISOMAP II"/"PCA II" and "ISOMAP III"/"PCA III" correspond to the result when the learning stops after 20000 and 50000 images are generated, respectively.

than PCA, indicating that ISOMAP recovers the rotation angle better. The difference is more pronounced when more images are used. The adaptive nature of the proposed algorithm allows us to utilize a larger amount of data while keeping the memory usage almost constant.

V. Discussion

We have presented algorithms to incrementally update the co-ordinates produced by ISOMAP. Our approach can be extended to other manifold learning algorithms; for example, creating an incremental version of Laplacian eigenmap requires the update of the neighborhood graph and the leading eigenvectors of a matrix (graph Laplacian) derived from the neighborhood graph.

The convergence of geodesic distance is guaranteed since the geodesic distances are maintained exactly. Subspace iteration used in co-ordinate update is provably convergent if a sufficient number of iterations is used, assuming all eigenvalues are simple, which is generally the case. The fact that we only run subspace iteration once can be interpreted as trading off guaranteed convergence with empirical efficiency. Since the change in target inner product matrix is often small, the eigenvector improvement due to subspace iterations with different number of points is aggregated, leading to the low approximation error as shown in Figures 5 and 10.

While running the proposed incremental ISOMAP is much faster than running the batch version repeatedly, it is more efficient to run the batch version once using all the data points if only the final solution is desired (compare Tables I and II, as well as Tables III and IV). It is because maintaining intermediate geodesic distances and co-ordinates accurately requires extra computation. The incremental algorithm can be made faster if the geodesic distances are updated upon seeing p subsequent points, $p > 1$. We first embed $\mathbf{y}_{n+1}, \ldots, \mathbf{y}_{n+p}$ independently by the method in section III-A.3. The geodesic distances among the existing points are not updated, and the same set of $\mathbf{x}_i$ is used to find $\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+p}$. After that, all the geodesic distances are updated, followed by the update of $\mathbf{x}_1, \ldots, \mathbf{x}_{n+p}$ by subspace iteration. This strategy makes the incremental algorithm almost p-times faster, because the time to embed the new points is very small (see the time for "computing $\mathbf{x}_{n+1}$" in Tables I and III). On the other hand, the quality of the embedding will deteriorate because the embedding of the existing points cannot benefit from the new points.

Also, for a fixed amount of memory, the solution obtained by the incremental version can be superior to that of the batch version. This is because the incremental version can perform vertex contraction, thereby obtaining a better geodesic distance estimate. The incremental version can be easily adopted to unbounded data stream when training data are generated by applying invariance transformation, too.

A. Variants of the Main Algorithms

Our incremental algorithm can be modified to cope with variable neighborhood definition, if the user is willing to do some tedious book-keeping. We can, for example, use ϵ-neighborhood with the value of ϵ re-adjusted whenever, say, 200 data points have arrived. The edges that need to be deleted or added because of the new neighborhood definition are calculated, before applying the proposed algorithm for update. The supervised ISOMAP algorithm in [51], which utilizes a criterion similar to Fisher discriminant for embedding, can also be converted to become incremental. The only change is that the subspace iteration method for solving a generalized eigenvalue problem is used instead. The conformal ISOMAP in [14] can also be made to be incremental. In conformal ISOMAP, the edge weight w_{ij} of the neighborhood graph is $\Delta_{ij}/\sqrt{M(i)M(j)}$, where $M(i)$ denotes the average distance of $\mathbf{y}_i$ from its k nearest neighbors. To convert this to its incremental counterpart, the sum of the weights of the k nearest neighbors of different vertices should be maintained. The update of geodesic distance and co-ordinate is carried in a manner similar to the proposed incremental ISOMAP.

B. Comparison With the Out-of-sample Extension

One problem closely related to incremental nonlinear mapping is the "out-of-sample extension" [3]: given the embedding $\mathbf{x}_1, \ldots, \mathbf{x}_n$ for a "training set" $\mathbf{y}_1, \ldots, \mathbf{y}_n$, what is the embedding result $(\mathbf{x}_{n+1})$ for a "testing" point $\mathbf{y}_{n+1}$? This is effectively the problem considered in section III-A.3. In incremental learning, however, we go one step further than obtaining $\mathbf{x}_{n+1}$: the co-ordinate estimates $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are also improved by the new point $\mathbf{y}_{n+1}$. In the case of incremental ISOMAP, this amounts to updating the geodesic distances and then applying subspace iteration.

The out-of-sample extension is faster because it skips the improvement step. However, it is less accurate, and cannot provide intermediate embedding with good quality as points are accumulated. Incremental ISOMAP, on the other hand, utilizes the new samples to continuously improve the co-ordinate estimates. The out-of-sample extension may be more appealing when a large number of samples have been accumulated and the geodesic distances and $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are reasonably accurate.

Even in this case, though, the strategy of updating $\mathbf{x}_1, \ldots, \mathbf{x}_n$ after embedding p new points (with $p > 1$) works equally as well. The geodesic distances and co-ordinates and updated infrequently in this case, and its amortized time cost is low.

Incremental ISOMAP is also preferable to the out-of-sample extension when there is a drifting of data characteristics. In the out-of-sample extension, the n points collected are assumed to be representative for all future data points. There is no way to capture the change of data characteristics. In incremental ISOMAP, however, we can easily maintain an embedding using a window of points recently encountered. Changes in data characteristics are captured as the geodesic distances and co-ordinate estimates are updated. Vertex contraction should be turned off if incremental ISOMAP is run in this mode, to ensure that the effect of old data points is erased.

C. Implementation Details

The subspace iteration in section III-A.4 requires that the eigenvalues corresponding to the leading eigenvectors have the largest absolute values. This can be violated if the target inner product has a large negative eigenvalue. To tackle this, we shift the spectrum and find the eigenvectors of $(\mathbf{B} + \alpha\mathbf{I})$ instead of $\mathbf{B}$. We empirically set $\alpha = \max(-0.7\lambda_{\text{min}}(\mathbf{B}) - 0.3\lambda_{d\text{-th}}(\mathbf{B}), 0)$, where $\lambda_{\text{min}}(\mathbf{B})$ and $\lambda_{d\text{-th}}(\mathbf{B})$ denote the current estimate of the smallest (most negative) and the d-th largest eigenvalues, respectively.

During the incremental learning, the neighborhood graph may be temporarily disconnected. A simple solution is to embed only the largest graph component. The excluded vertices are added back for embedding again when they become reconnected as additional data points are encountered. Alternatively, an edge can be added between the two nearest vertices to connect the two disconnected components in the neighborhood graph.

VI. Conclusions and Future Work

Nonlinear dimensionality reduction is an important problem with applications in pattern recognition, computer vision and machine learning. We have proposed an algorithm for incremental nonlinear mapping problem by modifying the ISOMAP algorithm. The core idea is to efficiently update the geodesic distances (a graph theoretic problem) and re-estimate the eigenvectors (a numerical analysis problem), using the previous computation results. Our experiments validate that the proposed method is almost as accurate as running the batch version, while saving significant computation time.

There is still room to improve the efficiency of the proposed algorithm. Data structures such as kd-tree, ball-tree or cover-tree [5] can be used to speed up the search of the k nearest neighbors. The update strategy for geodesic distance and co-ordinates can be more aggressive; we can sacrifice the theoretical convergence property in favor of empirical efficiency. For example, the geodesic distance can be updated approximately using a scheme analogous to the distance vector protocol in the network routing literature. Alternative algorithms [16], [37] can be explored for shortest path tree update. Co-ordinate update can be made faster if only a subset of the co-ordinates (such as those close to the new point) are updated at each iteration. The co-ordinates of every point would be finally updated if the new points come from different regions of the manifold. The sparseness of the change in geodesic distances may be exploited to improve the efficiency. Other methods such as the Lanczos method [22] can be adopted to further improve the accuracy of the embedded co-ordinates.

Acknowledgement

This research was supported by ONR contract # N00014-01-1-0266. The authors are grateful to Xiaoguang Lu and Dr. Yunhong Wang for the NLPR database. The authors also want to thank Tilman Lange and the anonymous reviewers for their helpful comments on the earlier versions of the manuscript.

References

[1] P. Baldi and K. Hornik. Neural networks and principal component analysis: learning from examples without local minima. *Neural Networks*, 2:53–58, 1989.

[2] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, June 2003.

[3] Y. Bengio, J.-F. Paiement, and P. Vincent. Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps, and spectral clustering. In *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

[4] M. Bernstein, V. de Silva, J. Langford, and J. Tenenbaum. Graph approximations to geodesics on embedded manifolds. Technical report, Department of Psychology, Stanford University, 2000.

[5] A. Beygelzimer, S.M. Kakade, and J. Langford. Cover trees for nearest neighbor. Technical report, 2005. `http://www.cis.upenn.edu/~skakade/papers/ml/cover_tree.pdf`.

[6] C.M. Bishop, M. Svensen, and C.K.I. Williams. GTM: the generative topographic mapping. *Neural Computation*, 10:215–234, 1998.

[7] M. Brand. Charting a manifold. In *Advances in Neural Information Processing Systems 15*, pages 961–968. MIT Press, 2003.

[8] A. Brun, H-J. Park, H. Knutsson, and Carl-Fredrik Westin. Coloring of DT-MRI fiber traces using Laplacian eigenmaps. In *Proc. the Ninth International Conference on Computer Aided Systems Theory*, volume 2809, February 2003.

[9] J. Bruske and G. Sommer. Intrinsic dimensionality estimation with optimally topology preserving maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):572–575, 1998.

[10] F. Camastra and A. Vinciarelli. Estimating the intrinsic dimension of data with a fractal-based method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(10):1404–1407, October 2002.

[11] Y. Chang, C. Hu, and M. Matthew Turk. Probabilistic expression analysis on manifolds. In *Proc. the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 520–527, 2004.

[12] J. Costa and A.O. Hero. Manifold learning using euclidean k-nearest neighbor graphs. In *Proc. IEEE International Conference on Acoustic Speech and Signal Processing*, volume 3, pages 988–991, Montreal, 2004.

[13] T.F. Cox and M.A.A. Cox. *Multidimensional Scaling*. Chapman & Hall, 2001.

[14] V. de Silva and J.B. Tenenbaum. Global versus local approaches to nonlinear dimensionality reduction. In *Advances in Neural Information Processing Systems 15*, pages 705–712. MIT Press, 2003.

[15] D. DeMers and G. Cottrell. Non-linear dimensionality reduction. In *Advances in Neural Information Processing Systems 5*, pages 580–587. Morgan Kaufmann, 1993.

[16] C. Demetrescu and G.F. Italiano. A new approach to dynamic all pairs shortest paths. *Journal of the ACM*, 51(6):968–992, November 2004.

[17] D.L. Donoho and C. Grimes. When does isomap recover natural parameterization of families of articulated images? Technical Report 2002-27, Department of Statistics, Stanford University, August 2002.

[18] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. John Wiley & Sons, New York, 2nd edition, 2001.

[19] A. Elgammal and C.S. Lee. Inferring 3D body pose from silhouettes using activity manifold learning. In *Proc. the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 681–688, 2004.

[20] A. Elgammal and C.S. Lee. Separating style and content on a nonlinear manifold. In *Proc. the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 478–489, 2004.

[21] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):214–225, February 2004.

[22] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.

[23] A. Hadid, O. Kouropteva, and M. Pietikainen. Unsupervised learning using locally linear embedding: experiments in face pose analysis. In *Proc. the 16th International Conference on Pattern Recognition*, pages I:111–114, 2002.

[24] T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84:502–516, 1989.

[25] G.E. Hinton, P. Dayan, and M. Revow. Modeling the manifolds of handwritten digits. *IEEE Transactions on Neural Networks*, 8(1):65–74, January 1997.

[26] O. Jenkins and M. Mataric. A spatio-temporal extension to Isomap nonlinear dimension reduction. In *Proc. the 21st International Conference on Machine Learning*, 2004.

[27] B. Kégl. Intrinsic dimension estimation using packing numbers. In *Advances in Neural Information Processing Systems 15*. MIT Press, 2003.

[28] B. Kégl, A. Krzyzak, T. Linder, and K. Zeger. Learning and design of principal curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(3):281–297, 2000.

[29] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, 2001. 3rd edition.

[30] M.H. Law, N. Zhang, and A.K. Jain. Non-linear manifold learning for data stream. In *Proc. SIAM International Conference for Data Mining*, pages 33–44, 2004.

[31] E. Levina and P.J. Bickel. Maximum likelihood estimation of intrinsic dimension. In *Advances in Neural Information Processing Systems 17*. MIT Press, 2005.

[32] S.Z. Li, R. Xiao, Z.Y. Li, and H.J. Zhang. Nonlinear mapping from multi-view face patterns to a gaussian distribution in a low dimensional space. In *Proc. IEEE ICCV Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-time Systems*, 2001.

[33] X. Lu and A.K. Jain. Ethnicity identification from face images. In *Proc. SPIE*, volume 5404, pages 114–123, 2004.

[34] J. Mao and A.K. Jain. Artificial neural networks for feature extraction and multivariate data projection. *IEEE Transactions on Neural Networks*, 6(2):296–317, March 1995.

[35] A. Martinez and R. Benavente. The AR face database. Technical Report 24, CVC, 1998. `http://rvl1.ecn.purdue.edu/~aleix/aleix_face_DB.html`.

[36] P. Narváez, K.-Y. Siu, and H.-Y. Tzeng. New dynamic algorithms for shortest path tree computation. *IEEE/ACM Transactions on Networking*, 8(6):734–746, December 2000.

[37] P. Narváez, K.-Y. Siu, and H.-Y. Tzeng. New dynamic SPT algorithm based on a ball-and-string model. *IEEE/ACM Transactions on Networking*, 9(6):706–718, December 2001.

[38] M. Niskanen and O. Silvn. Comparison of dimensionality reduction methods for wood surface inspection. In *Proc. 6th International Conference on Quality Control by Artificial Vision*, pages 178–188, 2003.

[39] K. Pettis, T. Bailey, A.K. Jain, and R. Dubes. An intrinsic dimensionality estimator from near-neighbor information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(1):25–36, January 1979.

[40] S.T. Roweis, L.K. Saul, and G.E. Hinton. Global coordination of local linear models. In *Advances in Neural Information Processing Systems 14*, pages 889–896. MIT Press, 2002.

[41] J.W. Sammon. A non-linear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18(5):401–409, May 1969.

[42] B. Schölkopf, A.J. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[43] P.Y. Simard, D. Steinkraus, and J. Platt. Best practice for convolutional neural networks applied to visual document analysis. In *Proc. the International Conference on Document Analysis and Recogntion*, pages 958–962, 2003.

[44] E. Sjöström. Singular value computations for toeplitz matrices. *Licentiate thesis*, 1996. `http://www.mai.liu.se/~evlun/pub/lic/lic.html`.

[45] A.J. Smola, S. Mika, B. Schölkopf, and R.C. Williamson. Regularized principal manifolds. *Journal of Machine Learning Research*, 1:179–209, June 2001.

[46] J.B. Tenenbaum, V. de Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[47] R. Tibshirani. Principal curves revisited. *Statistics and Computing*, 2:183–190, 1992.

[48] J.J. Verbeek, N. Vlassis, and B. Krose. Coordinating principal component analyzers. In *Proc. International Conference on Artificial Neural Networks*, pages 914–919, 2002.

[49] K.Q. Weinberger and L.K. Saul. Unsupervised learning of image manifolds by semidefinite programming. In *Proc. the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 988–995, 2004.

[50] J. Weng, Y. Zhang, and W.S. Hwang. Candid covariance-free incremental principal component analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):1034–1040, 2003.

[51] M.-H. Yang. Face recognition using extended isomap. In *IEEE International Conference on Image Processing*, pages II: 117–120, 2002.

[52] H. Zha and Z. Zhang. Isometric embedding and continuum isomap. In *Proc. the 20th International Conference on Machine Learning*, 2003.

[53] J. Zhang, Stan Z. Li, and J. Wang. Nearest manifold approach for face recognition. In *Proc. The 6th International Conference on Automatic Face and Gesture Recognition*, Seoul, Korea, May 2004.