Manifold Learning by Graduated Optimization

Michael Gashler, Dan Ventura, and Tony Martinez

***Abstract*—We present an algorithm for manifold learning called *manifold sculpting*, which utilizes graduated optimization to seek an accurate manifold embedding. An empirical analysis across a wide range of manifold problems indicates that manifold sculpting yields more accurate results than a number of existing algorithms, including Isomap, locally linear embedding (LLE), Hessian LLE (HLLE), and landmark maximum variance unfolding (L-MVU), and is significantly more efficient than HLLE and L-MVU. Manifold sculpting also has the ability to benefit from prior knowledge about expected results.**

***Index Terms*—Manifold learning, nonlinear dimensionality reduction, unsupervised learning.**

I. Introduction

LARGE dimensionality is a significant problem for many machine learning algorithms [1]. Dimensionality reduction algorithms address this issue by projecting data into fewer dimensions while attempting to preserve as much of the informational content in the data as possible.

Dimensionality reduction involves transforming data to occupy as few dimensions as possible so that the other dimensions may be eliminated with minimal loss of information. Nonlinear transformations not only have more flexibility to align the data with a few dimensional axes but also have more potential to disrupt the structure of the data in that process. Manifold learning algorithms seek a balance by prioritizing the preservation of data structure in local neighborhoods. A projection is deemed to be good if the relationships (typically, distances and/or angles) between neighboring points after the projection are very similar to the relationships between those same points before the projection.

Manifold learning therefore requires solving an optimization problem. In general, global optimization over a nonlinear error surface is an NP-hard problem [2]. Most popular manifold learning algorithms, such as Isomap [3] and locally linear embedding (LLE) [4], approach this problem by casting it as an overconstrained convex optimization problem in the low-dimensional space. Unfortunately, much is lost in casting the inherently nonconvex problem as a convex problem. The solution to the convex problem can typically be rapidly computed, but the results do not necessarily preserve the distances and angles between neighboring points as well as can be done in low-dimensional space. Algorithms that perform optimization in the high-dimensional space, such as maximum variance unfolding (MVU) [5], produce better results but tend to have unreasonably high computational costs.

Manuscript received July 7, 2010; revised January 11, 2011; accepted April 23, 2011. Date of publication May 27, 2011; date of current version November 18, 2011. This paper was recommended by Associate Editor P. S. Sastry.

The authors are with the Department of Computer Science, Brigham Young University, Provo, UT 84602 USA (e-mail: mike@axon.cs.byu.edu; ventura@axon.cs.byu.edu; martinez@cs.byu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCB.2011.2151187

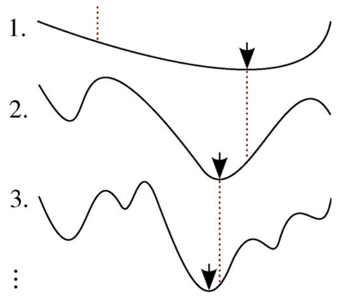

Fig. 1. With graduated optimization, the solution to each optimization problem gives a good starting point for the next harder problem. If the optimum of the first problem is found, and the solution to each problem is within the convex region around the optimum of the next problem, then graduated optimization will find the optimum of the final problem.

We make the novel observation that the optimization problem inherent in manifold learning can be solved using graduated optimization. Graduated optimization involves solving a sequence of successively more difficult optimization problems, where the solution to each problem gives a good starting point for the next problem, as illustrated in Fig. 1. This technique is commonly used with hierarchical pyramid methods for matching objects within images [6]. A related technique called numerical continuation [7] has been used to approximate solutions to parameterized equations in chaotic dynamical systems, molecular conformation, and other areas. To our knowledge, graduated optimization has not yet been recognized as being suitable for addressing the problem of manifold learning. With graduated optimization, if the first optimization problem in the sequence can be solved and if the solution to each problem falls within the locally convex region around the solution to the next problem, then it will find the globally optimal solution to the nonconvex optimization problem at the end of the sequence.

We present an algorithm for manifold learning called *manifold sculpting*, which discovers manifolds through a process of graduated optimization. Manifold sculpting approaches the optimization problem of manifold learning in a manner that enables it to solve the optimization problem in the original high-dimensional space, while only requiring the computational cost of optimizing in the reduced low-dimensional space. Furthermore, because graduated optimization is robust to local optima, it is not necessary to cast it as an overconstrained convex optimization problem. Instead, manifold sculpting directly optimizes to restore the relationships computed between neighboring points. This gives manifold sculpting the flexibility to operate using an arbitrary set of distance metrics or other relationship metrics. Additionally, manifold sculpting has the ability to benefit from prior knowledge about expected results and the ability to further refine the results from faster manifold learning algorithms. We report results from a variety of experiments, which demonstrate that manifold sculpting yields results that are typically about an order of magnitude more accurate than state-of-the-art manifold learning algorithms, including Hessian LLE (HLLE), and landmark MVU (L-MVU).

Section II discusses work that has previously been done in manifold learning. Section III describes the manifold sculpting algorithm in detail. Section IV reports the results of a thorough empirical analysis comparing manifold sculpting with existing manifold learning algorithms. Finally, Section V summarizes the contributions of manifold sculpting.

II. Related Work

Dimensionality reduction has been studied for a long time [8] but has only started to become a mainstay of machine learning in the last decade. More algorithms exist than we can mention, but we will attempt to give a summary of the major work that has been done in this field. Early nonlinear dimensionality reduction algorithms, such as nonlinear multidimensional scaling [9] and nonlinear mapping [10], have shown effectiveness but are unable to handle high nonlinearities in the data. Curvilinear component analysis [11] uses a neural network technique to solve the manifold embedding, and curvilinear distance analysis (CDA) [12] takes it a step further by using distance on the manifold surface as a metric for identifying manifold structure. These algorithms are unfortunately computationally demanding and suffer from the problems of local minima.

Isomap [3] uses the same metric as CDA but solves for the embedding into fewer dimensions using classic multidimensional scaling, which enables it to operate significantly faster. Unfortunately, it still struggles poorly sampled areas of the manifold. (See Fig. 2(a).) LLE [4] achieves even better speed by using only local vector relationships represented in a sparse matrix. It is more robust to sample holes but tends to produce quite distorted results when the sample density is nonuniform. (See Fig. 2(b).) With these algorithms, a flurry of new research in manifold learning began to produce numerous new techniques. L-Isomap is an improvement to the Isomap algorithm that uses landmarks to reduce the amount of necessary computation [13]. Other algorithms include kernel principal component analysis [14], Laplacian eigenmaps [15], manifold charting [16], manifold Parzen windows [17], HLLE [18], and there are many more [19]–[21]. HLLE preserves the manifold structure better than the other algorithms but is unfortunately very computationally expensive. (See Fig. 2(c).)

More recently, the MVU algorithm has become popular for manifold learning [5]. This algorithm seeks to maximize variance in the data points while preserving distances and angles in local neighborhoods. It finds the solution to this problem using semidefinite programming. Unfortunately, because it optimizes in the original high-dimensional space and because of the computational complexity of semidefinite programming, it is too inefficient to operate on large data sets. L-MVU [22] utilizes randomly chosen landmarks to reduce the computational complexity of MVU. (See Fig. 2(d).) Although this technique yields somewhat degraded results, it makes the algorithm more suitable for larger problems. Excessive computational complexity, however, is still the most significant drawback of L-MVU.

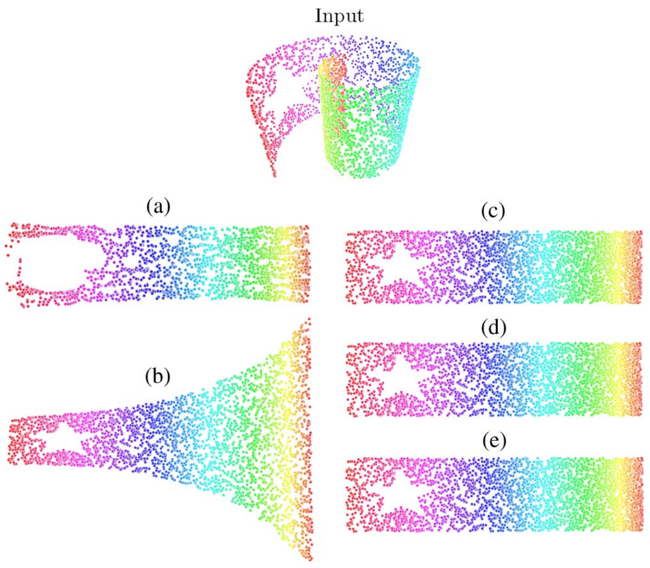

Fig. 2. Comparison of several manifold learners with a Swiss Roll manifold. Color is used to indicate how points in the results correspond to points on the manifold. Isomap has trouble with sampling holes. LLE has trouble with changes in sample density. HLLE, L-MVU, and manifold sculpting all produce very good results with this particular problem. (Results with other problems are presented in Section IV.) (a) Isomap. (b) LLE. (c) HLLE. (d) L-MVU with 56 landmarks. (e) Manifold sculpting.

Several recent manifold learning algorithms have also been presented with specialized capabilities. For example, LGGA [23] creates a continuous mapping such that out-of-sample points can efficiently be projected onto a learned manifold. We show that manifold sculpting can achieve a similar capability using a pseudoincremental technique. S-Isomap [24] has the ability to benefit from partial supervision. Manifold sculpting can also utilize supervision to improve its manifold embedding. TRIMAP [25] and the D-C Method [26] are manifold learning techniques that specifically seek to preserve class separability in their projections for classification tasks. Manifold sculpting is not specifically designed for this application.

The primary difference between other methods that have been presented and manifold sculpting is that others use various convex optimization techniques to approximate a solution to the nonconvex problem of preserving relationships in local neighborhoods, while the latter seeks to directly solve this nonconvex optimization problem with the use of graduated optimization. Manifold sculpting was first presented in [27]. In this paper, we present an improved version of the algorithm, demonstrate additional capabilities and show that it gives better results than modern algorithms. (Fig. 2(e) shows that results with the Swiss Roll manifold are visually similar to those of HLLE or L-MVU. In Section IV, we show that an empirical analysis with this and several other problems indicates that the results of manifold sculpting tend to be much more accurate than those of other algorithms.)

TABLE I
HIGH-LEVEL OVERVIEW OF THE MANIFOLD SCULPTING ALGORITHM

1	Find the k-nearest neighbors of each point.
2	Compute a set of relationships between neighboring points.
3	Pre-process the data with a faster dimensionality reduction algorithm. (This step is optional.)
4	Do until no improvement is made for 50 iterations: a. Scale the data in the non-preserved dimensions by a constant factor σ, where $\sigma < 1$. b. Restore the relationships computed in step 2 by adjusting the data points in the first t dimensions.
5	Drop the non-preserved dimensions from the data.

III. Manifold Sculpting Algorithm

An overview of the manifold sculpting algorithm is provided in Table I, and the detailed pseudocode is provided in Fig. 3. Let

d	original dimensionality of the data;
t	number of target dimensions into which the data will be projected;
k	number of neighbors used to define a local neighborhood;
$\mathbf{P}$	set of all data points represented as vectors in $\Re^d$ such that p_{ij} is the jth dimensional element of the ith point in $\mathbf{P}$;
$\mathbf{N}$	$\lvert\mathbf{P}\rvert \times k$ matrix such that n_{ij} is the index of the jth neighbor of point $\mathbf{p}_{i,*}$;
σ	constant scaling factor;
η	step size (which is dynamically adjusted).

A. Steps 1 and 2: Compute Local Relationships

Manifold sculpting can operate using custom distance/relationship metrics. In our implementation, we use Euclidean distance and local angles. We compute the k-nearest neighbors $\mathbf{n}_{ij}$ of each point. For each j (where $1 \leq j \leq k$), we compute the Euclidean distance δ_{ij} between $\mathbf{p}_{i,*}$ and each of its neighbor points. We also compute the angle θ_{ij} formed by the two line segments ($\mathbf{p}_{i,*}$ to point n_{ij}) and (point $\mathbf{n}_{ij}$ to point $\mathbf{m}_{ij}$), where point $\mathbf{m}_{ij}$ is the most colinear neighbor of point $\mathbf{n}_{ij}$ with $\mathbf{p}_{i,*}$. (See Fig. 4.) The most colinear neighbor is the neighbor point that forms the angle closest to π. The values of δ and θ define the relationships that the algorithm will seek to preserve during the transformation. The global average distance between all the neighbors of all points δ_{ave} is also computed so that distances may be normalized.

B. Step 3: Optionally Preprocess the Data

The data may optionally be preprocessed with another dimensionality reduction algorithm. Manifold sculpting will work without this step; however, preprocessing may result in even faster convergence. For example, a fast but imprecise algorithm, such as LLE, may be used to initially unfold the manifold, and then, manifold sculpting can further refine its results to obtain a better embedding. (This technique is demonstrated in Section IV-B.) Even preprocessing with a linear dimensionality reduction algorithm may give some speed benefit. For example, principal component analysis (PCA) can be used to rotate the dimensional axes to shift the information in the data into the first several dimensions. Manifold sculpting will then further compact the information by unfolding the nonlinear components in the data. Except where otherwise noted, we use PCA to preprocess data in this manner.

```
function manifold_sculpting($\mathbf{P}$)
       $t \leftarrow 2$, $k \leftarrow 18$, $\sigma \leftarrow 0.999$
  1    for $i$ from 0 to $|\mathbf{P}| - 1$:
           $\mathbf{n}_i \leftarrow$ the $k$ nearest neighbors of $\mathbf{p}_{i,*}$
  2    for $i$ from 0 to $|\mathbf{P}| - 1$:
           for $j$ from 0 to $k - 1$:
               $\delta_{ij} \leftarrow \mathbf{P}.\mathrm{distance}(i, \mathbf{n}_{ij})$
               $\theta_{ij} \leftarrow \max_{0<l\leq k} \mathbf{P}.\mathrm{angle}(i, n_{ij}, n_{jl})$
               $\mathbf{M}_{ij} \leftarrow \mathrm{argmax}_{0<l\leq k} \mathbf{P}.\mathrm{angle}(i, n_{ij}, n_{jl})$
       $\delta_{ave} \leftarrow$ average_neighbor_distance()
       $\eta \leftarrow \delta_{ave}$
  3    Call align_axes_with_principal_components($\mathbf{P}$)
  4    Until at least ($\log_\sigma 0.01$) iterations, and until no
        improvement is made for 50 iterations, do:
  4a       for $i$ from 0 to $|\mathbf{P}| - 1$:
               for $j$ from $t$ to $d - 1$:
                   $p_{ij} \leftarrow \sigma p_{ij}$
           while average_neighbor_distance() $< \delta_{ave}$:
               for $i$ from 0 to $|\mathbf{P}| - 1$:
                   for $j$ from 0 to $t - 1$:
                       $p_{ij} \leftarrow p_{ij}/\sigma$
  4b       $r \leftarrow$ random value, $0 \leq r < |\mathbf{P}|$
           add point $r$ to a queue
           $steps \leftarrow 0$
           $\mathcal{A} \leftarrow$ empty set
           while the queue is not empty, do:
               pop index $i$ from the queue
               if $i \notin \mathcal{A}$:
                   $steps \leftarrow steps +$ adjust_point($\mathbf{p}, \eta$)
                   add each neighbor of $\mathbf{p}_{i,*}$ to the queue
                   add $i \rightarrow \mathcal{A}$
           if $steps \geq |\mathbf{P}|$:
               $\eta \leftarrow \eta * 1.1$
           else
               $\eta \leftarrow \eta * 0.9$
  5    Drop all dimensions $\geq t$
```

Fig. 3. Pseudocode for the manifold sculpting algorithm. Note that the pseudocode for the align_axes_with_principal_components function is given in the Appendix, and the pseudocode for the adjust_point function is given in Fig. 6. A C++ implementation of manifold sculpting is available online at http://waffles.sourceforge.net.

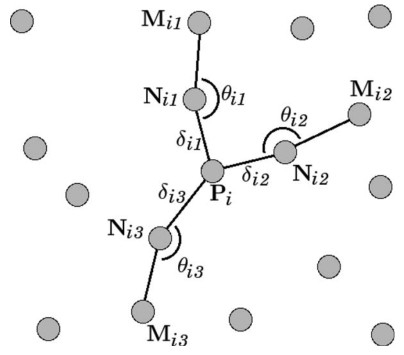

Fig. 4. δ and θ define the relationships that manifold sculpting seeks to preserve in the projection.

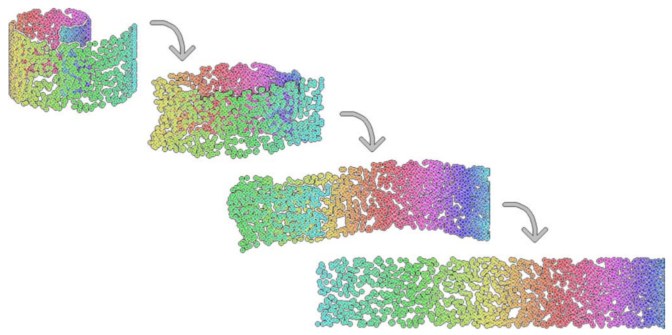

Fig. 5. Swiss Roll manifold shown at four stages of the iterative transformation. This experiment was performed with 2000 data points, $k = 14$, $\sigma = 0.99$, and iterations $= 300$.

Efficient PCA algorithms generally compute only the first few principal components and simultaneously project away the additional dimensions. In this case, however, it is preferable to rotate the axes to align the data with the first few principal components without projecting away the remaining dimensions, since that will be done later in step 5. The additional information in those dimensions is useful in step 4 for reducing tension in the graduated optimization step. The Appendix gives pseudocode for aligning axes with the first few principal components without projecting away the additional dimensions.

C. Step 4: Transform the Data

The data are iteratively transformed as shown in Fig. 5. This transformation continues until at least $log_{\sigma}(0.01)$ iterations are performed, and the sum error has not improved over a window of 50 iterations. The first criterion ensures that most (99%) of the variance is scaled out of the dimensions that will be dropped in the projection. The second criterion allows the algorithm to operate as long as it continues to improve the results. These criteria can be modified to suit the desired quality of results—if precision is important, a larger window may be used; if speed is important, early stopping may be appropriate.

1) Step 4a—Reduce the Variance in Nontarget Dimensions by Scaling: All the values in $\mathbf{P}$, except those in the first t dimensions, are scaled down by multiplying by a constant factor σ, where σ is slightly less than 1. This value controls the rate at which the optimization problem is graduated. A conservative value, such as $\sigma = 0.999$, will ensure that the error surface is slowly transformed, so that the global optimum can be followed with precision. A more liberal value, such as $\sigma = 0.99$, can be used to quickly obtain results, with some risk of falling into a local optimum. Except where otherwise indicated, we use the value $\sigma = 0.99$ for all of the experiments in this paper.

To compensate for this downscaling in the nonpreserved dimensions, the values in the first t dimensions are scaled up to keep the average neighbor distance equal to δ_{ave}. As the algorithm iterates, this will shift the variance out of the nonpreserved dimensions and into the preserved dimensions. Thus, when the projection is performed in step 5, very little information will be lost.

2) Step 4b—Restore Original Relationships: Next, the values in the first t dimensions in P are adjusted to recover the relationships that are distorted by scaling in the previous step.

```
function adjust_point($\mathbf{p}, \eta$)
  $s \leftarrow 0$
  loop:
      $\epsilon_0 \leftarrow$ compute_error($\mathbf{p}$)
      $j \leftarrow 0$
      for $i$ from $0$ to $t-1$:
          $p_i \leftarrow p_i + \eta$
          if compute_error($\mathbf{p}$) $< \epsilon_0$:
              $j \leftarrow j + 1$
          else
              $p_i \leftarrow p_i - 2\eta$
              if compute_error($\mathbf{p}$) $< \epsilon_0$:
                  $j \leftarrow j + 1$
              else
                  $p_i \leftarrow p_i + \eta$
      if $j = 0$:
          return $s$
      $s \leftarrow s + 1$
```

Fig. 6. Pseudocode for the adjust_point function, where compute_error is (1). This is a convex hill-climbing algorithm that moves a point to a locally optimal position and returns the number of steps required to get there.

This is the optimization phase of graduated optimization. A heuristic error value is used to measure the extent to which the current relationships between neighboring points differ from the original relationships

$$\epsilon_{p_i} = \sum_{j=0}^{k} w_{ij} \left(\left(\frac{\delta_{ij_0} - \delta_{ij}}{2\delta_{ave}} \right)^2 + \left(\frac{\max(0, \theta_{ij_0} - \theta_{ij})}{\pi} \right)^2 \right) \tag{1}$$

where δ_{ij} is the current distance to point n_{ij}, δ_{ij_0} is the original distance to point n_{ij} measured in step 2, θ_{ij} is the current angle, and θ_{ij_0} is the original angle measured in step 2. The denominators are normalizing factors that give each term approximately equal weight. When $\epsilon_{p_i} = 0$, the relationship metrics have been restored to their original values. We adjust the values in the first t dimensions of each point to minimize this error value. Since the equation for the true gradient of the error surface defined by this heuristic is complex and is $O(d^3)$ to compute, we use the simple hill-climbing technique of adjusting in each dimension in the direction that yields improvement until a local optimum is found. This technique is sufficient to follow the trough of the changing error surface. The pseudocode for our hill-climbing technique is given in Fig. 6.

Three performance optimizations can be used in this step to significantly speed convergence:

First, it is observed that the component of distances and angles in the nonpreserved dimensions does not change, except that it is scaled by σ in each iteration. These values can be cached (as long as the cached values are scaled by σ in each iteration) such that only the first t dimensions must be evaluated to compute the error heuristic. Since t tends to be a small constant value, this can have a significant impact on runtime performance.

Second, the step size η can be dynamically adjusted to keep the number of total steps low. We adjust η after each iteration to keep the total number of steps taken approximately equal to the total number of data points. If the number of steps is less than the number of data points, then $\eta \leftarrow \eta * 0.9$; otherwise, $\eta \leftarrow \eta * 1.1$. Experimentally, this technique was found to converge significantly faster than using a constant or decaying value for η.

Third, it is observed that the points that have already been adjusted in the current iteration have a more positive influence for guiding the movement of other points to reduce overall error than the points that have not yet been adjusted. Thus, we begin step 4b by starting with a randomly selected point, and we visit each point using a breadth-first traversal ordering. Intuitively, this may be analogous to how a person smoothes a crumpled piece of paper by starting at an arbitrary point and smoothing outward. Thus, a higher weight is given to the component of the error contributed by neighbors that have already been adjusted in the current iteration such that $w_{ij} = 1$ if point n_{ij} has not yet been adjusted in this iteration and $w_{ij} = 10$ if point n_{ij} has been adjusted. Intuitively, a large weight difference will promote faster unfolding, while a smaller weight difference should promote more stability. In our experiments, nearly identical results were obtained using values as low as $w_{ij} = 5$ or as high as $w_{ij} = 20$ for the case where a neighbor has already been adjusted, so we made no attempt to further tune w_{ij} in any of our experiments.

D. Step 5: Project the Data

At this point, nearly all of the variance is contained in the first t dimensions of P. The data are projected by simply dropping all dimensions $> t$ from the representation of the points.

E. Graduated Optimization

The optimization technique used by manifold sculpting warrants particular consideration. The algorithm relies on a simple hill climber to adjust the location of each point (in step 4b). It cycles through the dimensions, and for each dimension, it tries increasing and decreasing the value. It accepts whichever yields improvement (or leaves the point unchanged if neither yields improvement). By itself, this is an unsophisticated optimization technique that is highly susceptible to falling into local optima. The problem over which manifold sculpting ultimately seeks to optimize, however, is not convex. Thus, an additional component is necessary to ensure that manifold sculpting obtains good results.

The key observation of manifold sculpting is that after the relationships between neighboring points has been computed but before any transformation begins, the error value will be zero. No set of values for the data points can produce a lower error, so the system begins in a stable state. The hill climber need not seek a global optimum from a random starting point. Rather, it needs to only remain in a stable state while the variance is iteratively scaled out of the higher dimensions. Thus, by gradually transforming the problem from these initial conditions, a simple hill-climbing algorithm can be sufficient to follow the optimum.

Gradient-based optimization may be comparable to rolling a ball down a hill and hoping that it finds its way to a good local optimum. The optimization technique used by manifold sculpting, on the other hand, is more analogous to a ball following at the feet of a person walking across a trampoline. It begins adjacent to the person's feet and seeks to remain by them as the person moves slowly across the trampoline. As long as the ball never deviates very far from the person's feet, the topology of the rest of the trampoline is irrelevant to the final destination of the ball because the trampoline will always be locally convex around the person's feet.

To understand why this approach tends to be effective, let us first consider the hypothetical case where the manifold represented by the original data set is topologically close to the optimally transformed data. In other words, suppose the optimally transformed data with all variance in the first t dimensions can be obtained through a continuous transformation. Furthermore, suppose that there exists such a continuous transformation that would also preserve the relationships in local neighborhoods at all times during the transformation. It follows that there will be a globally optimal embedding (with an error of zero) at any time during the transformation. Furthermore, because the transformation is topologically continuous, that embedding will follow a continuous path through the error space, beginning at the point that represents the original data and ending with the optimally transformed data. At any given time, this embedding, which is known to be optimal with respect to the error surface at that time, will be found at the exact bottom of a locally convex basin. Thus, if the continuous transform is approximated with small enough values for σ (the scaling rate) and η (the step size), then the system can be certain to finally arrive at the globally optimal solution. This is not a proof that manifold sculpting will always yield optimal results, however, because there is no guarantee that there exists such a continuous transformation. On the other hand, it is not a pathological situation either. The Swiss Roll manifold, for example, can be trivially shown to meet all of these conditions.

In cases where neighborhood relationships must be temporarily violated to "unfold" a manifold, it is often still reasonable to expect that the lowest valley in the error surface will follow a continuous path through the error space, even if that valley does not remain at zero. Noncontinuous jumps in the error space correspond to "tearing" of a manifold structure to instantly separate previously adjacent parts. Intuitively, this is not typically a desirable behavior. Even in cases where the best transformation requires such jumps in the error space, it is still likely that it will yet arrive at a good local optimum. It should do no worse than techniques that simply use convex optimization. Furthermore, necessary rips in the manifold structure are not likely to be frequent events. If the system becomes temporarily separated from the global optimum, it may yet be able to find the optimum again as the error surface continues to change. The global optimum is the only optimum that is certain to remain a local optimum during the entire transformation.

F. Parameter Tuning

Although this algorithm has several parameters that could, in theory, be tuned to obtain better results with a particular problem, we have not found that it is typically necessary to do so. In the experiments reported in this paper, we have only adjusted k (the number of neighbors), t (the number of target dimensions), and σ (the scaling rate). The parameters k and t are common in all of the algorithms with which we compare. The scaling rate, however, is unique to manifold sculpting. For most problems, rapid convergence can be obtained using the value $\sigma = 0.99$, but for complex manifold topologies, a slower scaling rate, such as $\sigma = 0.999$, may be necessary to give the manifold more iterations to unfold.

G. Estimating Intrinsic Dimensionality

The error heuristic used by manifold sculpting is an indicator of how well the neighborhood structure has been preserved. In cases where the intrinsic dimensionality of a problem is not known *a priori*, the variance may be scaled out of the higher dimensions one at a time. The error heuristic will then indicate how well the manifold structure is preserved into each number of dimensions. In contrast with eigenvalues, this error heuristic will indicate the component of nonlinear variance in the manifold. When the error begins to rapidly climb, the intrinsic dimensionality of the manifold has been subceeded. This feature is particularly convenient for analysis in which the intrinsic dimensionality must be determined. When the intrinsic dimensionality is known, however, as is the case with the experiments in this paper, it is preferable for efficiency to simultaneously scale all of the extra dimensions.

IV. Empirical Analysis

We tested the properties of manifold sculpting with a wide variety of experiments and a range of manifolds. Section IV-A reports empirical measurements of accuracy using toy problems that have known ideal results. These results indicate that manifold sculpting is more accurate than Isomap, LLE, HLLE, and L-MVU with these problems. Section IV-B demonstrates the capabilities of manifold sculpting using several image-based manifolds. Section IV-C reports on an experiment with document-based manifolds. Section IV-D discusses using partial supervision and training manifold sculpting in a pseudoincremental manner.

A. Accuracy

Fig. 2 shows that manifold sculpting visually appears to produce results of higher quality than LLE and Isomap with the Swiss Roll manifold, a common visual test for manifold learning algorithms. A quantitative analysis shows that it also yields more precise results than HLLE and L-MVU. Since the actual structure of this manifold is known prior to using any manifold learner, we can use this prior information to quantitatively measure the accuracy of each algorithm.

We define a Swiss Roll in 3-D space with n points (x_i, y_i, z_i), where $0 \leq i < n$, as follows: Let $t = 8i/n + 2$, $x_i = t\sin(t)$, y_i be a random number $-6 \leq y_i < 6$, and $z_i = t\cos(t)$. In 2-D manifold coordinates, the corresponding target points are (u_i, v_i), such that $u_i = (\sinh^{-1}(t) + t\sqrt{t^2+1})/2$ and $v_i = y_i$. To emphasize the effect of poorly sampled areas, we also

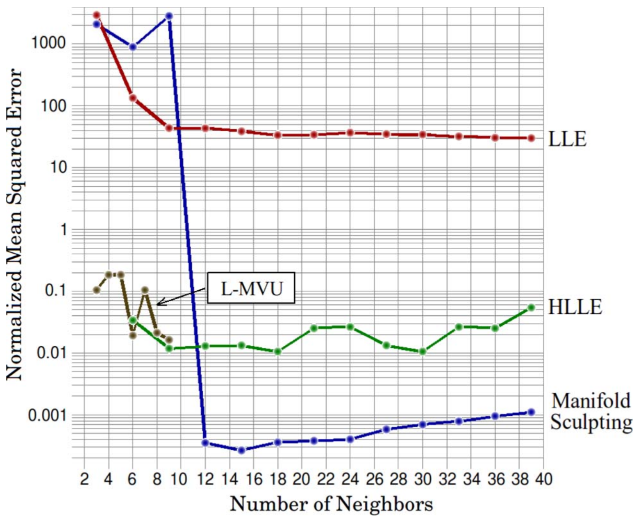

Fig. 7. Mean squared error of four algorithms for a Swiss Roll manifold using a varying number of neighbors k. The vertical axis is shown on a logarithmic scale. The excessive memory requirement of L-MVU and HLLE limited the range of neighbors over which we were able to test these algorithms, but L-MVU did very well with few neighbors. Manifold sculpting yielded the most accurate results when at least 12 neighbors were used.

removed samples that fell within a star-shaped region on the manifold, as shown in Fig. 2.

We created a Swiss Roll with 2000 data points and reduced the dimensionality to two with each of four algorithms. We tested how well the output of each algorithm is aligned with the target output values (u_i, v_i) by measuring the mean squared distance from each point to its corresponding target value. Since there is no guarantee how the results would be oriented, we used the affine transformation that most closely aligned the results with the expected results before measuring the mean squared distance. We then normalized the mean squared error by dividing by λ, where λ is the square of the average distance between each point in the target data set and its nearest neighbor. Thus, a normalized mean squared error larger than 1 would probably indicate that the results are significantly distorted, since the average deviation of a point from its ideal location is more than the distance between neighboring points.

Fig. 7 shows the normalized mean squared distance between each transformed point and its expected value. Results are shown with a varying number of neighbors k. Both axes are shown on a logarithmic scale. With L-MVU, 44 landmarks $(\approx \sqrt{n})$ were used. The resource requirements of L-MVU became unreasonable after nine neighbors, as they did for HLLE after 48 neighbors. LLE and manifold sculpting could easily handle many more neighbors, but neighbors begin to cut across manifold boundaries at that point. Isomap yielded very poor results and is not shown. Manifold sculpting did not yield good results until at least 12 neighbors were used. This may indicate that L-MVU is a better choice when so few samples are available that many neighbors would be unreasonable. With this problem, L-MVU, HLLE, and manifold sculpting can all produce results that are very close to the ideal results. Proportionally, however, the results from manifold sculpting are precise by more than an order of magnitude over the next-best algorithm.

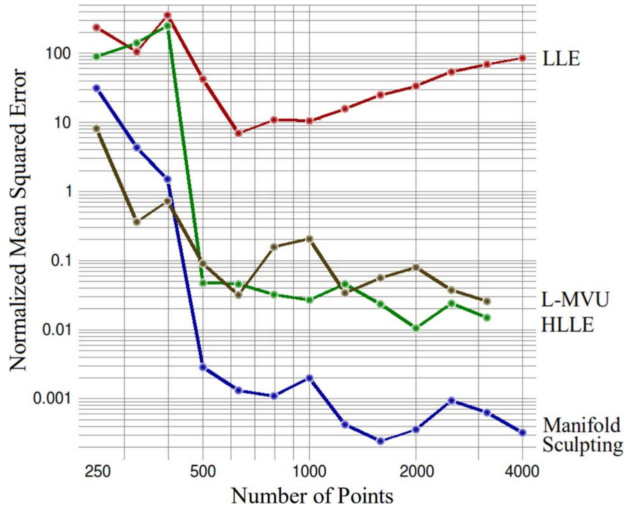

Fig. 8. Mean squared error of four algorithms with a Swiss Roll manifold using a varying number of points. Both axes are shown on a logarithmic scale.

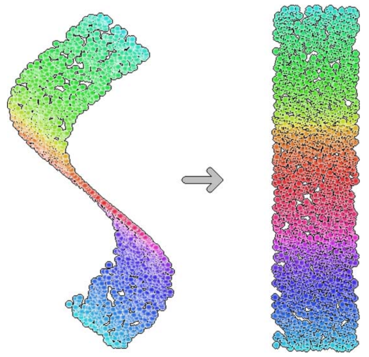

Fig. 9. Visualization of an S-curve manifold.

We repeated the experiment with the Swiss Roll manifold using a varying number of points to sample the manifold. Fig. 8 shows the results of this experiment. Eighteen neighbors were used for LLE, HLLE, and manifold sculpting because these algorithms all did well with this value in the previous experiment. For L-MVU, six neighbors were used to keep the resource requirements manageable and because it did well with this value in the previous experiment. We used the square root of the number of points (rounded down) for the number of landmarks. The memory requirements for HLLE and L-MVU became unreasonable after 3175 points. HLLE and L-MVU yielded very good results with this problem. When the manifold was sampled with at least 500 points, however, manifold sculpting produced proportionally more accurate results than the other algorithms. L-MVU yielded the best results when the manifold was sampled with fewer points, but none of the algorithms yielded very good results with fewer than 500 sample points. HLLE, L-MVU, and manifold sculpting all appear to exhibit the trend of producing better accuracy as the sample density is increased.

To verify that these results were not peculiar to the Swiss Roll manifold, we repeated this experiment using an S-curve manifold as depicted in Fig. 9. This manifold was selected because we could also compute the ideal results for it by integrating to find the distance over its surface. We defined the S-curve

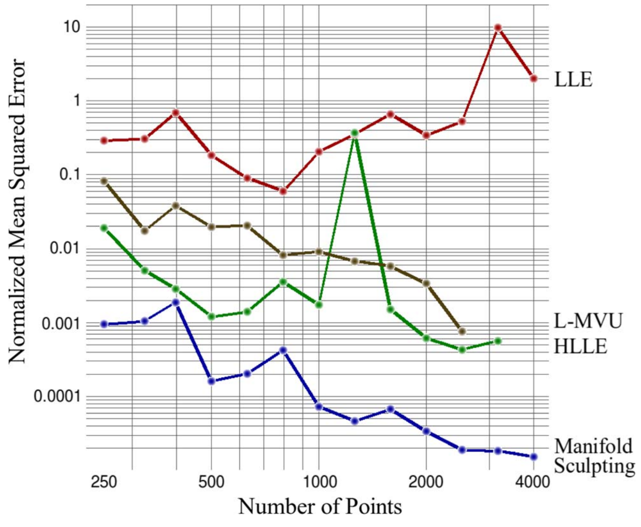

Fig. 10. Mean squared error of four algorithms for an S-curve manifold using a varying number of points. Manifold sculpting consistently outperformed all other algorithms on this problem and is more than an order of magnitude better than HLLE, which is the closest competitor. Both axes are shown on a logarithmic scale.

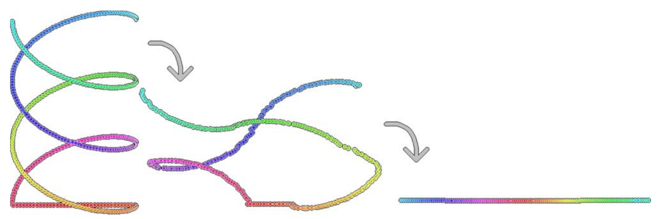

Fig. 11. Visualization of an entwined spirals manifold.

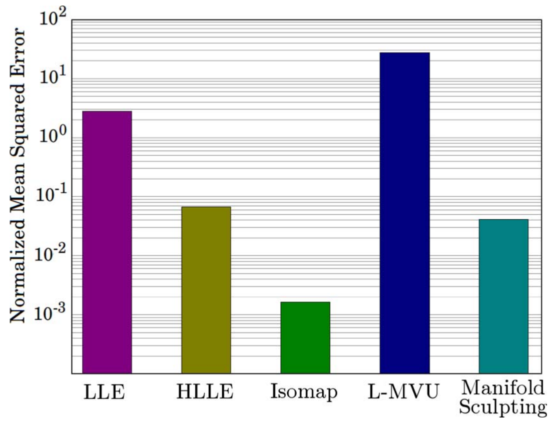

Fig. 12. Mean squared error for five algorithms with an entwined spirals manifold. Isomap does very well when the intrinsic dimensionality is exactly 1.

points in 3-D space with n points (x_i, y_i, z_i), where $0 \leq i < n$, as follows: Let $t = (2.2i - 0.1)\pi/n$, $x_i = t$, $y_i = \sin(t)$, and z_i be a random number $0 \leq z_i < 2$. In 2-D manifold coordinates, the corresponding target points are (u_i, v_i), such that $u_i = \int_0^t (\sqrt{\cos^2(w) + 1})dw$ and $v_i = z_i$. We measured accuracy in the same manner described in the previous two experiments. These results are shown in Fig. 10. Again, results are not shown for L-MVU or HLLE with very large numbers of points due to the demanding resource requirements of these algorithms. Consistent with the previous experiment, L-MVU, HLLE, and

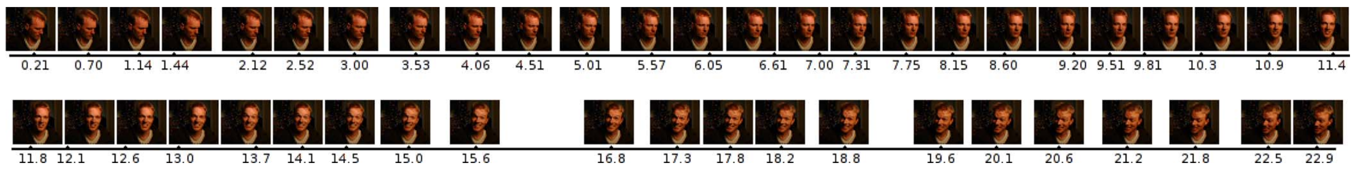

Fig. 13. Images of a face reduced by manifold sculpting into a single dimension. The values are shown here on two wrapped lines to fit the page. The original image is shown above each point.

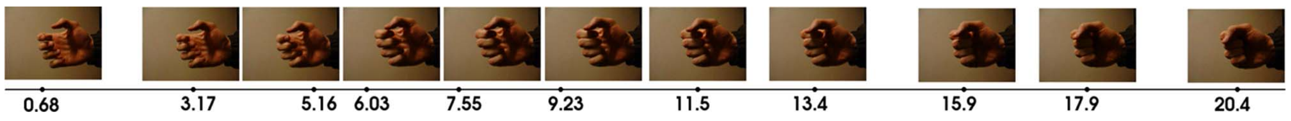

Fig. 14. Images of a hand reduced to a single dimension. The original image is shown above each point.

manifold sculpting all produced very good results with this simple manifold. The trends exhibited in these results were similar to those from the Swiss Roll manifold, with manifold sculpting producing results that were proportionally better.

A test was also performed with an entwined spirals manifold, as shown in Fig. 11. In this case, Isomap produced the most accurate results, even though it consistently had the poorest results for all manifolds with an intrinsic dimensionality greater than 1. (See Fig. 12.) This can be attributed to the nature of the Isomap algorithm. In cases where the manifold has an intrinsic dimensionality of exactly 1, a path from neighbor to neighbor provides an accurate estimate of isolinear distance. Thus, an algorithm that seeks to globally optimize isolinear distances will be less susceptible to the noise from cutting across local corners. When the intrinsic dimensionality is higher than 1, however, paths that follow from neighbor to neighbor produce a zig-zag pattern that introduces excessive noise into the isolinear distance measurement. In these cases, preserving local neighborhood relationships with precision yields better overall results than globally optimizing an error-prone metric. Consistent with this intuition, Isomap yielded very accurate results in our other experiments, which are reported hereafter, that involved a manifold with a single intrinsic dimension and yielded the poorest results with experiments in which the intrinsic dimensionality was larger than one.

B. Image-Based Manifolds

Many unsupervised learning problems do not have a corresponding set of ideal results. The Swiss Roll and S-curve manifolds are useful for quantitative analysis because expected results can be computed *a priori* but real-world applications are likely to involve many more than just three dimensions. We therefore performed several experiments to demonstrate that manifold sculpting is also accurate with problems that involve much larger initial dimensionality. Fig. 13 shows several frames from a video sequence of a person turning his head while gradually smiling. Each image was encoded as a vector of 1634 pixel intensity values. No single pixel contained enough information to characterize a frame according to the high-level concept of facial position, but this concept was effectively encoded in multidimensional space. These data were then reduced to a single dimension. (Results are shown on two separate lines to fit the page.) The one preserved dimension could then characterize each frame according to the high-level concept that was previously encoded in many dimensions. The dot below each image corresponds to the single-dimensional value in the preserved dimension for that image. In this case, the ordering of every frame was consistent with the ordering in the video sequence. Because the ideal results with this problem are not known, it is not possible to compute the accuracy of these results. We therefore did not compare with other algorithms using this problem, but the correctness of these results is somewhat visually apparent.

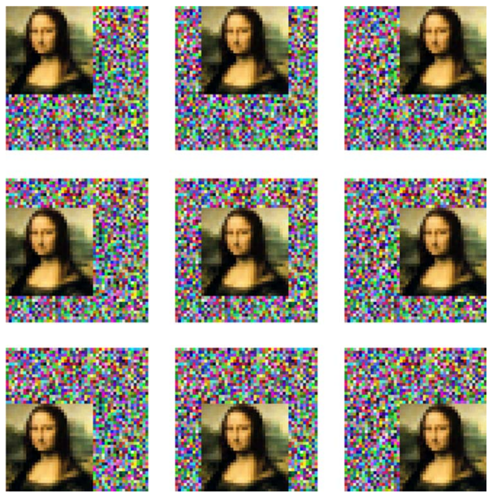

Fig. 15. Data set was generated by translating an image over a background of noise. Nine representative images are shown. Results from several algorithms using this data set are shown in Fig. 16.

Fig. 14 shows 11 images of a hand. These images were encoded as multidimensional vectors in the same manner. The high-level concept of “hand openness” was preserved into a single dimension. In addition to showing that manifold sculpting can work with real-world data, this experiment also shows the robustness of the algorithm to poorly sampled manifolds because these 11 images were the only images used for this experiment. Again, the ideal results with this problem are not known, so no accuracy measurement is given.

Another experiment involves a manifold that was generated by translating a picture over a background of random noise, as shown in Fig. 15. This figure shows a sample of nine images. The manifold was sampled with 625 images, each encoded as

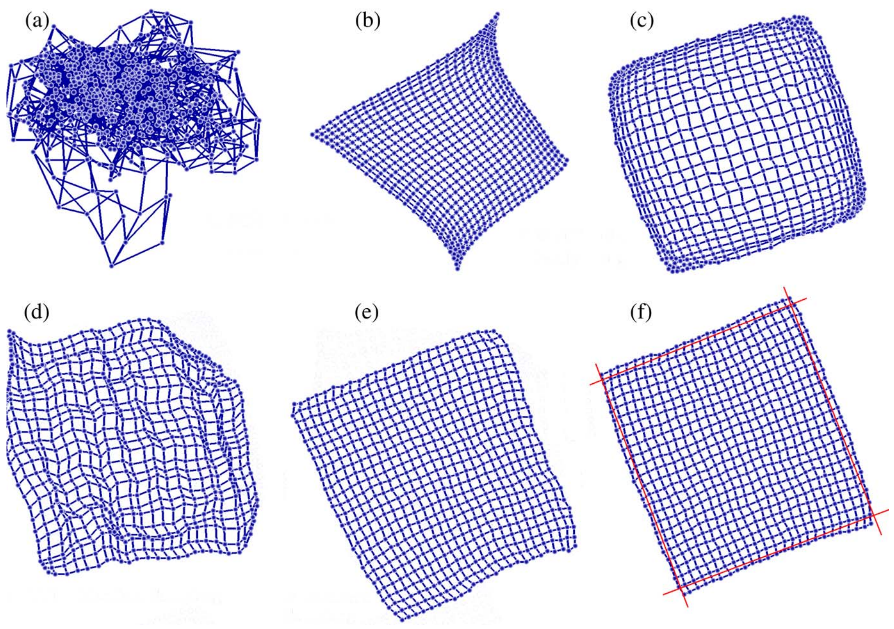

Fig. 16. Comparison of results with a manifold generated by translating an image over a background of noise. Eight neighbors were used with HLLE, L-MVU, and manifold sculpting. Four neighbors were used with LLE because it produced better results than the case with eight neighbors. Forty-nine landmarks were used with L-MVU. Results for manifold sculpting are shown with LLE preprocessing using the default scaling rate $\sigma = 0.99$, and with only PCA preprocessing using a slower scaling rate of $\sigma = 0.999$. The results from manifold sculpting are nearly linearly separable. (a) PCA. (b) LLE. (c) HLLE. (d) L-MVU. (e) LLE + manifold sculpting. (f) Manifold sculpting.

a vector of 2304 pixel intensity values. Because two variables (horizontal position and vertical position) were used to generate the data set, the data can be interpreted as sampling from a manifold with an intrinsic dimensionality of two in a space with an extrinsic dimensionality of 2304. Because the background is random, the average distance between neighboring points in the input space should be somewhat uniform; therefore, the ideal reduced-dimensionality result can be expected to occupy a space very close to square in shape. We therefore use this as a basis for empirically measuring the quality of results. Quantitative measurements with this problem may be a better indicator of the strengths of a manifold learning algorithm than the Swiss Roll or S-curve manifolds because: 1) this problem involves reduction from high-dimensional space; and 2) to our knowledge, highly accurate results have not yet been obtained with this problem.

Fig. 16 shows a comparison of results from various algorithms on this problem. For increased visibility of the inherent structure, each vertex is shown to be connected with the four nearest neighbors in the input space. Results are shown with PCA to demonstrate how very nonlinear this manifold is [see Fig. 16(a)]. We tested the other algorithms with four and eight neighbors (because the points lie on a grid-like structure) and report the best results for each algorithm. LLE did better with four neighbors. The other algorithms did better with eight neighbors.

To demonstrate the ability of manifold sculpting to benefit from the results of other dimensionality reduction algorithms, we substituted LLE for the preprocessing (step 3 of the manifold sculpting algorithm). These results, shown in Fig. 16(e), were rapidly obtained using the default scaling rate of $\sigma = 0.99$. The best results, however, were obtained using a slower scaling rate ($\sigma = 0.999$). These results are shown in Fig. 16(f).

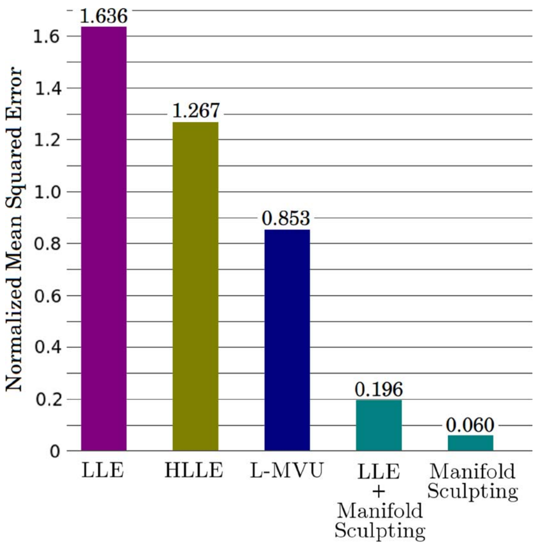

Fig. 17. Empirical measurement of the error of the results shown in Fig. 16. Error was computed as the smallest mean squared error from points evenly distributed over a square. An affine transformation was found to align results as closely as possible with the square before error was measured. Results from manifold sculpting are more than an order of magnitude better than the next closest competitor.

In this case, we used the default PCA preprocessing. With this problem, it is necessary to use a slower scaling rate when PCA is used for preprocessing so that the manifold has sufficient time to unfold.

We empirically measured the quality of the results obtained in this experiment by comparing results with points evenly distributed over a perfect square. These results are shown in Fig. 17. LLE and HLLE do poorly because their results tend to exhibit global distortions, which have a more significant impact

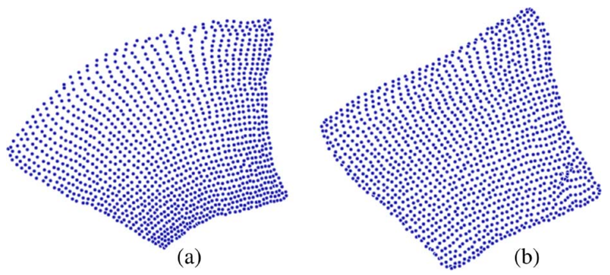

Fig. 18. (a) Manifold sculpting was used to reduce the dimensionality of a manifold generated from a collection of 1296 images, each represented as a vector of 3675 continuous pixel values. These images were generated by sliding a window over a larger picture of the Mona Lisa. This result is not square because the sliding window creates a bigger change in input distance when sliding over regions with a bigger gradient. (b) A custom distance-metric was used to normalize distance such that each neighborhood represents a uniform amount of total distance. This result better represents the intrinsic variables of this problem.

on the mean squared error. L-MVU achieves a normalized mean squared error less than 1. It exhibits a mixture of both global and local distortions. The mean squared error of the result from manifold sculpting is more than an order of magnitude smaller than that of L-MVU.

Another observation that can be made from the results shown in Fig. 16(f) is that the results are approximately linearly separable. For example, if it were desirable to classify these images into two classes such that class A contained all images in which the picture of the Mona Lisa is adjacent to the top of the image and class B contained all other images, this could be done using the results from manifold sculpting with a single linear division boundary. Since this data set was designed to have predictable results, these classes would only have pathological applications, but this demonstrates the significant potential of manifold sculpting to create separability of intrinsic concepts from otherwise complex data.

With many problems, however, distances in observation space are not uniformly scaled in relation to the intrinsic variables. For example, Fig. 18(a) shows the results of using manifold sculpting to reduce the dimensionality of a manifold generated by sliding a window over a larger picture of the Mona Lisa. This result is not square because some parts of the image exhibit different gradients than other parts. When the window slides over a region with a larger gradient, a bigger distance is measured. To obtain results that more correctly represent the intrinsic variables in this problem, we used a custom distance metric that normalized distances in local neighborhoods, such that each neighborhood represents a uniform amount of total distance. This result is shown in Fig. 18(b). These values are a better representation of the intrinsic values in this problem because they are less biased by the irrelevant gradient in the images. This experiment used 1296 images, each represented as a vector of 3675 continuous pixel values.

In addition to supporting custom relationship metrics, manifold sculpting is also flexible regarding the representation of intrinsic points. For example, if a subset of supervised points is available, these points can be clamped with their supervised values while manifold sculpting operates, and manifold sculpting will naturally find values for the other points that fit well in relation to the supervised points. This capability is useful for at least three potential applications. First, it can be used to improve runtime performance. With several problems, we were able to measure as much as a 50% speedup when as few as 5% of the points had supervised values. Since the supervised points are clamped with their known values, they tend to act as a force that pulls the other points directly toward their final destinations. Second, it can be used to align intrinsic values with the dimensional axes or to give the distances between them meaning with respect to a particular unit. This might be useful, for example, if the intrinsic values are used to estimate the state of some system. Third, it can be used to facilitate pseudoincremental manifold learning. If, for example, points arrive from a stream, pseudoincremental manifold learning may be useful to update the reduced-dimensional estimate for all of the points that have yet arrived. This is done in two steps. First, manifold sculpting is applied with values clamped to all the points that have known values. This rapidly computes values for the new incoming points. Second, manifold sculpting is applied again with all points unclamped but starting in the location of their reduced-dimensional values. This enables all of the points to be updated in consequence of the new information but also incurs very little computational cost since all points already begin in nearly optimal locations.

C. Document Manifolds

The utility of manifold learning algorithms for image processing applications has recently been recognized, but this is certainly not the only field that deals with multidimensional data. The vector space model [28], for example, is commonly used in the field of information retrieval to characterize web documents. Each web page is represented as a large vector of term weights. The number of dimensions in the vector corresponds to the number of unique word stems (about 57 000 in English), and the values in these dimensions correspond to a term weight computed from the number of occurrences of the term in a document. Search queries can be evaluated by finding the documents whose vector has the closest angle with the vector of the query. This representation bears some striking resemblances to the pixel representation used for processing images, so it seems likely that similar results could be obtained by applying manifold learning to this field.

To test this hypothesis, we implemented a simple application for refining the results obtained from a *Google* search. A typical search often yields many thousands of documents, but users rarely have patience to look at more than the first few. Our application downloads the first 100 documents, removes stop words, stems each term with the Porter stemming algorithm [29], and represents the documents with the vector space model. Next, it uses manifold sculpting to reduce the dimensionality of the vectors to a single dimension. It then divides this dimension into two halves at the point that minimizes the sum variance of the two halves. Finally, it extracts three terms to represent each of the two clusters by summing the vectors in the cluster and picking the three terms with the highest total weight. In theory, these two groups of terms should reflect the most significant range of context found in the query results, and a user should be able to refine his or her query by selecting which of the two halves is closer to the intended meaning of the query.

As an example, a query for the term "speaker" yielded for one set of terms "box," "amp," and "off," and for the other set, it yielded "hit," "baseball," and "bat." The first interpretation of the term "speaker" probably comes from references to audio devices. Prior to performing this experiment, we did not know that this term had anything to do with baseball, but a search for the refined query "speaker baseball" yields many documents with information about Tris Speaker who was elected to the baseball hall of fame in 1937. Not all queries yielded such distinctly separated results, but this is an area with potential for further research.

D. Semi-Supervision

Manifold learning and clustering have many things in common. Clustering collections of multidimensional vectors such as images, for example, is more effective when manifolds in the data are taken into account [30]. Manifold learning and clustering are both unsupervised operations. Unlike clustering, however, which groups vectors into a discrete number of buckets, manifold learning arranges them into a discrete number of continuous spectra. In some sense, clustering is to manifold learning what classification is to regression. It seems intuitive, therefore, that techniques that benefit clustering algorithms may have a corresponding counterpart for manifold learning algorithms. Clustering algorithms can be greatly enhanced with partial supervision [31]. Likewise, a small modification to the manifold sculpting algorithm makes it possible to perform semisupervised manifold learning.

Semisupervised clustering involves a subset of data points for which classification values or hints about those values are provided. Semisupervised manifold learning likewise requires final values or estimated final values to be provided for a subset of the data points. During scaling iterations (step 4 of the manifold sculpting algorithm), these points are clamped to their supervised values. The unsupervised points are free to move but will be influenced by their neighbors, some of which may be supervised. This results in more efficient and potentially more accurate manifold learning. Fig. 19 shows the amount of time required to learn the intrinsically 1-D manifold for each of the four video sequences with a varying percentage of supervised points. It can be observed in these results that the first 20% to 50% of supervised points tend to produce noticeable improvements in speed, but additional supervised points tend to make little difference.

In some cases, data may not be available all at once. In such cases, it may be desirable to learn a manifold as the data become available from a stream [32]. Pseudoincremental learning is naturally achieved with semisupervised manifold learning. The following two-step process is followed when new data points become available: First, the old points are clamped to their known reduced-dimensionality values, and the new points are allowed to settle. Second, all points are unclamped, and the entire data set is allowed to settle. The first step is very fast because most of the points are supervised. The second step is also fast because the manifold is already unfolded, and all of the points are likely to be very close to their final states. Significant computation is only required when new data cause the overall structure of the manifold to change.

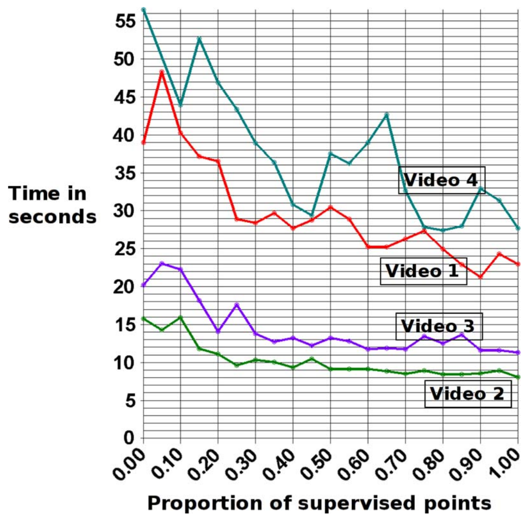

Fig. 19. Manifold learning is faster when more points are supervised. (When most of the points are supervised, the only significant cost is the neighbor-finding step of the algorithm.)

Often, semisupervised clustering is performed by having human experts classify a subset of the available data. When real values are required, however, it may only be reasonable to ask human experts to provide vague estimates. For example, humans might be able to easily sort a collection of images according to relevance to a particular topic. The sorted images could be assigned sequential output values, and these values could be used as estimates for the points in manifold coordinates. However, precise values may be difficult to obtain initially. Fortunately, even estimated output values can be useful in semisupervised manifold learning. This benefit is exploited by first clamping estimated values to guide the unfolding of the manifold and then unclamping all of the points during later iterations so that more precise refinements can be made to the data.

V. Conclusion

A significant contribution of this paper is the observation that the optimization step of manifold learning is suitable to be solved using graduated optimization, a technique that can rapidly find the global optimum with many otherwise-difficult optimization problems. We have demonstrated this by presenting a manifold learning algorithm called manifold sculpting, which uses graduated optimization to find a manifold embedded in high-dimensional space.

The collection of experiments reported in this paper indicate that that manifold sculpting yields more accurate results than other well-known manifold learning algorithms for most problems. We have collected empirical measurements using a Swiss Roll manifold with varying sample densities and with varying numbers of neighbors. We have also tested with an S-curve manifold, an entwined spirals manifold, a manifold generated by translating an image over a background of noise, and a few other manifolds. Isomap tends to be very strong when the

```
function align_axes_with_principal_components($\mathbf{P}$)
  $\mu \leftarrow$ compute_mean($\mathbf{P}$)
  for each $\mathbf{p}_{i,*} \in \mathbf{P}$:
      $\mathbf{p}_{i,*} \leftarrow \mathbf{p}_{i,*} - \mu$
  $\mathbf{Q} \leftarrow$ copy_of($\mathbf{P}$)
  $\mathbf{G} \leftarrow \{\hat{i}, \hat{j}, \hat{k}, \ldots\}$ such that $|\mathbf{G}| = t$
  for $k$ from $0$ to $t-1$
      $\mathbf{c} \leftarrow$ a random vector of size $t$
      do 20 times:
          $\mathbf{v} \leftarrow$ zero vector of size $t$
          for each $\mathbf{q}_i \in \mathbf{Q}$:
              $\mathbf{v} \leftarrow \mathbf{v} + (\mathbf{q}_i \cdot \mathbf{c})\mathbf{q}_i$
          $\mathbf{c} \leftarrow \frac{\mathbf{v}}{|\mathbf{v}|}$
      for each $\mathbf{q}_i \in \mathbf{Q}$:
          $\mathbf{q}_i \leftarrow \mathbf{q}_i - (\mathbf{c} \cdot \mathbf{q}_i)\mathbf{c}$
      $\mathbf{a} \leftarrow \mathbf{G}_k$
      $\mathbf{b} \leftarrow \frac{\mathbf{c} - (\mathbf{a} \cdot \mathbf{c})\mathbf{a}}{|\mathbf{c} - (\mathbf{a} \cdot \mathbf{c})\mathbf{a}|}$
      $\phi \leftarrow \arctan(\frac{\mathbf{b} \cdot \mathbf{c}}{\mathbf{a} \cdot \mathbf{c}})$
      for $j$ from $k$ to $t-1$
          $u \leftarrow \mathbf{a} \cdot \mathbf{G}_j$
          $v \leftarrow \mathbf{b} \cdot \mathbf{G}_j$
          $\mathbf{G}_j \leftarrow \mathbf{G}_j - u\mathbf{a}$
          $\mathbf{G}_j \leftarrow \mathbf{G}_j - v\mathbf{b}$
          $r \leftarrow \sqrt{u^2 + v^2}$
          $\theta \leftarrow \arctan(\frac{v}{u})$
          $u \leftarrow r\cos(\theta + \phi)$
          $v \leftarrow r\sin(\theta + \phi)$
          $\mathbf{G}_j \leftarrow \mathbf{G}_j + u\mathbf{a}$
          $\mathbf{G}_j \leftarrow \mathbf{G}_j + v\mathbf{b}$
  for each $\mathbf{p}_{i,*} \in \mathbf{P}$:
      for $j$ from $0$ to $t-1$:
          $p_{ij} \leftarrow \mathbf{p}_{i,*} \cdot \mathbf{G}_j + \mu_j$
```

Fig. 20. Pseudocode for the align_axes_with_principal_components function. This performs the same function as the axis rotation step of PCA, except this algorithm only aligns with the first $|D_{\text{preserved}}|$ principal components while preserving data in all dimensions.

intrinsic dimensionality of a problem is exactly 1, and L-MVU does well when very few sample points are available, but in all other cases, manifold sculpting yielded the most accurate results and is typically at least an order of magnitude more accurate than the closest competitor algorithm.

The results produced by manifold sculpting are robust to parameter choices, except for the scaling rate. We have shown that the default scaling rate ($\sigma = 0.99$) works well with most problems but that a slower scaling rate will yield good results with more complex problems. We have also demonstrated that manifold sculpting can benefit by preprocessing the data with other dimensionality reduction algorithms. In many cases, this enables accurate results to be rapidly obtained, without resorting to the use of a slower scaling rate.

Appendix
PCA for Graduated Optimization

This appendix provides pseudocode for the align_axes_with_principal_components function. This performs the same function as the axis rotation step of PCA, except that it preserves data in all dimensions while only aligning with the first t principal components. This differs from regular PCA, which aligns data with the first few principal components and then throws out all values in the remaining dimensions. With manifold sculpting, it may be preferable to preserve these values because they represent some component of the distances between points. Thus, this algorithm may be used instead of regular PCA for preprocessing the data prior to applying manifold sculpting with the slight advantage that this technique guarantees not to affect any of the distances in local neighborhoods. This pseudocode is given in Fig. 20.

References

[1] R. Bellman, *Adaptive Control Processes: A Guided Tour*. Princeton, NJ: Princeton Univ. Press, 1961.

[2] T. C. E. Cheng and M. Y. Kovalyov, "An unconstrained optimization problem is NP-hard given an oracle representation of its objective function: A technical note," *Comput. Oper. Res.*, vol. 29, no. 14, pp. 2087–2091, Dec. 2002.

[3] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, Dec. 2000.

[4] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, Dec. 2000.

[5] K. Q. Weinberger, F. Sha, and L. K. Saul, "Learning a kernel matrix for nonlinear dimensionality reduction," in *Proc. 21st ICML*, 2004, pp. 839–846.

[6] P. Burt, "Fast filter transforms for image processing," *Comput. Vis. Graph. Image Process.*, vol. 16, pp. 20–51, 1981.

[7] Z. Wu, "The effective energy transformation scheme as a special continuation approach to global optimization with application to molecular conformation," *SIAM J. Optim.*, vol. 6, no. 3, pp. 748–768, 1996.

[8] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *J. Educ. Psychol.*, vol. 24, no. 6, pp. 417–441, Sep. 1933.

[9] R. N. Shepard and J. D. Carroll, "Parametric representation of nonlinear data structures," in *Proc. Int. Symp. Multivariate Anal.*, vol. 2. New York: Academic, 1965, pp. 561–592.

[10] J. W. Sammon, "A nonlinear mapping for data structure analysis," *IEEE Trans. Comput.*, vol. C-18, no. 5, pp. 401–409, May 1969.

[11] P. Demartines and J. Hérault. (1997, Jan.). Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets. *IEEE Trans. Neural Netw.* [Online]. *8(1)*, pp. 148–154. Available: citeseer.ist.psu.edu/demartines96curvilinear.html

[12] J. Lee, A. Lendasse, N. Donckers, and M. Verleysen, "A robust non-linear projection method," in *Proc. ESANN*, Bruges, Belgium, 2000, pp. 13–20.

[13] V. de Silva and J. B. Tenenbaum, "Global versus local methods in nonlinear dimensionality reduction," in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, pp. 705–712.

[14] B. Schölkopf, A. J. Smola, and K.-R. Müller, "Kernel principal component analysis," in *Advances in Kernel Methods: Support Vector Learning*. Cambridge, MA: MIT Press, 1999, pp. 327–352.

[15] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in Neural Information Processing Systems, 14*. Cambridge, MA: MIT Press, 2001, pp. 585–591.

[16] M. Brand, "Charting a manifold," in *Advances in Neural Information Processing Systems, 15*, S. T. S. Becker and K. Obermayer, Eds. Cambridge, MA: MIT Press, 2003, pp. 961–968.

[17] P. Vincent and Y. Bengio, "Manifold Parzen windows," in *Advances in Neural Information Processing Systems 15*. Cambridge, MA: MIT Press, 2003, pp. 825–832.

[18] D. Donoho and C. Grimes, "Hessian eigenmaps: Locally linear embedding techniques for high dimensional data," *Proc. Nat. Acad. Sci.*, vol. 100, no. 10, pp. 5591–5596, May 2003.

[19] Y. Bengio and M. Monperrus, "Non-local manifold tangent learning," in *Advances in Neural Information Processing Systems 17*, L. K. Saul, Y. Weiss, and L. Bottou, Eds. Cambridge, MA: MIT Press, 2005, pp. 129–136.

[20] E. Levina and P. J. Bickel, "Maximum likelihood estimation of intrinsic dimension," in *Advances in Neural Information Processing Systems 17*, L. K. Saul, Y. Weiss, and L. Bottou, Eds. Cambridge, MA: MIT Press, 2005, pp. 777–784.

[21] Z. Zhang and H. Zha, "A domain decomposition method for fast manifold learning," in *Advances in Neural Information Processing Systems 18*, Y. Weiss, B. Schölkopf, and J. Platt, Eds. Cambridge, MA: MIT Press, 2006.

[22] K. Q. Weinberger, B. D. Packer, and L. K. Saul, "Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization," in *Proc. 10th Int. Workshop Artif. Intell. Statist.*, 2005, pp. 381–388.

[23] D. Huang, Z. Yi, and X. Pu, "Manifold-based learning and synthesis," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 3, pp. 592–606, Jun. 2009.

[24] X. Geng, D.-C. Zhan, and Z.-H. Zhou, "Supervised nonlinear dimensionality reduction for visualization and classification," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 35, no. 6, pp. 1098–1107, Dec. 2005.

[25] C. Chen, J. Zhang, and R. Fleischer, "Distance approximating dimension reduction of Riemannian manifolds," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 40, no. 1, pp. 208–217, Feb. 2010.

[26] D. Meng, Y. Leung, T. Fung, and Z. Xu, "Nonlinear dimensionality reduction of data lying on the multicluster manifold," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 4, pp. 1111–1122, Aug. 2008.

[27] M. Gashler, D. Ventura, and T. Martinez, "Iterative non-linear dimensionality reduction with manifold sculpting," in *Advances in Neural Information Processing Systems 20*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds. Cambridge, MA: MIT Press, 2008.

[28] V. V. Raghavan and S. K. M. Wong, "A critical analysis of vector space model for information retrieval," *J. Amer. Soc. Inf. Sci.*, vol. 37, no. 5, pp. 279–287, Sep. 1986.

[29] M. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[30] M. Breitenbach and G. Z. Grudic, "Clustering through ranking on manifolds," in *Proc. 22nd ICML*, 2005, pp. 73–80.

[31] A. Demiriz, K. Bennett, and M. Embrechts, "Semi-supervised clustering using genetic algorithms," in *Proc. Artif. Neural Netw. Eng.*, 1999, pp. 809–814.

[32] M. H. C. Law, N. Zhang, and A. K. Jain, "Nonlinear manifold learning for data stream," Dept. Comput. Sci. Eng., Michigan State Univ., East Lansing, MI, Tech. Rep. MSU-CSE-03-5. [Online]. Available: citeseer.ist.psu.edu/691968.html

Michael Gashler, photograph and biography not available at the time of publication.

Dan Ventura, photograph and biography not available at the time of publication.

Tony Martinez, photograph and biography not available at the time of publication.