A Compositional Model for Low-Dimensional Image Set Representation

Hossein Mobahi
MIT
Cambridge, MA
hmobahi@csail.mit.edu

Ce Liu
Microsoft Research
Cambridge, MA
celiu@microsoft.com

William T. Freeman
MIT
Cambridge, MA
billf@mit.edu

Abstract

Learning a low-dimensional representation of images is useful for various applications in graphics and computer vision. Existing solutions either require manually specified landmarks for corresponding points in the images, or are restricted to specific objects or shape deformations. This paper alleviates these limitations by imposing a specific model for generating images; the nested composition of color, shape, and appearance. We show that each component can be approximated by a low-dimensional subspace when the others are factored out. Our formulation allows for efficient learning and experiments show encouraging results.

1. Introduction

One of the fundamental problems in computer vision is to characterize the "***space***" of images in some meaningful parametrization. Ideally, the parameters of this representation would correspond to pertinent variations of the considered data. For example, thinking of the space of horse images, these parameters could control different color, texture, and pose of the horses.

Such representations can be useful for some image editing applications like image morphing [13], shape transfer [14], or pose manipulation [20]. It can also provide an efficient way to navigate large datasets by automatically ordering the set w.r.t. a specific variation parameter [4]. Finally, this may have potential applications in detection and recognition problems [1].

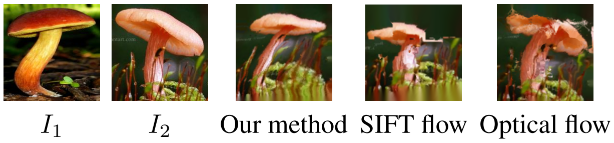
The stated problem naturally falls into the category of manifold learning for natural images. There are two classes of mainstream methods for this application. The first is to perform manifold analysis at the local level, e.g. using patches [6, 17, 23, 15]. However, this approach inevitably ignores the spatial structure of the image, which is crucial for representing images at the object level.

The second case is when manifold analysis is applied to the entire image. Such techniques have been successfully used to discover the manifold of pose changes [18]. However, they require very dense sampling of the image space. This is expected, because this approach does not make much structural assumption about how images are generated, other than forcing them to be on a low-dimensional manifold. However, the dense sampling constraint can be very restrictive. For example, a tremendously large number of images is required to densely capture all the gradual change of horse shapes and colors.

In order to cope with this problem, and yet work with an entire image, we propose a latent structure to constrain how images are generated. Our model takes into account variations in color, appearance, and shape in a nested and compositional way. In addition, the manifolds of shape and appearance are restricted to ***low-dimensional subspaces*** to further reduce sample complexity of the learning problem. We argue that this choice is a reasonable regularization for each of these components.

Each component used in our model is well-known and has been successfully used when other sources of variations are factored out. For example, in absence of shape and color variations, principal component analysis has been used to capture the variations in appearance [7]. When appearance and color are fixed, and only shape changes, optical flow [9, 22, 16] and its variants such as SIFT flow [14] can be used to infer geometric deformation across the images. Finally, if shape and appearance are not of any concern, global color transfer methods can be used to match the color spaces [21].

However, we believe it is the interaction of these three that makes the problem interesting and allows for less controlled scenarios. Our proposed scheme resembles some similarity to Active Appearance Model [7] and Morphable Models [10]. However, the former requires manually provided landmarks while the latter needs pre-aligned images. In contrast, our method is fully unsupervised. Another related work is Collection Flow [12] which provides an elegant formulation tailored for human faces. However, our proposed approach is designed to handle a larger class of image categories. Finally, we remind that Transformed Component Analysis (TCA) [8] also aims at cap-



I_1 I_2 Our method SIFT flow Optical flow

Figure 1. Our compositional model can transfer shape between quite dissimilar images, while optical flow and SIFT flow fail.

turing a low-dimensional appearance subspace by factoring out shape deformations. However, it works for shape models that have a small number of parameters (e.g. global translation). TCA does not scale to more complex motions as it must exhaustively process the entire (discretized) parameter space. In contrast, our method handles arbitrary motions represented by dense motion fields.

To better understand the problem addressed in this paper, consider the following example. Suppose we want to transfer the shape of image I_1 to image I_2 (see Figure 1). This may require knowing a dense correspondence field between two totally different objects, which cannot be obtained by classic solutions such as optical flow or even SIFT flow. None of these two methods know what the "space of mushrooms" looks like. In fact, their obtained solution drastically exists such space as seen in Figure 1. In contrast, our composition scheme is able to provide a reasonable shape transfer, as it relies on a learned model for the space of mushroom images in terms of their color, appearance and shape.

In this work, we present a simple algorithm for recovering low-dimensional representation simultaneously for color, appearance and shape. The problem is formulated as an optimization task. The compositional and subspace assumptions in our model provide very efficient update rules for optimization. Our quantitative and qualitative results in some related tasks such as image editing and image browsing are encouraging.

2. Compositional Low-Dimensional Model

2.1. Definitions

We denote sets by $\mathcal{X}$, vectors by $\boldsymbol{x}$, and matrices by $\boldsymbol{X}$. Scalar valued functions are denoted as $f(.)$, vector valued functions as $\boldsymbol{f}(.)$, and functionals as $F\,[\,.\,]$. We denote a subspace by $\mathscr{X}$. The symbol $\triangleq$ refers to equality by definition. A set $\{x_1,\dots,x_n\}$ is alternatively written as $\{x_k\}_{k=1}^n$. By $\|.\|$ we mean $\|.\|_2$. Define $|\mathcal{X}|$ to be the number of elements of $\mathcal{X}$ when $\mathcal{X}$ is discrete, and let $|\mathcal{X}| \triangleq \int_{\mathcal{X}} d\boldsymbol{x}$ when $\mathcal{X}$ is continuous.

The operator $\mathrm{vec}(f(\boldsymbol{x})\,,\,\mathcal{X}')$ is defined to take a map $f:\mathcal{X}\to\mathbb{R}$ and evaluate it on a uniformly spaced grid $\mathcal{X}'\subset\mathcal{X}$. It returns a vector $\boldsymbol{f}$ by concatenating all these evaluations. The size of this vector is obviously $|\mathcal{X}'|$. The operator $\mathrm{unvec}(\boldsymbol{f}\,,\,\mathcal{X})$ approximates the original map f on the domain $\mathcal{X}$ by interpolating[1] between the elements of $\boldsymbol{f}$.

When $\boldsymbol{f}(\boldsymbol{x})$ is multivalued, say consisting of r components $(f_1(\boldsymbol{x}),\dots,f_r(\boldsymbol{x}))$, we define $\mathrm{vec}(\boldsymbol{f}(\boldsymbol{x})\,,\,\mathcal{X}') \triangleq \big(\mathrm{vec}(f_1(\boldsymbol{x})\,,\,\mathcal{X}')\,\dots\,\mathrm{vec}(f_r(\boldsymbol{x})\,,\,\mathcal{X}')\big)$, which is simply concatenating all single valued components of $\boldsymbol{f}(\boldsymbol{x})$. The operator $\mathrm{unvec}(\boldsymbol{f}\,,\,\mathcal{X})$ in this case is defined similar to single valued maps, except that it approximates the original multivalued f by interpolating[2] between the elements of $\boldsymbol{f}$.

Throughout the paper, we drop the second argument of vec and unvec for brevity. It should be clear that $\mathcal{X}'$ is simply the pixel space of the images and $\mathcal{X}\subset\mathbb{R}^2$ is the tightest region that encloses $\mathcal{X}'$.

2.2. Model Description

Let $\mathcal{X}\subset\mathbb{R}^2$ and $\mathcal{C}\subset\mathbb{R}^3$ be ***location*** and ***color*** spaces respectively. Given a set of input color images $\mathcal{F}\triangleq\{\boldsymbol{f}_k(\boldsymbol{x})\}_{k=1}^n$, where each image is $\boldsymbol{f}_k:\mathcal{X}\to\mathcal{C}$. We model each input image $\boldsymbol{f}_k(\boldsymbol{x})\in\mathcal{F}$ by the following composition of functions,

$$\boldsymbol{f}_k(\boldsymbol{x}) \approx \boldsymbol{h}_k\Big(\boldsymbol{g}_k\big(\boldsymbol{u}_k(\boldsymbol{x})\big)\Big) \quad \text{for } k=1,\dots,n\,, \tag{1}$$

where the maps are $\boldsymbol{h}_k:\mathcal{C}\to\mathcal{C}$, $\boldsymbol{g}_k:\mathcal{X}\to\mathcal{C}$ and $\boldsymbol{u}_k:\mathcal{X}\to\mathcal{X}$. This nested form imposes a specific latent structure on how the images $\mathcal{F}$ are generated. Based on the nature of each map, we can think of them respectively as ***photometric***, ***appearance***, and ***geometric*** transforms.

The goal is to estimate these maps from the given data $\mathcal{F}$. We adopt ℓ_2 norm to quantify the approximation error. Hence, we aim at solving the following optimization task,

$$\min_{\{(\boldsymbol{h}_k,\boldsymbol{g}_k,\boldsymbol{u}_k)\}_{k=1}^n} \sum_{k=1}^n \int_{\mathcal{X}} \|\,\boldsymbol{h}_k\Big(\boldsymbol{g}_k\big(\boldsymbol{u}_k(\boldsymbol{x})\big)\Big) - \boldsymbol{f}_k(\boldsymbol{x})\,\|^2\,d\boldsymbol{x}\,. \tag{2}$$

The problem (2) is obviously ***ill-posed***; there are many different ways to choose equally good maps $\{\boldsymbol{h}_k(\boldsymbol{c}),\boldsymbol{g}_k(\boldsymbol{x}),\boldsymbol{u}_k(\boldsymbol{x})\}_{k=1}^n$ to approximate $\mathcal{F}$. Therefore, we need to restrict the solution space.

We control the capacity of the maps $\{\boldsymbol{g}_k(\boldsymbol{x})\}_{k=1}^n$ and $\{\boldsymbol{u}_k(\boldsymbol{x})\}_{k=1}^n$ by forcing them to be constructed from a small number of ***basis functions***. In addition, we regularize the problem by imposing ***spatial smoothness*** on geometric transforms. We also restrict each color transform function $\boldsymbol{h}_k$ to be a ***shift and rotation***, i.e. $\boldsymbol{h}_k(\boldsymbol{c})\triangleq\boldsymbol{A}_k\boldsymbol{c}+\boldsymbol{b}_k$. Here, the rotation matrix $\boldsymbol{A}_k$ is 3×3 and the vector $\boldsymbol{b}_k$ is 3×1. Hence, the regularized problem is the following,

[1] In this paper we always use bilinear interpolation, as our functions are $2D$.

[2] We interpolate each component independently. This is for computational advantage of being able to use bilinear interpolation again.

$$\min_{\{(\boldsymbol{A}_k,\boldsymbol{b}_k,\boldsymbol{g}_k,\boldsymbol{u}_k)\}_{k=1}^n} \sum_{k=1}^n \Big(\int_{\mathcal{X}} \| \boldsymbol{A}_k\, \boldsymbol{g}_k\big(\boldsymbol{u}_k(\boldsymbol{x})\big) + \boldsymbol{b}_k - \boldsymbol{f}_k(\boldsymbol{x}) \|^2 \, d\boldsymbol{x} + R\,[\,\boldsymbol{u}_k\,]\Big)$$

$$\text{s.t. } \forall k \;;\; \boldsymbol{g}_k(\boldsymbol{x}) \in \mathscr{G}\,,\; \boldsymbol{u}_k(\boldsymbol{x}) \in \mathscr{U}\,,\; \boldsymbol{A}_k \in SO(3)\,. \qquad (3)$$

Here R is a functional that penalizes spatial variations of its argument. The spaces $\mathscr{G}$ and $\mathscr{U}$ have the property that there exist for them d_g and d_u basis functions $\{\boldsymbol{\phi}_p(\boldsymbol{x})\}_{p=1}^{d_g}$ and $\{\boldsymbol{\psi}_q(\boldsymbol{x})\}_{q=1}^{d_u}$ such that $\mathscr{G} = \text{span}(\boldsymbol{\phi}_1(\boldsymbol{x}), \ldots, \boldsymbol{\phi}_{d_g}(\boldsymbol{x}))$ and $\mathscr{U} = \text{span}(\boldsymbol{\psi}_1(\boldsymbol{x}), \ldots, \boldsymbol{\psi}_{d_u}(\boldsymbol{x}))$.

In the following we discuss the rationale behind the regularization choices. Affine maps have shown to be rich enough for transferring color across natural images with a reasonable quality [21]. Inspired by this model, we work with a special case of affine transform where the matrix $\boldsymbol{A}$ is restricted to a rotation. This is merely for its numerical advantage; we will see in the next section that our method relies on $\boldsymbol{A}^{-1}$. By choosing $\boldsymbol{A}$ to be a rotation, it is guaranteed that $\boldsymbol{A}^{-1}$ can be stably computed.

Basic deformations are low-rank, e.g. motion field of global translation or global scaling is rank one. The object may have a mixture of basic motions (e.g. due to articulation, in a piecewise or combined fashion). The total rank of the motion is not more than sum of the rank of basic motions. The latter is typically much smaller than the dimension of the entire field. Finally, it is empirically observed that when similar objects are registered and brightness normalized, they lie on a low-dimensional subspace relative to the dimension of the pixel space [2]. This is also a key assumption in Active Appearance Models [7].

2.3. Illustrative Example

Suppose we are interested in parameterizing the space of mushrooms. These objects vary in their texture, color, and shape. Nevertheless, these variations are not arbitrary, and our goal is to approximate the space of these variations using the proposed model.

Let $\dim(\mathscr{G}) = 1$ and $\dim(\mathscr{U}) = 6$. Using the proposed scheme, we obtain a subspace for appearance and another for shape as shown in Figure 2. Interestingly, one can semantically interpret what each basis function tries to capture. For example, moving along the $\boldsymbol{\phi}_1(\boldsymbol{x})$ sweeps dark background and bright mushroom to the opposite case. In addition, $\boldsymbol{\psi}_1(\boldsymbol{x})$ controls height, $\boldsymbol{\psi}_2(\boldsymbol{x})$ changes cap's angle, $\boldsymbol{\psi}_3(\boldsymbol{x})$ chooses flatness of cap, $\boldsymbol{\psi}_4(\boldsymbol{x})$ leans mushroom toward left or right, and so forth. In addition, the evolution of the internal representation to the input image is depicted in Figure 3.

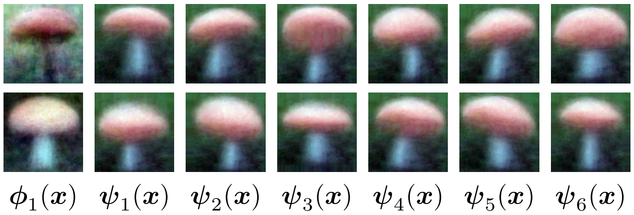

$\boldsymbol{\phi}_1(\boldsymbol{x})$
$\boldsymbol{\psi}_1(\boldsymbol{x})$
$\boldsymbol{\psi}_2(\boldsymbol{x})$
$\boldsymbol{\psi}_3(\boldsymbol{x})$
$\boldsymbol{\psi}_4(\boldsymbol{x})$
$\boldsymbol{\psi}_5(\boldsymbol{x})$
$\boldsymbol{\psi}_6(\boldsymbol{x})$

Figure 2. The effect of moving along each basis function in negative (top) and positive (bottom) directions.

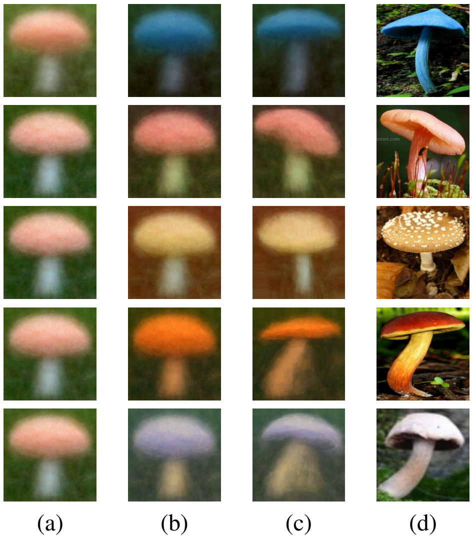


Figure 3. Evolution of mushrooms from $\boldsymbol{g}$ to $\boldsymbol{f}$. (a): $\boldsymbol{g}_k(\boldsymbol{x})$, (b): $\boldsymbol{A}_k\boldsymbol{g}_k(\boldsymbol{x}) + \boldsymbol{b}_k$, (c): $\boldsymbol{A}_k\boldsymbol{g}_k(\boldsymbol{u}_k(\boldsymbol{x}))+\boldsymbol{b}_k$, (d): $\boldsymbol{f}_k(\boldsymbol{x})$

3. Optimization

Finding the exact solution of the problem (3) is difficult, due to its non-convexity. We obtain an approximate solution to this problem by alternating between subproblems. Specifically, there are three sets of variables, namely color transform $\{(\boldsymbol{A}_k, \boldsymbol{b}_k)\}_{k=1}^n$, appearance $\{\boldsymbol{g}_k(\boldsymbol{x})\}_{k=1}^n$, and shape deformation $\{\boldsymbol{u}_k(\boldsymbol{x})\}_{k=1}^n$.

By focusing only on one category at a time (fixing the other two), we can derive efficient update rules that reduce the objective function (3). These update rules either have closed form expression, or there exist efficient algorithms to solve them. In the sequel, we study each of these subproblems and eventually propose a complete algorithm that puts the pieces together.

3.1. Solving for $\{\boldsymbol{g}_k(\boldsymbol{x})\}_{k=1}^n$

Remember from (1) that the model tries to satisfy $\boldsymbol{f}_k(\boldsymbol{x}) \approx \boldsymbol{A}_k\boldsymbol{g}_k\big(\boldsymbol{u}_k(\boldsymbol{x})\big) + \boldsymbol{b}_k$ for $k = 1, \ldots, n$. The rotation matrix $\boldsymbol{A}_k$ is obviously invertible. In addition, suppose the shape deformation is invertible or its inverse can be approximated (details at the end of this subsection). Now we alternatively express the model requirement as $\boldsymbol{g}_k(\boldsymbol{x}) \approx \boldsymbol{A}_k^{-1}\Big(\boldsymbol{f}_k\big(\boldsymbol{u}_k^{-1}(\boldsymbol{x})\big) - \boldsymbol{b}_k\Big)$. Since $\{(\boldsymbol{A}_k, \boldsymbol{b}_k)\}_{k=1}^n$

and $\{\boldsymbol{u}_k(\boldsymbol{x})\}_{k=1}^n$ are fixed, the optimization subproblem becomes as follows,

$$\begin{aligned}\min_{\{\boldsymbol{g}_k\}_{k=1}^n} \sum_{k=1}^n \int_{\mathcal{X}} \| \, \boldsymbol{g}_k(\boldsymbol{x}) \; - \; \boldsymbol{A}_k^{-1}\Big(\boldsymbol{f}_k\big(\boldsymbol{u}_k^{-1}(\boldsymbol{x})\big) - \boldsymbol{b}_k\Big) \, \|^2 \, d\boldsymbol{x} \\ \text{s.t.} \qquad \forall k \; ; \; \boldsymbol{g}_k(\boldsymbol{x}) \in \mathscr{G} \, . \qquad (4)\end{aligned}$$

Let $\boldsymbol{z}(\boldsymbol{x}) \triangleq \boldsymbol{A}_k^{-1}\Big(\boldsymbol{f}_k\big(\boldsymbol{u}_k^{-1}(\boldsymbol{x})\big) - \boldsymbol{b}_k\Big)$, which does not depend on $\boldsymbol{g}_k(\boldsymbol{x})$. The problem then becomes minimizing the ℓ_2 reconstruction error $\sum_{k=1}^n \int_{\mathcal{X}} \| \, \boldsymbol{g}_k(\boldsymbol{x}) - \boldsymbol{z}(\boldsymbol{x}) \, \|^2 \, d\boldsymbol{x}$ subject to subspace rank being d_g. The solution to this problem is known to be related to the eigenfunctions of the covariance operator of $\{\boldsymbol{z}_k(\boldsymbol{x})\}_{k=1}^n$.

The algorithm actually applies spectral decomposition to the covariance of the discretized $\boldsymbol{g}_k(\boldsymbol{x})$ and $\boldsymbol{z}_k(\boldsymbol{x})$, using the vec operator defined in Section 2.1. Let $\tilde{\boldsymbol{z}}_k \triangleq \text{vec}(\boldsymbol{z}_k(\boldsymbol{x}))$ and $\bar{\boldsymbol{z}} \triangleq \frac{1}{n}\sum_{k=1}^n \tilde{\boldsymbol{z}}_k$. Then, the top d_g eigenvectors of $\sum_{k=1}^n (\tilde{\boldsymbol{z}}_k - \bar{\boldsymbol{z}})(\tilde{\boldsymbol{z}}_k - \bar{\boldsymbol{z}})^T$, namely $(\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_{d_g})$ are computed. It then follows that $\tilde{\boldsymbol{g}}_k = \bar{\boldsymbol{z}} + \sum_{p=1}^{d_g} \boldsymbol{\phi}_p \boldsymbol{\phi}_p^T (\tilde{\boldsymbol{z}}_k - \bar{\boldsymbol{z}})$

The field $\boldsymbol{u}_k(\boldsymbol{x})$ is not necessarily invertible; there might be locations $\boldsymbol{x}$ where $\boldsymbol{u}_k^{-1}(\boldsymbol{x})$ is not defined, or has multiple values. Thus, instead of working with $\boldsymbol{u}_k^{-1}$, we define a function $\boldsymbol{u}_k^\sharp : \mathcal{X} \rightarrow \mathcal{X}$ that approximates $\boldsymbol{u}_k^{-1}$. We first resolve multi-valued issue of $\boldsymbol{u}_k^{-1}$ in the following way,

$$\forall \boldsymbol{x} \in \mathcal{X} \, ; \, |\boldsymbol{u}_k^{-1}(\boldsymbol{x})| \geq 2 \Rightarrow \boldsymbol{u}_k^\sharp(\boldsymbol{x}) = \arg \min_{\hat{\boldsymbol{u}} \in \boldsymbol{u}_k^{-1}(\boldsymbol{x})} \|\hat{\boldsymbol{u}} - \bar{\boldsymbol{u}}_{\boldsymbol{x}}\| \, ,$$

where $\bar{\boldsymbol{u}}_{\boldsymbol{x}} \triangleq \frac{1}{|\mathcal{U}_{\boldsymbol{x}}|} \sum_{\tilde{\boldsymbol{u}} \in \mathcal{U}_{\boldsymbol{x}}} \tilde{\boldsymbol{u}}, \, \mathcal{U}_{\boldsymbol{x}} \triangleq \cup_{\boldsymbol{y} \in \mathcal{N}(\boldsymbol{x})} \{\boldsymbol{u}_k^{-1}(\boldsymbol{y})\}$ and $\mathcal{N}(\boldsymbol{x})$ is some small neighborhood of $\boldsymbol{x}$. After handling multivalued points, we get to points $\boldsymbol{x}$ where $\boldsymbol{u}_k^{-1}(\boldsymbol{x})$ has no value. We fill in these points by interpolating them from the remaining $\boldsymbol{u}_k^\sharp$ that has value.

With abuse of notation, in the rest of the paper we write $\boldsymbol{u}^{-1}$ to refer to $\boldsymbol{u}^\sharp$, as the former better conveys the notion of inverse.

3.2. Solving for $\{\boldsymbol{u}_k(\boldsymbol{x})\}_{k=1}^n$

Since $\{(\boldsymbol{A}_k, \boldsymbol{b}_k)\}_{k=1}^n$ and $\{\boldsymbol{g}_k(\boldsymbol{x})\}_{k=1}^n$ are fixed, optimization (3) simplifies to the following,

$$\begin{aligned}\min_{\{\boldsymbol{u}_k\}_{k=1}^n} \sum_{k=1}^n \Big(\int_{\mathcal{X}} \| \, \boldsymbol{A}_k \, \boldsymbol{g}_k\big(\boldsymbol{u}_k(\boldsymbol{x})\big) + \boldsymbol{b}_k \; - \; \boldsymbol{f}_k(\boldsymbol{x}) \, \|^2 \, d\boldsymbol{x} \\ + R\,[\,\boldsymbol{u}_k\,]\Big) \\ \text{s.t.} \qquad \forall k \; ; \; \boldsymbol{u}_k(\boldsymbol{x}) \in \mathscr{U} \, . \qquad (5)\end{aligned}$$

For efficiency, we relax this task to two simpler subproblems. We first solve (5) without subspace constraint, and then project the solution of that onto the subspace. Specifically, in the first step we compute,

$$\begin{aligned}\hat{\boldsymbol{u}}_k(\boldsymbol{x}) \triangleq \arg\min_{\boldsymbol{u}_k} \int_{\mathcal{X}} \| \, \boldsymbol{A}_k \, \boldsymbol{g}_k\big(\boldsymbol{u}_k(\boldsymbol{x})\big) + \boldsymbol{b}_k \; - \; \boldsymbol{f}_k(\boldsymbol{x}) \, \|^2 \, d\boldsymbol{x} \\ + R\,[\,\boldsymbol{u}_k\,] \, , \qquad (6)\end{aligned}$$

and then solve the following,

$$\begin{aligned}\min_{\{\boldsymbol{u}_k\}_{k=1}^n} \sum_{k=1}^n \int_{\mathcal{X}} \| \, \boldsymbol{u}_k(\boldsymbol{x}) \; - \; \hat{\boldsymbol{u}}_k(\boldsymbol{x}) \, \|^2 \\ \text{s.t.} \qquad \forall k \; ; \; \boldsymbol{u}_k(\boldsymbol{x}) \in \mathscr{U} \, . \qquad (7)\end{aligned}$$

The task in (6) is a standard optical flow problem, for which there exist efficient algorithms. Specifically, we use the optical flow method of [22] which has shown to have a good performance. This method uses ***Charbonnier*** penalty function $\sqrt{x^2 + \epsilon^2}$ [5] to construct a robust spatial regularizer functional $R[\,.\,]$ (see [22] for details).

The subproblem (7) is again minimization of ℓ_2 reconstruction error, subject to subspace rank constraint. Similar to section 3.1, the solution of this problem can be derived by spectral decomposition.

3.3. Solving for $\{(\boldsymbol{A}_k, \boldsymbol{b}_k)\}_{k=1}^n$

The subproblem is as below,

$$\begin{aligned}\min_{\{\boldsymbol{A}_k, \boldsymbol{b}_k\}_{k=1}^n} \sum_{k=1}^n \int_{\mathcal{X}} \| \, \boldsymbol{A}_k \, \boldsymbol{g}_k\big(\boldsymbol{u}_k(\boldsymbol{x})\big) + \boldsymbol{b}_k - \boldsymbol{f}_k(\boldsymbol{x}) \, \|^2 \, d\boldsymbol{x} \\ \text{s.t.} \qquad \boldsymbol{A}_k \in SO(3) \, .\end{aligned}$$

This problem has a closed form answer known as ***Kabsch's solution*** [11]. Define $\boldsymbol{T}_k$ as the following,

$$\boldsymbol{T}_k \triangleq \Big(\int_{\mathcal{X}} \big(\boldsymbol{f}_k(\boldsymbol{x}) - \bar{\boldsymbol{f}}_k\big)\big(\boldsymbol{g}_k(\boldsymbol{u}_k(\boldsymbol{x})) - \bar{\boldsymbol{g}}_k\big)^T \, d\boldsymbol{x} \Big) \, ,$$

where $\bar{\boldsymbol{f}}_k \triangleq \frac{1}{|\mathcal{X}|}\int_{\mathcal{X}} \boldsymbol{f}_k(\boldsymbol{x})$ and $\bar{\boldsymbol{g}}_k \triangleq \frac{1}{|\mathcal{X}|}\int_{\mathcal{X}} \boldsymbol{g}_k(\boldsymbol{u}_k(\boldsymbol{x}))$. Let $\boldsymbol{T}_k = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$ be the ***singular value decomposition*** of the 3×3 matrix $\boldsymbol{T}_k$. Then it follows that,

$$\begin{aligned}\boldsymbol{A}_k &= \boldsymbol{U}\boldsymbol{V}^T \\ \boldsymbol{b}_k &= \bar{\boldsymbol{f}}_k - \boldsymbol{A}_k \bar{\boldsymbol{g}}_k \, .\end{aligned}$$

3.4. Algorithm

We can now construct an algorithm for computing the low-dimensional representation of image data. The algorithm essentially uses the ideas introduced above.

Algorithm 1 Computing Low Dimensional Representation of an Image Set.

```
Input: A set of images $\{\boldsymbol{f}_1(\boldsymbol{x}),\dots,\boldsymbol{f}_n(\boldsymbol{x})\}$.
  $\boldsymbol{u}_k(\boldsymbol{x}) = \boldsymbol{x}$    for $k = 1,\dots,n$
  $\boldsymbol{A}_k = \boldsymbol{I}, \boldsymbol{b}_k = \boldsymbol{0}$    for $k = 1,\dots,n$
  repeat
    for $k = 1,\dots,n$ do
      $\tilde{\boldsymbol{f}}_k(\boldsymbol{x}) = \boldsymbol{A}_k^{-1}\big(\boldsymbol{f}_k(\boldsymbol{u}_k^{-1}(\boldsymbol{x})) - \boldsymbol{b}_k\big)$
      $\tilde{\boldsymbol{f}}_k = \mathrm{vec}(\tilde{\boldsymbol{f}}_k(\boldsymbol{x}))$
    end for
    $\bar{\boldsymbol{f}} = \frac{1}{n}\sum_{k=1}^{n}\tilde{\boldsymbol{f}}_k$
    $(\boldsymbol{\phi}_1,\dots,\boldsymbol{\phi}_{d_g}) \leftarrow$ Top $d_g$ eigenvectors of $\sum_{k=1}^{n}(\tilde{\boldsymbol{f}}_k - \bar{\boldsymbol{f}})(\tilde{\boldsymbol{f}}_k - \bar{\boldsymbol{f}})^T$
    for $k = 1,\dots,n$ do
      $\tilde{\boldsymbol{g}} = \bar{\boldsymbol{f}} + \sum_{p=1}^{d_g}\boldsymbol{\phi}_p\boldsymbol{\phi}_p^T(\tilde{\boldsymbol{f}}_k - \bar{\boldsymbol{f}})$
      $\boldsymbol{g}_k(\boldsymbol{x}) = \mathrm{unvec}(\tilde{\boldsymbol{g}})$
      $\hat{\boldsymbol{u}}_k(\boldsymbol{x}) = \arg\min_{\boldsymbol{u}} \int_{\mathcal{X}} \Big(\parallel \boldsymbol{A}_k\boldsymbol{g}_k(\boldsymbol{u}(\boldsymbol{x})) + \boldsymbol{b}_k - \boldsymbol{f}_k(\boldsymbol{x}) \parallel^2 + R(\boldsymbol{u})\Big)\, d\boldsymbol{x}$
      $\hat{\boldsymbol{u}}_k = \mathrm{vec}(\hat{\boldsymbol{u}}_k(\boldsymbol{x}))$
    end for
    $\bar{\boldsymbol{u}} = \frac{1}{n}\sum_{k=1}^{n}\hat{\boldsymbol{u}}_k$
    $(\boldsymbol{\psi}_1,\dots,\boldsymbol{\psi}_{d_u}) \leftarrow$ Top $d_u$ eigenvectors of $\sum_{k=1}^{n}(\hat{\boldsymbol{u}} - \bar{\boldsymbol{u}})(\hat{\boldsymbol{u}} - \bar{\boldsymbol{u}})^T$
    for $k = 1,\dots,n$ do
      $\tilde{\boldsymbol{u}} = \bar{\boldsymbol{u}} + \sum_{p=1}^{d_u}\boldsymbol{\psi}_p\boldsymbol{\psi}_p^T(\hat{\boldsymbol{u}} - \bar{\boldsymbol{u}})$
      $\boldsymbol{u}_k(\boldsymbol{x}) = \mathrm{unvec}(\tilde{\boldsymbol{u}})$
      $\bar{\boldsymbol{f}}_k = \frac{1}{|\mathcal{X}|}\sum_{\boldsymbol{x}\in\mathcal{X}}\boldsymbol{f}_k(\boldsymbol{x})$
      $\bar{\boldsymbol{g}}_k = \frac{1}{|\mathcal{X}|}\sum_{\boldsymbol{x}\in\mathcal{X}}\boldsymbol{g}_k(\boldsymbol{x})$
      $\boldsymbol{T} = \sum_{\boldsymbol{x}\in\mathcal{X}}\big(\boldsymbol{f}_k(\boldsymbol{x}) - \bar{\boldsymbol{f}}_k\big)\big(\boldsymbol{g}_k(\boldsymbol{x}) - \bar{\boldsymbol{g}}_k\big)^T$
      $(\boldsymbol{U},\boldsymbol{S},\boldsymbol{V}) = svd(\boldsymbol{T})$
      $\boldsymbol{A}_k = \boldsymbol{U}\boldsymbol{V}^T$
      $\boldsymbol{b}_k = \bar{\boldsymbol{f}}_k - \boldsymbol{A}_k\bar{\boldsymbol{g}}_k$
    end for
  until Convergence
Output: $\{\boldsymbol{g}_1(\boldsymbol{x}),\dots,\boldsymbol{g}_n(\boldsymbol{x})\}$, $\{\boldsymbol{u}_1(\boldsymbol{x}),\dots,\boldsymbol{u}_n(\boldsymbol{x})\}$,
  $\{\boldsymbol{\phi}_1(\boldsymbol{x}),\dots,\boldsymbol{\phi}_{d_g}(\boldsymbol{x})\}$, and $\{\boldsymbol{\psi}_1(\boldsymbol{x}),\dots,\boldsymbol{\psi}_{d_u}(\boldsymbol{x})\}$.
```

4. Quantitative Evaluation

We can quantitatively evaluate our method based on the accuracy of its estimated deformation fields $\{\boldsymbol{u}_k(\boldsymbol{x})\}_{k=1}^{n}$. We compare the quality of these fields against those estimated by robust optical flow and SIFT flow.

Datasets : We test our method on four object categories, namely, horses, mushrooms, flowers, and birds. For horses, we use publicly available Weizmann dataset [3] that consists of 327 color images. For each remaining category, we collected 120 images by Google's image search. In all these four sets, the border of the image almost coincides with the bounding box of the object. Each dataset $\mathcal{F}$ comes with a ground truth image set $\mathcal{B}$, i.e. for each $\boldsymbol{f}(\boldsymbol{x}) \in \mathcal{F}$ there is a corresponding map $b(\boldsymbol{x}) \in \mathcal{B}$, where $b : \mathcal{X} \to \{0, 1\}$.

Performance Criteria :

Each of the compared methods produces a deformation field $\boldsymbol{u}_{jk}(\boldsymbol{x})$, which provides dense correspondence from locations in image $\boldsymbol{f}_j$ to locations in image $\boldsymbol{f}_k$. For optical flow and SIFT flow, one needs to run these algorithms for each pair of i, j by providing $(\boldsymbol{f}_j, \boldsymbol{f}_k)$ as input in order to get $\boldsymbol{u}_{jk}(\boldsymbol{x})$ as output. We used publicly available codes for computing robust optical flow [22] and SIFT flow [14]. We used the default parameters shipped with these packages. In the implementation of our algorithm, we use the same optical flow package of [22] with the same parameters. For our method, $\boldsymbol{u}_{jk}(\boldsymbol{x})$ can be constructed for any pair (i, j) using the output set $\{\boldsymbol{u}_k(\boldsymbol{x})\}$. Specifically, it follows that $\boldsymbol{u}_{jk}(\boldsymbol{x}) = \boldsymbol{u}_j^{-1}\big(\boldsymbol{u}_k(\boldsymbol{x})\big)$ (see Section 5.1 for details).

To evaluate a method, we consider two performance criteria, namely region based and boundary based. Each criterion, is a functional z that takes two binary maps and measures similarity or dissimilarity of the pair, i.e. it is of form $z\big(b_j(.), b_k(.)\big)$. Since both region based and boundary based criteria operate on image pairs, their performance on the entire set is defined as their average value over all possible pairs,

$$\bar{z} \triangleq \frac{1}{n^2}\sum_{j=1}^{n}\sum_{k=1}^{n} z\big(b_j(\boldsymbol{u}_{jk}(.)), b_k(.)\big)\,.$$

The region based criterion for a pair of binary maps is defined as below,

$$z_R(b_j(.), b_k(.)) \triangleq \frac{\sum_{\boldsymbol{x}} b_j(\boldsymbol{x}) b_k(\boldsymbol{x})}{\sum_{\boldsymbol{x}} b_j(\boldsymbol{x})}\,.$$

This essentially captures what fraction of figure points in b_j coincide with figure points in b_k. Observe that when $b_j = b_k$, then $z_R(b_j(.), b_k(.)) = 1$. Hence, the larger values of z_R are better, as they indicate larger overlap.

The boundary based criterion z_B measures boundary displacement error. Let $\mathcal{Q}_k$ the set of boundary coordinates $\boldsymbol{x}$ in the mask $b_k(\boldsymbol{x})$. Then z_B is defined as below,

$$z_B(b_j(.), b_k(.)) \triangleq \frac{1}{|\mathcal{Q}_j|}\sum_{\boldsymbol{x}\in\mathcal{Q}_j}\min_{\boldsymbol{y}\in\mathcal{Q}_k}\|\boldsymbol{x} - \boldsymbol{y}\|\,.$$

Result : Tables 1 and 2 summarize the performance of each method. For all cases of our method, we set $d_g = 1$ and $d_u = 6$, which was empirically observed to work well jointly for all of the data we used here[3]. The proposed

[3] This setting was selected by exhaustively searching the 10×10 space of (d_g, d_u), where each dimensionality ranges from 1 to 10.

	Mushroom	Flowers	Birds	Horses
Optical Flow	**0.76**	0.61	0.69	0.62
SIFT Flow	0.70	0.62	0.67	0.59
Proposed Method	0.73	**0.68**	**0.71**	**0.64**

Table 1. Region based criterion $\bar{z}_R$ (higher is better).

	Mushroom	Flowers	Birds	Horses
Optical Flow	11.54	15.34	14.27	9.62
SIFT Flow	13.73	15.39	14.31	9.72
Proposed Method	**5.69**	**5.65**	**6.10**	**4.61**

Table 2. Boundary based criterion $\bar{z}_B$ (lower is better).

method outperforms via boundary displacement error and achieves good performance w.r.t. region based score.

5. Qualitative Evaluation

5.1. Scene Alignment

Suppose we want to align two scenes $\boldsymbol{f}_1(\boldsymbol{x})$ and $\boldsymbol{f}_2(\boldsymbol{x})$ via non-parametric deformation. Specifically, we are looking for a deformation field $\boldsymbol{u}^*(\boldsymbol{x})$ so that the deformed image $\boldsymbol{f}_1(\boldsymbol{u}^*(\boldsymbol{x}))$ ***"looks like"*** $\boldsymbol{f}_2(\boldsymbol{x})$. If the two scenes are quite similar, e.g. they are two consecutive frames of a video, the problem is well-defined. In fact, in this case the solution $\boldsymbol{u}^*(\boldsymbol{x})$ can be often recovered by computing the ***optical flow*** between the two images.

Raising the dissimilarity can break down the "brightness constancy" assumption and hence optical flow cannot be used. If the changes, however, are invariant to some feature-based representation, one can try to establish correspondence between pair of features instead of pixel intensity values. In this regime, tools such as ***SIFT flow*** [14] can be used. However, using the proposed scheme, we can even go beyond this, i.e. when there is no clear match solely using local invariant features.

The key here is that, rather than working with just two images, we work with a large set of images $\mathcal{F}$. The pair of interest, $\boldsymbol{f}_1(\boldsymbol{x})$ and $\boldsymbol{f}_2(\boldsymbol{x})$, are just two elements of $\mathcal{F}$. The set $\mathcal{F}$ can regularize our choice of $\boldsymbol{u}^*(\boldsymbol{x})$, because all elements of $\mathcal{F}$ are supposed to be related to each other after factoring out their color and geometry. The latter point is an important implication of working with a large set of related images, as opposed to just having two images.

Specifically, our model provides a common coordinate system in latent space, where pixels correspond to each other, regardless of their color and geometric variations. That is, for any pair of images $\boldsymbol{f}_j$ and $\boldsymbol{f}_k$ in $\mathcal{F}$, it holds that $\boldsymbol{g}_j(\boldsymbol{x}) \approx \boldsymbol{g}_k(\boldsymbol{x})$, where $\boldsymbol{g}_j(\boldsymbol{x}) \approx \boldsymbol{A}_j^{-1}\boldsymbol{f}_j\big(\boldsymbol{u}_j(\boldsymbol{x})\big) - \boldsymbol{b}_j$ ($\boldsymbol{g}_k(\boldsymbol{x})$ is defined in a similar fashion). This provides a correspondence between images of $\mathcal{F}$; for every possible $\boldsymbol{x}$, the location $\boldsymbol{u}_j^{-1}(\boldsymbol{x})$ in $\boldsymbol{f}_j$ corresponds to location $\boldsymbol{u}_k^{-1}(\boldsymbol{x})$ in $\boldsymbol{f}_k$. Hence, our model easily provides the solution $\boldsymbol{u}^*(\boldsymbol{x}) = \boldsymbol{u}_1^{-1}\big(\boldsymbol{u}_2(\boldsymbol{x})\big)$. See Figure 5 for some examples.

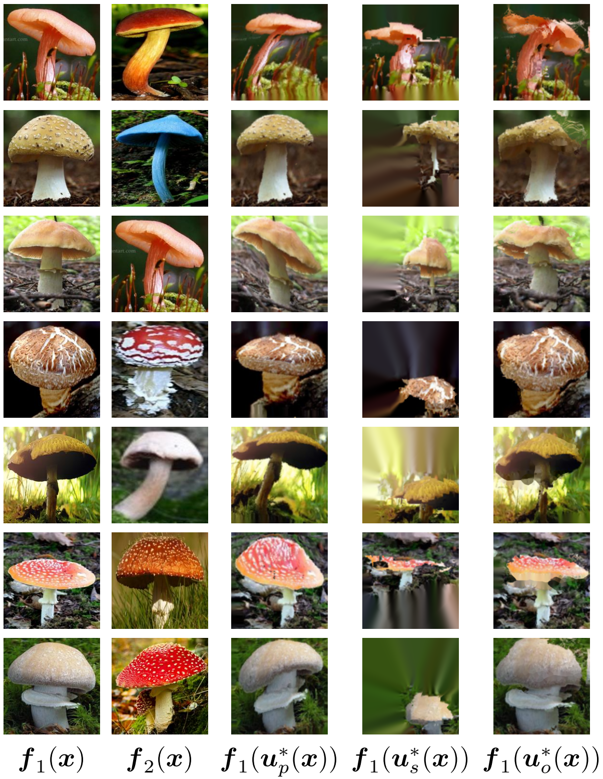

$\boldsymbol{f}_1(\boldsymbol{x})$
$\boldsymbol{f}_2(\boldsymbol{x})$
$\boldsymbol{f}_1(\boldsymbol{u}_p^*(\boldsymbol{x}))$
$\boldsymbol{f}_1(\boldsymbol{u}_s^*(\boldsymbol{x}))$
$\boldsymbol{f}_1(\boldsymbol{u}_o^*(\boldsymbol{x}))$

Figure 5. Example alignments by our proposed method $\boldsymbol{u}_p^*(\boldsymbol{x})$ against SIFT Flow $\boldsymbol{u}_s^*(\boldsymbol{x})$ and Optical Flow $\boldsymbol{u}_o^*(\boldsymbol{x})$.

5.2. Image Morphing

The goal of morphing is to gradually convert an image $\boldsymbol{f}_1(\boldsymbol{x})$ to another image $\boldsymbol{f}_2(\boldsymbol{x})$, such that the transition looks natural, e.g. without tearing apart the objects or exhibiting other artifacts. There are two conditions required for high quality morphing. First, a meaningful correspondence is needed between the pixels of $\boldsymbol{f}_1(\boldsymbol{x})$ and pixels of $\boldsymbol{f}_2(\boldsymbol{x})$. This determines where in $\boldsymbol{f}_2(\boldsymbol{x})$ each pixel of $\boldsymbol{f}_1(\boldsymbol{x})$ should land at. For example, when morphing a pair of mushroom images, we want to morph a cap to the other's cap, and not to the other's stem. The other important factor is that intermediate images must look natural. This requires knowing the space of images that $\boldsymbol{f}_1(\boldsymbol{x})$ and $\boldsymbol{f}_2(\boldsymbol{x})$ live in, so that the intermediate images are maintained in that space.

Both of these conditions are difficult to fulfill when $\boldsymbol{f}_1(\boldsymbol{x})$ and $\boldsymbol{f}_2(\boldsymbol{x})$ are not similar. Our proposed scheme, however, can benefit both of these conditions . First, estimating the low-dimensional representation from a collection of images $\mathcal{F}$ restricts the space of transformations to those needed for construction of $\mathcal{F}$. Second, while finding correspondence across two dissimilar images is difficult or even not well-defined, the collective relationship of ele-

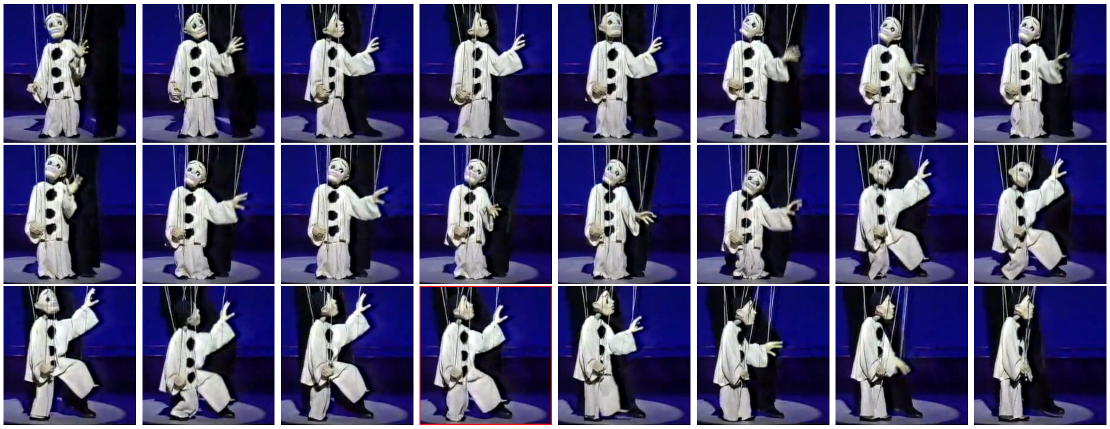


Figure 4. Some images from the puppet sequence.

ments in $\mathcal{F}$ can guide how $\boldsymbol{f}_1(\boldsymbol{x})$ and $\boldsymbol{f}_2(\boldsymbol{x})$ are related in particular.

We now discuss how the learned model can be used for morphing. Remember that $\boldsymbol{g}_k(\boldsymbol{x}) = \boldsymbol{A}_k^{-1}\boldsymbol{f}_k\big(\boldsymbol{u}_k(\boldsymbol{x})\big) - \boldsymbol{b}_k$ and thus for any $\boldsymbol{x}$, the location $\boldsymbol{u}_j^{-1}(\boldsymbol{x})$ in $\boldsymbol{f}_j$ corresponds to location $\boldsymbol{u}_k^{-1}(\boldsymbol{x})$ in $\boldsymbol{f}_k$. We can make this correspondence gradual by introducing a time parameter t and varying it from 0 to 1,

$$\boldsymbol{f}_{\text{morph}}\big((1-t)\boldsymbol{u}_1^{-1}(\boldsymbol{x})+t\boldsymbol{u}_2^{-1}(\boldsymbol{x})\big) = (1-t)\boldsymbol{f}_1(\boldsymbol{x})+t\boldsymbol{f}_2(\boldsymbol{x})\,.$$

Observe that at $t = 0$ we have $\boldsymbol{f}_{\text{morph}}(\boldsymbol{u}_1^{-1}(\boldsymbol{x})) = \boldsymbol{f}_1(\boldsymbol{x})$ and at $t = 1$ we have $\boldsymbol{f}_{\text{morph}}(\boldsymbol{u}_2^{-1}(\boldsymbol{x})) = \boldsymbol{f}_2(\boldsymbol{x})$. By varying t between 0 and 1 we achieve morphing while the deformation is constructed from the subspace-restricted learned deformations $\{\boldsymbol{u}_k(\boldsymbol{x})\}_{k=1}^n$. Obviously, since the first two methods are inferior in establishing desired correspondence, as shown in scene alignment section, they also fail for morphing.

5.3. Image Browsing

Consider browsing a large collection of images $\mathcal{F}$ to find a specific item, or to learn about variations across the images in the dataset. Doing these tasks by sequentially navigating through an unordered sequence of images like $\mathcal{F}$ is tedious and inefficient. Instead, we would like to browse an ordered sequence, where the ordering of each sequence is associated with some dominant structure in the data $\mathcal{F}$.

Our proposed method naturally provides a solution to this problem by discovering a low dimensional representation of $\mathcal{F}$. In fact, we can order the elements of $\mathcal{F}$ based on the values they attain when projected to a subspace axis.

For example, consider Weizmann horse dataset [3]. Let $d_g = 1$ and $d_u = 6$. Our method discovers the basis shown in Figure 6. Similar to mushroom example presented earlier, the subspace axes seem to be semantically meaningful.

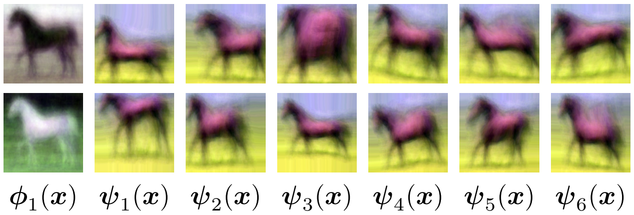

$\boldsymbol{\phi}_1(\boldsymbol{x})$
$\boldsymbol{\psi}_1(\boldsymbol{x})$
$\boldsymbol{\psi}_2(\boldsymbol{x})$
$\boldsymbol{\psi}_3(\boldsymbol{x})$
$\boldsymbol{\psi}_4(\boldsymbol{x})$
$\boldsymbol{\psi}_5(\boldsymbol{x})$
$\boldsymbol{\psi}_6(\boldsymbol{x})$

Figure 6. The effect of moving along each basis function in negative (top) and positive (bottom) directions.

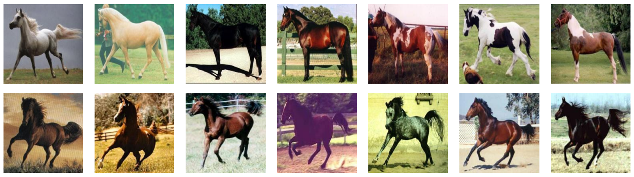


Figure 7. Seven images from $\mathcal{F}$ with smallest (top) and largest (bottom) projections $\boldsymbol{u}_k$ onto $\boldsymbol{\psi}_5$.

For example $\boldsymbol{\psi}_5(\boldsymbol{x})$ turns a horse from profile view to head facing camera. We can now assign to k'th element of $\mathcal{F}$ a scalar c_k obtained by projecting $\boldsymbol{u}_k(\boldsymbol{x})$ onto $\boldsymbol{\psi}_5(\boldsymbol{x})$. This allows to order images in $\mathcal{F}$ based on their associated projection values $\{c_q\}$ for $q = 1, \ldots, d_u$ (see Figure 7).

5.4. Articulation Learning

Consider an articulated object appearing in different poses in a set of images $\mathcal{F}$. The goal of articulation learning is to provide a few parameters, by which you can manipulate the object. We show that the proposed framework can be used to achieve this goal at a reasonable quality.

We used the Youtube video of a puppet as the input. We extracted a portion of the video and sampled 100 frames from it to construct $\mathcal{F}$ (see Figure 4). Although these images were taken from a video with known image sequence order, our algorithm does not rely on this information. Therefore, the concept described here is also applicable to an unordered image sequence.

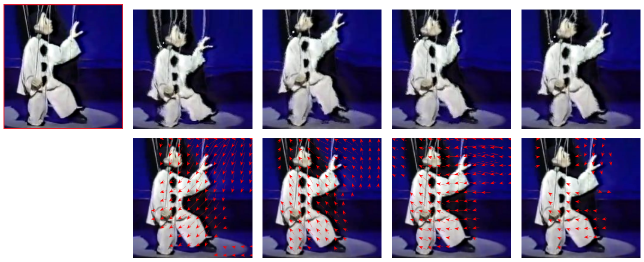


Figure 8. Novel synthesized poses. Top Left: Original image. Top Middle: Manipulation via $\boldsymbol{\psi}_1(\boldsymbol{x})$. Top Right: Manipulation via $\boldsymbol{\psi}_2(\boldsymbol{x})$. Bottom: Corresponding Motion Fields.

By applying our algorithm to this set, we obtain the set $\{\boldsymbol{u}_k(\boldsymbol{x})\}_{k=1}^n$, and consequently the inverse maps $\{\boldsymbol{u}_k^{-1}(\boldsymbol{x})\}_{k=1}^n$, as well as the basis functions $\{\boldsymbol{\psi}_q(\boldsymbol{x})\}_{q=1}^{d_u}$. The geometric deformations of the puppet are captured by the learned subspace $\mathscr{U}$. Hence, the coefficient c_q each basis function $\boldsymbol{\psi}_q(\boldsymbol{x})$ serves as a parameter which can manipulate the puppet along its dominant deformations within the sequence.

Now suppose we want to manipulate the puppet at the k'th image of $\mathcal{F}$, e.g. the one highlighted by red in Figure 4. We can achieve that using the following equation,

$$\boldsymbol{f}^{\dagger}_k(\boldsymbol{x}\,;c_1,\ldots,c_{d_u}) = \boldsymbol{f}_k\Big(\boldsymbol{u}_k^{-1}\big(\boldsymbol{u}_k(\boldsymbol{x})+\sum_{q=1}^{d_u} c_q\boldsymbol{\psi}_q(\boldsymbol{x})\big)\Big)\,,$$

where $\boldsymbol{f}^{\dagger}_k$ is the updated pose of $\boldsymbol{f}_k$, and c_q determines the contribution of the q'th basis function. Note that at the origin, i.e. $c_q = 0$ for all $q = 1,\ldots,d_u$, we have $\boldsymbol{f}^{\dagger}_k(\boldsymbol{x}\,;c_1,\ldots,c_{d_u}) = \boldsymbol{f}_k(\boldsymbol{x})$.

Figure 8 shows images synthesized this way.We can capture up/down and left/right movements of the hand. These give us completely new images; the synthesized images are novel and do not exist in the original set.

6. Conclusion

In this work we presented a simple compositional model of color, shape, and appearance to approximate image sets. The model is regularized by having shape and appearance representations be on a low-dimensional subspace, and having color be a global shift and rotation. The learned representation was applied to establish dense correspondence across instances of some object categories. The proposed method significantly outperforms robust optical flow and SIFT flow.

An interesting observation in our experiments is that the dimensionality that worked jointly well on all of our data is larger for shape relative to appearance. This is well aligned with recent theories that claim the sample complexity of visual learning is mainly due to the pose variations, not the appearance [19].

Acknowledgment

This research is partially funded by Shell Research.

References

[1] A. Asthana, C. Sanderson, T. D. Gedeon, and R. Gocke. Learning-based face synthesis for pose-robust recognition from single image. In *BMVC*, 2009. 1

[2] M. J. Black and A. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *IJCV*, 26(1):63–84, 1998. 3

[3] E. Borenstein and S. Ullman. Class-specific, top-down segmentation. ECCV '02, pages 109–124, 2002. 5, 7

[4] P. Brivio, M. Tarini, and P. Cignoni. Browsing large image datasets through voronoi diagrams. *IEEE Trans. Vis. Comput. Graph.*, 16(6):1261–1270, 2010. 1

[5] A. Bruhn, J. Weickert, and C. Schnörr. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. 61(3):211–231, 2005. 4

[6] G. Carlsson, T. Ishkhanov, V. D. Silva, and A. Zomorodian. On the local behavior of spaces of natural images. *IJCV*, 2006. 1

[7] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. In *ECCV (2)*, pages 484–498, 1998. 1, 3

[8] B. J. Frey and N. Jojic. Transformed component analysis: Joint estimation of spatial transformations and image components. In *ICCV*, pages 1190–1196, 1999. 1

[9] B. K. P. Horn and B. G. Schunk. Determining Optical Flow. *Artificial Intelligence*, 17:185–203, 1981. 1

[10] M. J. Jones and T. Poggio. Multidimensional morphable models: A framework for representing and matching object classes. *IJCV*, 29(2):107–131, 1998. 1

[11] W. Kabsch. A Solution for the Best Rotation to Relate Two Sets of Vectors. *Acta Crystallographica*, 32:922–923, 1976. 4

[12] I. Kemelmacher-Shlizerman and S. M. Seitz. Collection flow. In *CVPR*, pages 1792–1799, 2012. 1

[13] S. Lee, G. Wolberg, and S. Y. Shin. Polymorph: Morphing among multiple images. *IEEE Computer Graphics and Applications*, 18(1):58–71, 1998. 1

[14] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. T. Freeman. Sift flow: Dense correspondence across different scenes. In *ECCV*, pages 28–42, 2008. 1, 5, 6

[15] H. Mobahi, S. Rao, and Y. Ma. Data-driven image completion by image patch subspaces. In *Picture Coding Symposium*, 2009. 1

[16] H. Mobahi, C. L. Zitnick, and Y. Ma. Seeing through the blur. In *CVPR*, 2012. 1

[17] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, June 1996. 1

[18] R. Pless and R. Souvenir. A survey of manifold learning for images. *IPSJ Transactions on Computer Vision and Applications*, 1:83–94, March 2009. 1

[19] T. Poggio, J. Leibo, J. Mutch, and L. Rosasco. The computational magic of the ventral stream: Towards a theory. Technical Report MIT-CSAIL-TR-2012-035, Dec. 2012. 8

[20] D. A. Ross, D. Tarlow, and R. S. Zemel. Learning articulated structure and motion. *Int. J. Comput. Vision*, 88(2):214–237, 2010. 1

[21] D. L. Ruderman, T. W. Cronin, and C.-C. Chiao. Statistics of cone responses to natural images: Implications for visual coding. *JOSA A*, 15(8):2036–2045, 1998. 1, 3

[22] D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. In *CVPR*, pages 2432–2439, 2010. 1, 4, 5

[23] S. C. Zhu, Y. N. Wu, and D. Mumford. Minimax entropy principle and its application to texture modeling. *Neural Computation*, 9(8):1627–1660, 1997. 1