Published in final edited form as:

*IEEE Trans Pattern Anal Mach Intell*. 2017 December ; 39(12): 2496–2509. doi:10.1109/TPAMI.2016.2646685.

# Transformations Based on Continuous Piecewise-Affine Velocity Fields

**Oren Freifeld [Member, IEEE]**,
Department of Computer Science, Ben-Gurion University Be'er Sheva 84105, Israel

**Søren Hauberg [Member, IEEE]**,
Section for Cognitive Systems, DTU Compute, Kongens Lyngby 2800, Denmark

**Kayhan Batmanghelich [Member, IEEE]**, and
Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology, Cambridge, MA 02319

**Jonn W. Fisher III [Member, IEEE]**
Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology, Cambridge, MA 02319

## Abstract

We propose novel finite-dimensional spaces of well-behaved $\mathbb{R}^n \to \mathbb{R}^n$ transformations. The latter are obtained by (fast and highly-accurate) integration of continuous piecewise-affine velocity fields. The proposed method is simple yet highly expressive, effortlessly handles optional constraints (e.g., volume preservation and/or boundary conditions), and supports convenient modeling choices such as smoothing priors and coarse-to-fine analysis. Importantly, the proposed approach, partly due to its rapid likelihood evaluations and partly due to its other properties, facilitates tractable inference over rich transformation spaces, including using Markov-Chain Monte-Carlo methods. Its applications include, but are not limited to: monotonic regression (more generally, optimization over monotonic functions); modeling cumulative distribution functions or histograms; time-warping; image warping; image registration; real-time diffeomorphic image editing; data augmentation for image classifiers. Our GPU-based code is publicly available.

### Index Terms

Spatial transformations; continuous piecewise-affine velocity fields; diffeomorphisms; tessellations; priors; MCMC

## 1 Introduction

Diffeomorphisms are important in many fields such as computer vision, medical imaging, graphics, and robotics. Unfortunately, current representations of *highly-expressive* diffeomorphism spaces are overly complicated. Thus, despite their potential and mathematical beauty, their applicability is limited, especially in large datasets or when computing time is restricted. Moreover, in such spaces, owing to their complexity, using powerful inference tools, e.g., Monte Carlo Markov Chain (MCMC), still presents challenges, although some encouraging recent progress has been made in this active field of research (e.g., [1], [2], [3]). Lastly, seemingly-formidable mathematical preliminaries render these spaces accessible to only a small group of geometry experts. This hinders the exchange of ideas between communities and unnecessarily limits the potential impact of diffeomorphism-based methods; e.g., while certain machine-learning areas can benefit from such methods, little work has been done in this direction, partly since practical computational tools have yet to become available.

Motivated by practicalities of probabilistic modeling and statistical inference as well as a desire to make diffeomorphisms broadly accessible, in this work, which expands [4], we propose a *representation* that (as we will show in Section 4) combines simplicity, expressiveness, and efficiency. Particularly, *we propose new spaces of transformations that are based on (fast, highly-accurate) integration of Continuous Piecewise-Affine (CPA) velocity fields*. Importantly, as we will show, their benefits go beyond speed and accuracy.

Possible applications of the proposed representation are numerous, as we demonstrate here with: image editing and shape manipulation; unconstrained optimization over monotonic functions; modeling of Cumulative Distribution Functions (CDFs) and histograms with order-preserving geometry; time warping; image registration; landmark-based image warping/animation; “prettifying” results of (non-diffeomorphic) dense-correspondence tools. Moreover, we recently used the proposed representation as a key component in a *learned* data-augmentation scheme [5], improving the results of image classifiers. Finally, our code is available at https://github.com/freifeld/cpabDiffeo.

## 2 Related Work

### Pattern Theory and Differential Geometry

Representing objects via transformations acting on them is a *Pattern-Theoretic* cornerstone [6]. Our work is influenced by many impressive works in this field, primarily in the geometry-oriented areas of computer vision and medical imaging. Due to space limits, we can mention only a few: [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19]. *Most* of these works are based on complicated, usually $\infty$-dimensional, spaces, and the associated representations and computations are, in practice, discretized and/or otherwise approximated. We take a more practical approach and start from a finite-dimensional space, in which discretizing the representation is unneeded, while computations require no approximations in the 1D case and almost no approximations in higher dimensions. The result is a simple, efficient, and practical machinery for working with a rich space of diffeomorphisms. The rapidness and high accuracy of Algorithm 1 (see Section 4), together with the fact that the machinery is easily implemented on GPU, let us evaluate transformations fast. This, in turn, together with the fact that the representation effortlessly supports coarse-to-fine analysis and the use of smoothness priors, allows the use of general-purpose powerful inference methods such as MCMC. See also other recent approaches for MCMC on diffeomorphisms, e.g., [1], [2], [3].

In applications involving landmark pairs, most methods above can deal with only a small number of landmarks as their inference complexity grows super-linearly with the number of landmarks. Consequently, they cannot leverage the success of popular tools for dense-correspondence extraction (e.g., [20], [21]), highlighting a disconnect with the larger computer-vision community. Our algorithms have linear complexity and are (embarrassingly) parallelizable, yielding sub-linear running times in practice.

Note that our representation, despite the fact it involves tessellations, is *not* not based on control points (such as, e.g., [2], [19]); e.g., in cases where we either impose volume preservation or use type-II tessellations (see Section 4), one cannot simply define arbitrary velocities at the tessellation vertices as there are too few degrees of freedom.

## PA and CPA Affine Maps in Computer Vision

Cootes et al. [22] use PA transformations. The latter, while simple and efficient, are neither continuous nor invertible. Several other works use CPA transformations (e.g.,[23]; see also [24]), sometimes with additional constrains (e.g., [25]). Common to those works is the *direct* use of PA (or CPA) maps as transformations. Those are not differentiable (and are often not invertible), hence not diffeomorphisms. In contrast, by *integrating* CPA *velocity fields*, we obtain transformations–which themselves are *not* PA (hence not CPA)–that are (orientation-preserving) diffeomorphisms. Lin et al. [26] work with CPA velocity fields but, rather than integrating them, they use them to model motion pattens.

Closer to ours is the elegant *log-Euclidean polyaffine* method proposed by Arsigny et al. [27] (see also [28], [29]) that, similar to ours, uses finitely-many affine building blocks to build flexible velocity fields. They use a spatial averaging with smoothly-decaying weights of the blocks to ensure a smooth velocity field whose integration yields trajectories that define a diffeomorphism. However, as their integral has no closed form they must use an approximation throughout; i.e. they must numerically approximate the entire trajectory. This is computationally demanding and can cause substantial approximation errors. Their method is based on approximating the integral of a weighted sum of affine velocity fields via a weighted sum of the affine diffeomorphisms associated with these fields. To reduce errors, they divide the field by a large $2^{\#\text{steps}}$ as the approximation holds only for near-zero velocity. To keep the number of steps not too large, they smartly generalize the scaling-and-squaring method. The result is, however, still exact only if a single affine component is used. For expressiveness, however, a larger number is needed, and accuracy drops. It is thus unsurprising they focus on a small number of affine components, and that to achieve reasonable timings, they use their method only in the last stage of inference; till then they resort to a non-diffeomorphic fusion. In contrast, we use affine building blocks in a different way; i.e., we use them in a piecewise manner, but, using linear constraints, force them to yield everywhere-continuous velocity fields. This seemingly-subtle difference has profound implications. In 1D, it yields a closed-form integration; in higher dimensions it lets us integrate almost the entire trajectory in closed form using large (hence few) steps which are exact, and only in small portions of the trajectory do we resort to a numerical solver. This virtually eliminates numerical issues and allows construction of substantially more expressive transformations; e.g., Arsigny et al. [27] report using 7 affine blocks, while we routinely use tens or hundreds while neither accuracy nor computational cost becomes an issue. Though this indirect comparison of *integration methods* (as their method is inapplicable to our velocity fields and vice versa, a direct comparison is impossible) favors ours, it is just part of the story. Our *representation* has additional advantages over theirs, including simplicity, better suitability for a GPU implementation, trivial handling of boundary constraints and volume preservation, and simpler encoding of statistical priors and coarse-to-fine analysis.

## Discrete Representations and Approximations

Allassonnière et al. [13] efficiently *approximate* diffeomorphisms. Unlike our transformations, their non-differentiable transformations are not diffeomorphisms. While we focus on a general-purpose representation, diffeomorphic demons [16] is a registration method, popular due to its speed, based on discretely-defined fields. As its authors note, these may be inconsistent with a diffeomorphic framework and may not preserve orientation. They also cannot easily impose volume preservation, though some success was reported [30]. Also, a computer representation of a discrete-field sequence needs plenty of memory. These issues can be obviated by adapting their method to use our compact and continuously-defined fields. More generally, approximations based on discretely-defined fields and/or discrete diffeomorphisms are widely used, e.g., in medical imaging [16], [31], robotics [32], [33], [34], geometric modeling [35] and fluid dynamics [36]. Unlike these works, both our fields and transformations are continuously-defined and more compact.

## Statistics on Manifolds and Tangent Spaces

Like many authors (including of some aforementioned works), we handle the nonlinearity of a space via the linearity of the tangent space at the identity [11], [27], [29], [37], [38], [39], [40], [41]. Other tools (not explored here) for statistics on manifolds that either use other tangent spaces or work on the manifold itself [42], [43], [44], [45], [46], [47], [48] may also be applied to our spaces. Particularly, parallel-transport tools may be especially relevant here [49], [50], [51], [52], [53], [54], [55].

## CDF/Histogram Modeling

In modeling distributions, it is better to work with *cummulative distribution functions* over densities since the latter might not exist and inter-density $L_p$ (or sphere-based) distances can be arbitrarily large even if their *probability measures* are essentially the same. One approach to CDF representation uses $p$-Wasserstein spaces, usually $p \in \{1, 2\}$, the $p = 1$ case is tied to Earth Mover's Distance [56]. A limitation of this approach is that it needs bounded $p$th-moments and hard computations. 1-Wasserstein methods also lack easy synthesis of new points that are valid histograms/CDFs. While it is less of an issue for spherical methods [57], they suffer from two issues. 1) They do not respect the ordering of the bins or of $\mathbb{R}$. While, for histograms, bin ordering is *sometimes* immaterial, the ordering of $\mathbb{R}$ matters. 2) Large moves on the sphere lead to CDFs/histograms with negative values. The problems above do not exist in our representation.

### Image Warping and Shape Manipulation

Related to ours are works on image warping ([58], [59], [60], [61], [62]) and shape manipulation (e.g., [63]). Unlike most methods, ours is fast, invertible and handles constraints effortlessly.

### Benefits of the Proposed Representation

To summarize, existing spaces of diffeomorphisms offer only subsets of the following list: 1) high expressiveness; 2) ease of implementation; 3) modest mathematical preliminaries (basic linear algebra and ODE); 4) ease of handling optional constraints (e.g., volume preservation); 5) convenient modeling choices (coarse to fine, easy-to-use smoothness priors); 6) finite dimensionality; 7) fast *and* highly-accurate computations. These benefits, especially the last three, render more tractable the use of inference tools that are usually too expensive in the context of rich diffeomorphisms.

## 3 High-Level Summary

The section provides a summary of the proposed representation, laying the ground for the formal treatment (Section 4). Let Ω be either $\mathbb{R}^n$ or a certain type, to be defined later, of a proper subset of $\mathbb{R}^n$ (i.e., $\Omega \subsetneq \mathbb{R}^n$). A popular way to obtain a diffeomorphism, $T : \Omega \to \Omega$, is via the integration of velocity fields; see Fig. 1a. The choice of velocity-field family affects the dimensionality, structure, and expressiveness of the space of the resulting diffeomorphisms, as well as the accuracy and computational complexity of the integration. Thus, this choice crucially affects which probabilistic models can be used and the tractability of the statistical inference.

### CPA Velocity Fields

We base our representation on spaces of $\Omega \to \mathbb{R}^n$ CPA velocity fields (Fig. 1). The term 'piecewise' is w.r.t. a certain *tessellation* (Section 4.1), denoted by □. Let $□_{\Omega, □}$ be such a space. While $□_{\Omega, □}$ depends on Ω and □, we will usually notationally suppress these dependencies, and will just write □. One appeal of these spaces is that they are *finite-dimensional* and *linear* (although their elements, i.e., the velocity fields, are usually nonlinear). Let $d = \mathrm{dim}(□)$. The spaces $\mathbb{R}^d$ and □ are identified with each other (as we will explain in Section 4.2, Eq. (11)), where every $\theta \in \mathbb{R}^d$ is identified with exactly one element of □, denoted by $v^\theta$, and vise versa. Symbolically, we write

$$\theta \leftrightarrow v^\theta \text{ where } v^\theta □ □, \theta □ □^d. \quad (1)$$

Likewise, $\theta + \theta' \leftrightarrow v^\theta + v^{\theta'} □ v^{\theta + \theta'}$ and $\alpha\theta \leftrightarrow \alpha v^\theta \triangleq v^{\alpha\theta}$ where $\theta; \theta' \in \mathbb{R}^d$ and $\alpha \in \mathbb{R}$. Note that $d$ depends on □ (and typically grows with $n$). A finer □ implies a higher $d$ and richer velocity fields and vice versa (Figs. 4, 2, and 3).

**Remark 1—**There are many finite-dimensional linear spaces of continuous velocity fields (e.g., [27] or other spaces based on splines). We will show that CPA spaces, however, have additional useful properties in our context.

### From CPA Velocity Fields to Trajectories

Modulo a detail (to be explained in Section 4.4) related to the case $\Omega \subsetneq \mathbb{R}^n$, any continuous $\Omega \to \mathbb{R}^n$ velocity field, whether Piecewise-Affine (PA) or not, defines differentiable $\mathbb{R} \to \Omega$ *trajectories*. If $x \in \Omega$ then $v^\theta \in \mathcal{V}$ defines a trajectory, $t \mapsto \phi^\theta(x, t)$, such that $\phi^\theta(x, 0) = x$ and $\phi^\theta(x, t)$ solves the *integral equation*

$$\phi^\theta(x,t) = x + \int_0^t v^\theta(\phi^\theta(x,\tau))d\tau \quad \text{where } v^\theta \in \mathcal{V}. \qquad (2)$$

The equivalent ODE (with an initial condition $x$) is

$$d\phi^\theta(x,t)/dt = v^\theta(\phi^\theta(x,t)). \qquad (3)$$

**Remark 2—**Eq. (2), whose unknown $\phi^\theta(x, \cdot)$ is both inside and outside the integral, should not be confused with the piecewise-quadratic $\Omega \to \mathbb{R}^n$ map, $y \mapsto \int_{0_{n\times 1}}^{y} v^\theta(x)dx$. The latter, a popular tool in computer-vision [24] and numerical analysis, is unrelated to our work. Particularly, both $x \mapsto \phi^\theta(x, t)$ and $t \mapsto \phi^\theta(x, t)$ are *not* piecewise quadratic.

### CPA-Based (CPAB) Transformations

Modulo that detail, any continuous $\Omega \to \mathbb{R}^n$ velocity field, whether PA or not, defines a *transformation*; i.e., a map whose input and output are viewed as points, not vectors. Letting $x$ vary and fixing $t$, $x \mapsto \phi^\theta(x, t)$ is an $\Omega \to \Omega$ transformation. Without loss of generality (Section 4), we may set $t = 1$ and define

$$T^\theta(\cdot) \triangleq \phi^\theta(\cdot, 1), \quad \theta \in \mathbb{R}^d. \qquad (4)$$

Since we integrate CPA velocity fields, we coin our transformations CPA-*Based*.

**Remark 3—**CPAB transformations are *not* CPA (unless the CPA field is affine): while $T^\theta$ is continuous, it is not PA.

The notation $T^\theta = \exp(v^\theta)$ indicates that $T^\theta$ is defined by integrating $v^\theta \leftrightarrow \theta$ for $t = 1$ (see Section 4.9). We let

$$M \triangleq \exp(\mathcal{V}) \triangleq \left\{\exp(v^\theta) : v^\theta \in \mathcal{V}\right\} \qquad (5)$$

denote the space of CPAB transformations, where we notationally suppressed that both $\mathcal{M}$ and □ depend on both Ω and □ (i.e., more formally we should write $\mathcal{M}_{\Omega,\,□}$). Note that $\mathcal{M}$ is nonlinear: i.e., both $T^{\theta} + T^{\theta'}$ and $\alpha T^{\theta}$ are usually not in $\mathcal{M}$. The linearity of □, together with $\exp: \mathcal{M} \to □$, however, provides a convenient way to handle this nonlinearity.

### CPAB Transformations are Well Behaved

E.g., they are *diffeomorphisms*; moreover, $(T^{\theta})^{-1} \in \mathcal{M}$ and the inversion is simple. Appealingly, useful subsets of $\mathcal{M}$ (e.g., volume-preserving CPAB transformations) are easily obtained via linear subspaces of □ (we will return to this point in Section 4).

**Remark 4—**$x \mapsto \phi^{\theta}(x, t = 1)$ should not be confused with (the non-diffeomorphism, parametric optical-flow-like representation) $x \mapsto x + v^{\theta}(x)$, the latter being only a *Taylor approximation* of the former. Thus, the way we utilize flexible parametrized vector fields is different from, e.g., [64], [65].

### Integration of CPA Velocity Fields

Integral equations usually lack analytic solutions. Since CPA velocity fields are Lipschitz continuous and almost-everywhere smooth, generic integration solvers are quite effective for them. However, we can do even better. *One of our contributions is showing that integration of such fields is given in either closed form* ($n = 1$) *or almost closed form* ($n > 1$). Besides its obvious pluses (accuracy, computing time), this analytic solution makes it easier to interpret the resulting trajectories/transformation and is key to the theorems in Section 4.

### A Specialized Numerical Solver

There is a shortcoming to that solution: as we will explain after Theorem 2, it requires tedious bookkeeping (if $n > 1$) and invoking certain routines (easy if $n = 1$ but hard if $n > 1$). This is especially a hurdle with GPU. We thus propose a practical alternative, *a specialized solver for integrating CPA velocity fields*, which is faster *and* more accurate than non-specialized solvers.

### Convenient Modeling, Tractable Inference

Smoothness priors on $\mathcal{M}$ are easy to build and use. As is common with nonlinear spaces of nonlinear transformations, the nonlinearity of $(\theta, x) \mapsto T^{\theta}(x)$ prohibits closed forms for the posterior or Maximum Likelihood (ML) estimation. However, since our transformations are evaluated fast, we rapidly evaluate $\theta \mapsto p(\text{data}|T^{\theta})$. This, together with the relatively-modest dimensionality and ease of using priors, facilitates the use of inference methods that require multiple likelihood evaluations (e.g., MCMC). Lastly, spaces of CPA velocity fields support coarse-to-fine approaches.

## 4 The Mathematical Representation

Let Ω be a Cartesian product of $n$ compact intervals; i.e., Ω is a closed *n-orthotope*, also called a (closed) hyperrectangle. The lemmas/theorems below are proved in our supplemental material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2016.2646685, henceforth referred to as Sup. Mat., available online.

### 4.1 Tessellations

A (finite) tessellation, denoted by $□ = \{U_c\}_{c=1}^{N_□}$ (where $N_□$ is a positive integer), is a set of $N_□$ closed subsets of Ω, also called *cells*, such that their union is Ω and the intersection of any pair of adjacent cells is their shared border (see Figs. 2 and 3). Henceforth we will always assume the cells are *convex n-polytopes* (namely, a subset of $\mathbb{R}^n$ whose sides are flat). We define 3 cell types. Type-I cells have $n + 1$ vertices; i.e., intervals (if $n = 1$), triangles ($n = 2$), tetrahedral ($n = 3$), etc. Type-II cells are *n-orthotopes* (i.e., hyperrectangles). Note intervals fit both these types. Type-III includes all other convex $n$-polytopes. We say that □ is of type I (respectively, II, III) if all its cells are of type I (II, III). A type-II tessellation is called *regular* if its cells are evenly spaced along each of the $n$ dimensions. As we will show, type-I tessellations are special, having benefits possessed by neither type II nor III. Mathematically, there is no difference (for our purposes) between types II and III; all theoretical results below that apply to type-II cells also apply to type-III cells. Type-II cells, however, lead to faster computations and simpler implementations than both other types, especially when compared with type III. Thus, unless stated otherwise, we restrict our attention to types I and II. Let

$$□_{\text{vert}} = \{\xi : \xi \text{ is a vertex of some } U_c □ □\} \quad (6)$$

and let $N_V$ denote its cardinality; e.g., see Fig. 2 and Table 1. Henceforth we will view $□_{\text{vert}}$ as an ordered $N_V$-tuple; the ordering is arbitrary, but assumed fixed. Let $N_□' > N_□$. An $N_□'$-cell tessellation, $\mathrm{P}' = \{U_{c'}\}_{c'=1}^{N_{\mathrm{P}'}}$, is a *refinement* of □, a relation denoted $□ □ \mathrm{P}'$, if each $U_{c'} □ \mathrm{P}'$ is a (possibly-improper) subset of some $U_c □ □$.

**Algorithm 1**

Integrating $v^\theta □ □$. See Text for Details:

```
Input: $□ = \{C_c\}_{c=1}^{N_□}, (A_{1,\theta}, \dots, A_{N_□,\theta})$; $U$ = a sequence of $N_{\text{pts}}$ points in Ω; $t > 0$; $N_{\text{steps}}, n_{\text{steps}} □ □^+$
Output: $\phi^\theta(U,t) □ \{\phi^\theta(x,t)\}_{x □ U} □ Ω$
1:  $\Delta_t \leftarrow t/N_{\text{steps}}$; $\delta_t \leftarrow \Delta_t/n_{\text{steps}}$
2:  for $c □ \{1, \dots, N_□\}$ do in parallel
3:      $T_{A_{c,\theta},\Delta_t} \leftarrow \exp(\Delta_t A^{□}_{c,\theta})$
```

```
4:      for $x \in U$ do in parallel
5:        $x_0 \leftarrow x$
6:        for $i \in \{1, \ldots, N_{\text{steps}}\}$ do
7:          $\overset{□}{x}_{\text{temp}} = [x_{\text{temp}}^T \;\; 1]^T \leftarrow T_{A_{\gamma(x_{i-1}),\theta},\Delta_t}\overset{□}{x}_{i-1}$
8:          if $\gamma(x_{i-1}) == \gamma(x_{\text{temp}})$ then
9:            $x_i \leftarrow x_{\text{temp}}$ // analytic update
10:         else
              // inter-cell bdry crossed
11:           $x_i \leftarrow$ genericODEsolver(ini.con.=$x_{i-1}$, step size=$\delta_t$, $n_{\text{steps}}$, $v(□): x$ □ $A_{\gamma(x),\theta}\overset{□}{x}$)
12:       $(\phi)^{\theta}(x, t) \leftarrow x_i$
```

## 4.2 CPA Velocity Fields

Fix □, let $x \in \Omega$, and define the membership function $\gamma: \Omega \to \{1, \ldots, N_□\}$, $\gamma: x \mapsto \min\{c: x \in U_c\}$. i.e, if $x$ is not on an inter-cell border, then $\gamma(x) = c \Leftrightarrow x \in U_c$, while a border point is (arbitrarily) assigned to the cell of the lower index.

**Definition 1—**A map, $f: \Omega \to \mathbb{R}^n$, is called $PA(w.r.t.\ □)$ if $\left\{f|_{U_c}\right\}_{c=1}^{N_□}$ are *affine*; i.e., $f(x) = A_{\gamma(x)}\overset{□}{x}$ where

$$\overset{□}{x} □ \begin{bmatrix} x \\ 1 \end{bmatrix} □ □^{n+1},\ A_c □ □^{n \times (n+1)} □ c □ \{1, \ldots, N_□\}. \quad (7)$$

**Definition 2—**f is called CPA if it is continuous and PA.

**Fact 1—**If f is CPA w.r.t. □ it is CPA w.r.t. any P′ □ □.

A *vector field* $v$ (on Ω) is an $\Omega \to \mathbb{R}^n$ map viewed as the mapping of points to vectors; i.e., if $x \in \Omega$ then $v(x)$ is viewed as an $n$-dimensional "arrow". The terms *velocity field* and *vector field* will be used interchangeably.

**Definition 3—**A vector-field space is a linear space whose elements are vector fields.

Let $□'_{\Omega,□}$ and $□_{\Omega,□}$ be the spaces of PA and CPA velocity fields on Ω w.r.t. □. Note $□_{\Omega,□}$ □ $□'_{\Omega,□}$. We will often write □′ and □, suppressing the dependencies on Ω and □.

**Lemma 1—** □′ and □ are linear spaces, $D$ □ $\dim(□') = (n^2 + n) \times N_□$, $d$ □ $\dim(□) \le nN_v$ with equality if and only if □ is of type $I$, and $d \le D$ with equality if and only if $N_□ = 1$.

The values of $D$ and $d$ for the tessellations in Fig. 2 appear in Table 1. A generic element of $\square'$ is denoted by $v_A$ where $A \square (A_1, \ldots, A_{N_\square})$ consists of its associated matrices (see Eq. (7)). If $A \in \mathbb{R}^{n\times(n+1)}$, then $\text{vec}(A) \square \square^{n^2+n}$ is its *row-by-row* flattening to a column vector. Likewise, $\mathbf{v}\text{ec}(A) \square [(\text{vec}(A_1))^T \square (\text{vec}(A_{N_\square}))^T]^T \square \square^D$. Both vec(·) and **v**ec(·) are linear bijections. An inner product on $\square'$ is defined by $\left\langle v_{A_1}, v_{A_2} \right\rangle = \mathbf{v}\text{ec}(A_1)^T \mathbf{v}\text{ec}(A_2)$.

We now explain how to build *an* orthonormal basis for $\square$, how its elements are parametrized by $\theta \in \mathbb{R}^d$, and how PA velocity fields are projected onto $\square$.

**Lemma 2—**An element of $\square$ is any $v_A \square \square'$ such that **v**ec($A$) satisfies a linear system of constraints, denoted by $L\,\mathbf{v}\text{ec}(A) = 0$.

**Proof—**For concreteness, we prove it for $n = 2$ and type-I tessellations, the other cases being similar. Let $v_A \square \square'$. While $v_A$ is continuous on every cell, it is (usually) discontinuous on cell boundaries. Consider two adjacent cells, $U_i$ and $U_j$. Let $x_a$ and $x_b$ be their 2 shared vertices and let $A_i$ and $A_j$ denote the corresponding 2 × 3 matrices. Continuity of $v_A$ at $x_a$ implies 2 (more generally, $n$) linear constraints on $A_i$ and $A_j$. Similarly, continuity at $x_b$ implies another constraint pair. Thus, the continuity at both $x_a$ and $x_b$ implies the following 4 linear constraints:

$$\underbrace{\begin{bmatrix} \overset{\square}{x}_a^T & 0_{1\times3} & -\overset{\square}{x}_a^T & 0_{1\times3} \\ 0_{1\times3} & \overset{\square}{x}_a^T & 0_{1\times3} & -\overset{\square}{x}_a^T \\ \overset{\square}{x}_b^T & 0_{1\times3} & -\overset{\square}{x}_b^T & 0_{1\times3} \\ 0_{1\times3} & \overset{\square}{x}_b^T & 0_{1\times3} & -\overset{\square}{x}_b^T \end{bmatrix}}_{4\times12} \underbrace{\begin{bmatrix} \text{vec}(A_i) \\ \hline \text{vec}(A_j) \end{bmatrix}}_{12\times1} = 0_{4\times1}. \quad (8)$$

Continuity at $x_a$ and $x_b$ implies continuity throughout their join, i.e., $A_i\overset{\square}{x}_a = A_j\overset{\square}{x}_a$ and $A_i\overset{\square}{x}_b = A_j\overset{\square}{x}_b$ imply ($\forall\lambda \in [0, 1]$)

$$A_i(\lambda\overset{\square}{x}_a + (1-\lambda)\overset{\square}{x}_b) = A_j(\lambda\overset{\square}{x}_a + (1-\lambda)\overset{\square}{x}_b). \quad (9)$$

Any $v_A$ whose $A_i$ and $A_j$ satisfy Eq. (9) is thus continuous on $U_i \cup U_j$. Similar constraints for other pairs of adjacent cells can be stacked together in an analogous equation:

$$\underset{4N_e\times6N_\square}{L}\ \underset{6N_\square\times1}{\mathbf{v}\text{ec}(A)} = 0_{4N_e\times1} \quad (10)$$

where $N_e$ is the number of shared line segments in □ and $L$ is the constraint matrix. Any $v_A$ whose $A$ satisfies Eq. (10) is thus everywhere continuous. We conclude that the null space of $L$, denoted by null($L$), coincides with □. □

Let the columns of $B = [B_1 \dots B_d] \in \mathbb{R}^{D \times d}$ denote the particular orthonormal basis of null($L$) which is obtained via SVD of $L$. Let $\theta = [\theta_1 \dots \theta_d]^T \in \mathbb{R}^d$. If $\mathbf{vec}(A) = B\theta$ then $v_A : \Omega \to \mathbb{R}^n$ is CPA. Particularly, for every $j \in \{1, \dots, d\}$, $v_{\mathbf{vec}^{-1}(B_j)} : \Omega \to □^n$ is CPA and $\left\{ v_{\mathbf{vec}^{-1}(B_j)} \right\}_{j=1}^{d}$ is *a* basis for □. For a visualization of this basis, see Sup. Mat., available online Regardless of whether $v_A$ □ □′ is CPA or not, $v_{\mathbf{vec}^{-1}(BB^T \mathbf{vec}(A))}$, its projection on □, is CPA. When earlier we denoted a generic element of □ by $v^\theta$ we meant that $\theta$ stands for the coefficients w.r.t. $\left\{ v_{\mathbf{vec}^{-1}(B_j)} \right\}_{j=1}^{d}$:

$$v^\theta(x) = A_{\gamma(x),\theta}\tilde{x} = \sum_{j=1}^{d} \theta_j v_{\mathbf{vec}^{-1}(B_j)} \tilde{x} \quad (11)$$

where $A_\theta$ □ $(A_{1,\theta}, \dots, A_{N_□,\theta})$ □ $\mathbf{vec}^{-1}(\sum_{j=1}^{d} \theta_j B_j)$. This basis is orthonormal w.r.t. the inner product

$$\langle ⋅, ⋅\rangle_B : □ \times □ \to □, \ (v^{\theta_1}, v^{\theta_2}) \ □ \ \theta_1^T \theta_2 . \quad (12)$$

We refer to it as a *global* basis since each $v_{\mathbf{vec}^{-1}(B_j)}$ impacts $v^\theta$ on the entirety of Ω. We will soon describe another orthonormal basis which is more local in nature.

Let $v^\theta_{\text{vert}}$ □ $\left\{ v^\theta(\xi) \right\}_{\xi \,□\, □_{\text{vert}}}$ denote the (ordered) $N_v$-tuple of $n$-dimensional values $v^\theta$ takes at the $N_v$ points in $□_{\text{vert}}$.

**Lemma 3—**Let $L_{\theta \,□\, v^\theta_{\text{vert}}}$ denote the map that sends $\theta$ to $v^\theta_{\text{vert}}$. $L_{\theta \,□\, v^\theta_{\text{vert}}}$ is a linear injection and its associated $d \times (nN_v)$ matrix is given in closed form. If □ is of type I, the map is invertible. In which case, we denote its inverse by $L_{v^\theta_{\text{vert}} \,□\, \theta}$.

Let ∂Ω be the border of Ω and $n : \partial\Omega \to S^{n-1}$ be the unit normal to it. We now discuss linear subspaces of □.

**Lemma 4—**Optional constraints such as $\langle v^{\theta}(x), n(x)\rangle = 0$ on $\partial\Omega$ and/or $\{\mathrm{tr}(A_c) = 0\}_{c=1}^{N_\square}$, are linear and can thus extend $L$ (from Lemma 2) to have more rows. The null space of the extended $L$ is a linear subspace (whose dimension is denoted by $d' < d$) of the null space of the original $L$. Reusing the symbol $B = [B_1 \ldots B_{d'}] \square \square^{D \times d'}$ to denote the particular orthonormal basis of this new null space which is obtained via SVD of the extended $L$, we get a $d'$-dimensional basis of CPA velocity fields, $\left\{v_{\mathrm{vec}^{-1}(B_j)}\right\}_{j=1}^{d'}$, whose elements satisfy the new constraint(s).

The corresponding linear subspaces of $\square$ are denoted $\square^{\partial}$ and $\square^{\mathrm{tr}}$. We also define their intersection, $\square^{\partial,\mathrm{tr}} \square \square^{\partial} \cap \square^{\mathrm{tr}}$. We reuse the symbol $v^{\theta}$ to denote an element of any of these subsets, with the understanding that then $\theta \in \mathbb{R}^{d'}$ for some $d' < d$. We now define, similarly to Eq. (5), subsets of $M$:

$$M^{\partial} \square \left\{\exp(v^{\theta}) : v^{\theta} \square \square^{\partial}\right\} \square M; \quad (13)$$

$$M^{\mathrm{vp}} \square \left\{\exp(v^{\theta}) : v^{\theta} \square \square^{\mathrm{tr}}\right\} \square M; \quad (14)$$

$$\mathrm{M}^{\partial,\mathrm{vp}} \square \left\{\exp(v^{\theta}) : v^{\theta} \square \square^{\partial,\mathrm{tr}}\right\} \square M. \quad (15)$$

If $\square$ is of type I (so $d = nN_v$) then, by viewing $v^{\theta}_{\mathrm{vert}}$ as a point in $\mathbb{R}^d$, the standard basis for $\mathbb{R}^d$ defines another basis for $\square$ (there is a linear bijection between the $(n+1)$-tuple of the $n$-dimensional velocities at the vertices of a type-I $U_c$ and the $n \times (n+1)$ matrix $A_c$). Naturally, we refer to this vertex-based basis as a *localized* basis of $\square$. It is orthonormal w.r.t. the following inner product, obtained by summing the dot products between pairs of velocities at the vertices:

$$\langle \square, \square\rangle_{\mathrm{vert}} : (v^{\theta_1}, v^{\theta_2}) \square \sum_{\xi : \xi \square \square_{\mathrm{vert}}} v^{\theta_1}(\xi) \square v^{\theta_2}(\xi). \quad (16)$$

For $\square^{\partial}$, we can build a similar ($d'$-dimensional) localized basis; however, for $\square^{\mathrm{tr}}$ (and $\square^{\partial,\mathrm{tr}}$) this is not possible. We conclude Section 4.2 with the following fact.

**Algorithm 2**

A Greedy Algorithm for (Landmark-based) Inference over a Transformation Obtained by Integrating a Non-Stationary Field where the Field is Piecewise-Constant w.r.t. Time. See Text for Details:

```
Input : $N_{\text{pts}}$ pairs of Ω-valued points: $\{(x_i, y_i)\}_{i=1}^{N_{\text{pts}}}$, $\varepsilon$; $k_{\max}$
1:  for $i \in \{1, \ldots, N_{\text{pts}}\}$ do in parallel
2:      $x_i^{[0]} \leftarrow x_i$
3:  for $k \in \{1, \ldots, k_{\max}\}$ do
4:      $\text{obs}^k \leftarrow \{(x_i^{k-1}, y_i)\}_{i=1}^{N_{\text{pts}}}$ // observations
5:      $\theta^k \leftarrow \arg\max_\theta p(\theta^k) L_v(v^\theta; \text{obs}^k)$ where $L_v(v^{\theta^k}; \text{obs}^k)$ □ $e^{-\frac{1}{\sigma^2}\sum_{i=1}^{N_{\text{pts}}}\left\|x_i^{k-1} + v^\theta(x_i^{k-1}) - y_i\right\|^2}$
    // closed-form optimization; see text
6:      $\alpha_k \leftarrow \arg\max_\alpha p(\alpha\theta^k) L_T(T^{\alpha\theta^k}; \text{obs}^k)$ where $L_T(T^{\alpha\theta^k}; \text{obs}^k)$ □ $e^{-\frac{1}{\sigma^2}\sum_{i=1}^{N_{\text{pts}}}\left\|T^{\alpha\theta^k}(x_i^{k-1}) - y_i\right\|^2}$
    /* the 1D optimization can be done by, e.g., a line search;
    $\{T^{\alpha\theta^k}(x_i^{k-1})\}_{i=1}^{N_{\text{pts}}}$ is computed by Algorithm 1
7:      if $\|\theta^k \alpha_k\| < \varepsilon$ then break
Output $\hat{T}$ □ $(T^{\alpha_k\theta^k}$ □ □ □ $T^{\alpha_1\theta^{[1]}})$
// $\hat{T}$ is a concatenation of points in $M$
```

**Fact 2—**$\langle\cdot,\cdot\rangle_B$ and $B$ vary with translations of Ω while $\langle\cdot,\cdot\rangle_{\text{vert}}$ and the localized basis are invariant to them.

## 4.3 Smoothness Priors on CPA Velocity Fields

Adopting a Bayesian approach, we now present two complementary methods to construct priors on CPA fields.

**I: Project a prior from □′ onto □—**On □′, it is easy to build a smoothness prior that also penalizes large values; e.g., we use a zero-mean Gaussian with a $D \times D$ covariance, $\Sigma_{\text{PA}}$, whose correlations decay with inter-cell distances, and write $\text{vec}(A) \sim □(0_{D\times 1}, \Sigma_{\text{PA}})$. Next, we use the □′ → □ projection, $A \mapsto \theta = B^T\text{vec}(A)$, to induce a prior on □:

$$p(\theta) = \square\left(0_{d \times 1}, \sum{}_{\text{CPA}}\right) \quad (17)$$

where $\Sigma_{\text{CPA}} = B^T \Sigma_{\text{PA}} B$. See Fig. 4 for samples from $p(\theta)$. By Fact 2, $p(\theta)$ depends on the origin of Ω.

**II: Let a prior on $v_{\text{vert}}^{\theta}$ induce a prior on $v^{\theta} \square \square$—**If the prior on $v_{\text{vert}}^{\theta}$ is invariant to the origin of Ω then so is the resulting $p(\theta = L_{v_{\text{vert}}^{\theta} \square \theta} v_{\text{vert}}^{\theta})$; e.g., such is the case for either a zero-mean Gaussian whose covariance is fully determined by *inter-vertex* distances, or for an MRF of the form $\prod_{i,j:\xi_i \underset{\sim}{\text{nbrs}} \xi} \psi(v^{\theta}(\xi_i), v^{\theta}(\xi_j))$ where $\psi$ is some pairwise potential. Method II does not require the computation of $B$. Unlike method I, however, it is applicable only if $\square$ is of type *I and* volume preservation is not imposed. Another advantage of method I over method II (or, for that matters, over most regularizers/priors people often place on velocity fields in the context of diffeomorphisms) is that by defining the prior in terms of the $A_c$'s (as opposed to the $v_{\text{vert}}^{\theta}$ values they generate), *we avoid introducing fictitious variance*. For example, if $v^{\theta}$ is purely affine then the $A_c$'s are all the same. Thus, w.r.t. the prior defined using the global basis (i.e., method I), $v^{\theta}$ is considered very smooth (in the machine-learning sense, not calculus one). However, w.r.t. the localized basis (i.e., method II), $v^{\theta}$ is less smooth since the values of $v_{\text{vert}}^{\theta}$ vary with the vertex location.

### 4.4 CPA-Based Transformations

Recall from Section 3 that, via Eq. (2), $v^{\theta} \square \square$ implies $T^{\theta} \in M$, a relation denoted $T^{\theta} = \exp(v^{\theta})$ (see Eq. (5)). The detail alluded to in Section 3 is that for Eq. (2) to be well defined, $\phi^{\theta}(x, \tau)$ must always be in Ω, the domain of $v^{\theta}$. This trivially holds if $\Omega = \mathbb{R}^n$. It also holds if $\Omega \subsetneq \mathbb{R}^n$ *and* $v^{\theta} \square \square^{\partial}$. If $\Omega \subsetneq \mathbb{R}^n$ and $v^{\theta} \square \square \backslash \square^{\partial}$, it might not. Also, some of the results below require both the continuity and PA properties of $v^{\theta}$. This brings us to the following lemma.

**Lemma 5—**If $\Omega \subsetneq \mathbb{R}^n$, we can extend $\square$ and constrain the velocity fields such that we will obtain a linear subspace of $\square$ whose elements extend to CPA fields on the entirety of $\mathbb{R}^n$. In which case, we also redefine $\Omega$ to be equal to (the whole of) $\mathbb{R}^n$.

Henceforth, if $\Omega \subsetneq \mathbb{R}^n$ *and* we use $\square$ (or $\square^{\text{tr}}$) and not $\square^{\partial}$ (or $\square^{\partial,\text{tr}}$), we will assume the procedure from Lemma 5 has been applied. Thus, Eq. (2) is well defined, $T^{\theta}$ is always an $\Omega \to \Omega$ map and $v^{\theta}$ is indeed CPA. Either way, $\phi^{\theta}(x, 0) = x$ for every $x \in \Omega$ while $\{x : v^{\theta}(x) = 0\}$ are *fixed points* of $T^{\theta}$.

**Fact 3—**If $A \in \mathbb{R}^{n\,(n+1)}$, let $\overset{\square}{A} = \begin{bmatrix} A \\ 0_{1 \times (n+1)} \end{bmatrix} \square \square^{(n+1) \times (n+1)}$. The last row of **expm**$(\overset{\square}{A})$ (where **expm** is the matrix exponential) is $[0_{1\times n}\ 1]$. Also, det **expm**$(\overset{\square}{A}) > 0$.

If $n = 1$ (so $\tilde{A}$ is 2-by-2) or if $A$ has a special structure (e.g., if $A_{1:n,1:n+1}$ is either diagonalizable, idempotent, or a 3-by-3 skew-symmetric matrix), then $\mathbf{expm}(\tilde{A})$ has a closed form. Otherwise, since $n$ is usually small and due to the structure of $\tilde{A}$, a generic **expm** routine typically approximates it well. Let $t \in \mathbb{R}$ and let $\psi^t_{\theta,c} : \Omega \to \mathbb{R}^n$ be the solution to an ODE with an $\mathbb{R}^n \to \mathbb{R}^n$ *affine* velocity field, $\xi \mapsto A_{c,\theta}\tilde{\xi}$, and an initial condition, $x$:

$$\begin{bmatrix} \psi^t_{\theta,c}(x) \\ 1 \end{bmatrix} \triangleq T_{A_{c,\theta},t}\tilde{x}, \quad T_{A_{c,\theta},t} \triangleq \mathbf{expm}(t\tilde{A}_{c,\theta}) . \quad (18)$$

We note that the solution to Eq. (2) is the composition of a finite number, denoted by $m$, of such solutions:

$$\phi^\theta(x,t) = (\psi^{t_m}_{\theta,c_m} \circ \ldots \circ \psi^{t_2}_{\theta,c_2} \circ \psi^{t_1}_{\theta,c_1})(x) . \quad (19)$$

As mentioned earlier, $T^\theta$ is defined via $T^\theta(x \triangleq \phi^\theta(x, 1)$. The compact form of Eq. (19) hides a difficulty: the number of the trajectory segments, $m$, their durations, $\{t_i\}_{i=1}^m$, and the indices of the cells involved, $\{c_i\}_{i=1}^m$ (where a cell may appear more than once), *all depend on x*. Except the index of the first cell, $c_1 = \gamma(x)$, they also depend on $\theta$ and $t$. Thus, a more precise and cumbersome notation would be:

$$\left(\psi^{t_{m_{x,\theta,t}}(x,\theta,t)}_{\theta,c_{m_{x,\theta,t}}(x,\theta,t)} \circ \ldots \circ \psi^{t_2(x,\theta,t)}_{\theta,c_2(x,\theta,t)} \circ \psi^{t_1(x,\theta,t)}_{\theta,c_1(x)}\right)(x) .$$

**Remark 5—**While invertible affine matrices form a group, $\phi^\theta(\cdot, t) : \Omega \to \Omega$ is *not* affine, exactly because of the $x$-dependencies mentioned above. In fact, since *the above quantities vary with x even within a cell*, the continuous $x \mapsto \phi^\theta(x, t)$ is not PA, hence not CPA. the term *CPA-based*.

A map, $T : \Omega \to \Omega$, is a called a ($C^1$) *diffeomorphism* (on $\Omega$) if $T^{-1}$ exists and both $T$ and $T^{-1}$ are differentiable. Let $G$ be the space of (orientation-preserving) diffeomorphisms on $\Omega$. Let $H \subset G$ be its restriction to those diffeomorphisms that can be obtained, via integration, from uniformly-continuous stationary velocity fields; i.e., $T(\cdot) = \phi(\cdot,t)$ where $\phi(x,t) = x + \int_0^t v(\phi(x,\tau))d\tau$ for a uniformly-continuous $v : \Omega \to \mathbb{R}^n$. Both $G$ and $H$ are $\infty$–dimensional nonlinear spaces.

**Theorem 1—**(i) If $\alpha \in \mathbb{R}$ then $v^\theta = v^{\alpha\theta}/\alpha$ and $\phi^\theta(\cdot; t) = \phi^{\alpha\theta}(\cdot, t/\alpha)$. (ii) $(T^\theta)^{-1}$ exists, is in $\mathcal{M}$, and equals to $T^{-\theta}$. (iii) $\mathcal{M} \subset G$. (iv) If $T^\theta \in \mathcal{M}^{vp}$ then $T^\theta$ is volume-preserving (hence the superscripted $vp$). (v) $\mathcal{M}$ is a $d$-dimensional nonlinear space; similar statements hold for $\mathcal{M}^\partial$,

$M^{\mathrm{vp}}$ and $M^{\partial,\mathrm{vp}}$, with their respective values of $d'$. (vi) If $\square_1 \square \square_2 \square \ldots$ is a tessellation sequence such that eventually all the cells become arbitrarily small, then $M_{\Omega, \square_k} \xrightarrow[k \to \infty]{} H$.

**Remark 6.3—**Since $M$ is nonlinear, it is easier to work in the linear □ and then, via exp, move from □ to $M$. This (tangent-space-based) approach is popular in the context of nonlinear differentiable manifolds and, particularly, is used extensively in various spaces of diffeomorphisms (e.g., [27]).

**Corollary 1—**If $n = 1$, $T^\theta$ is increasing. Let $J \subset \mathbb{R}$ be an interval. If $F_0 : J \to \Omega$ is either non-decreasing, increasing, right-continuous, continuous, differentiable, a diffeomorphism, or a step function (7 non-mutually-exclusive cases), then so is $F^\theta \triangleq T^\theta \circ F_0 : J \to \Omega$. If, in addition, $\Omega = [0, 1]$, $v^\theta \square \square^\partial$ and $F_0$ is a CDF, then $F^\theta$ is a CDF. The (trivial) proof is omitted.

This gives an unconstrained parametrization of large classes of increasing functions and CDFs/histograms via a finite-dimensional linear space (Figs. 5 and 6). In the CDF case, $F_0$ need have neither a density nor finite moments.

## 4.5 Integration Details

Equation (19) justifies and simplifies the proof of Theorem 1, provides an insight into the structure of $t \mapsto \phi^\theta(x, t)$, and, for $n = 1$, yields a closed-form solution for $x \mapsto T^\theta(x)$. If $n > 1$, it suggests a practical, *essentially*-exact solution.

**Theorem 2—**If $n = 1$ then $m_{x,\theta,t}$, $\left\{t_i(x; \theta, t)\right\}_{i=1}^{m_{x,\theta,t}}$, and $\left\{c_i(x; \theta, t)\right\}_{i=2}^{m_{x,\theta,t}}$ have closed forms. Thus, so does $T^\theta(x)$.

**Proof—**The notation below drops the $\theta$-dependencies. Without loss of generality, let $t > 0$. For $c \square \left\{c_i(x; t)\right\}_{i=2}^{m_{x,t}}$, let $A_c = [a_c, b_c]$ and let $U_c = \left[x_c^{\min}, x_c^{\max}\right]$. It can be shown that

$$\psi_c^t(x) = \begin{cases} e^{ta_c}x + \dfrac{b_c(e^{ta_c} - 1)}{a_c} & \text{if } a_c \neq 0 \\ x + tb_c & \text{if } a_c = 0 \end{cases}. \quad (20)$$

Note $c_1 = \gamma(x)$ and suppose $v(x) > 0$. Let $t_1(x)$ be the hitting time of the right boundary of $U_{c_1}$:

$$t_1 \square \min\left\{t : \psi_{c_1}^t(x) = x_{c_1}^{\max}\right\} \quad (21)$$

where, by convention, the minimum of an empty set is ∞. If $v(x_{c_1}^{\max}) \le 0$ then $t_1 = \infty$. Otherwise,

$$t_1 = \begin{cases} \frac{1}{a_{c_1}} \log\left(\frac{x_{c_1} + \frac{b_{c_1}}{a_{c_1}}}{x + \frac{b_{c_1}}{a_{c_1}}}\right) & \text{if } a_{c_1} \ne 0 \\ \frac{x_{c_1} - x}{b_{c_1}} & \text{if } a_{c_1} = 0 \end{cases} . \quad (22)$$

If $t_1 > t$, we are done: $\phi^\theta(x, t) = \psi_{c_1}^t(x)$. Otherwise, we entered the next interval to the right, $U_{c_1+1}$ (so $c_2 = c_1 + 1$), and the process (i.e., solving for $t_2$, but this time with $x_{c_1}^{\max}$ as the new starting point instead of $x$, with $t - t_1$ instead of $t$, and $c_2$ instead of $c_1$) is redone iteratively till convergence (the number of such steps, $m_{x,t}$, is finite since $t$ is finite). A loose upper bound on $m_{x,t}$ is $N_{□} - c_1 + 1$. The case where $v(x) < 0$ is handled similarly. Taken together,

$$\phi(x; t) = (\psi_{c_m}^{t_m} \circ \dots \circ \psi_{c_2}^{t_2} \circ \psi_{c_1}^{t_1})(x) \quad (23)$$

where for $2 \le i \le m_{x,t}$, $c_i = c_{i-1} + \operatorname{sign}(v(x))$, while $m_{x,t}$ and $\{t_i\}_{i=1}^{m_{x,t}}$ are found as describe above. □

If $n > 1$ then $T^\theta(x)$ is given *essentially* in closed from in the sense that Eq. (19) still holds, but there is the mild issue that a generic **expm** is needed (see also the matrix decomposition suggested in [27]) and, much more importantly, to find the quantities from Theorem 2 one needs, for every $x$, to sequentially solve, an $x$-dependent number of problems of the form $\arg\min_{t>0} \gamma([\mathbf{exp\,m}(t\tilde{A})]_{1:n,1:n+1}\tilde{\xi}) \ne \gamma(\xi)$ where the pair $(\xi, A) \in \Omega \times \mathbb{R}^{n\times(n+1)}$ is (in general) different in each problem. Doing this for $n = 1$ is easy, fast, and accurate. But for $n \ge 2$, partly since $\phi^\theta(x, \cdot)$ *may reenter a cell* (prohibiting, e.g., a binary-search method), this requires tedious bookkeeping, multiple **expm** calls, and multiple invocations of a numerical solver. This yields slow approximated solutions that are also hard to implement in GPU.

Inference over diffeomorphisms (CPAB included) often requires evaluating $T^\theta$ for multiple values of $\theta$ (e.g., 10,000), and a high $N_{\text{pts}}$, where $N_{\text{pts}}$ is number of points to be transformed (e.g., the number of image pixels). Preferring a more practical alternative, we propose a specialized solver for integrating CPA fields. This solver, summarized in Algorithm 1, alternates between the analytic solution (with neither bookkeeping nor need to solve explicitly for $m_{x,\theta,1}$, $\{c_i(x;\theta,1)\}_{i=2}^{m_{x,\theta}}$ and $\{t_i(x;\theta,1)\}_{i=1}^{m_{x,\theta}}$) and a generic solver. Thus, the proposed specialized solver is both faster and more accurate than a generic solver since most of the trajectory is solved exactly, while only in small portions of the trajectory does our solver resort to the generic one. Importantly, in Algorithm 1, the required number of **expm** calls is constant, $N_{\mathcal{P}}$, and grows with neither $N_{\text{pts}}$ nor $t$. In fact, since Algorithm 1 is highly accurate, fast, and simple, and since usually $N_{\mathcal{P}} \ll N_{\text{pts}}$, we prefer to use it even when $n = 1$. Lower/upper bounds on the algorithm's complexity are $O(C_1) + O(C_2 \times N_{\text{steps}})$ and $O(C_1) + O(C_2 \times N_{\text{steps}} \times n_{\text{steps}})$ where $C_2 = N_{\text{pts}}/\#\text{cores}$ and $C_1 = \text{expm's cost} \times N_{\mathcal{P}} / \#\text{ cores}$. Usually, the $O(C_1)$ term is negligible. $N_{\mathcal{P}}$ and $\theta$ affect inter-cell moves hence which bound is tighter; usually it is the lower one.

**Remark 7—**Consecutive analytic updates (for the same $\Delta_t\tilde{A}_c$) amount to taking powers of **expm** $(\Delta_t\tilde{A}_c)$. While $(\mathbf{exp\,m}(\Delta_t\tilde{A}_c))^k = \mathbf{exp\,m}(k\Delta_t\tilde{A}_c)$, on a computer numerical errors accrue and the equality may not hold [66], [67]. This issue is most noticeable when the matrices are large (and dense) and when $k$ is large. Here, however, this issue does not arise in practice since both the matrices (which also have zeros in the last row) and $k$ are small.

In Algorithm 1, $\gamma$ is called often (though far less often than when using a generic solver throughout). In our implementation, we decided against a lookup-table approach since, once the trajectories start evolving, the argument to $\gamma$ takes values in the continuum. Besides their simplicity, a key advantage of type-II cells, most notably when compared with type-III cells, is that evaluating $\gamma$ is easy and fast, especially when $\mathcal{P}$ is also regular (since we can then use rounding and modulo operations). Thus, e.g., we construct our type-I tessellations as a refinement of regular type-II tessellation. If $n = 2$, we split each rectangle into 4 triangles while if $n = 3$ we split each box into 5 tetrahedra. When evaluating $\gamma$, we first find the "parent" hyperrectangle and then pick the correct type-II cell among those within that hyper-rectangle. For $n > 3$, we avoided implementing type-I cells; rather, in such cases we use only type-II tessellations.

### 4.6 Adaptive Tessellations

It is possible to use CPAB transformations with a data-adaptive $\mathcal{P}$; e.g., placing more cells in certain regions. The potential expressiveness gain, however, needs to be compared against: 1) the slowdown in evaluating $\gamma$ and thus in evaluating $x \mapsto T^{\theta}(x)$; 2) the need to recompute $B$ (unless one is content with a localized basis and priors obtained via method II). In any case, while for high values of $n$ (say, $n > 3$) this may be a reasonable approach to mitigate the curse of dimensionality, we prefer, to keep computing time low and implementation simple, to use fixed tessellations and handle scarce-data regions via the smoothness prior.

### 4.7 Computing Derivatives of $T^{\theta}(x)$ w.r.t. $\theta$

One of our main motivations was facilitating tractable MCMC inference over latent diffeomorphisms. While we empirically found MCMC methods to be easier to tune and to produce better results than gradient-based optimization, for completeness we provide the details for computing $dT^\theta(x)/d\theta$. While one can also use finite differences, the derivation below leads to better accuracy and some computational savings (e.g., no additional **expm** calls are needed). Let $\theta_j$ denote the $j$th component of $\theta$. Define $y(x,t) \triangleq \frac{d}{d\theta_j}\phi^\theta(x,t)$. It can be shown that $y(x, t)$ satisfies

$$\underbrace{y(x,t)}_{n\times 1} = \int_0^t \underbrace{B_{c,j}}_{n\times(n+1)} \underbrace{\widetilde{\phi^\theta(x,\tau)}}_{(n+1)\times 1} + \underbrace{A_{c,\theta}}_{n\times(n+1)} \underbrace{\widetilde{y(x,\tau)}}_{(n+1)\times n} \, d\tau \quad (24)$$

where $c = \gamma(\phi^\theta(x, \tau))$ and $B_{c,j}$ is the $c$th matrix in $\mathrm{vec}^{-1}(B_j) = (B_{1,j}, \ldots, B_{N_\mathcal{P},j})$. Setting $t = 1$, the solution of these $d$ equations (for $j \in \{1, \ldots; d\}$) yields $\frac{dT^\theta(x)}{d\theta}$. Any generic numerical solver for integral equations can be used to solve these equations, where the required values of $\phi^\theta(x, \tau)$ can be computed using Algorithm 1. We omit the details, but our implementation includes this derivative.

### 4.8 Time-Dependent Fields and Their Inference

This paper focuses on stationary CPA fields but, more generally, $\theta$ may vary with $t$. In which case, however, Algorithm 1 and Eq. (19) no longer apply (but note that most of the other advantages of the proposed approach will still apply; e.g., relatively-low dimensionality, compact representation, etc.). While we are unaware of a way to exploit the structure of general non-stationary CPA fields for better integration, we can still apply any generic integration technique to them. That said, one useful case of non-stationary fields is special: if $\theta(t)$ is piecewise constant w.r.t. $t$, then we (trivially) apply Algorithm 1 consecutively, once within each time interval. Here is a usage example. Given $N_\text{pts}$ pairs of Ω-valued points, $\left\{(x_i, y_i)\right\}_{i=1}^{N_\text{pts}}$, consider an optical-flow-like representation, $x \mapsto x + v^\theta(x)$, together with the following Gaussian likelihood model,

$$y_i \overset{\text{I.I.D.}}{\sim} \mathcal{N}(x_i + v^\theta(x_i), \sigma^2 I_{n\times n}) \quad (25)$$

($\sigma > 0$), and a Gaussian prior for $\theta$ as in, e.g., Eq. (17). The following $\Omega \times \mathbb{R}^d \to \mathbb{R}^n$ map,

$$(x,\theta) \mapsto x + v^\theta(x) \overset{\text{Eqn. (11)}}{=} x + \sum_{j=1}^{d} \theta_j v_{\mathrm{vec}^{-1}(B_j)} \tilde{x}, \quad (26)$$

is affine w.r.t. $\theta = [\theta_1, \ldots, \theta_d]^T$. It follows that there is a closed-form MAP solution for $\theta$ (obtained by applying the standard result of computing the conditional mean of one random variable given the other, when both are jointly Gaussian). Let $\hat{\theta}_\text{MAP}$ denote this solution. Importantly, the $\Omega \to \mathbb{R}^n$ map,

$$x \;\square\; x + v^{\hat{\theta}_{\mathrm{MAP}}}(x), \quad (27)$$

is (usually) not a diffeomorphism; however, for any real scalar $\alpha$, the $\Omega \to \Omega$ map,

$$x \;\square\; T^{\alpha\hat{\theta}_{\mathrm{MAP}}}(x) = \exp(v^{\alpha\hat{\theta}_{\mathrm{MAP}}})(x), \quad (28)$$

is a (CPAB) diffeomorphism. This suggests a practical greedy approach for a quick-and-dirty inference over (a transformation obtained by the integral of) a CPA velocity field which is piecewise-constant w.r.t. time. This method, summarized in Algorithm 2, may be adjusted to the case of no-correspondences image registration setting.

### 4.9 Some Differential-Geometric Aspects

Readers less interested in differential geometry may safely skip this section. As many readers must have recognized, □ is the tangent space at the identity to $M$; i.e., $\square = T_I M$. Like the transformation space from [27], $M$ is not a Lie group (lack of closure under composition); equivalently, □ is not a Lie algebra (lack of Lie-bracket closure). Note that □ is a finite-dimensional linear subspace of the ($\infty$-dimensional) Lie algebra, $T_I G$ (the Lie group $G$ was defined before Theorem 1), and that $\exp : \square \to M$ is the restriction of the Lie-group exponential map, $\exp : T_I G \to G$, to $\square \to T_I G$. This approach, of restricting a Lie group exponential map to a linear subspace (of the Lie algebra) lacking a Lie-bracket closure, was found useful, e.g., in the context of Symmetric Positive-Definite (SPD) matrices (SPD is not closed under the standard binary operation of matrix Lie groups; likewise, $T_I$SPD is not closed under the Lie Bracket.) [39]. The inner product (on $\square = T_I M$) in Eq. (16) is close in spirit to the oft-used inner product (on $T_I G$):

$$\left\langle v_1, v_2 \right\rangle = \int_\Omega v_1(\xi) \,\square\, v_2(\xi) d\xi \quad v_1, v_2 \;\square\; T_I G\,. \quad (29)$$

As we pointed out, one of the advantages of Eq. (12) over Eq. (16), is related to eliminating fictitious noise. We are unaware of works (including those using Sobelev spaces) suggesting an inner product on $T_I G$ with a similar property. While here we adopt an approach exploiting (the restriction of) the Lie group exponential of $G \supset M$, future work should explore Riemannian geometries of $M$ (with their Riemannian exponential maps) and their relations to those of $G$. While one can define $M$-specific Riemannian metrics, note that $M$ may also inherit a Riemannian metric from $G$.

## 5 Experiments

Sections 5.1, 5.2 and 5.3 focus on synthesis, Section 5.4 pertains to analysis by synthesis, Section 5.5 addresses large numbers of landmarks, Section 5.6 is related to Algorithm 2, and Section 5.7 illustrates representation of CDFs.

### 5.1 Integration Accuracy and Timings

The two parameters in Algorithm 1, $N_{\text{steps}}$ and $n_{\text{steps}}$, imply two step sizes, one large, $\Delta_t = t/N_{\text{steps}}$, used for exact updates, and one small, $\delta_t = \Delta_t = n_{\text{steps}}$, used for the generic-solver subroutine. The reason we can make large (hence few) steps is that we use an exact analytic update when possible. In our experiments, the generic solver was the modified Euler Method, which requires (i) the ability to evaluate a velocity field at a given location and (ii) specifying the step size and number of steps. By construction, Algorithm 1 is more accurate (due to the analytic updates) and faster (due to the smaller number of steps) than any generic solver used as its subroutine. We also verified this empirically, comparing the algorithm against simply running the generic solver step size $\delta_t$ and $N_{\text{steps}} \times n_{\text{steps}}$ steps. For accuracy, in 1D, we compared the integration error (that we can compute this error at all, is by the virtue of our closed-form solution; see Theorem 2) averaged over random CPA velocity fields; i.e., for each value of $N_{\square}$, we drew 100 CPA fields form the prior, $\left\{v^{\theta_i}\right\}_{i=1}^{100}$, sampled (uniformly) 1000 points in Ω, $\left\{x_j\right\}_{j=1}^{1000}$ and then computed the error:

$$\varepsilon = \frac{1}{100}\sum_{i=1}^{100}\frac{1}{1000}\sum_{i=j}^{1000}|T_{\text{exact}}^{\theta}(x_j) - T_{\text{solver}}^{\theta}(x_j)|. \qquad (30)$$

Table 2 shows the relative improvements, $\frac{\varepsilon_{\text{generic}} - \varepsilon_{\text{algorithm}}}{\varepsilon_{\text{generic}}}$, for different values of $N_{\square}$ (the number of intervals). For higher dimensions, where closed form is unavailable, such error analysis is impossible. The results of our timing experiments appear in the Sup. Mat., available online. Note that our implementation can be further optimized. e.g., use a C+ + wrapper instead of a Python one. Moreover, most of the computing time is spent on evaluating $\gamma$ ($\gamma$ is continuously-defined, so discretely-defined lookup tables will not do); with some tricks, the number of such calls can be drastically reduced.

### 5.2 Real-Time Diffeomorphic Image Editing

To highlight the speed at which CPAB transformations are computed, our code includes a GUI where a user chooses velocities at some/all vertices of $\square_4$. Using Lemma 3, the Gaussian priors on $v^{\theta}$, and the fast integration, we compute, in real time, the conditional warp, $\exp(E(v^{\theta}|\text{user's choice}))$. For a demo, please see the **video** in the Sup. Mat., available online.

### 5.3 Expressiveness

While simple, CPAB transformations capture a wide range of diffeomorphisms (obviously, like any other work on diffeomorphisms, we make no claims about capturing non-diffeomorphisms such as occlusions). One way to see this is by inspecting samples from $p(\theta)$ (Fig. 4). $\mathcal{M}$ also contains what is sometimes called large deformations (an overloaded term: used in some texts for elements of $G$ outside its identity component). One way to show a nominal space captures these is a rectangle-to-horseshoe morphing, achieved easily here (see Sup. Mat., available online). Expressiveness is also exemplified by the experiments below (Figs. 7 and 8) and the demo (Section 5.2). As another example, Fig. 9 shows a volume-preserving transformation sampled from the prior applied to a 3D shape. Here, $T^{\theta} \in \mathcal{M}^{\partial,\mathrm{vp}}$, where □ consists of $N_{□} = 320$ cells (a layout of 4×4×4 cubes, each divided into 5 tetrahedra) and $N_v = 125$. The dimension, $\dim(\mathcal{M}^{\partial,\mathrm{vp}})$, is 20 (and not $375 = 3N_v$) due to the additional constraints.

### 5.4 Inference: Analysis-by-Synthesis

Real-world signal analysis requires 3 oft-confounded parts: a *representation*; a probabilistic/deterministic *model* (e.g., a likelihood); *inference*. While the first affects the second which in turn affects the third, when judging a new choice for any of these, it is crucial to distinguish between them. For example, as noted in [68], Horn and Schunk's representation+model of optical flow had been regarded highly inaccurate, till modern inference rendered it almost state-of-the-art. Since *the present work is focused on representations*, we avoid advocating a particular likelihood model or inference method, and instead use, in a coarse-to-fine manner (recall that if $v^{\theta}$ is CPA *w.r.t.* □ it is CPA w.r.t. P′ □ □), simple choices such as Gaussian likelihood and either conjugate-gradient optimization or the MCMC Metropolis algorithm (where, in the proposal step, we use either global or local moves, utilizing Lemma 3). Without claiming these choices are optimal, these turned out to be effective.

Latent transformations arise in two typical settings: with and without known correspondences. In the first, also called (exact/inexact) diffeomorphic point matching [10], [14], one has a set of landmark pairs, $\{(x_i, y_i)\}_{i=1}^{N_{\text{pts}}}$ and seeks $T : \Omega \to \Omega$ such that $\{T(x_i) \approx y_i\}_{i=1}^{N_{\text{pts}}}$. We observe that for $n = 1$, by Corollary 1, our ability to easily solve this problem using CPAB transformations lets us solve a *monotonic regression* problem, as Fig. 5 demonstrates. Particularly, by adding boundary constraints and simulated annealing, we can well fit a CPAB transformation to any reasonable CDF. More generally, but still for $n = 1$, our method can be used to solve statistical/optimization problems over monotonic functions, CDFs and (cumulative sums of) histograms/filters while respecting bin ordering. The case of $n > 1$ appears, e.g., in landmark-based image warping. As shown in Fig. 7, using real images and landmarks from an online hands dataset (http://www.imm.dtu.dk/pubdb/p.php?403), we infer a transformation between different hands in different poses. We then apply it to the whole source image, creating new hands, whose appearance belongs to the source hands, with geometry akin to the destination hands. We also animate the source image by evaluating $\phi^{\theta}$ for different $t$'s (including outside of [0, 1]); e.g., Fig. 7e shows animated frames for the example in the second row of Fig. 7a–7d.

In the no-known-correspondences case, there are two subsets of Ω, denoted $U_{\text{src}}$ and $U_{\text{dst}}$, and two feature maps, $I_{\text{src}}: U_{\text{src}} \rightarrow □$ and $I_{\text{dst}}: U_{\text{dst}} \rightarrow □$, where □ is some feature space (e.g., color), and one seeks $T: \Omega \longrightarrow \Omega$ such that $I_{\text{dst}} \circ T^{-1} \approx I_{\text{src}}$ (or such that $I_{\text{src}} \circ T \approx I_{\text{dst}}$). This is, e.g., the case for time warping (when $n = 1$) and landmark-free image registration ($n > 1$). Fig. 8 (left and middle) shows inferred time warps for motion-capture data from [69]. A similar procedure can be applied to 2D images, in which case it is known as image registration. Fig. 10 shows some results for registrations of images of digits using a pixelwise Gaussian likelihood model and MCMC inference over CPAB transformations. We used 10K iterations for each single registration which took 10 [sec] (i.e., ~ 0.001 [sec] per each MCMC iteration).

As an example for statistics on the inferred transformations, Fig. 8 (right) shows a 1D warping of $\bar{u}$ (defined in the figure's caption) along the 1st eigen warp (computed using PCA on inferred velocity fields); i.e., we use euclidean statistics in the linear □ and map the results to the nonlinear $M$. For a 2D example in the same spirit, we first performed thousands of within-class pairwise registrations of images of digits as described above. Overall we estimated ~ 30, 000 transformations i.e., ~ 3, 000 per class (class of 1's, class of 2's, etc.). Next, we computed PCA on the inferred CPA velocity fields; Fig. 11 shows the first eigen wrap of each class. The Sup. Mat., available online includes a **video** showing the result of applying the first 3 eigen warps (per class) applied to 100 random images from the training set.

## 5.5 Handling a Large Number of Landmarks

In the hand experiment we showed inference for landmark-based warping using a relatively small set of landmarks on real data. The inference technique used there was MAP estimation using the Metropolis algorithm. Here we show, using synthetic data, that the framework can easily utilize a large set of landmark for inference over the latent transformation between the landmark sets; see Fig. 12. The technique used here is ML estimation using the Metropolis algorithm. It is not surprising that with more data the results improve, suggesting empirical evidence that the estimation procedure is *consistent*. However, our point here is that most other methods cannot easily utilize a large set of landmarks.

## 5.6 Inference over a CPA Field Piecewise w.r.t. *t*

As a proof-of-concept for the utility of Algorithm 2, we used it to "prettify" the output of a dense-correspondence tool. We extracted dense correspondences, $\left\{(x_i, y_i)\right\}_{i=1}^{N_{\text{pts}}}$, between two images using SIFT-Flow [20]. Then, we discarded the images and treated the correspondences as *noisy observations* fed as input to Algorithm 2. Finally, for visualization purposes, we used the output of Algorithm 2 to warp the source image. Fig. 14 shows a typical result.

## 5.7 Representing a CDF Using a CPAB Transformation

Fig. 13 shows results (using Metropolis' algorithm) for fitting $F^{\theta} \triangleq T^{\theta} \circ F_0$, where $F_0$ is the CDF of the uniform distribution on $J = [-7, 7]$, to a 3-component Gaussian mixture. Note $F^{\theta}$ well approximates the target CDF.

## 6 CONCLUSION

We proposed new classes of diffeomorphisms that are simple, expressive, efficient, and have many applications. We showed these objects are evaluated fast and with high accuracy (exact in 1D) and that they are highly expressive. While we required Ω to be a hyperrectangle it is not hard to see that, more generally, Ω may be any finite intersection of closed half spaces of $\mathbb{R}^n$. Like the space in [27], our spaces lack closure under composition; but since $T^{\theta_2} \circ T^{\theta_1} \notin G$, our representation can still be used within frameworks that apply consecutive transformations [16]. In fact, Algorithm 2 deals exactly with such scenarios. While this work focused on representation it was primarily motivated by modeling-and-inference considerations: e.g., the ability to quickly run MCMC inference over diffeomorphisms is a hallmark of the method, showing that simplicity pays off.

## Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

## Acknowledgments

O.F. and J.W.F. are partially supported by U.S. Office of Naval Research MURI program, award N000141110688, and VITALITE, which receives support from U.S. ARO MURI, award W911NF-11-1-0391. S.H. was funded by the Danish Council for Independent Research, Natural Sciences.

## Biographies

**Oren Freifeld** received the BSc and MSc degrees in biomedical engineering from Tel-Aviv University, in 2005 and 2007, respectively. He received the ScM and PhD degrees in applied mathematics from Brown University, in 2009 and 2013, respectively. Later, he was a postdoc with MIT Computer Science & Artificial Intelligence Laboratory. He is currently an assistant professor in the Department of Computer Science, Ben-Gurion University. His main fields of research are computer vision, statistical inference, and machine learning.

**Søren Hauberg** received the PhD degree in computer science from the University of Copenhagen, in 2011. He has been a visiting scholar with UC Berkeley (2010), and a post doc in the Perceiving Systems Department, Max Planck Institute for Intelligent Systems (2012–2014). He is currently a post doc in the Section for Cognitive Systems, Technical University of Denmark. He works in the span of geometry and statistics.

**Kayhan Batmanghelich** received the PhD degree in electrical and system engineering from the University of Pennsylvania, in 2012. He is currently a postdoctoral associate with MIT Computer Science & Artificial Intelligence Laboratory. He works in the span of machine learning, medical imaging, and computational biology.

**John W. Fisher III** received a PhD degree in electrical and computer engineering from the University of Florida, Gainesville, Florida in 1997. He is a Senior Research Scientist in the Computer Science and Artificial Intelligence Laboratory at the Massachusetts Institute of Technology. Prior to joining the Massachusetts Institute of Technology he was a member of the research staff with the Electronic Communications Laboratory at the University of Florida from 1987 to 1997, during which time he conducted research in the areas of ultra-wideband radar for ground and foliage penetration applications, radar signal processing, and automatic target recognition algorithms. His current area of research focus includes information theoretic approaches to signal processing, multi-modal data fusion, machine learning and computer vision.

## References

1. Allassonnière S, Kuhn E, Trouvé A. Construction of Bayesian deformable models via a stochastic approximation algorithm: A convergence study. Bernoulli. 2010; 16:641–678.
2. Allassonniere S, Durrleman S, Kuhn E. Bayesian mixed effect atlas estimation with a diffeomorphic deformation model. SIAM J Imag Sci. 2015; 8:1367–1395.
3. Zhang, M., Fletcher, PT. Algorithmic Advances in Riemannian Geometry and Applications. Berlin, Germany: Springer; 2016. Bayesian statistical shape analysis on the manifold of diffeomorphisms.
4. Freifeld O, Hauberg S, Kayhan B, Fisher JW III. Highly-expressive spaces of well-behaved transformations: Keeping it simple. Proc IEEE Int Conf Comput Vis. 2015:2911–2919.

5. Hauberg, S., Freifeld, O., Larsen, A., Fisher, J., Hansen, L. presented at the 19th Int Conf Artif Intell Statist. Cadiz, Spain: 2016. Dreaming more data: Class-dependent distributions over diffeomorphisms for learned data augmentation.

6. Grenander, U. General Pattern Theory: A Mathematical Study of Regular Structures. Oxford, U.K.: Clarendon; 1993.

7. Grenander U, Miller M. Representations of knowledge in complex systems. J Roy Statistical Soc Series B: Methodological. 1994; 56:549–603.

8. Dupuis P, Grenander U, Miller MI. Variational problems on flows of diffeomorphisms for image matching. Quart Appl Math. 1998; 56:587–600.

9. Trouvé A. Diffeomorphisms groups and pattern matching in image analysis. Int J Comput Vis. 1998; 28:213–221.

10. Joshi SC, Miller MI. Landmark matching via large deformation diffeomorphisms. IEEE Trans Image Process. Aug; 2000 9(8):1357–1370. [PubMed: 18262973]

11. Vaillant M, Miller MI, Younes L, Trouvé A. Statistics on diffeomorphisms via tangent space representations. NeuroImage. 2004; 23:S161–S169. [PubMed: 15501085]

12. Beg MF, Miller MI, Trouvé A, Younes L. Computing large deformation metric mappings via geodesic flows of diffeomorphisms. Int J Comput Vis. 2005; 61:139–157.

13. Allassonnière S, Trouvé A, Younes L. Geodesic shooting and diffeomorphic matching via textured meshes. Proc 5th Int Conf Energy Minimization Methods Comput Vis Pattern Recognit. 2005:365–381.

14. Guo, H., Rangarajan, A., Joshi, S. Handbook of Mathematical Models in Computer Vision. Berlin, Germany: Springer; 2006. Diffeomorphic point matching.

15. Grenander, U., Miller, M. Pattern Theory: From Representation to Inference. London, U.K.: Oxford Univ. Press; 2007.

16. Vercauteren T, Pennec X, Perchant A, Ayache N. Diffeomorphic demons: Efficient non-parametric image registration. NeuroImage. 2009; 45:S61–S72. [PubMed: 19041946]

17. Mumford, D., Desolneux, A. Pattern Theory: The Stochastic Analysis of Real-World Signals. Natick, MA, USA: AK Peters; 2010.

18. Younes L. Spaces and manifolds of shapes in computer vision: An overview. Image Vis Comput. 2012; 30:389–397.

19. Durrleman S, Allassonnière S, Joshi S. Sparse adaptive parameterization of variability in image ensembles. Int J Comput Vis. 2013; 101:161–183.

20. Liu C, Yuen J, Torralba A. SIFT flow: Dense correspondence across scenes and its applications. IEEE Trans Pattern Anal Mach Intell. May; 2011 33(5):978–994. [PubMed: 20714019]

21. Kim J, Liu C, Sha F, Grauman K. Deformable spatial pyramid matching for fast dense correspondences. Proc IEEE Conf Comput Vis Pattern Recognit. 2013:2307–2314.

22. Cootes TF, Twining CJ, Petrovic V, Schestowitz R, Taylor CJ. Groupwise construction of appearance models using piecewise affine deformations. Proc Brit Mach Vis Conf. 2005:879–888.

23. Fawzi A, Frossard P. Measuring the effect of nuisance variables on classifiers. Proc Brit Mach Vis Conf. 2016

24. Szeliski, R. Computer Vision: Algorithms and Applications. Berlin, Germany: Springer; 2010.

25. Galun M, Amir T, Hassner T, Basri R, Lipman Y. Wide baseline stereo matching with convex bounded distortion constraints. Proc IEEE Int Conf Comput Vis. 2015:2228–2236.

26. Lin D, Grimson E, Fisher J. Modeling and estimating persistent motion with geometric flows. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit. 2010:1–8.

27. Arsigny, V., Commowick, O., Pennec, X., Ayache, N. Biomedical Image Registration. Berlin, Germany: Springer; 2006. A log-Euclidean polyaffine framework for locally rigid or affine registration.

28. Arsigny V, Pennec X, Ayache N. Polyrigid and polyaffine transformations: A novel geometrical tool to deal with non-rigid deformations–application to the registration of histological slices. Med Image Anal. 2005; 9:507–523. [PubMed: 15948656]

29. Arsigny V, Commowick O, Pennec X, Ayache N. A log-Euclidean framework for statistics on diffeomorphisms. Med Image Comput Comput Assisted Intervention. 2006; 9:924–931.

30. Mansi T, Pennec X, Sermesant M, Delingette H, Ayache N. iLogDemons: A demons-based registration algorithm for tracking incompressible elastic biological tissues. Int J Comput Vis. 2011; 92:92–111.
31. Zhang M, Fletcher PT. Finite-dimensional Lie algebras for fast diffeomorphic image registration. Inf Process Med Imag. 2015; 24:249–259.
32. Censi A, Murray RM. Learning diffeomorphism models of robotic sensorimotor cascades. Proc IEEE Int Conf Robot Autom. 2012:3657–3664.
33. Censi A, Nilsson A, Murray RM. Motion planning in observations space with learned diffeomorphism models. Proc IEEE Int Conf Robot Autom. 2013:2860–2867.
34. Nilsson A, Censi A. Accurate recursive learning of uncertain diffeomorphism dynamics. Proc IEEE/RSJ Int Conf Intell Robots Syst. 2013:1208–1215.
35. Kilian M, Mitra N, Pottmann H. Geometric modeling in shape space. ACM Trans Graph. 2007; 26 Art. no. 64.
36. Gawlik ES, Mullen P, Pavlov D, Marsden JE, Desbrun M. Geometric, variational discretization of continuum theories. Physica D: Nonlinear Phenomena. 2011; 240:1724–1760.
37. Fletcher P, Lu C, Joshi S. Statistics of shape via principal geodesic analysis on Lie groups. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit. 2003:I-95–I-101.
38. Miller EG, Chefd'hotel C. Practical non-parametric density estimation on a transformation group for vision. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit. 2003:II-114–II-121.
39. Porikli F, Tuzel O, Meer P. Covariance tracking using model update based on Lie algebra. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit. 2006:728–735.
40. Lin D, Grimson E, Fisher J. Learning visual flows: A Lie algebraic approach. Proc IEEE Conf Comput Vis Pattern Recognit. 2009:747–754.
41. Freifeld O, Black M. Lie bodies: A manifold representation of 3D human shape. Proc 12th Eur Conf Comput Vis. 2012
42. Pennec X. Probabilities and statistics on Riemannian manifolds: Basic tools for geometric measurements. Proc Nonlinear Signal Image Process. 1999:194–198.
43. Fletcher P, Lu C, Pizer S, Joshi S. Principal geodesic analysis for the study of nonlinear statistics of shape. IEEE Trans Med Imag. Aug; 2004 23(8):995–1005.
44. Sommer S, Lauze F, Hauberg S, Nielsen M. Manifold valued statistics, exact principal geodesic analysis and the effect of linear approximation. Proc 11th Eur Conf Comput Vis. 2010:43–56.
45. Subbarao R, Meer P. Nonlinear mean shift over riemannian manifolds. Int J Comput Vis. 2009; 84:1–20.
46. Srivastava A, Joshi SH, Mio W, Liu X. Statistical shape analysis: Clustering, learning, and testing. IEEE Trans Pattern Anal Mach Intell. Apr; 2005 27(4):590–602. [PubMed: 15794163]
47. Hinkle J, Fletcher PT, Joshi S. Intrinsic polynomials for regression on Riemannian manifolds. J Math Imag Vis. 2014; 50:32–52.
48. Harandi MT, Salzmann M, Hartley R. From manifold to manifold: Geometry-aware dimensionality reduction for SPD matrices. Proc 13th Eur Conf Comput Vis. 2014:17–32.
49. Lorenzi, M., Ayache, N., Pennec, X. Information Processing in Medical Imaging. Berlin, Germany: Springer; 2011. Schild's ladder for the parallel transport of deformations in time series of images.
50. Taheri S, Turaga P, Chellappa R. Towards view-invariant expression analysis using analytic shape manifolds. Proc IEEE Int Conf Autom Face Gesture Recognit Workshops. 2011:306–313.
51. Lorenzi M, Pennec X. Geodesics, parallel transport & one-parameter subgroups for diffeomorphic image registration. Int J Comput Vis. 2013; 105:111–127.
52. Xie Q, Kurtek S, Le H, Srivastava A. Parallel transport of deformations in shape space of elastic surfaces. Proc IEEE Int Conf Comput Vis. 2013:865–872.
53. Fletcher P. Geodesic regression and the theory of least squares on Riemannian manifolds. Int J Comput Vis. 2012; 105:171–185.
54. Jayasumana S, Hartley R, Salzmann M, Li H, Harandi M. Kernel methods on the Riemannian manifold of symmetric positive definite matrices. Proc IEEE Conf Comput Vis Pattern Recognit. 2013:73–80.

55. Freifeld O, Hauberg S, Black MJ. Model transport: Towards scalable transfer learning on manifolds. Proc IEEE Conf Comput Vis Pattern Recognit. 2014:1378–1385.
56. Rubner Y, Tomasi C, Guibas L. Metric for distributions with applications to image databases. Proc 6th Int Conf Comput Vis. 1998:59–66.
57. Srivastava A, Jermyn I, Joshi S. Riemannian analysis of probability density functions with applications in vision. Proc IEEE Conf Comput Vis Pattern Recognit. 2007:1–8.
58. Bookstein FL. Principal warps: Thin-plate splines and the decomposition of deformations. IEEE Trans Pattern Anal Mach Intell. Jun; 1989 11(6):567–585.
59. Arad N, Dyn N, Reisfeld D, Yeshurun Y. Image warping by radial basis functions: Application to facial expressions. CVGIP: Graph Models Image Process. 1994; 56:161–172.
60. Liu C, Shum H-Y, Zhang C-S. A two-step approach to hallucinating faces: Global parametric model and local nonparametric model. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit. 2001:I-192–I-198.
61. Chen T-L, Geman S. Image warping using radial basis functions. J Appl Statist. 2014; 41:242–258.
62. Nielsen M, Johansen P, Jackson A, Lautrup B, Hauberg S. Brownian warps for non-rigid registration. J Math Imag Vis. 2008; 31 Art. no. 221.
63. Weber O, Poranne R, Gotsman C. Biharmonic coordinates. Comput Graph Forum. 2012; 31:2409–2422.
64. Nir T, Bruckstein AM, Kimmel R. Over-parameterized variational optical flow. Int J Comput Vis. 2008; 76:205–2016.
65. Wulff J, Black M. Efficient sparse-to-dense optical flow estimation using a learned basis and layers. Proc IEEE Conf Comput Vis Pattern Recognit. 2015:120–130.
66. Moler C, Loan C Van. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. SIAM Rev. 2003; 43:3–49.
67. Al-Mohy AH, Higham NJ. Computing the action of the matrix exponential, with an application to exponential integrators. SIAM J Sci Comput. 2011; 33:488–211.
68. Sun D, Roth S, Black MJ. Secrets of optical flow estimation and their principles. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit. 2010:2432–2439.
69. Ionescu C, Papava D, Olaru V, Sminchisescu C. Human3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments. IEEE Trans Pattern Anal Mach Intell. Jul; 2014 39(7):1325–1339.

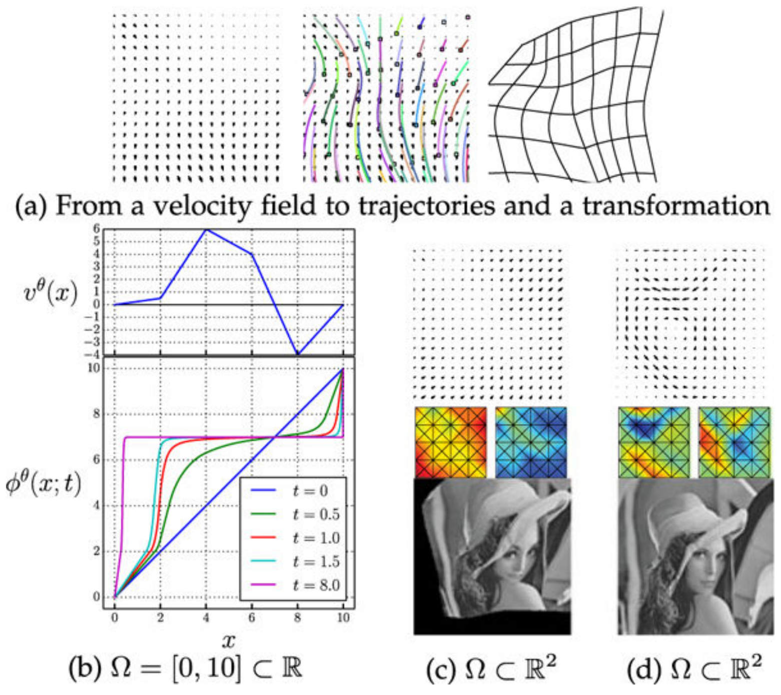

**Fig. 1.**

(a) Integration of sufficiently-nice velocity fields is widely used to generate well-behaved nonlinear transformations. The choice of using CPA velocity fields, among other benefits, reduces computational costs, increases integration accuracy, and simplifies modeling and inference. A CPAB transformation, $x \mapsto \phi^\theta(x, t)$, is one that is *based* (via integration) on a CPA velocity field, $v^\theta$. (b) A 1D example. (c–d) Two 2D examples, where in (d) there are also additional constraints. Top row: a continuously-defined $v^\theta$ in select locations. Middle: Visualizing the horizontal ( $v_h^\theta$, left) and vertical ( $v_v^\theta$, right) components as heat maps highlights the CPA property; blue = −λ, green=0, and red=λ where $\lambda = \max_{x \in \Omega} \max(|v_h^\theta(x)|, |v_v^\theta(x)|)$. Bottom: $I_{\text{src}} \circ \phi^\theta(\cdot, 1)$.

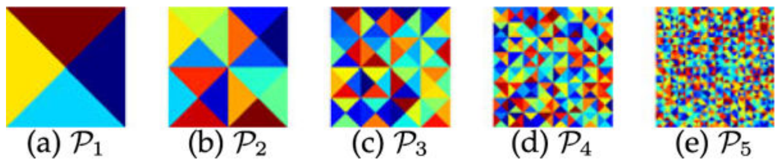

**Fig. 2.**

Several type-I tessellations of a 2D region.

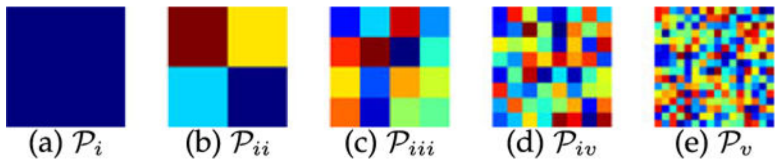

**Fig. 3.**
Several type-II tessellations of a 2D region.

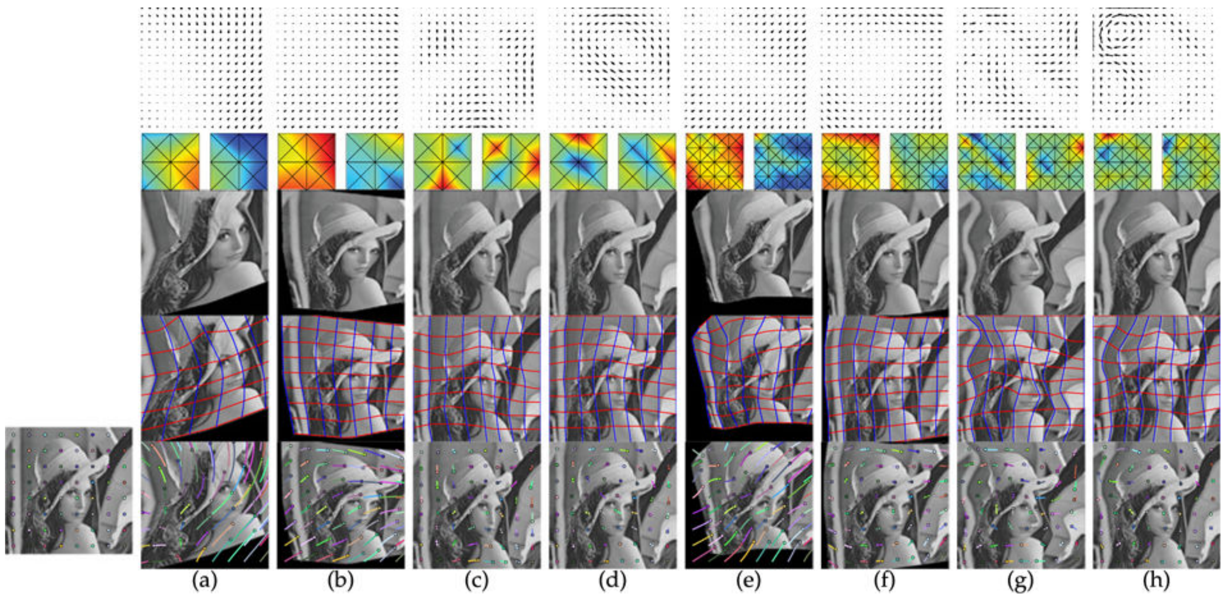

**Fig. 4.**

Samples from the prior (Section 4). (a–h) The top 3 rows echo those in Fig. 1c and Fig. 1d. The 4th row shows a deformed grid overlaid on the image. The 5th row shows select *trajectories*. Note that the trajectories and transformations are differentiable (hence continuous) but *not* piecewise affine. The tessellation in (e–h) is a *refinement* of the one in (a–d). (a) & (e): $T^{\theta} \in \mathcal{M}$. (b) & (f): $T^{\theta} \in \mathcal{M}^{\text{vp}}$. (c) & (g): $T^{\theta} \in \mathcal{M}^{\partial}$. (d) & (h): $T^{\theta} \in \mathcal{M}^{\partial,\text{vp}}$. See Section 4 for details.

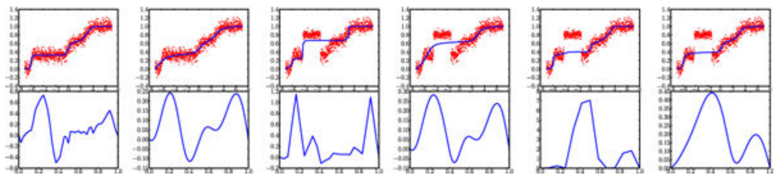

**Fig. 5.**

Monotonic regression on synthetic data (top) by inferring CPA fields (bottom), defined over 100 equal-length cells. We used a squared loss in the 4 leftmost columns and a robust one (Geman-McClure) in the 2 others. Col. 1, 3, and 5 show ML solutions. Col. 2, 4 and 6 show MAP solutions with a smoothness prior. In all cases, the function is increasing, as obtained effortlessly from the representation.

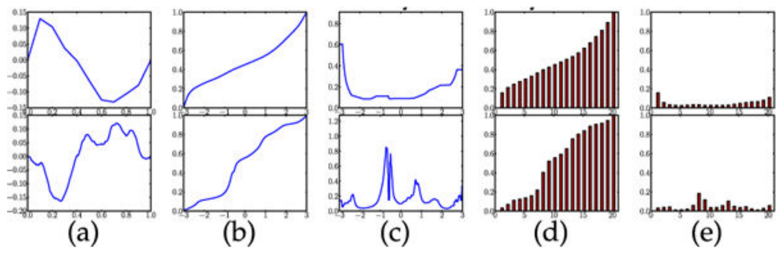

**Fig. 6.**
CDF/histogram representation. 1st row: $N_{\square} = 10$. 2nd row: $N_{\square} = 100$. $F_0$, not shown, is a CDF of a uniform distribution on $J = [-3, 3]$. (a) $v^{\theta} \square \square^{\partial}([0, 1], \square)$. (b) $F = T^{\theta} \circ F_0$. (c) $\frac{d}{dx}F$. Note it is *not* piecewise constant. $H_0$, not shown, is a *cumsum* of a 20-bin uniform histogram. (e) $H = T^{\theta} \circ H_0$, a *cumsum* of a new histogram $h$ (f) $h$.

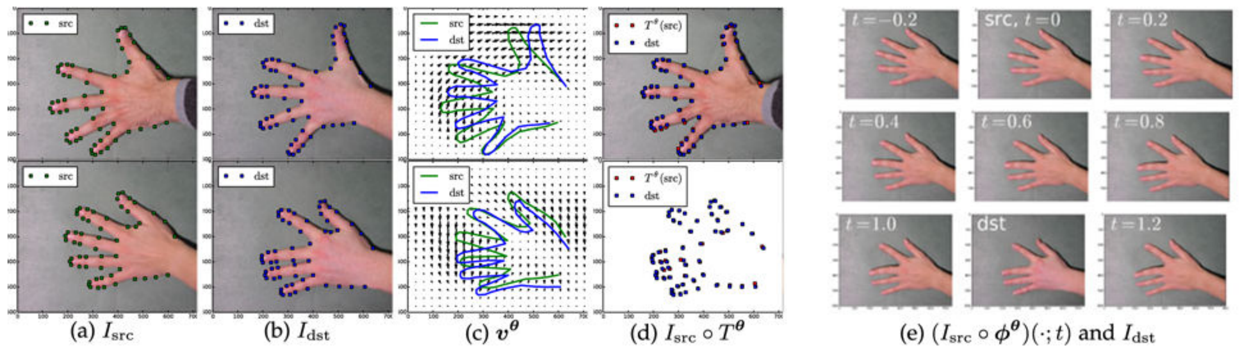

**Fig. 7.**

Landmark-based warping. One example in each row. (a) $I_{\text{src}}$ (and src landmarks). (b) $I_{\text{src}}$ (and src/dst landmarks). (c) The inferred $v^\theta$. (d) A new image is synthesized by warping $I_{\text{src}}$ using $T^\theta$ (and dst & $T^\theta_{(\text{src})}$ landmarks). (d) Animation.

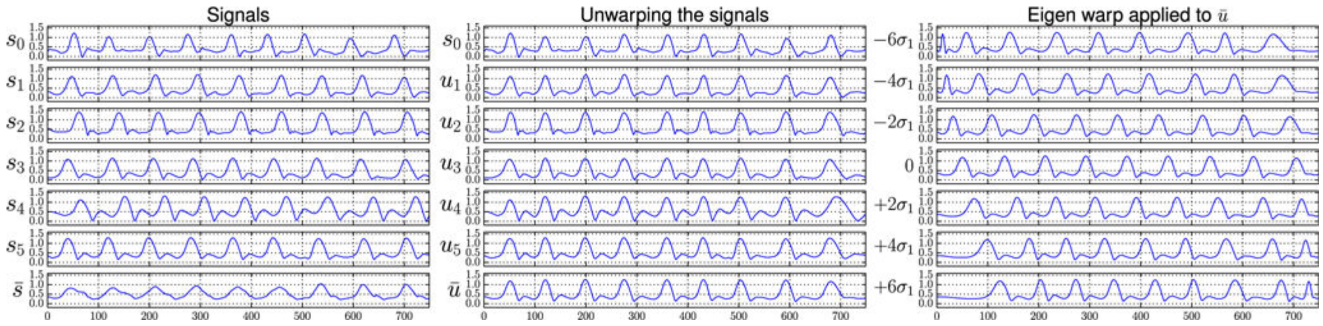

**Fig. 8.**

Nonlinear time warping in real MoCap Data. Left: a reference signal, $s_0$, 5 other signals, $\{s_i\}_{i=1}^{5}$, and their mean, $\bar{s} = \frac{1}{5}\sum_{i=1}^{5} s_i$. The mean is smeared since $\{s_i\}_{i=1}^{5}$ are misaligned. Center: having inferred the warps, $\left\{T^{\theta_i}\right\}_{i=1}^{5}$, we unwarp the signals, setting $u_i = s_i \circ T^{-\theta_i}$. Note $\{u_i\}_{i=1}^{5}$ are better aligned, as is evident by the details preserved in their mean, $\bar{u} = \frac{1}{5}\sum_{i=1}^{5} u_i$. Right: warping $\bar{u}$ by $T^{\theta + j\sigma_1\xi_1}$ where $\theta = \frac{1}{5}\sum_{i=1}^{5}\theta_i$, $j \in \{-6, -4, -2, 0, 2, 4, 6\}$, $\xi_1$ is the first principal component of $\{\theta_i - \bar{\theta}\}_{i=1}^{5}$, and $\sigma_1$ is the corresponding standard deviation.

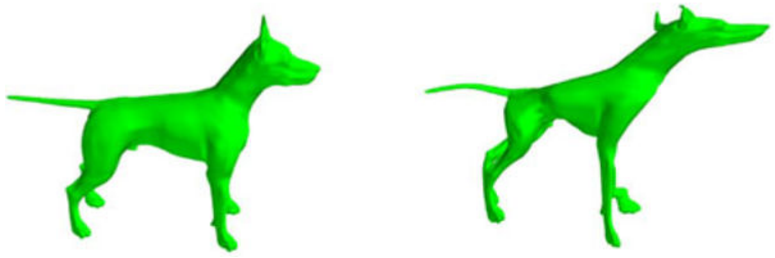

**Fig. 9.**
A 3D example. Left: Source shape (taken from the Tosca Dateset: http://tosca.cs.technion.ac.il). Right: Result of applying a volume-preserving transformation, sampled from the prior, to the source.

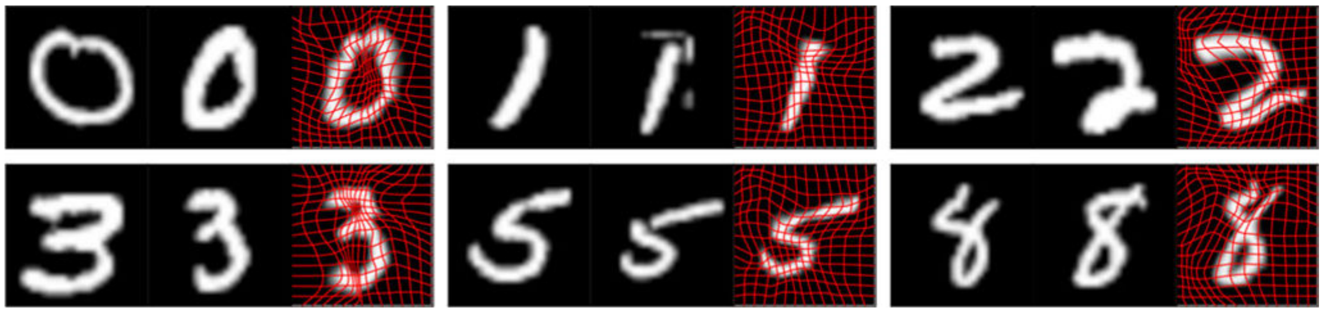

**Fig. 10.**
Example registrations of images from the MNIST dataset.

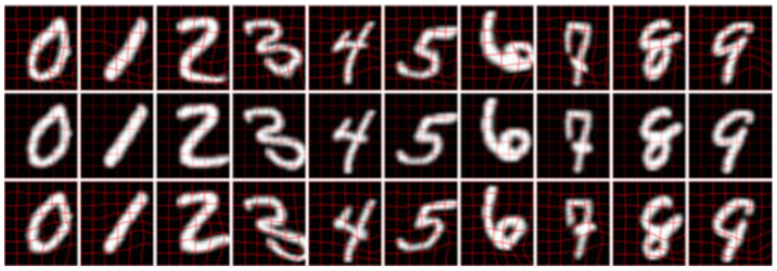

**Fig. 11.**

The 1st eigen wrap of class. *Center row:* the mean (identity) transform. *Top row:* the mean plus 3 standard devidations. *Bottom row:* the mean minus 3 standard deviations.

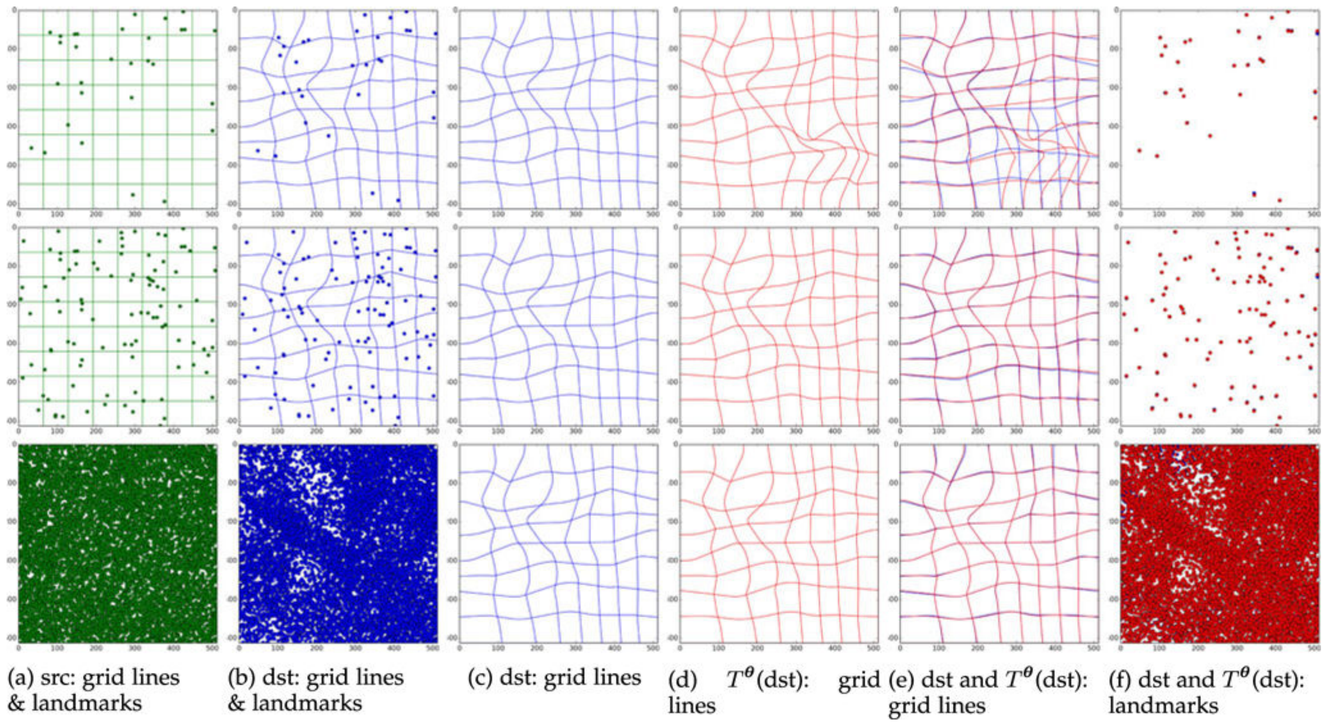

**Fig. 12.**

Landmark-based Maximum Likelihood (ML) estimates of latent transformations. The dimension of the transformation space in this example is $d = 38$. Rows differ in the number of landmarks, $N_{\text{pts}}$. Top row: $N_{\text{pts}} = 30$. Middle: $N_{\text{pts}} = 100$. Bottom: $N_{\text{pts}} = 10,000$. Colors: green=src, blue=dst, red=$T^{\theta}$(src), where $T^{\theta}$ is the ML transformation, found by Metropolis' algorithm. (a) original grid lines + src landmarks; (b) grid lines deformed by the Ground-Truth (GT) transformation + dst landmarks. (c) grid lines deformed by the GT transformation. (d) grid lines deformed by the ML transformation. (e) grid lines deformed by the GT transformation (blue) and grid lines deformed by the ML transformation (red). (f) dst landmarks (blue) and the src landmarks transformed by the ML transformation (red).

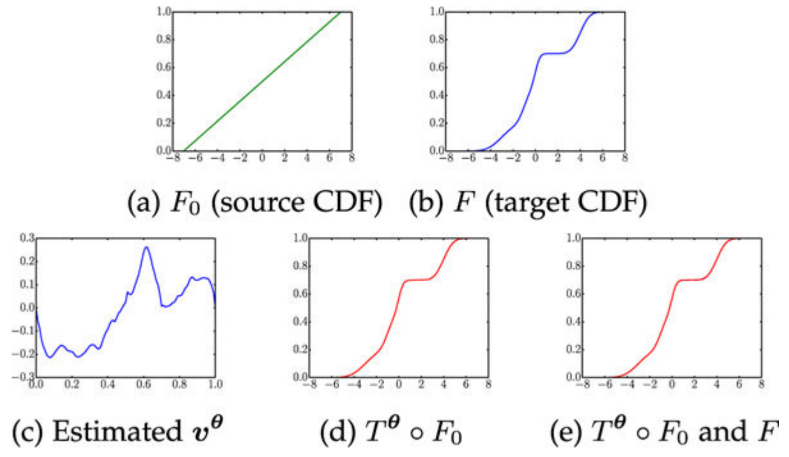

**Fig. 13.**

CDF Representation. The source is the CDF of the uniform distribution. The target is the CDF of a 3-component Gaussian mixture. The setting and the inference procedure, where we used a Maximum Likelihood (ML) solution obtained using MCMC, are virtually the same as in the monotonic-regression experiment, except here we also used simulated annealing, gradually reducing the standard deviation of the "noise" to zero. Typical to an ML solution, the inferred $v^{\theta}$ is non-smooth. MAP estimation (not shown) leads to a smoother field, analogously to the results in Fig. 5.

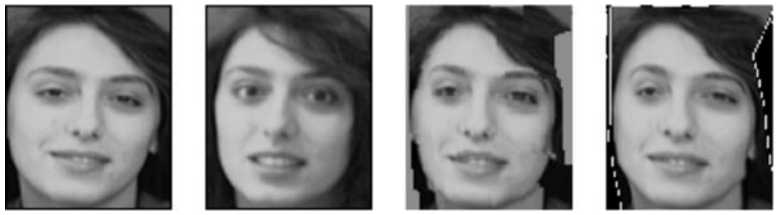

**Fig. 14.**
"Prettifying" the result of a dense-correspondence tool. From left to right: $I_{\mathrm{src}}$; $I_{\mathrm{dst}}$; warping $I_{\mathrm{src}}$ according to SIFT-Flow [20]; warping $I_{\mathrm{src}}$ using the output of Algorithm 2 where the SIFT-Flow was the input.

**TABLE 1**

Values of $N_{□}$, $N_v$, $D = \dim(□'_{\Omega,□})$, and $d = \dim(□_{\Omega,□})$ for the □'s shown in Fig. 2. See also Section 4.

| □ | $N_{□}$ | $N_v$ | $D = 6N_{□}$ | $d = 2N_v$ |
|---|---|---|---|---|
| $□_1$ | 4 | 5 | 24 | 10 |
| $□_2$ | 16 | 13 | 96 | 26 |
| $□_3$ | 64 | 41 | 384 | 82 |
| $□_4$ | 256 | 145 | 1536 | 290 |
| $□_5$ | 1024 | 1025 | 6144 | 1090 |

**TABLE 2**

Accuracy: Improvement, in %, of Algorithm 1 ($N_{\text{steps}} = 10$, $n_{\text{steps}} = 10$) Over the Generic Solver (100 Steps). $\Omega = [0, 1]$ Was Tessellated into $N_{\square}$ Equal-Length Intervals

| $N_{\square}$ | **5** | **25** | **45** | **65** | **85** | **105** |
|---|---|---|---|---|---|---|
| improvement [%] | 2.99 | 12.56 | 11.26 | 9.98 | 3.29 | 3.73 |