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 outperform 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

Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)
0-7695-2597-0/06 $20.00 © 2006 IEEE

Figure 1. Two image sets of articulating toy figures that have been generated independently. The top and bottom rows show selected pairs from each image dataset which are related to each other by the underlying body pose of the arms and legs. From this training set, we would like to learn the correspondence between all of the image datasets. Details are described in Section 4.

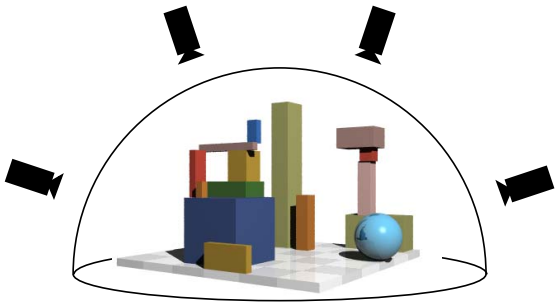

Figure 2. Multiple images of an object are obtained by a rotating camera. The underlying manifold structure of the camera pose space can be used to estimate the poses of the camera images.

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 problems at once, 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 $\{x_1, x_2, \cdots, x_{n_x}\}$ and $\{y_1, y_2, \cdots, y_{n_y}\}$. Additionally, n pairs of matched data are given: $\{(u_i, v_i)\},\ k = 1, \cdots, n$ where $n \ll n_x,\ n \ll n_y$. We assume the data X and Y are samples of Euclidean submanifolds $\mathcal{M} \subset \mathbb{R}^{d_x}$ and $\mathcal{N} \subset \mathbb{R}^{d_y}$ of dimension d with $d \ll d_x,\ d \ll d_y$, and also that $\mathcal{M}$ and $\mathcal{N}$ 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 $\mathcal{M}$ and $\mathcal{N}$ be two topological manifolds of the same dimension d. Suppose $\{(U_i, \phi_i) | i \in \mathcal{I}\}$ is an atlas for $\mathcal{M}$ and $\{(V_j, \psi_j) | j \in \mathcal{J}\}$ is an atlas for $\mathcal{N}$, where U_i is an open subset of $\mathcal{M}$ and ϕ_i is the coordinate map $\phi_i : U_i \to \mathbb{R}^d$, and similarly V_j is an open subset of $\mathcal{N}$ and ψ_j is the coordinate map $\psi_j : V_j \to \mathbb{R}^d$. The topological manifolds $\mathcal{M}$ and $\mathcal{N}$ are also smooth manifolds, if the atlas of each has a smooth structure, that is, $\phi_i \circ \phi_k^{-1}$ is smooth when $U_i \cup U_k \neq \emptyset$, and similarly for $\psi_j \circ \psi_l^{-1}$.

Smooth map A map $f : \mathcal{M} \to \mathcal{N}$ is smooth, if for each $p \in \mathcal{M}$, there is a chart (U_i, ϕ_i) containing p and (V_j, ψ_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, ϕ_i), such that $\phi_i(U_i)$ is an open subset of $T_p\mathcal{M}$, and ϕ_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. 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 α_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 manifold structure and 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 α_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 ϕ_i as the projection itself. Notice that we need not have an explicit form of ϕ_i and can still compute the projection:

$$\phi_i(x) = S_i(x - u_i), \tag{1}$$

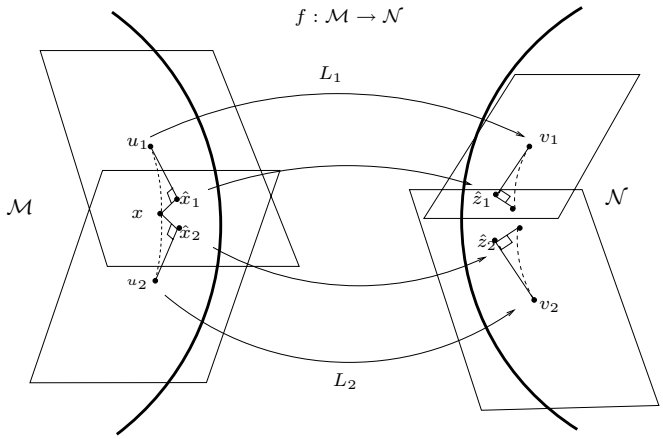

Figure 3. $\hat{x}_i := S_i(x - u_i)$ is the projection of x into i-th tangent space. On the tangent space, the smooth map $f : \mathcal{M} \to \mathcal{N}$ is approximated by a linear map $L_i := f_* : T_{u_i}\mathcal{M} \to T_{v_i}\mathcal{N}$, where $\{(u_i, v_i = f(u_i))\}$ are the given training points. $\hat{z}_i$ is the image of $\hat{x}_i$ under the local map: $\hat{z}_i = T_i^T L_i \hat{x}_i + v_i$. These local maps are glued together by the partition of unity α_i as follows : $f(x) \approx \sum_i \alpha_i(x) \left(T_i^T L_i \hat{x}_i + v_i\right)$. The objective of the optimization is to choose $\{L_i\}$ which minimizes the disagreement of images $\{\hat{z}_i\}$ of the same point x, for all $x \in X$.

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 ψ is approximated by

$$\psi_j(y) = T_j(y - v_j), \tag{2}$$

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

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

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

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 ϕ_i in the previous section, we have the global map between two manifolds

$$\begin{aligned} f(x) &= \sum_i \alpha_i(x) f_i(x) \qquad (4)\\ &\approx \sum_i \alpha_i(x) \left(T_i^T L_i \phi_i(x) + v_i\right) \\ &\approx \sum_i \alpha_i(x) \left(T_i^T L_i S_i (x - u_i) + v_i\right). \end{aligned}$$

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

$$\hat{z}_i = T_i^T L_i \hat{x}_i + v_i. \tag{5}$$

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 ϕ_i andf_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 $\mathcal{C}_x(L_1, L_2, \cdots, L_n)$ for x as the matrix norm of weighted covariance of the images $\hat{z}_i$ via different linear maps $\{L_i\}$:

$$\begin{aligned} \mathcal{C}_x &= \text{tr}\left(\sum_i \alpha_i \hat{z}_i \hat{z}_i^T - \sum_i \alpha_i \hat{z}_i \sum_j \alpha_j \hat{z}_j^T\right) \qquad (6)\\ &= \text{tr}\left(\sum_i \alpha_i \{T_i^T L_i \hat{x}_i + v_i\}\{T_i^T L_i \hat{x}_i + v_i\}^T\right) \\ &- \text{tr}\left(\sum_i \alpha_i \{T_i^T L_i \hat{x}_i + v_i\} \sum_j \alpha_j \{T_j^T L_j \hat{x}_j + v_j\}^T\right). \end{aligned}$$

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

$$\mathcal{C}(L_1, \cdots, L_n) := \sum_{x \in X} \mathcal{C}_x(L_1, \cdots, L_n). \tag{7}$$

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

$$\frac{\partial \mathcal{C}}{\partial L_i} = 0, \quad \text{for } i = 1, 2, \cdots, n, \tag{8}$$

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

$$\begin{aligned} &T_k T_k^T L_k S_k A_k S_k^T + T_k B_k S_k^T \\ &\quad - \sum_j T_k T_j^T L_j S_j C_{jk} S_k^T - T_k D_k S_k^T = 0, \end{aligned} \tag{9}$$

where A_k, B_k, C_{jk} 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:

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

where u_i is the i-th training data, and $d(\cdot, \cdot)$ is the usual distance in $\mathbb{R}^{d_x}$. 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 be 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 α_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 $\sum_i \alpha_i(x) = 1$ holds for all x. The above definition of α 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, $i = 1, \cdots, n_x$, $j = 1, \cdots, n$,

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 $\beta_j(y) \propto e^{-d(y,v_j)^2/2s^2}$.

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* $A_{jj} = W_{i_j,i}$.
3. *Normalize* $A_{jj} \leftarrow A_{jj} / \sum_j A_{jj}$
4. *Find the local basis* S_i *by Singular-Value Decomposition:*

$$S_i \Sigma D^T = [\bar{x} - u_i] A \tag{11}$$

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

$$A_k = \sum_i W_{ik}(x_i - u_k)(x_i - u_k)^T \tag{12}$$

$$B_k = \sum_i W_{ik} v_k (x_i - u_k)^T \tag{13}$$

$$D_k = \sum_i (\sum_j W_{ij} v_j) W_{ik} (x_i - u_k)^T \tag{14}$$

$$C_{jk} = \sum_i W_{ij} W_{ik} (W_i - u_j)(x_i - u_k)^T. \tag{15}$$

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 $\mathrm{vec}(\cdot)$, we compute the new matrices

$$\overline{A}_k = S_k A_k^T S_k^T \otimes T_k T_k^T \tag{16}$$

$$\overline{B}_k = \mathrm{vec}(T_k B_k S_k^T) \tag{17}$$

$$\overline{D}_k = \mathrm{vec}(T_k D_k S_k^T) \tag{18}$$

$$\overline{C}_{jk} = S_k C_{jk}^T S_j^T \otimes T_k T_j^T \tag{19}$$

$$m_k = \mathrm{vec}(L_k). \tag{20}$$

These turn (9) into the vector equation

$$\overline{A}_k m_k + \overline{B}_k - \sum_j \overline{C}_{jk} m_j - \overline{D}_k = 0, \quad k = 1, \cdots, n. \tag{21}$$

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

$$\overline{A} = \begin{bmatrix} \overline{A}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \overline{A}_n \end{bmatrix}. \tag{22}$$

$\overline{B}$ and $\overline{D}$ 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:

$$\overline{B}^T = \left[\overline{B}_1^T \cdots \overline{B}_n^T\right], \tag{23}$$

$$\overline{D}^T = \left[\overline{D}_1^T \cdots \overline{D}_n^T\right]. \tag{24}$$

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

$$\overline{C} = \begin{bmatrix} \overline{C}_{11} & \cdots & \overline{C}_{n1} \\ \vdots & \ddots & \vdots \\ \overline{C}_{1n} & \cdots & \overline{C}_{nn} \end{bmatrix}. \tag{25}$$

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

$$\overline{L}^T = [m_1^T m_2^T \cdots m_n^T]. \tag{26}$$

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

1. *For each* $j = 1, \cdots, n$, $k = 1, \cdots, n$,
   (a) *Compute* A_k, B_k, D_k *and* C_{jk} *from (12)-(15)*
   (b) *Compute* $\overline{A}_k, \overline{B}_k, \overline{D}_k$, *and* $\overline{C}_{jk}$ *from (16)-(19)*
2. *Compute* $\overline{A}, \overline{B}, \overline{D}$, *and* $\overline{C}$ *from (22)-(25)*
3. *Compute* $\overline{L}$ *from matrix inversion*

$$\overline{L} = (\overline{A} - \overline{C})^{-1} (\overline{D} - \overline{B}). \tag{27}$$

4. *Get each* L_i *by rearranging the elements of* $\overline{L}$.

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 $\hat{z}$:

$$\begin{aligned} y_{\text{match}} &= \text{argmax}_{y \in Y} \, \|\hat{y} - \hat{z}(x)\| \\ &= \text{argmax}_{y \in Y} \, \|\hat{y} - \sum_i \alpha_i(x)\hat{z}_i(x)\|, \quad (28) \end{aligned}$$

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

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

$$\hat{z} = \sum_i \alpha_i(x)\hat{z}_i(x). \quad (30)$$

Note the best match (28) is defined for each x. To get matches for multiples points, we simply repeat (28) pointwise. The search for this closest point is accelerated by the following trick: for a given x, limit the candidates of $\tilde{y}$ to those points who belong to the same charts $U_1, U_2, \cdots$ 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 $(0 - 360°)$ and nonnegative elevation angle $(0 - 90°)$, as depicted in Figure 2. The angles were generated to have approximately $3°$ 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 40×40 pixels. The scenes showed occasional self-occlusions and occlusions from limited field of view.

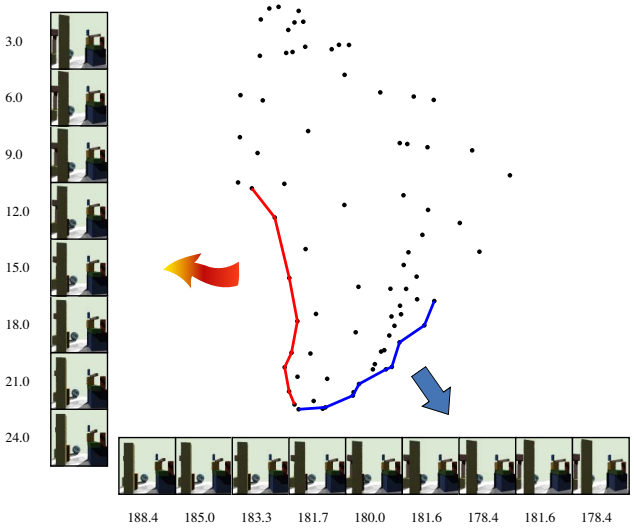

Figure 4. Images projected on a local tangent space of dimension 2. To see how images vary on the tangent space, we sampled the images along two arbitrary curves and displayed on the vertical and horizontal sequences. The vertical images vary mainly in the elevation angle of view (true elevation angles are given on the left for reference), and the horizontal images vary mainly in the azimuthal angle (true azimuthal angles are shown on the bottom).

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 figurine

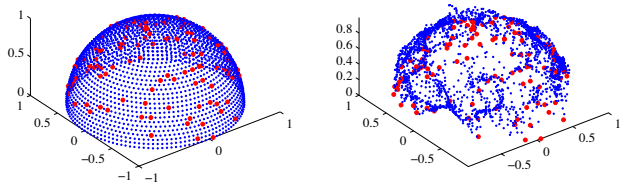

Figure 5. The two dimensional pose space of camera is considered as a hemisphere embedded in 3D as shown in the left figure. On the right is the estimated pose $\hat{z}$ of images, using the randomly chosen training points $\{v_i\}$ indicated by big red dots.

has 8 independent joints among which we used four: left / right arms and left / right legs (hip joints). Combinations of joint angles are generated by moving the arms -40 to160° with 20° separation, and the legs from -30 to 30° with 10° separation. Zero angles correspond to the neutral stand up position. A total of $5929 = 11^2 \times 7^2$ images are generated for each figure. Images were rendered with pixel sizes of 54×30 for figurine 1 and 48×30 for figurine 2. A small fraction (5 percent = 296) of the images were chosen with matched labelling, examples of which are shown in Figure 1.

We learned the manifolds of each image and the direct map between the two. For visualization, we defined two interesting curves on the manifold of figurine 1, which mimic a 'walking' and a 'hurray' motions to get corresponding animations of figurine 2, shown in Figure 7. This demonstrate the capability of our algorithm to synthesize a novel sequence in a data-driven way.

In this example, we have used pointwise matches to generate the animation. However, we can interpolate the temporal sequence to get a smoother animation. When the manifold is known exactly, interpolation is done analytically [6, 18]. Although not shown here, we also have preliminary results of smoothing out a sequence on each tangent space of the data manifold. This could be improved further by considering a dynamic model on manifolds [12].

4.3. Matching lip images

In this experiment, we demonstrate "lip sync" of real images through matching (refer to [5] for references therein.) We have acquired a sequence of lip images during continuous utterance of vowels. For the training set, a subject pronounced eight vowels 'ah', 'aa','ae','er','ih','o','uh','wu', and 'silence', and repeated the vowels five times to get a total of 45 images (Figure 6). For the test set, the subject freely made arbitrary vowel sound to get 900 images for each dataset. The second dataset was collected from the same procedure, from the same subject but under different

Figure 6. Training data for lip images. The subject pronounced eight vowels 'ah', 'aa','ae','er','ih','o','uh','wu', and 'silence', and repeated the vowels five times. The images in the top row and the bottom row are obtained under different conditions and times.

camera poses and at different times. To roughly register the images of each data respectively, we have tracked three markers on the forehead of the subject, and cropped lip regions after linear transformations. The resulting images were of size 50×53 and 45×45 pixels. The CCD camera showed a temporal change of the color tone, which was corrected by histogram equalization and Gaussian blurring. The dimensionality of the image manifold was empirically chosen to be $d = 2$.

The result of mapping is shown in Figure 8. The first row is a portion of test set of the first data, given as a query. The second row shows the corresponding weighted average $\tilde{z} = \sum_i \alpha_i(x)\hat{z}_i(x)$ from the learned map. The best matches (28) to the second row is given in the bottom row. Since the test sets are lip images of arbitrary vowel sounds, we do not have annotations to evaluate the matching. However from the visual inspection, the images of bottom row displays a satisfactory lip sync to the images in the top row.

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

[1] C. G. Atkeson, S. A. Schaal, and A. Moore. Locally weighted learning. *AI Review*, 11:11–73, 1997.

[2] P. N. Belhumeur and D. J. Kriegman. What is the set of images of an object under all possible illumination conditions? *International Journal of Computer Vision*, 28(3):245–260, 1998.

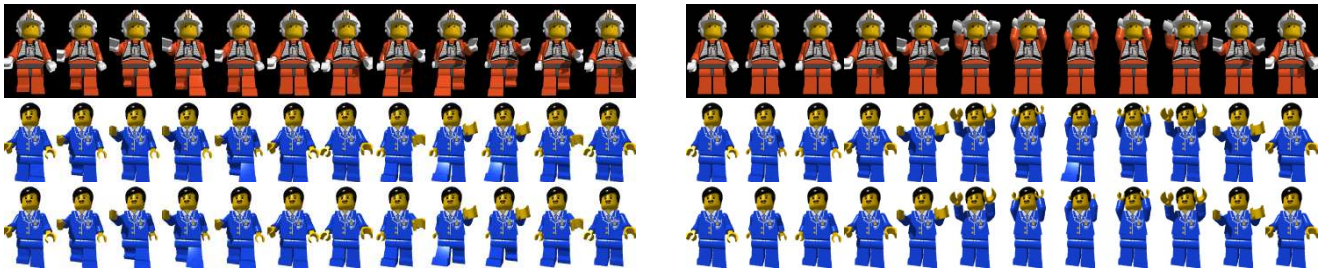

Figure 7. The top row shows two sequences 'walking' and 'hurray' of figurine 1, used as queries. The middle row shows the best matching images of figurine 2 from the learned map. The estimated poses are reasonably similar to the true poses in the bottom row.

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.

[3] C. M. Bishop, M. Svensén, and C. K. I. Williams. GTM: The generative topographic mapping. *Neural Computation*, 10(1):215–234, 1998.

[4] Y. Chang, C. Hu, and M. Turk. Probabilistic expression analysis on manifolds. In *CVPR (2)*, pages 520–527, 2004.

[5] T. Chen. Audiovisual speech processing. *IEEE Signal Processing*, 19(1):9–21, 2001.

[6] P. Crouch, G. Kun, and F. S. Leite. The De Casteljau algorithm on lie groups and spheres. *Journal of Dynamical and Control Systems*, 5(3):397–429, 1999.

[7] D. L. Donoho and C. Grimes. Hessian eigenmaps: locally linear embedding techniques for high-dimensional data. In *Proceedings of National Academy of Science, 100 (10)*, pages 5591–5596, 2003.

[8] D. L. Donoho and C. Grimes. Image manifolds which are isometric to euclidean space. *Journal of Mathematical Imaging and Vision*, 23(1):5–24, July 2005.

[9] A. M. Elgammal and C.-S. Lee. Inferring 3D body pose from silhouettes using activity manifold learning. In *CVPR (2)*, pages 681–688, 2004.

[10] X. He and P. Niyogi. Locality preserving projections. In *NIPS*, 2003.

[11] J. M. Lee. *Introduction to smooth manifolds*, volume 218 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, Berlin, Heidelberg, 2003.

[12] J. S. Marques, J. M. Lemos, and A. J. Abrantes. A HMM approach to the estimation of random trajectories on manifolds. *Signal Process.*, 82(9):1205–1214, 2002.

[13] H. Murase and S. K. Nayar. Visual learning and recognition of 3-D objects from appearance. *International Journal of Computer Vision*, 14(1):5–24, 1995.

[14] L. K. Saul and S. T. Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research, 4*, pages 119–155, 2003.

[15] B. Schölkopf, A. Smola, and K.-R. Mller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[16] H. S. Seung and D. D. Lee. The manifold ways of perception. *Science*, 290:2268–2269, 2000.

[17] C. Shan, S. Gong, and P. McOwan. Appearance manifold of facial expression. In *IEEE International Workshop on Human-Computer Interaction*, October 2005.

[18] K. Shoemake. Animating rotation with quaternion curves. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 245–254, New York, NY, USA, 1985. ACM Press.

[19] R. L. Tai-Peng Tian and S. Sclaroff. Articulated pose estimation in a learned smooth space of feasible solutions. Technical report, Department of Computer Science, Boston University, July 2005.

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

[21] F. W. Warner. *Foundations of Differentiable Manifolds and Lie Groups*, volume 94 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, Berlin, Heidelberg, 1983.

[22] C. Zhang, J. Wang, N. Zhao, and D. Zhang. Reconstruction and analysis of multi-pose face images based on nonlinear dimensionality reduction. *Pattern Recognition*, 37(2):325–336, 2004.