Eurographics Symposium on Geometry Processing 2016
Maks Ovsjanikov and Daniele Panozzo
(Guest Editors)

Volume 35 (2016), Number 5

Advection-Based Function Matching on Surfaces

Omri Azencot Orestis Vantzos Mirela Ben-Chen

Technion – Israel Institute of Technology

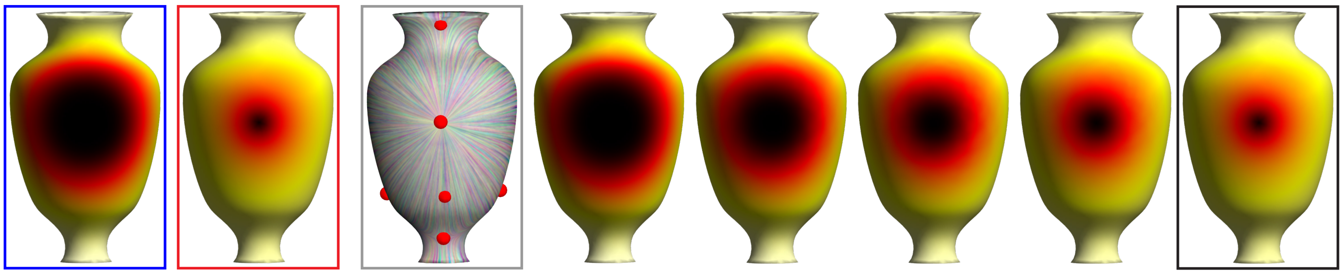

Figure 1: Our method takes a source function (blue frame) and a target function (red frame) and finds a single vector field (gray frame) whose associated flow map advects the source function to a function which matches the target function at the end time (black frame). In addition, our method yields a smooth interpolation of functions by advecting the source function for different times (5 frames from the right).

Abstract

A tangent vector field on a surface is the generator of a smooth family of maps from the surface to itself, known as the flow. *Given a scalar function on the surface, it can be transported, or* advected, *by composing it with a vector field's flow. Such transport is exhibited by many physical phenomena, e.g., in fluid dynamics. In this paper, we are interested in the inverse problem: given source and target functions, compute a vector field whose flow advects the source to the target. We propose a method for addressing this problem, by minimizing an energy given by the advection constraint together with a regularizing term for the vector field. Our approach is inspired by a similar method in computational anatomy, known as LDDMM, yet leverages the recent framework of* functional vector fields *for discretizing the advection and the flow as operators on scalar functions. The latter allows us to efficiently generalize LDDMM to curved surfaces, without explicitly computing the flow lines of the vector field we are optimizing for. We show two approaches for the solution: using linear advection with multiple vector fields, and using non-linear advection with a single vector field. We additionally derive an approximated gradient of the corresponding energy, which is based on a novel* vector field transport *operator. Finally, we demonstrate applications of our machinery to intrinsic symmetry analysis, function interpolation and map improvement.*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

1. Introduction

Finding correspondences between geometric objects is a fundamental problem in geometry processing. In many cases, the map between the objects can be represented through correspondences between *scalar functions*. A Gaussian distribution centered at the location of the object [SNB*12], an intensity function representing medical data [BMTY05] or a geometric descriptor [OBCS*12], are all examples utilizing this approach. In fact, matching functions is a *more* general problem, as functions are not restricted to encode shapes, but can represent alternative information, such as distortion information [OBCCG13], appearance properties [BVDPPH11] or texture coordinates.

A natural extension to the function correspondence problem is to additionally compute an *interpolation* between the given functions, namely a time varying function which starts from the source and smoothly interpolates to the target. One possible approach then, is to recast the problem as finding a set of vector fields, whose associated *flow maps* are composed to yield an interpolation between the functions. The flow map of a vector field is computed in any point of the domain by "traveling" from that point and following the trajectory of the vector field (known as the *flow line*), for a specified time. *Advection* of a function is then achieved by composing it with the *inverse* of the flow map (see Figure 2), where interpolation is computed by advecting the source function for various times, and the target function is attained at the final time.

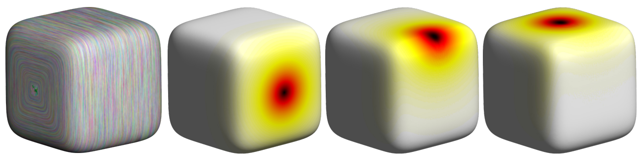

Figure 2: The flow map of a vector field (left, shown with the Line Integral Convolution method [PZ11]) is used to advect a function (middle left) for various different times (middle right and right).

This approach to function interpolation has long been considered in medical imaging where anatomical images are deformed from one to another. Several methods designed for solving this problem are currently available [SDP13] of which the Large Deformation Diffeomorphic Metric Mapping (LDDMM) algorithm [BMTY05] is widely adopted. LDDMM tackles the problem by minimizing an objective function, which combines the advection constraint with a regularizing term on the vector fields. In practice, energy minimization is computed by explicitly constructing the flow map and its Jacobian, or *deformation gradient*. Therefore, carrying over this framework to *curved* domains is challenging, as these quantities are difficult to represent and compute on such domains. We suggest to overcome these difficulties by reformulating the energy using the framework of functional vector fields [ABCCO13], leading to a novel advection-based method for spatial interpolation between real-valued functions on curved triangle meshes.

The main component required for our method is an advection operator acting on curved domains. Previously, advection has been employed to simulate fluids [SY04], and more recently to compute stable shock filters [PK15]. However, these methods rely on the explicit computation of flow lines which is algorithmically complicated, unstable and error-prone on curved triangle meshes. Alternatively, tangent vector fields can be encoded as directional derivatives of functions, and thus as linear operators acting on the space of functions. Adopting this approach, [ABCCO13] showed that on triangle meshes, discrete tangent vector fields can be encoded as sparse matrices, whose *exponential* represents their associated flow map. Further, the composition of a map with a function is given in this setup as a matrix-vector multiplication, and hence advection can be efficiently approximated by computing the action of the matrix exponential on a vector [AMH11]. Notice that since functions are directly mapped to functions, the explicit computation of flow lines and its inherent difficulties is completely avoided in this setup.

In this paper, we facilitate the functional advection technique for solving the function matching problem. Initially, we propose to optimize for a set of vector fields and use *linear* transport, i.e., taking only the first two terms of the matrix exponential. This approach works well in scenarios as fluid simulation [AWO*14, AVW*15] where the Courant–Friedrichs–Lewy (CFL) condition limits propagation speeds and thus restricts the dynamic time step to be small. However, it is sometimes necessary to find a *single* vector field; a constraint that is rarely attainable with the linearized formulation as the required time step for matching might be too big. For instance, assume that the target function is in fact the advected version of the source function, as is the case in *optical flow* problems [SRB14]. In this context, one hopes to reconstruct the underlying vector field that governs the motion. Thus, we generalize our energy functional to include the *full* matrix exponential, for cases where it is crucial to interpolate using a single velocity field.

Unfortunately, the associated *directional derivative* of the matrix exponential is computationally intractable in most of our problems. Recently, Corman et al. [COC15] noticed that this derivative is in fact a block in the matrix exponential of a bigger operator, and thus used a sum of matrix exponentials of bigger matrices. However, they worked in a *reduced* spectral basis, allowing them to facilitate this observation which requires the computation of a large number of matrix exponentials (as many as the number of basis vectors). We, on the other hand, work with the full basis, thus their approach is less applicable in our case. Instead, we observe that an approximation of the matrix exponential derivative can also be formulated in terms of a *Lie bracket* operator acting on vector fields. Thus, we propose a novel discrete bracket and exploit the relation between vector fields and matrices to arrive at a *tractable* derivative for the matrix exponential. Overall, we emphasize that through the entire computation of the functional's gradient, we never store or explicitly compute the matrix exponential, but only its *action* on vectors.

We demonstrate our machinery in several applications as spatial interpolation between various functions and reconstruction of the governing velocity in an optical flow type scenario. Moreover, we construct a continuous symmetric map based on two descriptors. Finally, we show that our method can be used to extract the point-to-point map that is related to a given functional map.

1.1. Related Work

We discuss here various approaches which either solve the same problem on Euclidean domains, solve a related problem on triangle meshes, or target similar applications as ours.

Computational anatomy. The problem of matching consecutive medical images is a classical problem in computational anatomy, and one of the common solutions uses the flow of one or more vector fields, see [SDP13], for a recent review. Among the plethora of such methods, LDDMM [BMTY05] is extremely popular, and has been extended to many settings, though not to curved triangle meshes. On flat domains, discretizations of LDDMM use semi-Lagrangian techniques for vector field integration, and for computing discrete mappings and their differentials, using simple interpolation rules. However, on curved meshes these computations are more challenging, as trajectories should be constrained to remain on the curved surface. Our discretization, on the other hand, is based on the functional approach, thus functions can be advected without explicit computations of mappings and their differentials.

Optical flow. vector field based registration is also popular in computer vision, where it is known as *optical flow*, see [SRB14] for a recent review. In the classical formulation, when two images are given, the goal is to find a smooth displacement vector field which matches the first image to the second. This is in fact the linearized version of advection on Euclidean domains, highly appropriate in optical flow since the change between consecutive images is small. Optical flow has been generalized in many ways, and was recently adapted to advection-based matching of a series of functions on triangle meshes [LB08]. There, however, *multiple* samplings of the interpolated function are given as input, i.e., not only the initial and final functions as in our setup, inherently assuming that the deformation between two consecutive functions is small. Furthermore, they compute multiple vector fields which realize the flow, and their advection approach exhibits far more diffusion than ours.

Optimal transport. Matching between distributions is a prevalent objective in optimal transportation (OT) methods [Vil03, Vil08]. In fact, the Benamou–Brenier [BB00] formulation shares some similarities with our matching approach, with the important difference that their associated vector field is *time-dependent* in general. In the special case when distances are raised to the power of 1 [SRGB14], instead of a general power p, OT is formulated using a single vector field. However, advecting the function on the resulting vector field leads to a trivial pointwise linear interpolation between the source and target functions, whereas our method yields spatial displacements. Alternatively, using *squared* distances [SDGP*15] (i.e., $p = 2$) yields interpolation results that are closer to ours, yet requires regularization for computational efficiency which leads to blurring, and does not output a single vector field.

Related Applications in Computer Graphics. Our method can be classified as on-surface interpolation of functions, and there exist a small number of works addressing similar problems in Computer Graphics. Perhaps the closest to our approach is the method for continuous matching [COC15]. There, the authors improve a given point-to-point map, by optimizing for a vector field such that the composition of its flow map with a given input map approximates another known map. However, optimization in their setup requires working in a reduced spectral basis in order to be computationally feasible due to their usage of explicit matrix exponentials. In addition, their obtained field is smooth and thus does not account for high frequency deformations whereas our method does. Deformation of functions on surfaces is also addressed in [RTD*10], by computing a map fulfilling some point constraints and composing its inverse with the function to be deformed. This problem is in some sense simpler than the one we address, since the constraints imply that the correspondence between the source and target functions is known, and only interpolation is needed.

1.2. Contributions

Our main contribution is a method for solving the inverse problem of computing a vector field whose flow advects one function to another on curved triangle meshes, where the functions are not required to be similar or have overlapping support. To this end we:

- Reformulate the LDDMM energy using functional operators. We explore linear and non-linear advection formulations and provide the associated gradients (Sections 3–5).
- Present a novel Lie bracket operator on vector fields (Section 6) which is instrumental for efficiently computing the derivative of the advection operator (Appendix C).
- Present applications of this machinery to optical flow on curved domains, interpolation of scalar functions, extraction of a point-to-point map from a given functional map, and realization of intrinsic symmetry maps. (Section 8).

2. Vector Fields and Flows

To formally specify our objective we briefly describe the following definitions for vector fields and their flows. In differential geometry, it is well-known [Fra11] that tangent vector fields are fully encoded through their *action* on smooth scalar functions. Given a surface M with its associated metric $\langle\cdot,\cdot\rangle$ and a smooth function $f : M \to \mathbb{R}$, the action of a vector field v on f is given by the *directional derivative* of the function:

$$v(f) = D_v(f) = \langle v, \nabla f\rangle \,, \tag{1}$$

where the inner product is computed per point $p \in M$. Vector fields and mappings are tightly linked as any tangent vector field v defines a one-parameter family of self-maps ϕ_v^t, known as the *flow* of v, which satisfies:

$$\frac{\mathrm{d}}{\mathrm{d}t}\phi_v^t = v \circ \phi_v^t, \quad \phi_v^0 = \mathrm{id} \;.$$

Advection of scalar functions is then achieved by composing f with the inverse of the flow map, i.e., $f(t) = f \circ \phi_v^{-t}$. Thus, $f(t)$ is the unique solution of the following partial differential equation,

$$\frac{\mathrm{d}}{\mathrm{d}t} f(t) = -D_v(f(t)), \quad f(0) = f \;. \tag{2}$$

The operator of advection plays a key role in our method since it allows to match between functions. We proceed by describing our approach for function matching on surfaces.

3. Advection-based Function Matching (ABFM)

A straightforward (and computationally trivial) approach to interpolating functions is to linearly blend them. However, pointwise interpolation is independent of the global structure of the functions and the underlying geometry. Moreover, if the functions are spatial deformations of one another, i.e., an associated field generates the functions (as in optical flow), linear interpolation would not produce satisfying results. These issues motivate a different approach.

Given a surface M and two scalar functions $f, g : M \to \mathbb{R}$, we seek for a time-varying tangent vector field $v(t)$ whose associated flow advects f onto g. In general, this problem is ill-defined, as

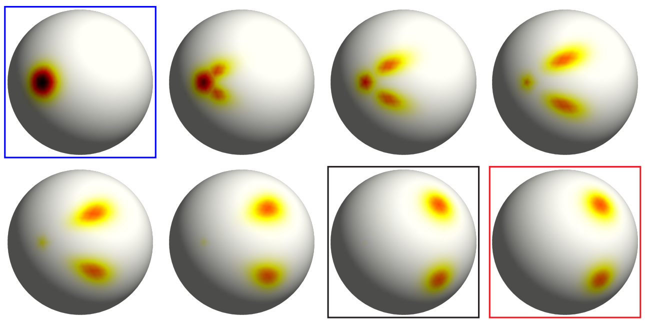
Figure 3: Given source and target functions (blue and red frames), our method matches the advected source to the target (black frame). The resulting interpolation is obtained by advecting for different times leading to spatial displacement of values.

there can be many such fields, therefore some additional regularization on the vector field is required. To this end, we design an energy functional which includes two terms: a data term that promotes the advection constraint, and a regularization term which enforces smoothness on the vector field. Our approach is inspired by the popular LDDMM framework [BMTY05], which enforces a similar functional. In Figure 3, we show matching and interpolation results computed using our method.

We propose to solve the following optimization problem:

$$\arg\min_v \frac{1}{2}\int_0^\tau \|v(t)\|_\alpha^2 \,\mathrm{d}t + \frac{1}{2\sigma^2}\|f\circ\phi_v^{-\tau} - g\|_\beta^2 \,. \qquad (3)$$

Namely, our data term seeks to minimize the norm of $f(\tau)-g$, where $f(\tau)$ satisfies Eq. (2) when advecting f using the optimized velocity. The scaling parameter σ weighs the matching against the penalty due to the regularization of the velocity, i.e., we treat the matching as a *weak* constraint.

We choose a function norm that is more suitable for measuring distances between *disjoint* functions, i.e., functions whose supports (where $f,g\neq 0$) are disjoint. Specifically, increasing the parameter β allows for interpolating functions that are farther apart. The term which regularizes vector fields also uses a modified norm, see e.g., [BMTY05], which promotes a smoother velocity as α grows. Thus, we define the following norms:

$$\|f\|_\beta^2 = \int_M f(x)\,C_\beta\, f(x)\,\mathrm{d}x\,, \qquad \|v\|_\alpha^2 = \int_M \langle v(x), D_\alpha\, v(x)\rangle\,\mathrm{d}x\,,$$

where C_β is defined as $C_\beta = \mathrm{id} - \beta\Delta_{LB}$ with Δ_{LB} the *negative-definite Laplace–Beltrami* operator. Similarly, the operator D_α is given by $D_\alpha = \mathrm{id} + \alpha\Delta_H$, where Δ_H is the *Hodge Laplacian*.

4. Discretization

The main challenge in the discretization of our objective function on triangle meshes is computing the flow map ϕ_v^t. This requires computing the flow lines of v, which is known to be a non-trivial and error prone problem on curved meshes, requiring combinatoric decisions (e.g., to which triangle should the flow line continue). Furthermore, it is not clear how one could compute the gradient of our objective functional if using such a direct approach for computing the flow lines. In the following, we propose an alternative method, which involves only the advected functions and does not require the computation of the flow lines, based on the functional representation of vector fields [ABCCO13].

Notation. We are given a triangle mesh with face set $\mathcal{F}$ and vertex set $\mathcal{V}$. We define our discretizations in matrix notation, and thus represent functions as vectors of length $|\mathcal{V}|$, and vector fields as vectors of length $3|\mathcal{F}|$. Vertex and face areas are respectively denoted by $A_\mathcal{V}\in\mathbb{R}^{|\mathcal{V}|}$ and $A_\mathcal{F}\in\mathbb{R}^{|\mathcal{F}|}$, where the area of vertex i is computed by one third of the total area of its adjacent triangles. We use diagonal mass matrices given by $G_\mathcal{V} = [A_\mathcal{V}] \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ for vertices and $G_\mathcal{F} = [A_\mathcal{F}] \in \mathbb{R}^{3|\mathcal{F}|\times 3|\mathcal{F}|}$ for faces. The bracket $[\cdot]$ operator converts vectors in $\mathbb{R}^{|\mathcal{V}|}$ and $\mathbb{R}^{|\mathcal{F}|}$ to diagonal matrices in $\mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ and $\mathbb{R}^{3|\mathcal{F}|\times 3|\mathcal{F}|}$ respectively (replicating each entry 3 times for the latter). In addition, we define the interpolation matrix $I_\mathcal{V}^\mathcal{F}\in\mathbb{R}^{|\mathcal{V}|\times|\mathcal{F}|}$ to average quantities from faces to the vertices, i.e.,

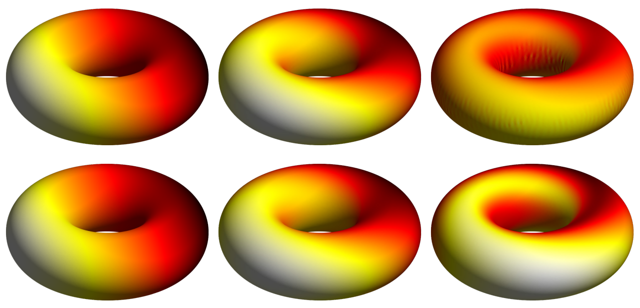

Figure 4: Linear advection of an input function (top left) works well for short times (top middle), but discretization errors in the form of oscillations appear for longer times (top right). For comparison, the non-linear transport (bottom) better approximates the flow and it yields a smooth result, even for long times (bottom right).

$I_\mathcal{V}^\mathcal{F}(i,j) = \frac{A_\mathcal{F}(j)}{3A_\mathcal{V}(i)}$, iff vertex i belongs to face j and 0 otherwise. Our differential operators, grad $\in\mathbb{R}^{3|\mathcal{F}|\times|\mathcal{V}|}$ and div $\in\mathbb{R}^{|\mathcal{V}|\times 3|\mathcal{F}|}$, are the standard ones as defined in [BKP*10, Chapter 3].

Functional Vector Fields. Following the construction introduced in [ABCCO13] and using our notation, we have that D_v and its *dual* version $\overline{D}_f$, required for derivative computations, can be computed as follows

$$D_v = I_\mathcal{V}^\mathcal{F}[v]_\bullet^T\,\mathrm{grad}\,, \quad \overline{D}_f = I_\mathcal{V}^\mathcal{F}[\mathrm{grad}\, f]_\bullet^T\,,$$

where $\overline{D}_f$ is a matrix of size $|\mathcal{V}|\times|\mathcal{F}|$ defined as the operator which satisfies $\overline{D}_f(v) = D_v(f)$. Here, $[\cdot]_\bullet\in\mathbb{R}^{3|\mathcal{F}|\times|\mathcal{F}|}$ is a block diagonal matrix which encodes a pointwise multiplication of a vector field by a face-based function, and its transpose evaluates a pointwise inner product.

Functional Advection. A particularly useful property of D_v is that discrete advection of a function f can be computed using the action of the matrix exponential on f, namely,

$$f_t = \exp(-t\,D_v)\,f = \sum_{k=0}^\infty \frac{(-t)^k}{k!} D_v^k\, f\,, \qquad (4)$$

where the action is computed in an efficient manner with methods as [AMH11]. When viewed as an operator that maps functions to their directional derivatives, the discrete D_v with the relation (4) is closely related to the functional maps framework [OBCS*12]. In this context, the operator $\exp(-t\,D_v)$ is in fact the functional map associated with the flow map of v. Finally, there are cases (e.g., in fluid simulation) where it is sufficient to use the *linearized* version of advection, i.e.,

$$f_t = (\mathrm{id} - t\,D_v)\,f\,. \qquad (5)$$

In Figure 4, we show a comparison between the linear and non-linear versions of advection. Starting from the same initial function (top and bottom, left), the linear computation (top) exhibits discretization noise, i.e., oscillationst (top right), while the non-linear discretization (bottom) provides a smooth result (bottom right).

Discrete Energy. To fully discretize problem (3), we break the time parameter into N segments of equal size $\delta\tau = \tau/N$. Thus, we optimize for a finite set of vector fields $\{v_j\}_{j=1}^N$ that are constant per time segment. Using the above definitions and matrix notations, we arrive at the following discrete optimization problem

$$\underset{\{v_j\}}{\arg\min} \left(\frac{\delta\tau}{2} \sum_{j=1}^{N} v_j^T G_{\mathcal{F}} D_\alpha v_j + \frac{1}{2\sigma^2} \delta g^T G_{\mathcal{V}} C_\beta \delta g \right) , \tag{6}$$

where $\delta g = f \circ \phi_v^{-\tau} - g$. The map $\phi_v^{-\tau}$ is obtained by composing the flow maps of the different velocities, i.e.,

$$f \circ \phi_v^{-\tau} = f \circ \phi_{v_1}^{-\delta\tau} \circ .. \circ \phi_{v_N}^{-\delta\tau} ,$$

where composition is achieved through matrix multiplication.

The transported function can be computed using the linearized flow (5) or the non-linear flow (4). In general, the linearized method is preferred in cases where the overall smoothness of advection is of less importance (see applications in Section 8 that are related to Figures. 11, 12), since the composition of discrete mappings may induce some error. Also, the linearized method yields reasonable results when the underlying deformation is small or the flow time is sufficiently short. On the other hand, non-linear advection is crucial when one requires a single vector field that generates a smooth deformation (see e.g., Figures 9, 10). In the following, we consider two scenarios: linear flows with multiple vector fields and non-linear flows with a single vector field. Hence, we have

$$f \circ \phi_v^{-\tau} = \left(\prod_{j=1}^{N} (\mathrm{id} - \delta\tau D_{v_j}) \right) f ,$$

$$\text{or} \quad f \circ \varphi_v^{-\tau} = \exp(-\tau D_v) f ,$$

where we used the notation $\varphi_v^{-\tau}$ in the non-linear case to distinguish it from the linear case. Finally, we use the notation f_t to denote the advected version of f to time t, i.e., $f_t = f \circ \phi_v^{-t}$ or $f_t = f \circ \varphi_v^{-t}$, depending on the associated flow, and $f_0 = f$.

Discrete Gradient. To solve the discrete problem (6), the gradient of the energy functional is required. Given that D_α and C_β are *self-adjoint* operators, i.e., in our case these are symmetric matrices with respect to the corresponding inner product, the derivative of the energy functional is given by

$$\frac{\partial}{\partial v_j} E = \delta\tau G_{\mathcal{F}} D_\alpha v_j + \frac{1}{\sigma^2} \left(\frac{\partial}{\partial v_j} f_t \right)^T G_{\mathcal{V}} C_\beta \delta g . \tag{7}$$

The full derivation of the gradient appears in Appendix A. Note, that the gradient depends on $\partial_{v_j} f_t$, and thus on the choice of advection operator. We provide in Appendices B and C the derivative of the advection for the linearized and non-linear flows, respectively.

5. Linear and Non-linear Advection-based Function Matching

The first scenario we consider takes $N > 1$ and employs linearized flows, i.e., uses the advection $f \circ \phi_v^{-\tau}$ as described in Section 4. To compute the directional derivative of the energy functional (7), we derive the component $\frac{\partial}{\partial v_j} f_t$ and obtain

$$\frac{\partial}{\partial v_j} f_t = -\delta\tau \left(\prod_{i=j+1}^{N} (\mathrm{id} - \delta\tau D_{v_i}) \right) \overline{D}_{f_{(j-1)\delta\tau}} , \tag{8}$$

where $f_{(j-1)\delta\tau}$ is the (partial) advection of f_0 to time $(j-1)\delta\tau$. We refer to this method as Linearized Advection-based Function Matching (LABFM) and Figures 11, and 12 were generated using this method.

While the LABFM method is fast and simple to implement, there are certain applications in which $N = 1$ is a design requirement. Thus, we extend the former method to include non-linear flows, i.e., $f \circ \varphi_v^{-\tau}$, and to optimize for a *single* vector field. The modified energy becomes

$$E(v) = \frac{\tau}{2} v^T G_{\mathcal{F}} D_\alpha v + \frac{1}{2\sigma^2} \delta g^T G_{\mathcal{V}} C_\beta \delta g ,$$

where, as before, we need to re-derive the suitable gradient of the component $\partial_v f_t$. We emphasize that computing the derivative of this expression is more involved compared to the former case. In particular, $\exp(-\tau D_v) \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is a *dense* matrix, thus using finite differencing methods or extracting a block in the exponential of a bigger operator [COC15] is possible only for small problems. Instead, we derive in Appendix C an approximation of the gradient by exploiting the relation between matrices and vector fields. We arrive at the following expression,

$$\frac{\partial}{\partial v} f_t = -\frac{\tau}{k+1} \exp(-\tau D_v) \overline{D}_{f_0} \sum_{s=0}^{k} \exp\left(\frac{s\tau}{k} \mathrm{ad}_v \right) , \tag{9}$$

where k is a scalar used to approximate a continuous integral, and ad_v is the *Lie bracket* [Fra11], an operator that acts on vector fields, i.e., $\mathrm{ad}_v u = [v,u]$, where u is a vector field. Intuitively, the bracket measures the amount of change u exhibits with respect to the flow lines of v. We defer the discussion on the operator ad_v and its exponentiated version to the next section.

6. Lie bracket of Vector Fields

The Lie derivative evaluates the change of a vector field over the flow of another vector field. Given two tangent fields v and u, we say that their flows ϕ_v^t and ϕ_u^t *commute* when their bracket is zero. Geometrically, it means that one can apply ϕ_v^t and then ϕ_u^t or the other way around and arrive at the same point. Formally, the bracket is given in operator form by:

$$D_{[v,u]} = D_v D_u - D_u D_v , \tag{10}$$

where $[v,u]$ denotes the associated vector field. Notice that under the bracket operation, vector fields form a group (since second derivatives cancel), i.e., $D_{[v,u]}$ is a directional derivative operator. We show in Figure 5 an example of the smoothest vector field u (right) that commutes with v (left).

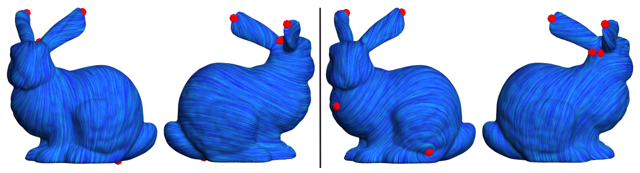

Figure 5: Given a vector field v (left), the kernel of its operator ad_v consists many vector fields that commute with v, where we show the smoothest one (right).

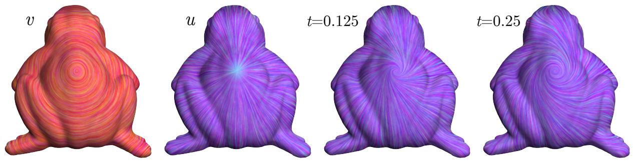

Figure 6: Transport (pushforward) of a vector field u (middle left) over the flow lines of v (left) is shown for various times $t = 0.125$ and $t = 0.25$ (middle right and right).

Representing and computing the Lie derivative on surfaces is an on-going challenge and several methods try to solve this problem. For instance, [AOCBC15] exploit the functional approach and offer an efficient representation of the bracket in a reduced spectral basis. We choose to follow [ABCCO13] as their bracket discretization is closely related to the directional derivative operators we use. Specifically, the discrete version of Eq. (10) is computed in [ABCCO13] by taking the commutator of the respective matrices. However, the resulting matrix acts on scalar functions, whereas $\text{ad}_v \in \mathbb{R}^{3|\mathcal{F}|\times 3|\mathcal{F}|}$ is an operator that takes a vector field and returns a vector field.

Nevertheless, we observe that a vector field $[v,u]$ can be extracted from its directional derivative $D_{[v,u]}$ by applying the operator on the coordinate functions x, y and z. For instance, to reconstruct the x-coordinate we compute $D_{[v,u]}(x) = [v,u]_x$. Repeating this procedure for y and z (the derivation appears in Appendix D) yields:

$$\text{ad}_v = \begin{pmatrix} \mathcal{D}_v & 0 & 0 \\ 0 & \mathcal{D}_v & 0 \\ 0 & 0 & \mathcal{D}_v \end{pmatrix} - \begin{pmatrix} \overline{\mathcal{D}}_{v_x} \\ \overline{\mathcal{D}}_{v_y} \\ \overline{\mathcal{D}}_{v_z} \end{pmatrix} \in \mathbb{R}^{3|\mathcal{F}|\times 3|\mathcal{F}|} \,, \qquad (11)$$

where $\mathcal{D}_v = [v]_\bullet^T \, \text{grad}\, I_{\mathcal{V}}^{\mathcal{F}}$ and $\overline{\mathcal{D}}_f = [\text{grad}\, I_{\mathcal{V}}^{\mathcal{F}} f]_\bullet^T$.

A novel property of ad_v is its relation to the *differential* of the flow map ϕ_v^t. The differential of a self-map ϕ_v^t generates a self-map $\text{Ad}_{\phi_v^t}$ on the tangent bundle, i.e., $\text{Ad}_{\phi_v^t}$, also known as the *push-forward*, transports vector fields to fields. For finite matrices, our representation is extremely useful since we can discretize $\text{Ad}_{\phi_v^t}$ by taking the exponential of ad_v [Hal15]:

$$\text{Ad}_{\phi_v^t}\, u = \exp(t\, \text{ad}_v)\, u \,, \qquad (12)$$

where the term $\exp(t\,\text{ad}_v)$ appears in our gradient computations (9), and we evaluate it using Eq. (11) and the matrix exponential. In Figure 6, we demonstrate the action of $\text{Ad}_{\phi_v^t}$ associated with v (left) on the field u (middle left) for several times (middle right and right).

7. Implementation Details and Limitations

We implemented our method in MATLAB using the `minFunc` routine [Sch05] which employs a quasi-Newton algorithm with L-BFGS (limited memory) updating. In Algorithm 1, we provide the function handle that computes the energy and derivative of NLABFM. This pseudo-code includes calls to exp and exp_tspan which compute the action of a matrix exponential on a vector for a specific time and for a range, respectively (see [AMH11] for further details). Notice that the gradient computation includes the transpose of the matrices $D_v, \overline{D}_f$ and ad_v. This is due to the use of $(\partial_v f_t)^T$ in the gradient (7) and since $\exp(A)^T = \exp(A^T)$ for any matrix A.

```
function NLABFM($\mathbf{f}, \mathbf{g}, \mathbf{v}$)
    // Energy computation
    $f_\tau \leftarrow \exp(-\tau, D_{\mathbf{v}}, \mathbf{f})$
    $\delta g \leftarrow f_\tau - \mathbf{g}$
    $\mathbf{E} \leftarrow \frac{\tau}{2}\mathbf{v}^T G_{\mathcal{F}} D_\alpha \mathbf{v} + \frac{1}{2\sigma^2}\delta g^T G_{\mathcal{V}} C_\beta \delta g$        // Eq. (6)

    // Gradient computation
    $d \leftarrow \overline{D}_{\mathbf{f}}^T \exp(-\tau, D_{\mathbf{v}}^T, G_{\mathcal{V}} C_\beta \delta g)$
    $d_s \leftarrow \text{exp\_tspan}(\text{ad}_{\mathbf{v}}^T, d, 0, \tau, k)$
    $\mathbf{D} \leftarrow \tau G_{\mathcal{F}} D_\alpha \mathbf{v} - \frac{\tau}{(k+1)\sigma^2}\sum_s d_s$        // Eq. (9)

    return E, D
```

Algorithm 1: Energy and gradient calculation for the Quasi-Newton iteration of the non-linear ABFM algorithm. The routines exp and exp_tspan are described in [AMH11].

Our advection method depends on the end time parameter τ. Unfortunately, the computation of the action of the matrix exponential is not stable for long times. Consequently, the optimization does not find a descent direction and stops immediately. We handle this limitation with a simple modification to the definition of the non-linear flow, i.e.,

$$f \circ \phi_v^{-\tau} = \exp(-\tau D_v) f = \left(\prod_{j=1}^{n} \exp(-\delta\tau D_v) \right) f \,,$$

where $\delta\tau = \tau/n$. Notice that we use the *same* vector field v in the product and the above relation holds up to machine precision. However, the gradient also changes in a way that is not equivalent in the discrete setting, i.e.,

$$\frac{\partial}{\partial v} f_t = -\frac{\delta\tau}{k+1} \sum_{r=1}^{n} \exp\left(-(n-r)\delta\tau D_v\right) \overline{D}_{f_{(r-1)\delta\tau}} \sum_{s=0}^{k} \exp\left(\frac{s\,\delta\tau}{k}\,\text{ad}_v\right) \,.$$

The main difference in the above expression compared to Eq. (9) is that we advect for a range of times instead of advecting only for the end time, and, in particular, advection for shorter times contributes to the computation.

Table 1 shows the parameters we used in all of our experiments. For small deformations, taking $\tau = 1$ is sufficient to achieve good results, whereas large deformations require larger τ. Increasing the parameters α and β corresponds to smoother fields and functions,

Figure	τ	α	β	σ	N	n
Fig. 1	1	1	0	$1e-2^*$	–	10
Fig. 3	2	$1e+3$	$1e-5$	$1e-4$	–	40
Fig. 7	1	$1e+3$	0	$1e-1^*$	20	–
Fig. 8	10	$1e+3$	0	$1e-2^*$	–	200
Fig. 9	1	1	$1e-4$	$5e-3$	–	20
Fig. 10	1	$1e+3$	0	$1e-2^*$	–	20
Fig. 11	1	1	0	$1e-2^*$	10	–
Fig. 12	1	1	0	$1e-2^*$	10	–

Table 1: The parameters used in our experiments. See the text for details about the effect of each parameter on the obtained results.

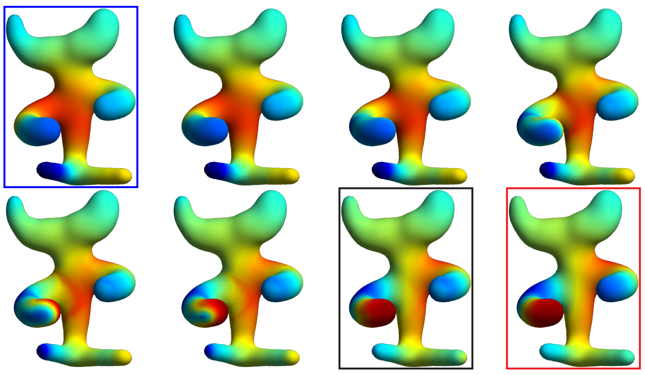

Figure 7: Interpolation results between the eigenfunctions 7 and 8 of the Laplace–Beltrami operator (blue and red frames). Notice that our result (black frame) highly matches the target function and that the interpolation path is smooth.

respectively. Finally, decreasing σ weighs the matching constraint higher with respect to the vector field constraint. In practice, we employ a “cooling” procedure for σ (denoted with asterisk in the table) to achieve better matching, i.e., σ is divided by 10 per a fixed amount of iterations.

8. Results

Spatial interpolation of functions. Fig. 3 shows an intuitive spatial interpolation between a large smooth Gaussian (blue frame) and two small smooth Gaussians (red frame). Our method yields a result (black frame) which matches the target function, where the rest of the frames are obtained by advecting the source over the resulting velocity field for different times. While our result is similar in nature to those obtained with optimal transportation techniques [SDGP*15], we stress that our method is not designed for probability distributions. In particular, our differential operators and their integrated versions do not exhibit a *maximum principle*, i.e., the advected functions are not guaranteed to be probability distributions, even if they originated from a probability distribution. Nevertheless, the drift can be minimized by taking a small σ parameter. Similarly, we show in Fig. 7 a smooth interpolation between the eigenfunctions 7 and 8 of the Laplace–Beltrami operator. Matching between eigenfunctions is important for improving maps between surfaces, as was shown in [COC15]. In addition, several mapping techniques rely on associating scalar geometric descriptors [OBCS*12]. Therefore, our method can serve as a building block in such scenarios.

Optical flow on surfaces. In cases when the target function is the advected version of the source function, one common objective is to reconstruct the underlying vector field which generated the motion. For instance, in the context of optical flow, registration between consecutive frames of a movie allows to up-sample the given signal. In Fig. 8, we show optical flow on curved surfaces where the source function (blue frame) is advected to time $\tau = 10$ (red frame) over the velocity field (top left, showing its LIC visualization and top middle left, showing its norm). Our method matches

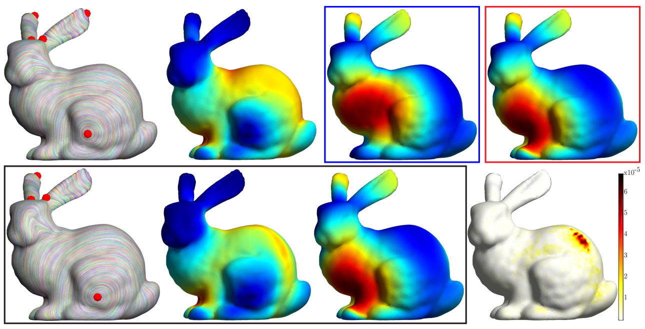

Figure 8: Our method takes as input a function (blue frame) and its advected version (red frame) computed using a vector field (top left). The output of our method is the matched function (black frame, right) and the corresponding vector field (black frame, left and middle). Notice that the resulting field highly matches the original field and the error between the function (bottom right) is very small. See the text for additional details.

the target function (black frame, right) with a vector field (black frame, left and middle) that is very close to the original field. In addition, we show the absolute error between the target function g and the matched function $f \circ \varphi_v^{-10}$, i.e., we compute pointwise $|g - f \circ \varphi_v^{-10}|$ (bottom right). Notice that both f and g are on the scale of 1, thus our matching exhibits significantly small error.

Continuous matching of symmetric surfaces. Given a surface and its symmetry map, our goal is to infer a one-parameter family of maps which continuously matches the surface to its symmetries. In Fig. 9, we take a source function (blue frame), map it with the symmetry map (red frame), and we optimize for a single vector field which matches between those functions (black frame, shown with LIC and norm of the field). The bottom row shows an RGB color coding of the coordinate functions mapped with the flow map of the velocity. The initial geometry with two points marked on top of it (left) is advected to time $\tau/2$ (middle left) and then to time τ

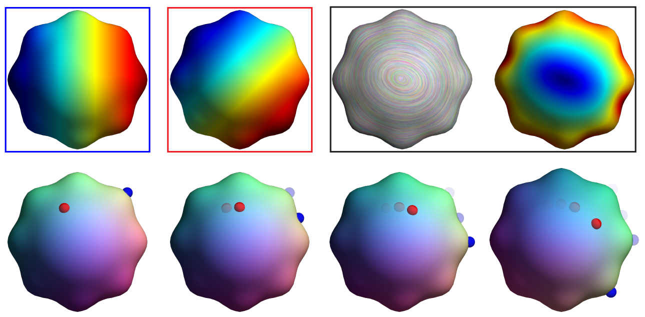

Figure 9: Our method handles large deformations between the source function (blue frame) and the target function (red frame), and succeeds in finding a single velocity field whose flow represents the continuous symmetry map (black frame). We show an RGB color coding of the surface mapped under the resulting flow for times $0, \tau/2, \tau$ and 2τ (bottom, left to right).

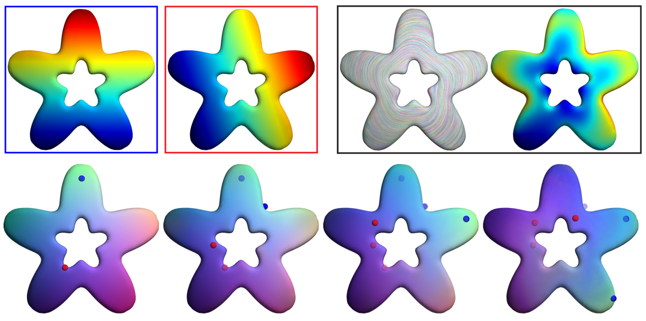

Figure 10: An experiment similar to one shown in Fig. 9 on a more complex geometry that contains creases. Nevertheless, our method finds a smooth field whose flow generates a continuous symmetric mapping.

(middle right). Notice that the points and the coordinate functions are smoothly mapped to the correct values. We additionally show an *extrapolation* of the advection to time 2τ (right), where some drift is noticeable, yet it is relatively small. Fig. 10 shows a similar experiment on a much more complex data since the geometry contains creases, thus it is not clear that the required vector field even exists. Yet, our method yields a reasonable continuous symmetric mapping which maintains the general behavior.

Function Matching for Mapping. The functional map framework [OBCS*12] provides the basis for many mapping algorithms. In this framework, one can infer pose constraints requiring functions to correspond, and compute a map taking functions to functions, which best fulfills these constraints. The effectivity of this framework is somewhat hindered by the fact that it is sometimes difficult to extract a corresponding point-to-point map if it is required. Using our framework, when working with *self-maps*, it is possible to leverage the functional map idea (mapping between corresponding functions), while *simultaneously* maintaining a point-to-point correspondence, defined using the flow map of the computed vector field. This approach was suggested in [COC15], yet as we work in the hat basis both for functions and vector fields, we are not limited to a small subspace of functions and vector fields.

We extend the setting using linearized advection of multiple vector fields to match multiple functions by summing over the errors. Figure 11 demonstrates that we can indeed match successfully between two non-smooth functions (blue frame, visualized as transported texture coordinates using a given map from the model on the left) to a smooth version of these functions (red frame, obtained by transportation with the corresponding functional map), and achieve the required functions (black frame). Moreover, we compute the pointwise sum of absolute errors between the target functions $\{g_i\}_{i=1}^2$ and the matched functions $\{f_i \circ \phi_v^\tau\}_{i=1}^2$, i.e., we plot the pointwise difference $\sum_i |g_i - f_i \circ \phi_v^\tau|$ (right). In Figure 12 we repeat this experiment on two models from the SCAPE dataset, using the functional map obtained from the ground truth correspondence to transport the functions we use as targets (which are the coordinate functions of the source mesh, two of them visualized as texture coordinates). Computing the flow using one function constraint (gray frame) and 3 function constraints (black frame) - we again match the target functions. Furthermore, we compute the point-to-point map corresponding to our flow and compare the error with respect to the ground truth, with the output of [COC15] on the same inputs. The resulting errors are shown in Figure 13, demonstrating that we improve the output point-to-point map.

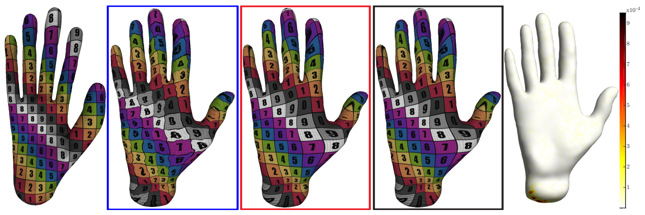

Figure 11: Matching between two non-smooth functions (blue frame) to two smooth functions (red frame) shown as texture coordinates. See the text for additional details.

9. Conclusion and Future Work

We have presented a novel method for matching scalar functions on curved triangle meshes that is based on the vector field's flow. We designed an energy minimization framework which is inspired by the well-known LDDMM algorithm and facilitates the machinery of functional vector fields. Our unique approach avoids the problematic explicit computation of flow lines and allows to advect in a linear and non-linear fashion. We showed that our matching method is applicable in scenarios of small and large deformations. We also demonstrated its effectiveness in optical flow problems, continuous matching of symmetric surfaces and in the context of the functional maps framework.

Numerous problems which are related to geometry processing can be posed as function matching problems. Thus, we believe that the generality of our framework will make it a valuable tool in many practical scenarios. In particular, we would like to extend our machinery to match and interpolate between tangent vector fields, a problem whose solutions are useful in fluid simulation techniques. Moreover, we would like to generalize our method for computing barycenters or weighted averages of scalar functions. Finally, we believe that our method can be also considered in the context of geometry-aware texture synthesis interpolation.

Acknowledgements. We thank AIM@Shape and SCAPE for the models. The authors acknowledge ISF grant 699/12, Marie Curie CIG 303511, the German-Israeli Foundation for Scientific Research and Development, Grant No: I-2378-407.6, and an Adams Fellowship.

References

[ABCCO13] AZENCOT O., BEN-CHEN M., CHAZAL F., OVSJANIKOV M.: An operator approach to tangent vector field processing. In *Computer Graphics Forum* (2013), vol. 32, Wiley Online Library, pp. 73–82. 2, 4, 6

[AMH11] AL-MOHY A. H., HIGHAM N. J.: Computing the action of the matrix exponential, with an application to exponential integrators. *SIAM journal on scientific computing 33*, 2 (2011), 488–511. 2, 4, 6

[AOCBC15] AZENCOT O., OVSJANIKOV M., CHAZAL F., BEN-CHEN M.: Discrete derivatives of vector fields on surfaces–an operator approach. *ACM Transactions on Graphics (TOG) 34*, 3 (2015), 29. 6

[AVW*15] Azencot O., Vantzos O., Wardetzky M., Rumpf M., Ben-Chen M.: Functional thin films on surfaces. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2015), ACM, pp. 137–146. 2

[AWO*14] Azencot O., Weissmann S., Ovsjanikov M., Wardetzky M., Ben-Chen M.: Functional fluids on surfaces. In *Computer Graphics Forum* (2014), vol. 33, Wiley Online Library, pp. 237–246. 2

[BB00] Benamou J.-D., Brenier Y.: A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem. *Numerische Mathematik 84*, 3 (2000), 375–393. 3

[BKP*10] Botsch M., Kobbelt L., Pauly M., Alliez P., Lévy B.: *Polygon mesh processing*. CRC press, 2010. 4

[BMTY05] Beg M. F., Miller M. I., Trouvé A., Younes L.: Computing large deformation metric mappings via geodesic flows of diffeomorphisms. *International journal of computer vision 61*, 2 (2005), 139–157. 1, 2, 4

[BVDPPH11] Bonneel N., Van De Panne M., Paris S., Heidrich W.: Displacement interpolation using lagrangian mass transport. In *ACM Transactions on Graphics (TOG)* (2011), vol. 30, ACM, p. 158. 1

[COC15] Corman E., Ovsjanikov M., Chambolle A.: Continuous matching via vector field flow. In *Computer Graphics Forum* (2015), vol. 34, Wiley Online Library, pp. 129–139. 2, 3, 5, 7, 8, 9

[Fra11] Frankel T.: *The geometry of physics: an introduction*. Cambridge University Press, 2011. 3, 5

[Hal15] Hall B. C.: *Lie groups, Lie algebras, and representations: an elementary introduction*, vol. 222. Springer, 2015. 6, 10

[LB08] Lefèvre J., Baillet S.: Optical flow and advection on 2-Riemannian manifolds: a common framework. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 30*, 6 (2008), 1081–1092. 3

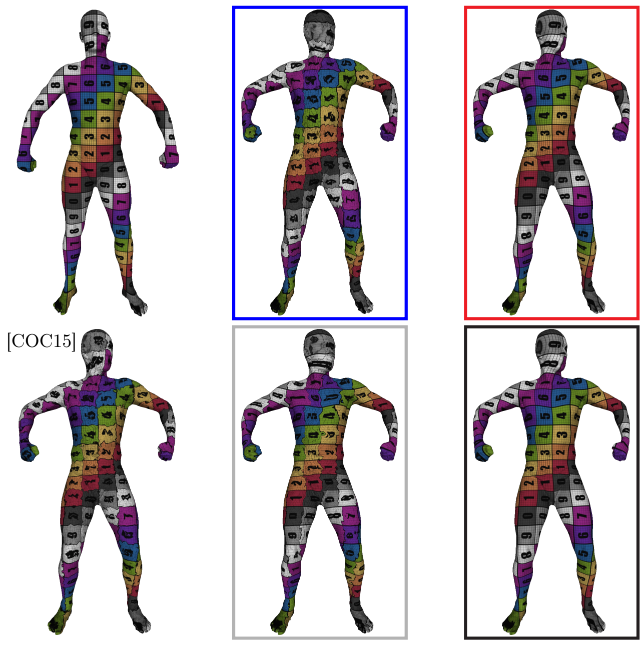

Figure 12: A similar experiment to Fig. 11 where given a set of non-smooth source functions (blue frame) and smooth target functions (red frame), we optimize for a single match (gray frame) and three matches (black frame). We compare our result to matching obtained with the method of [COC15] (bottom left).

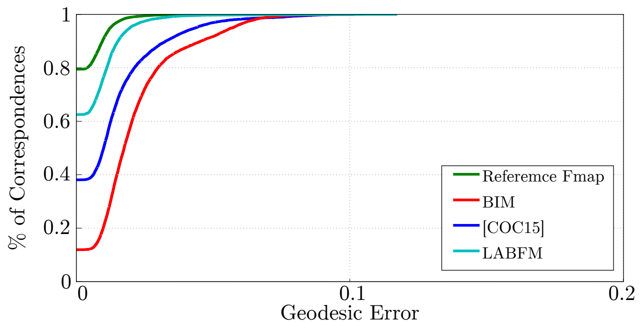

Figure 13: Our method matches only three functions on the data of Fig. 12, yet it provides a better reconstruction of the reference functional map when compared to the method [COC15].

[OBCCG13] Ovsjanikov M., Ben-Chen M., Chazal F., Guibas L.: Analysis and visualization of maps between shapes. In *Computer Graphics Forum* (2013), vol. 32, Wiley Online Library, pp. 135–145. 1

[OBCS*12] Ovsjanikov M., Ben-Chen M., Solomon J., Butscher A., Guibas L.: Functional maps: a flexible representation of maps between shapes. *ACM Transactions on Graphics (TOG) 31*, 4 (2012), 30. 1, 4, 7, 8

[PK15] Prada F., Kazhdan M.: Unconditionally stable shock filters for image and geometry processing. In *Computer Graphics Forum* (2015), vol. 34, Wiley Online Library, pp. 201–210. 2

[PZ11] Palacios J., Zhang E.: Interactive visualization of rotational symmetry fields on surfaces. *Visualization and Computer Graphics, IEEE Transactions on 17*, 7 (2011), 947–955. 2

[RTD*10] Ritschel T., Thormählen T., Dachsbacher C., Kautz J., Seidel H.-P.: Interactive on-surface signal deformation. In *ACM Transactions on Graphics (TOG)* (2010), vol. 29, ACM, p. 36. 3

[Sch05] Schmidt M.: minfunc: unconstrained differentiable multivariate optimization in matlab. `http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html`. 6

[SDGP*15] Solomon J., De Goes F., Peyré G., Cuturi M., Butscher A., Nguyen A., Du T., Guibas L.: Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics (TOG) 34*, 4 (2015), 66. 3, 7

[SDP13] Sotiras A., Davatzikos C., Paragios N.: Deformable medical image registration: A survey. *IEEE Transactions on Medical Imaging 32*, 7 (2013), 1153–1190. 2

[SNB*12] Solomon J., Nguyen A., Butscher A., Ben-Chen M., Guibas L.: Soft maps between surfaces. In *Computer Graphics Forum* (2012), vol. 31, Wiley Online Library, pp. 1617–1626. 1

[SRB14] Sun D., Roth S., Black M. J.: A quantitative analysis of current practices in optical flow estimation and the principles behind them. *International Journal of Computer Vision 106*, 2 (2014), 115–137. 2

[SRGB14] Solomon J., Rustamov R., Guibas L., Butscher A.: Earth mover's distances on discrete surfaces. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 67. 3

[SY04] Shi L., Yu Y.: Inviscid and incompressible fluid simulation on triangle meshes. *Computer Animation and Virtual Worlds 15*, 3-4 (2004), 173–181. 2

[Vil03] Villani C.: *Topics in optimal transportation*. No. 58. American Mathematical Soc., 2003. 3

[Vil08] Villani C.: *Optimal transport: old and new*, vol. 338. Springer Science & Business Media, 2008. 3

Appendix A: Directional derivatives of the discrete energy (6).

We take variations of E with respect to the velocities v_j, and identify the coefficient of v_j with the partial derivative $\frac{\partial}{\partial v_j}E$:

$$\begin{aligned}\frac{\partial}{\partial t}E(v_j+t\delta v_j)\Big|_{t=0} &= \frac{\delta\tau}{2}\,\delta v_j^T G_{\mathcal{F}} D_\alpha v_j + \frac{\delta\tau}{2}\, v_j^T G_{\mathcal{F}} D_\alpha \delta v_j \\ &+ \frac{1}{2\sigma^2}\left(\frac{\partial f_t}{\partial v_j}\delta v_j\right)^T G_{\mathcal{V}} C_\beta \,\delta g \\ &+ \frac{1}{2\sigma^2}\,\delta g G_{\mathcal{V}} C_\beta \left(\frac{\partial f_t}{\partial v_j}\delta v_j\right) \\ &= \delta\tau\,\delta v_j^T G_{\mathcal{F}} D_\alpha v_j + \frac{1}{\sigma^2}\,\delta v_j^T \left(\frac{\partial}{\partial v_j} f_t\right)^T G_{\mathcal{V}} C_\beta\, \delta g \\ &\equiv \delta v_j^T\left(\frac{\partial}{\partial v_j}E\right)\ .\end{aligned}$$

Notice that the above holds in our setup since D_α and C_β are self-adjoint operators with respect to the inner products defined by $G_{\mathcal{F}}$ and $G_{\mathcal{V}}$ respectively.

Appendix B: Directional derivatives of $f \circ \phi_v^{-\tau}$ (linear advection).

The key insight for deriving the gradient for $f \circ \phi_v^{-\tau}$ is to employ the dual operator $\overline{D}_f$ in order to extract the particular v_j. As in Appendix A, we have:

$$\begin{aligned}\frac{\partial}{\partial t} f\circ\phi_{v+t\delta v}^{-\tau}\Big|_{t=0} &= \frac{\partial}{\partial t}\prod_{j=1}^{N}\left(\mathrm{id}-\delta\tau D_{v_j+t\delta v_j}\right) f\Big|_{t=0} \\ &= \frac{\partial}{\partial t}\prod_{j=1}^{N}\left(\mathrm{id}-\delta\tau D_{v_j}-t\,\delta\tau D_{\delta v_j}\right) f\Big|_{t=0} \\ = \sum_{j=1}^{N}&\left(\prod_{i=j+1}^{N}(\mathrm{id}-\delta\tau D_{v_i})\right)(-\delta\tau D_{\delta v_j})\left(\prod_{i=1}^{j-1}(\mathrm{id}-\delta\tau D_{v_i})\right) f \\ &= -\delta\tau\sum_{j=1}^{N}\left(\prod_{i=j+1}^{N}(\mathrm{id}-\delta\tau D_{v_i})\right) D_{\delta v_j} f_{(j-1)\delta\tau} \\ &= -\delta\tau\sum_{j=1}^{N}\left(\prod_{i=j+1}^{N}(\mathrm{id}-\delta\tau D_{v_i})\right)\overline{D}_{f_{(j-1)\delta\tau}}\delta v_j \\ &\equiv \sum_{j=1}^{N}\frac{\partial f\circ\phi_v^{-\tau}}{\partial v_j}\delta v_j\end{aligned}$$

Appendix C: Directional derivative of $f \circ \varphi_v^{-\tau}$ (non-linear advection).

In what follows, we describe our approximation to the derivative of $f \circ \varphi_v^{-\tau}$. Notice that for finite matrix groups, a direct differentiation is available (see e.g., [Hal15]). However, the resulting expression is not computationally tractable as it contains an exponential of a $9|\mathcal{F}|^2 \times 9|\mathcal{F}|^2$ matrix. We, on the other hand, facilitate the discrete relation between vector fields and matrices and thus obtain an efficient yet approximate expression for the derivative.

$$\begin{aligned}&\frac{\partial}{\partial t}\left(\frac{1}{2\sigma^2}\left\|\exp(-\tau D_{v+t\,\delta v})f-g\right\|_\beta^2\right)\Big|_{t=0} = \\ &\frac{1}{\sigma^2}\left\langle \exp(-\tau D_{v+t\,\delta v})f-g, \left[\frac{\partial}{\partial t}\left(\exp(-\tau D_{v+t\,\delta v})f-g\right)\right]\Big|_{t=0}\right\rangle_\beta = \\ &\frac{1}{\sigma^2}\left\langle \exp(-\tau D_v)f-g, \left[\frac{\partial}{\partial t}\exp(-\tau D_{v+t\,\delta v})\right]\Big|_{t=0} f\right\rangle_\beta =^{(1)} \\ &\frac{1}{\sigma^2}\left\langle \delta g, \exp(-\tau D_v)\left[\int_0^1 \exp(-\,\mathrm{ad}_{-s\tau D_v}) D_{-\tau\delta v}\,\mathrm{d}s\right] f\right\rangle_\beta =^{(2)} \\ &\frac{-\tau}{\sigma^2}\left\langle \delta g, \exp(-\tau D_v)\left[\int_0^1 D_{\exp(s\tau\,\mathrm{ad}_v)\delta v}\,\mathrm{d}s\right] f\right\rangle_\beta = \\ &\frac{-\tau}{\sigma^2}\left\langle \delta g, \exp(-\tau D_v)\int_0^1 \overline{D}_f \exp(s\tau\,\mathrm{ad}_v)\delta v\,\mathrm{d}s\right\rangle_\beta = \\ &\frac{-\tau}{\sigma^2}\left\langle \delta g, \exp(-\tau D_v)\overline{D}_f\int_0^1 \exp(s\tau\,\mathrm{ad}_v)\,\mathrm{d}s\delta v\right\rangle_\beta =^{(3)} \\ &\frac{-\tau}{(k+1)\sigma^2}\left\langle \delta g, \exp(-\tau D_v)\overline{D}_f\sum_{s=0}^{k}\exp\left(\frac{s\tau}{k}\mathrm{ad}_v\right)\delta v\right\rangle_\beta\ .\end{aligned}$$

The proof for (1) is given in [Hal15] and (3) is a simple averaging rule for approximating the continuous integral with a finite sum. The pass in (2) states $\exp(\mathrm{ad}_{D_v})D_u = D_{\exp(\mathrm{ad}_v)u}$, i.e., applying this operation to the matrices D_v and D_u is the same as acting the on vector fields v and u. In the discrete setting this relation does not hold, thus pass (2) can be considered as an approximation of the required computation.

Appendix D: Construction of the operator ad_v.

To derive Eq. (11), we employ the following observations. As the current D_v operates on vertex-based functions, yet ad_v is expected to act on vector fields, we define $\mathcal{D}_v = [v]_\bullet^T\,\mathrm{grad}\,I_{\mathcal{V}}^{\mathcal{F}}$, an operator on face-based functions. Moreover, a vector field $v = (v_x, v_y, v_z)$ can be reconstructed by applying its directional derivative operator on the coordinate functions of the surface. Namely,

$$\mathcal{D}_v(x) = v_x\ ,\ \mathcal{D}_v(y) = v_y\ ,\ \mathcal{D}_v(z) = v_z\ ,$$

where any point $p \in M$ is given by $(x_p, y_p, z_p) \in \mathbb{R}^3$. Thus, using the above observations we obtain,

$$\begin{aligned}\mathcal{D}_{[v,u]}(x) &= \mathcal{D}_v\mathcal{D}_u(x) - \mathcal{D}_u\mathcal{D}_v(x) \\ &= \mathcal{D}_v(u_x) - \mathcal{D}_u(v_x) \\ &= \mathcal{D}_v(u_x) - \overline{\mathcal{D}}_{v_x}(u)\ ,\end{aligned}$$

where $\overline{\mathcal{D}}_f = [\mathrm{grad}\,I_{\mathcal{V}}^{\mathcal{F}} f]_\bullet^T$. Finally, applying the above argument to the coordinate functions y and z yields,

$$\begin{aligned}\mathrm{ad}_v(u) &= [v,u] \\ &= \left(\mathcal{D}_{[v,u]}(x), \mathcal{D}_{[v,u]}(y), \mathcal{D}_{[v,u]}(z)\right) \\ &= \left(\mathcal{D}_v(u_x) - \overline{\mathcal{D}}_{v_x}(u), \mathcal{D}_v(u_y) - \overline{\mathcal{D}}_{v_y}(u), \mathcal{D}_v(u_z) - \overline{\mathcal{D}}_{v_z}(u)\right)\ ,\end{aligned}$$

where Eq. (11) is the matrix form of the above computation.