Learning a Manifold-Constrained Map between Image Sets: Applications to Matching and Pose Estimation

Jihun Ham  
*University of Pennsylvania, jhham@seas.upenn.edu*

Ikkjin Ahn  
*University of Pennsylvania, ikkjin@seas.upenn.edu*

Daniel D. Lee  
*University of Pennsylvania, ddlee@seas.upenn.edu*

Follow this and additional works at: [http://repository.upenn.edu/grasp_papers](http://repository.upenn.edu/grasp_papers)  
Part of the [Engineering Commons](http://repository.upenn.edu/grasp_papers)

Recommended Citation  

Suggested Citation:  

©2006 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

This paper is posted at ScholarlyCommons. [http://repository.upenn.edu/grasp_papers/58](http://repository.upenn.edu/grasp_papers/58)  
For more information, please contact libraryrepository@pobox.upenn.edu.
Abstract
This paper proposes a method for matching two sets of images given a small number of training examples by exploiting the underlying structure of the image manifolds. A nonlinear map from one manifold to another is constructed by combining linear maps locally defined on the tangent spaces of the manifolds. This construction imposes strong constraints on the choice of the maps, and makes possible good generalization of correspondences between all of the image sets. This map is flexible enough to approximate an arbitrary diffeomorphism between manifolds and can serve many purposes for applications. The underlying algorithm is a non-iterative efficient procedure whose complexity mainly depends on the number of matched training examples and the dimensionality of the manifold, and not on the number of samples nor on the dimensionality of the images. Several experiments were performed to demonstrate the potential of our method in image analysis and pose estimation. The first example demonstrates how images from a rotating camera can be mapped to the underlying pose manifold. Second, computer generated images from articulating toy figures are matched using the underlying 4 dimensional manifold to generate image-driven animations. Finally, two sets of actual lip images during speech are matched by their appearance manifold. In all these cases, our algorithm is able to obtain reasonable matches between thousands of large-dimensional images, with a minimum of computation.

Disciplines
Engineering

Comments
Suggested Citation:

©2006 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

This conference paper is available at ScholarlyCommons: http://repository.upenn.edu/grasp_papers/58
Learning a manifold-constrained map between image sets: applications to matching and pose estimation

Jihun Ham, Ikkjin Ahn, and Daniel Lee
GRASP Laboratory, University of Pennsylvania
{jhham,ikkjin,ddlee}@seas.upenn.edu

Abstract

This paper proposes a method for matching two sets of images given a small number of training examples by exploiting the underlying structure of the image manifolds. A nonlinear map from one manifold to another is constructed by combining linear maps locally defined on the tangent spaces of the manifolds. This construction imposes strong constraints on the choice of the maps, and makes possible good generalization of correspondences between all of the image sets. This map is flexible enough to approximate an arbitrary diffeomorphism between manifolds and can serve many purposes for applications. The underlying algorithm is a non-iterative efficient procedure whose complexity mainly depends on the number of matched training examples and the dimensionality of the manifold, and not on the number of samples nor on the dimensionality of the images. Several experiments were performed to demonstrate the potential of our method in image analysis and pose estimation. The first example demonstrates how images from a rotating camera can be mapped to the underlying pose manifold. Second, computer generated images from articulating toy figures are matched using the underlying 4 dimensional manifold to generate image-driven animations. Finally, two sets of actual lip images during speech are matched by their appearance manifold. In all these cases, our algorithm is able to obtain reasonable matches between thousands of large-dimensional images, with a minimum of computation.

1. Introduction

Objects under continuously varying illumination or pose, give rise to a geometrical structure in camera images known as image manifolds [2, 13, 16]. From preliminary analytic work on the subject [8], the concept of image manifolds has recently been used for computer vision applications such as pose estimation problems [9, 19, 22] and facial expressions analysis [4, 17]. Although these results may not yet out-perform conventional model-based approaches, they point to a new direction in understanding images by learning the structure of images from their appearance alone.

The current interest and potential for rapid progress in this area can be attributed to several different factors. First, thousands of images of an object or a scene are easily collected by camera rigs, arrays of cameras, or by an autonomous mobile robot. Second, a series of nonlinear dimensionality reduction techniques have been developed recently, including Kernel PCA [15], Isomap [20], LLE [14], and LLP [10], which reduce the dimensionality of the image data, and output a low-dimensional representation that preserves certain geometrical properties of the original data. These algorithms are unsupervised, that is, no prior knowledge is used to guide the process of dimensionality reduction. Consequently, the resulting representations do not directly reflect parameters of interest such as pose parameters or joint angles. To relate these features to the desired parameters, a separate learning problem needs to be addressed. Typically, the dimensionality reduction algorithms are used as a preprocessing front-end, yielding low-dimensional features for subsequent algorithms to build upon.

Another problem with current nonlinear dimensionality reduction algorithms is the difficulty of simultaneously learning the inverse map from low-dimensional parameters to the high-dimensional image space. Some of the algorithms mentioned above have adopted generic radial basis functions or nonlinear regression methods to fit the inverse map. However, due to the nonlinearity, high dimensionality, and lack of training samples, the inverse mapping problem remains an obstacle for image-based approaches.

In this paper we propose a semi-supervised learning of a direct map between two images manifolds, constrained by the fact that the map should be smooth on the manifolds. This map is learned from a large number of unlabelled images and a small fraction of images that are labelled with known correspondences. The algorithm simultaneously learns local low dimensional representations along with the associated mapping between the manifolds. We illustrate how this method can be used to efficiently map
between one image dataset to another image dataset, and between an image data to a pose parameter space.

In the first application, we consider the following example as depicted in Figure 1. A set of images of a subject under varying poses is collected, and another set of images of a different subject is similarly obtained. Among these unordered sets, we assume that we are given a few paired examples which contain images of the two subjects with the same pose. We would like to generalize the rule exemplified by the training set, so that we can find matches to the rest of the images. Our manifold mapping method directly solves this matching problem by considering the correspondence problem as an explicit map between the underlying image manifolds.

The second application is similar to the first, in that the pose parameter space, such as free rotations in 3D, is simply another manifold that needs to be mapped to. Consider the situation in Figure 2, where images of an object are obtained by taking pictures from camera positions in a sphere around the object. If the camera poses of a small subset of these images are known, we would like to estimate the poses of the rest of the images. Our method does this by constructing an explicit map between the image manifold and the pose parameter manifold.

The remainder of the paper is organized as follows. Section 2 provides a background to smooth manifolds, and formally explains the geometric ideas of the proposed learning method. Section 3 presents algorithms for computing the map from image data. In Section 4, results of experiments on applications to matching articulated figure images, estimating rotating camera poses, and on lip images are described. Finally, we conclude with a discussion in Section 5.

2. Learning a manifold-constrained map

In this section we introduce a semi-supervised method of approximating a smooth map between manifolds. The difficulty of learning such a map is mostly due to the high dimensionality of the embedding space. However, by working in the low-dimensional tangent spaces of the manifolds, we can constrain the class of possible maps allowing for effective generalization. We begin by introducing necessary concepts in smooth manifold theory, and then show how to explicitly construct such a map from data.

2.1. Notation

Let \( X \subset \mathbb{R}^{d_x} \) and \( Y \subset \mathbb{R}^{d_y} \) be the two sets of images. For each data set we have \( n_x \) and \( n_y \) samples \( \{x_1, x_2, \ldots, x_{n_x}\} \) and \( \{y_1, y_2, \ldots, y_{n_y}\} \). Additionally, \( n \) pairs of matched data are given: \( \{(u_k, v_k)\}, \ k = 1, \ldots, n \) where \( n \ll n_x, n \ll n_y \). We assume the data \( X \) and \( Y \) are samples of Euclidean submanifolds \( M \subset \mathbb{R}^{d_x} \) and \( N \subset \mathbb{R}^{d_y} \) of dimension \( d \) with \( d \ll d_x, d \ll d_y \), and also that \( M \) and \( N \) are diffeomorphic.

2.2. Mathematical preliminaries

We provide mathematical preliminaries of smooth manifolds to make this paper self-contained. The following definitions can be found in differential geometry textbooks such as [11, 21].

**Smooth manifold** Let \( M \) and \( N \) be two topological manifolds of the same dimension \( d \). Suppose \( \{(U_i, \phi_i)\}_i \) is an atlas for \( M \) and \( \{(V_j, \psi_j)\}_j \) is an atlas for \( N \), where \( U_i \) is an open subset of \( M \) and \( \phi_i \) is the coordinate map \( \phi_i : U_i \rightarrow \mathbb{R}^d \), and similarly \( V_j \) is an open subset of \( N \) and \( \psi_j \) is the coordinate map \( \psi_j : V_j \rightarrow \mathbb{R}^d \). The topological manifolds \( M \) and \( N \) are also smooth manifolds, if the atlas of each has a smooth structure, that is, \( \phi_i \circ \phi_k^{-1} \) is smooth when \( U_i \cap U_k \neq \emptyset \), and similarly for \( \psi_j \circ \psi_l^{-1} \).

**Smooth map** A map \( f : M \rightarrow N \) is smooth, if for each \( p \in M \), there is a chart \( (U_i, \phi_i) \) containing \( p \) and \( (V_j, \psi_j) \) containing \( f(p) \), and the composition map...
$g : \phi_i(U_i) \to \psi_j(V_j)$ defined by $g = \psi_j \circ f \circ \phi_i^{-1}$ is smooth in the ordinary sense.

**Tangent space** For an embedded submanifold $\mathcal{M}$ of Euclidean spaces, a tangent space $T_p\mathcal{M}$ of $\mathcal{M}$ centered at $p$, is identifiable with the $d$-dimensional subspace $\mathbb{R}^d$, whose origin is at $p$ and is in normal direction at $p$. Furthermore, for each point $p$, there exists a chart $(U_i, \phi_i)$, such that $\phi_i(U_i)$ is an open subset of $T_p\mathcal{M}$, and $\phi_i$ is the projection $\phi_i = \pi : U_i \to T_p\mathcal{M}$. By the Whitney embedding theorem, an abstract manifold can be smoothly embedded in a finite Euclidean space, and the geometric characterization of tangent spaces applies to abstract manifolds as well.

**Smooth partition of unity** By paracompactness of manifolds, there exist a smooth partition of unity $\{\alpha_i : \mathcal{M} \to \mathbb{R}\}$ subordinate to the given atlas:

1. $\alpha_i(p) \geq 0$, $\forall p \in \mathcal{M}$, $i \in \mathcal{I}$
2. $\text{supp} \alpha_i \subset U_i$, $\forall i \in \mathcal{I}$
3. $\sum_i \alpha_i(p) = 1$, $\forall p \in \mathcal{M}$.

In the last sum, there are only finitely many $\alpha_i$'s that are nonzero at each point $p$.

### 2.3. Locally linear maps on tangent spaces

Given training data $X$, $Y$, and the $n$ matched pairs, the goal of the algorithm is to learn the mapping between the manifolds from data.

#### 2.3.1 Manifold learning

The manifold $\mathcal{M}$ modeling the data $X$ is fully characterized by the atlas $\{(U_i, \phi_i)i \in \mathcal{I}\}$. In general, the index set $\mathcal{I}$ need not be finite or countable. However, we will only consider charts centered at the training samples $\{u_i\}$, and assume they are regularly sampled on $\mathcal{M}$. To define the chart $U_i$ around $u_i$, we use a partition of unity $\{\alpha_i(x)\}$, which serves as a membership ‘weight’ assigned to a point $x$ with respect to $u_i$. We say $x$ belongs to the chart $U_i$ if $\alpha_i(x) > 0$. The choice for the function $\alpha_i$ will be detailed in Section 3.

From the discussion of tangent spaces in the previous section, we regard each tangent space $T_{u_i}\mathcal{M}$ as an affine space centered at $u_i$. For each of the spaces, we can define an orthonormal frame from the Riemannian metric induced from the Euclidean space. For a small enough $U_i$, the projection from $U_i$ to $T_{u_i}$ is a diffeomorphism by the inverse function theorem, and therefore we can define the coordinate function $\phi_i$ as the projection itself. Notice that we need not have an explicit form of $\phi_i$ and can still compute the projection:

$$\phi_i(x) = S_i(x - u_i),$$

where $S_i$ is the matrix whose columns form an orthonormal basis of the tangent space $T_{u_i}\mathcal{M}$. A related discussion for finding the orthonormal frame is in [7].

The manifold $\mathcal{N}$ modeling the second data $Y$, is characterized in an analogous way. If $\{(V_j, \psi_j)\}$ is the atlas for $\mathcal{N}$, then $V_j$ is defined around the training point $v_j$ by another partition of unity $\{\beta_j(y)\}$. The map $\psi$ is approximated by

$$\psi_j(y) = T_j(y - v_j),$$

where $T_j$ is an orthonormal basis of the tangent space $T_{v_j}\mathcal{N}$.

Numerically, the tangent spaces are computed from the local PCA around $u_i$’s and $v_j$’s. By pooling enough number of points and using their weights, we can find the principal subspaces reliably in the presence of moderate noise. More will be explained in Section 3.

#### 2.3.2 Nonlinear function learning

Next, we describe how the smooth map $f$ between the manifolds can be learned. Note the problem is extremely difficult, since all we know is $f(u_i) = v_i$ for a small number of training data, which does not constrain much the possible choices of $f$. However, we incorporate the knowledge that $X$ and $Y$ have manifold structure as follows.

First consider the Euclidean counterpart $g : \phi_i(U_i) \to \psi(V_j)$ of a smooth function $f$ between manifolds, for some $i$ and $j$. To simplify the situation, we only consider the map from $\phi_i(U_i)$ to $\psi(V_j)$. Given the map

\[ f : \mathcal{M} \to \mathcal{N} \]

\[ L_1 \]

\[ L_2 \]

\[ M \]

\[ N \]

\[ f \]

\[ v_1 \]

\[ z_1 \]

\[ v_2 \]

\[ z_2 \]

\[ u_1 \]

\[ u_2 \]

\[ x \]

\[ x_1 \]

\[ x_2 \]
\[ g_i : \phi(U_i) \to \psi(V_i) \], the corresponding map \( f_i : U_i \to V_i \) is

\[ f_i(x) = \psi_i^{-1} \circ g_i \circ \phi_i(x). \] (3)

A global map \( f \) on \( \mathcal{M} \) is defined by pasting the locally defined maps \( f_i \) together.

The simplest approximation of the map \( g_i \) is a linear transform \( L_i \), represented as a matrix with respect to the orthonormal bases \( S_i \) and \( T_i \).

Composed with the linearization of \( \phi_i \) in the previous section, we have the global map between two manifolds

\[
    f(x) = \sum_i \alpha_i(x) f_i(x) \\
    \approx \sum_i \alpha_i(x) \left( T_i^T L_i \phi_i(x) + v_i \right) \\
    \approx \sum_i \alpha_i(x) \left( T_i^T L_i S_i(x - u_i) + v_i \right).
\]

In short, \( f \) is the weighted sum of \( i \)-th local maps, each of which sends \( x \) to the estimated image \( \hat{z}_i \)

\[
    \hat{z}_i = T_i^T L_i \hat{x}_i + v_i. \] (5)

The local maps \( \{ f_i(x) \} \) are simply linear, which is preferable to complex models from a generalization point of view. However, when they are combined with nonlinear weights \( \alpha(x) \) that conform to the underlying nonlinear manifold structure, they are capable of approximating a smooth nonlinear map. A geometric picture of this description is shown in Figure 3.

### 2.3.3 Optimization

How do we find the linear maps \( \{ L_i \} \) which are most consistent with the given data? Under the proposed scheme, each projection \( \hat{x}_i \) of \( x \) maps to different \( \hat{z}_i \)'s. If it were not for linear approximations of \( \phi_i \) and \( f_i \), the images \( \{ f_i(x) \} \) of a fixed \( x \) should agree exactly regardless of which charts are used to present the map. Otherwise, \( f \) is not a well-defined map. Therefore the natural choice of \( \{ L_i \} \) is that which makes the different estimates \( \{ \hat{z}_i \} \) agree most. We define the penalty \( C_x(L_1, L_2, \cdots, L_n) \) as the matrix norm of weighted covariance of the images \( \hat{z}_i \) via different linear maps \( \{ L_i \} \):

\[
    C_x = \text{tr} \left( \sum_i \alpha_i \hat{z}_i \hat{z}_i^T - \sum_i \alpha_i \hat{z}_i \sum_j \alpha_j \hat{z}_j^T \right) \\
    = \text{tr} \left( \sum_i \alpha_i \left( T_i^T L_i \hat{x}_i + v_i \right) \left( T_i^T L_i \hat{x}_i + v_i \right)^T \right) \\
    - \text{tr} \left( \sum_i \alpha_i \left( T_i^T L_i \hat{x}_i + v_i \right) \sum_j \alpha_j \left( T_j^T L_j \hat{x}_j + v_j \right)^T \right). \] (6)

The total cost is the sum of (6) for all \( x \in X \):

\[
    C(L_1, \cdots, L_n) := \sum_{x \in X} C_x(L_1, \cdots, L_n). \] (7)

It is readily shown that the cost is a convex function of the matrices \( L_1, \cdots, L_n \), and the minimizer is found by setting

\[
    \frac{\partial C}{\partial L_i} = 0, \quad i = 1, 2, \cdots, n, \] (8)

which results in \( n \) linear matrix equations for \( k = 1, \cdots, n \):

\[
    T_k^T T_j L_k S_j A_k S_k^T + T_k B_k S_k^T \\
    - \sum_j T_k^T T_j L_j S_j C_j k S_k^T - T_k D_k S_k^T = 0, \] (9)

where \( A_k, B_k, C_j k \) and \( D_k \) are matrices computed from the data. The solution to (9) is given uniquely by the matrices.

### 3. Algorithm

#### 3.1. Computing weights

We assign weights for a smooth partition using the following function:

\[
    \alpha_i(x) \propto e^{-d(x, u_i)^2/2s^2}, \] (10)

where \( u_i \) is the \( i \)-th training data, and \( d(\cdot, \cdot) \) is the usual distance in \( \mathbb{R}^d \). Roughly speaking, \( x \) has a large value of \( \alpha_i(x) \) if \( x \) is close to \( u_i \), and a vanishing value if \( x \) is far from \( u_i \). However, because of the curse of dimensionality, the standard distance is not a very sensitive measure for far points. We occasionally have had better experimental results using geodesic distances, approximated by shortest-paths on a nearest-neighbor graph as in the Isomap algorithm [20]. For each \( x \) we choose the constant \( s \) adaptively as \( s = \min_{i} d(x, u_i) \). To make \( \alpha_i \) have compact support, we can either 1) truncate small values of the weights, or 2) keep only the \( l \)-largest weights for \( x \). The second scheme guarantees fast sparse computation in subsequent procedures. Finally we normalize the weights so that \( \sum \alpha_i(x) = 1 \) holds for all \( x \). The above definition of \( \alpha \) may not technically be a smooth partition of unity, but makes no difference in numerical computations. Below is a summary of the procedure:

For each \( u_i \) and \( x_j, \quad i = 1, \cdots, n_x, \quad j = 1, \cdots, n \),

1. Compute \( W_{ij} \propto e^{-d(x_j, u_i)^2/2s^2} \)
2. Make \( W \) sparse.
3. Normalize \( W_{ij} \leftarrow W_{ij} / \sum_j W_{ij} \).

The weights for the second data \( Y \) is computed similarly by the exponential decay rule \( \beta_j(y) \propto e^{-d(y, u_j)^2/2s^2} \).
3.2. Local basis computation

The local tangent space at a point \( x \) can be determined by the sample covariance of points in the neighborhood of \( x \). Choosing the wrong size of the neighborhood can affect the resulting computations. Here, we use the weights computed from the previous section to discount the contribution of a point to the covariance around \( x \). Atkeson et al. discusses the advantages of using locally weighted estimates in [1]. The local tangent space is computed as follows:

For each point \( u_i \), \( i = 1, \cdots, n \),

1. Form a matrix \( \bar{x} = [x_{i_1}, x_{i_2}, \cdots, x_{i_m}] \) whose column vectors are \( m \) points in the neighborhood of \( u_i \).
2. Form the diagonal matrix \( A_{jj} = W_{j,i,i} \).
3. Normalize \( A_{jj} \rightarrow A_{jj}/\sum_j A_{jj} \).
4. Find the local basis \( S_i \) by Singular-Value Decomposition:
   \[
   S_i \Sigma D^T = [\bar{x} - u_i]A
   \]

The local bases \( \{T_j\} \) for \( Y \) is similarly computed.

3.3. Computing \( \{L_i\} \)

The matrices \( A_k, B_k, C_{jk} \) and \( D_k \) in (9) are defined as

\[
A_k = \sum_i W_i(x_i - u_k)(x_i - u_k)^T \quad (12)
\]

\[
B_k = \sum_i W_i v_k(x_i - u_k)^T \quad (13)
\]

\[
D_k = \sum_i \left( \sum_j W_i j v_j \right) W_i(x_i - u_k)^T \quad (14)
\]

\[
C_{jk} = \sum_i W_i j W_i(x_i - u_j)(x_i - u_k)^T \quad (15)
\]

Equation (9) is linear in \( \{L_i\} \), hence we can solve this by writing the equation as a tensor product of matrices.

Using the tensor operator \( \otimes \) and vectorization operator \( \text{vec}(\cdot) \), we compute the new matrices

\[
\bar{A}_k = S_k A_k^T S_k^T \otimes T_k T_k^T \quad (16)
\]

\[
\bar{B}_k = \text{vec}(T_k B_k S_k^T) \quad (17)
\]

\[
\bar{D}_k = \text{vec}(T_k D_k S_k^T) \quad (18)
\]

\[
\bar{C}_{jk} = S_k C_{jk}^T S_j^T \otimes T_j T_j^T \quad (19)
\]

\[
m_k = \text{vec}(L_k) \quad (20)
\]

These turn (9) into the vector equation

\[
\bar{A}_k m_k + \bar{B}_k - \sum_j \bar{C}_{jk} m_j - \bar{D}_k = 0, \quad k = 1, \cdots, n. \quad (21)
\]

To solve for \( \{L_k\} \) simultaneously, we construct larger matrices as follows: \( \bar{A} \) is the block-diagonal matrix of size \( (nd^2 \times nd^2) \):

\[
\bar{A} = \begin{bmatrix}
\bar{A}_1 & 0 & 0 \\
0 & \ddots & 0 \\
0 & 0 & \bar{A}_n
\end{bmatrix}. \quad (22)
\]

\( \bar{B} \) and \( \bar{D} \) are the vectors of size \( (nd^2 \times 1) \) and \( (nd^2 \times 1) \), obtained by stacking \( B_k \)’s and \( D_k \)’s along the columns:

\[
\bar{B}^T = \begin{bmatrix}
\bar{B}_1^T & \cdots & \bar{B}_n^T
\end{bmatrix}, \quad (23)
\]

\[
\bar{D}^T = \begin{bmatrix}
\bar{D}_1^T & \cdots & \bar{D}_n^T
\end{bmatrix}. \quad (24)
\]

\( \bar{C} \) is a full matrix of size \( (nd^2 \times nd^2) \):

\[
\bar{C} = \begin{bmatrix}
\bar{C}_{11} & \cdots & \bar{C}_{1n} \\
\vdots & \ddots & \vdots \\
\bar{C}_{1n} & \cdots & \bar{C}_{nn}
\end{bmatrix}. \quad (25)
\]

Let \( \bar{L} \) be a \( (nd^2 \times 1) \) vector of the following form:

\[
\bar{L}^T = [m_1^T m_2^T \cdots m_n^T]. \quad (26)
\]

The final solution of (9) is given as follows:

1. For each \( j = 1, \cdots, n, \quad k = 1, \cdots, n, \)
   
   (a) Compute \( A_k, B_k, D_k \) and \( C_{jk} \) from (12)-(15).
   
   (b) Compute \( \bar{A}_k, \bar{B}_k, \bar{D}_k \), and \( \bar{C}_{jk} \) from (16)-(19).

2. Compute \( \bar{A}, \bar{B}, \bar{D}, \) and \( \bar{C} \) from (22)-(25).

3. Compute \( \bar{L} \) from matrix inversion
   
   \[
   \bar{L} = (\bar{A} - \bar{C})^{-1}(\bar{D} - \bar{B}). \quad (27)
   \]

4. Get each \( L_i \) by rearranging the elements of \( \bar{L} \).

3.4. Remarks

The desired \( \{L_i\} \) which minimize (6) is computed from inversion of a matrix of size \( nd^2 \), which is the main cost of computation. Since the number of training sets is much smaller than the total number of data, \( (n \ll n_x, n \ll n_y) \) and the dimensionality of the manifold is also much smaller than the dimensionality of data \( (d \ll d_x, d \ll d_y) \), the computation is very efficient. Moreover, other computations that scale linearly or quadratically with respect to \( n_x \), are reduced to a small fraction by virtue of sparse weights. In practice, the computation time for each example in Section 4 was at most five minutes with a Pentium 4 desktop running Matlab scripts. Compared to other nonlinear
mapping methods such as the Generative Topographic Mapping [3], this optimization is global and no iterations are required.

The algorithm allows us to directly map a new test point not in the given set $X$. In this case, we only had to re-evaluate the weights $\alpha_i(x)$ and compute its projections $\hat{x}_i$ for the test point $x$. It is not necessary to compute $S_i, T_i,$ or $L_i$ again.

4. Applications

In this section we describe how to use the manifold-constrained map to analyze example image data, and to find matches between the image datasets.

Once we have computed $\{L_i\}$, we define the best matching $y \in Y$ given $x$, as the one whose approximation $\hat{y}$ of itself on $\mathcal{N}$ is closest to the estimated image $\hat{z}$:

$$y_{\text{match}} = \arg \max_{y \in Y} \| \hat{y} - \hat{z}(x) \|$$

$$= \arg \max_{y \in Y} \| \hat{y} - \sum_i \alpha_i(x)\hat{z}_i(x) \|,$$  \hspace{1cm} (28)

where $\hat{y}$ and $\hat{z}$ is computed from

$$\hat{y} = \sum_j \beta_j(y) \left( T_j^T T_j(y - v_j) + v_j \right)$$ \hspace{1cm} (29)

$$\hat{z} = \sum_i \alpha_i(x)\hat{z}_i(x).$$ \hspace{1cm} (30)

Note the best match (28) is defined for each $x$. To get matches for multiples points, we simply repeat (28) point-wise. The search for this closest point is accelerated by the following trick: for a given $x$, limit the candidates of $\hat{y}$ to those points who belong to the same charts $U_1, U_2, \ldots$ as the charts $x$ belongs to. This makes the set of candidates significantly smaller than the whole set $Y$.

We learn manifold-constrained maps from three different types of data: 1) images from a rotating camera, 2) articulating toy figure images, and 3) lip images during speech, and examine the performance of matching.

For these experiments, each image is simply represented as a single vector of gray-level intensity in $\mathbb{R}^{d_x}$ where $d_x$ equals the number of image pixels.

4.1. Estimating camera pose

Scenes of geometric objects in 3D were realistically rendered for a rotating camera. The camera was allowed to orbit around the object with azimuthal angle $(0 - 360^\circ)$ and nonnegative elevation angle $(0 - 90^\circ)$, as depicted in Figure 2. The angles were generated to have approximately $3^\circ$ sampling resolution for both azimuth and elevation. The total number of samples were 2368, and 5 percent $(= 118)$ of randomly chosen samples were used as training samples with known poses. Each image in the set consisted of $40 \times 40$ pixels. The scenes showed occasional self-occlusions and occlusions from limited field of view.

Figure 4 shows tangent space projections $\hat{x}_i$ of images which are in the neighborhood of a particular training point $u_i$. One sees that the projection axes of PCA is highly correlated with the underlying camera viewpoint angles. This provides strong evidence for manifold structure in the images even if we didn’t know how the images were collected. In this case, we have assumed the dimensionality $d$ of manifolds is fixed beforehand. In case $d$ is not known, one can analyze the dropoff of eigenvalues of the singular value decomposition (11) to estimate $d$.

To learn the direct map from images to pose space, we assume the pose space of a hemisphere of dimension 2, embedded in $\mathbb{R}^3$. Tangent spaces on the pose space can be derived analytically or computed from uniform points on the sphere. Using the matched training set, we learn the map $\{L_i\}$ and the estimates $\hat{z}$. The estimated poses are shown in Figure 5. Although the result is not comparable to the state-of-the-art results, we present the naive result as a proof of concept. Note that the poses were determined using only the raw pixel values of the images, without any knowledge of object geometry nor camera properties.

4.2. Generating a matched animation

We also matched images of two articulating LEGO™ figurines available from a public database. Each figure
by considering a dynamic model on manifolds [12]. This could be improved further by considering a dynamic model on manifolds [12].

5. Conclusion

Image-based approaches complement model-based approaches by their ability to automatically extract information from a large collection of images. In this paper, we proposed a new algorithm that differs from other dimensionality reduction techniques currently used in image-based approaches: our method constructs a direct map between two high-dimensional image sets, and the map generalizes well with a relatively small number of training samples. From examples of synthetic and real images, we demonstrated the potential of our method in image analysis and pose estimation using only appearance. We anticipate continued development of this algorithm will demonstrate additional applications of the methods on other types of data.

References

Figure 7. The top row shows two sequences ‘walking’ and ‘hurray’ of figurine 1, used as queries. The middle row shows the best matching images of figurine 2 from the learned map. The estimated poses are reasonably similar to the true poses in the bottom row.

Figure 8. The top row shows a sequence of lip images as a query. The sequence is a part of a continuous utterance of arbitrary vowels, and the ground truth for matching is not known. The middle row shows the corresponding weighted average $\tilde{z} = \sum_i \alpha_i(x)\hat{z}_i(x)$ from the learned map. The best matching images to the averaged images, are shown in the bottom row.


