Published in final edited form as:
SIAM J Sci Comput. 2017 ; 39(5): B860–B885. doi:10.1137/17M1114132.

A LAGRANGIAN GAUSS–NEWTON–KRYLOV SOLVER FOR MASS- AND INTENSITY-PRESERVING DIFFEOMORPHIC IMAGE REGISTRATION

ANDREAS MANG[*] and **LARS RUTHOTTO**[†]

[*]Institute for Computational Engineering and Sciences, University of Texas at Austin, Austin, Texas, USA. (AM is now with the Department of Mathematics at the University of Houston)

[†]Department of Mathematics and Computer Science, Emory University, Atlanta, Georgia, USA

Abstract

We present an efficient solver for diffeomorphic image registration problems in the framework of *Large Deformations Diffeomorphic Metric Mappings* (LDDMM). We use an optimal control formulation, in which the velocity field of a hyperbolic PDE needs to be found such that the distance between the final state of the system (the transformed/transported template image) and the observation (the reference image) is minimized. Our solver supports both stationary and non-stationary (i.e., transient or time-dependent) velocity fields. As transformation models, we consider both the transport equation (assuming intensities are preserved during the deformation) and the continuity equation (assuming mass-preservation).

We consider the reduced form of the optimal control problem and solve the resulting unconstrained optimization problem using a discretize-then-optimize approach. A key contribution is the elimination of the PDE constraint using a Lagrangian hyperbolic PDE solver. Lagrangian methods rely on the concept of characteristic curves. We approximate these curves using a fourth-order Runge-Kutta method. We also present an efficient algorithm for computing the derivatives of the final state of the system with respect to the velocity field. This allows us to use fast Gauss-Newton based methods. We present quickly converging iterative linear solvers using spectral preconditioners that render the overall optimization efficient and scalable. Our method is embedded into the image registration framework FAIR and, thus, supports the most commonly used similarity measures and regularization functionals. We demonstrate the potential of our new approach using several synthetic and real world test problems with up to 14.7 million degrees of freedom.

Keywords

Diffeomorphic Image Registration; Large Deformation Diffeomorphic Metric Mapping; Optimal Control; PDE-Constrained Optimization; Lagrangian Methods

AMS subject classifications

68U10; 49J20; 35Q93; 65M32; 76D55; 65K10

1. Introduction

In this paper, we present efficient numerical methods for diffeomorphic image registration in the framework of *Large Deformation Diffeomorphic Metric Mapping* (**LDDMM**) [65, 23, 9]. We use an optimal control formulation similar to the one in [11, 12, 10, 48, 50, 49]. Here, the task is to find a smooth velocity field v such that the distance between two images (or densities), $\mathcal{T}$ (the *template image*) and $\mathcal{R}$ (the *reference image*), is minimized, subject to some regularization norm for v and a transformation model, given by a hyperbolic partial differential equation (**PDE**) that models the deformation of $\mathcal{T}$. We consider the *transport equation* (assuming intensities are related at corresponding points) and the *continuity equation* (assuming that mass is preserved) as constraints. The connection to traditional image registration formulations [53, 54] is that a sufficiently smooth velocity field v gives rise to a diffeomorphism y via the method of characteristics. Vice versa, representing diffeomorphisms through velocity fields has been used for efficient statistical analysis; see, e.g., [3].

Solving the variational problem associated with LDDMM is, in theory, known to yield a diffeomorphic transformation y if v is sufficiently smooth [23, 65, 9]. Although, the theory of diffeomorphic registration using LDDMM is well explored [52, 70, 71], efficient numerical optimization is not. Until recently [5, 39, 48, 50, 49] mostly first-order optimization methods were used; see, e.g., [9, 67, 17, 58]. A key component in LDDMM is the numerical method for solving the hyperbolic PDE. Hyperbolic PDE solvers can be roughly divided into Eulerian (in which the density is discretized at the same locations for each time point) and Lagrangian (in which the grid moves over time along the characteristic curves) solvers; see also [45, 25, 46]. Intermediates are Semi-Lagrangian (**SL**) methods, which follow the velocity for a short time step and then estimate the density at fixed points. In this work, we use a Lagrangian solver.

It is well known that explicit Eulerian methods for hyperbolic PDEs require the size of the time step to be sufficiently small to ensure numerical stability. The maximal admissible time step size depends on the accuracy of the spatial discretization and the magnitude of the velocities. In optimal control problems, like ours, the velocity field v is not known a priori and, thus, it is difficult to come up with an efficient and stable choice of the time step. SL and Lagrangian solvers are explicit methods that, unlike explicit Eulerian methods, are stable without a restriction on the maximal admissible time step size. One drawback of SL methods is their memory requirements. For efficient derivative (or sensitivity) computations in Gauss–Newton type optimization schemes, we have to store intermediate images [51]. Further, SL methods require a repeated interpolation of the initial image and may therefore introduce severe dissipation if implemented naively.[1] As we will see, Lagrangian methods

[1]We note, that interpolation errors and numerical diffusion can be minimized, by, e.g., applying high-order interpolation schemes and/or evaluating the interpolation on a finer grid; this is costly.

require only one image interpolation at the final time. Secondly, derivatives of the Lagrangian solver can be obtained efficiently, without storing intermediate variables. A feature of SL and Lagrangian methods is that they can be easily modified to solve intensity- and mass-preserving problems, since the characteristic curves coincide.

The key idea of our work is to use a *discretize-then-optimize* strategy based on a Lagrangian hyperbolic PDE solver to efficiently solve the reduced formulation of the PDE-constrained optimization problem arising in LDDMM. We show that Lagrangian methods lead to a finite dimensional optimization problem that can be solved using an inexact Gauss–Newton method. Our PDE solver requires numerical computation of the characteristic curves. We use a fourth-order Runge–Kutta (**RK4**) method to numerically approximate the transformation y that is associated with a, in general non-stationary, velocity field v. As we show, derivatives of the transformation with respect to v can be derived analytically and computed efficiently. Due to the hyperbolic nature of the PDE, the derivatives can be represented as sparse matrices; the procedure can be paralellelized: characteristics starting at different points can be computed independently. Given these characteristics, the hyperbolic PDEs can be solved by a single interpolation step (for the advection equation) or the particle-in-cell method (for the continuity equation).

1.1. Contributions

- We propose a discretize-optimize method for solving LDDMM using a Lagrangian hyperbolic PDE solver. Our scheme is based on an RK4 method to approximate the characteristic curves and we derive an efficient algorithm for computing the derivative of the solver with respect to the velocities.
- The storage requirement of our method is independent on the number of time steps used in the numerical solver. Also, the Hessian of the objective function can be build explicitly at moderate costs, which is useful, e.g., to accelerate matrix-vector products. In this work, we use and numerically study the convergence of spectral preconditioners to iteratively solve the Gauss–Newton system.
- We extend the LDDMM framework to mass-preserving registration, which has been proved to be an adequate model for many relevant biomedical applications involving with density images, e.g., in [16, 59, 21, 30, 61].
- We derive a flexible framework supporting both stationary and non-stationary velocity fields. Our methods are embedded into the FAIR framework [54]. This allows us to consider different regularization norms and distance measures. Our implementation is freely available as an add-on to FAIR at: https://github.com/C4IR/FAIR.m/tree/master/add-ons/LagLDDMM
- We provide detailed numerical experiments on four different data sets that demonstrate the flexibility and effectiveness of our method. We show that our prototype implementation is competitive to state-of-the-art packages for diffeomorphic image registration [15, 60]. We study registration quality for

synthetic benchmark problems and real-world applications leading to optimization problem with up to 14.7 million degrees of freedom.

1.2. Related Work

We limit this review to work closely related to ours. For a general insight into the area of image registration, its applications, and its formulation we refer to [53, 54, 63]. Our work builds upon the LDDMM framework described in [23, 65, 9], which is based on the pioneering work on velocity-based fluid registration described in [20]. We adopt an optimal control point of view; we also do not directly invert for the transformation y but for its velocity v. We arrive at a hyperbolic PDE-constrained optimization problem. We refer to [31, 43, 41, 13] for a general introduction into optimal control theory and developments in PDE-constrained optimization. Related optimal control formulations for diffeomorphic image registration can, e.g., be found in [11, 12, 38, 10, 17, 48, 50, 49, 51]. Other formulations for velocity-based diffeomorphic image registration are described in [4, 66, 5]. Our work also shares characteristics with optical flow formulations [11, 12, 17]. Our formulation for mass-preserving registration problems is related to the Monge–Kantorovich functional arising in optimal mass transport [10, 35].

Most work on large deformation diffeomorphic image registration still considers first-order methods for numerical optimization (see, e.g., [6, 7, 8, 11, 9, 38, 44, 17, 67]); the exceptions are [5, 48, 50, 49, 66]. First-order schemes for numerical optimization do in general require a larger number of iterations than Newton type optimization schemes.[2] The work in [5] uses geodesic shooting and estimates the initial value of a non-stationary velocity field that parameterizes the diffeomorphism y. Other approaches that reduce the size of the optimization problem are based on stationary velocity fields; see, e.g., [40, 47, 50, 49, 51].

PDE-constrained optimization commonly requires a repeated solution of the forward problem. Thus, the design of an efficient forward solver is critical. The approaches described in [10, 17, 48, 50, 49, 38] are based on an Eulerian formulation. They employ explicit high-order schemes [11, 12, 48, 50, 38, 58], which suffer from a restriction on a maximally admissible time step, implicit schemes [10], or explicit SL schemes [49, 51, 17]. The latter were originally proposed in the context of weather prediction [64]. They are a hybrid between Lagrangian and Eulerian schemes, and unconditionally stable. SL schemes have been used in the context of Lagrangian formulations for diffeomorphic image registration (to compute the characteristics) [9, 40]. Conditionally stable schemes require small time steps, which can result in a significant amount of memory that needs to be allocated to store the time-space fields necessary to evaluate the gradient or Hessian. This makes a direct application of these type of methods to large-scale 3D problems challenging. One remedy is to turn to parallel architectures and use sophisticated checkpointing schemes in time to reduce the amount of memory that has to be allocated; see, e.g., [1]. Implicit schemes typically suffer from severe numerical diffusion. The same is true for straightforward implementations of SL schemes. Pure Lagrangian schemes for diffeomorphic image

[2]We note that in LDDMM most implementations use a gradient descent scheme in the Sobolev space induced by the regularization operator (dual space). This leads to a significant speedup compared to standard gradient descent approaches. However, it has been demonstrated experimentally that Gauss–Newton–Krylov methods are superior [48].

registration have been described in [6, 7, 8]. The time integration for computing the characteristics in [6, 7, 8] is based on a first-order explicit scheme.

What sets our work apart is the *numerical solver* and the generalization of our formulation to both the *transport* and the *continuity equation*. Most existing works on optimal control formulations for diffeomorphic image registration consider an optimize-then-discretize approach [17, 48, 50, 49, 51]. We use a discretize-then-optimize strategy instead.[3] Similar to [48], we describe a method that can handle stationary and non-stationary velocity fields. Our numerical scheme is, likewise to [6, 7, 8], based on a purely Lagrangian approach. We consider a reduced formulation of the PDE-constrained optimization problem arising in LDDMM, i.e., we eliminate the hyperbolic PDE constraint (state equation) from the variational problem. We use a Lagrangian solver to parameterize the final state in terms of the velocity. In doing so we avoid many of the complications we reviewed above: our method is unconditionally stable, limits numerical diffusion, and does not require the storage of multiple space-time fields for the evaluation of the gradient or Hessian operators. The work in [6, 7, 8] uses first-order information for numerical optimization and a first-order accurate explicit time integrator to compute the characteristic. We use a fourth-order, explicit RK scheme instead. We derive expressions for the exact derivative of the characteristics. We arrive at a Gauss–Newton–Krylov scheme that—combined with an efficient iterative linear solver—yields an approximate solution within a few iterations, and has an overall algorithmic complexity of $\mathcal{O}(n\log(n))$, where n is the dimension of the discretized velocity field (i.e., the number of unknowns).

2. Mathematical Formulation

We describe the variational optimal control formulation of the LDDMM problem next. We denote the image domain by $\Omega \subset \mathbb{R}^d$, where $d \in \{2, 3\}$ represents the spatial dimension. We assume that the *template* and the *reference image*, denoted by $\mathcal{T}: \Omega \to \mathbb{R}$ and $\mathcal{R}: \Omega \to \mathbb{R}$, are compactly supported on Ω and continuously differentiable. Given these two images, the task of image registration is to find a *plausible* transformation $y: \Omega \to \mathbb{R}^d$ so that the transformed template image $\mathcal{T} \circ y$ becomes *similar* to the reference image $\mathcal{R}$ [54]. The definitions of *plausibility*, *similarity*, and the *transformation model* depend on the context; see [54, 53, 63] for examples. Many relevant applications, e.g., in medical imaging, require that plausible transformations are diffeomorphic, i.e., smooth mappings with a smooth inverse. One framework that contains the most commonly used definitions of these three terms within the medical imaging application domain is LDDMM [23, 65, 9]. The variational optimal control formulation of the LDDMM problem can, in general format, be stated as follows: min

$$\min_{v,u}\{\mathcal{J}(v,u) := \mathcal{D}(u(\cdot,1),\mathcal{R}) + \alpha\mathcal{S}(v)\} \quad \text{subject to} \quad \mathcal{C}(v,u) = 0, \tag{2.1}$$

[3]Advantages and disadvantages of these two techniques are discussed, e.g., in [31].

where $\mathscr{D}$ is a distance (or similarity) measure, $\mathscr{S}$ is a regularizer (smoother), $v : \Omega \times [0, 1] \to \mathbb{R}^d$ is the sought after velocity field, and $u : \Omega \times [0, 1] \to \mathbb{R}$ is a time series of images. In an optimal control context, v is commonly referred to as the *control variable* and u as the *state variable*. Here, $\alpha > 0$ is a regularization parameter that balances between minimizing the image distance and the smoothness of the velocity field (and consequently controls the properties of the resulting transformation). In the current work, we assume that α is chosen by the user and refer to [36, 34, 68] for some works on automatic selection of the parameters.[4]

In this work, we assume that the constraint $\mathscr{C}$, which describes the transformation model, is given either by the *advection* (also called *transport*) equation

$$\mathscr{C}(u,v) = \begin{cases} \partial_t u(x,t) + v(x,t) \cdot \nabla u(x,t) = 0, \\ u(x,0) = \mathscr{T}(x) \end{cases} \qquad (2.2)$$

or the *continuity* equation

$$\mathscr{C}(u,v) = \begin{cases} \partial_t u(x,t) + \nabla \cdot (u(x,t) v(x,t)) = 0, \\ u(x,0) = \mathscr{T}(x). \end{cases} \qquad (2.3)$$

The former assumes intensity values are preserved during the transformation; the latter preserves the overall mass of the image. The choice of the transformation model depends on the application. Intensity preservation is commonly used, e.g., for registration of medical images acquired from different subjects [56]. Mass-preservation has been successfully used, e.g., for motion correction in position emission tomography (**PET**) [21, 30] or artifact correction of magnetic resonance imaging (**MRI**) [16, 61].

The models in (2.2) and (2.3) can be used to establish point-to-point correspondences between the template and the reference image. One way of showing this is the method of characteristics [46, 24]. To better illustrate this, we consider the advection equation (2.2), for which the intensity u is constant along the characteristics. This means that, for all $y_0 \in \Omega$ and $t \in [0, 1]$, it holds that $u(y(v, y_0, 0, t), t) = \mathscr{T}(y_0)$ where the characteristic curve $t \mapsto y(v, y_0, 0, t)$ satisfies

$$\partial_t y(v, y_0, 0, t) = v(y(v, y_0, 0, t), t) \quad \text{and} \quad y(v, y_0, 0, 0) = y_0. \qquad (2.4)$$

Similarly, the characteristics can be traced backwards in time to compute the state at some point $y_1 \in \mathbb{R}^d$ at $t = 1$. The position of the point is given by $t \mapsto y(v, y_1, 1, t)$, which satisfies

[4]Examples for an automatic selection of the regularization parameter in the context of image registration can, e.g., be found in [33, 48].

(2.4) with final time condition $y(v, y_1, 1, 1) = y_1$. Clearly, both operations are inverse to one another. That is, for all $x \in \mathbb{R}^n$ it holds that $y(v, y(v, x, 0, 1), 1, 0) = x$, i.e., a composition of both maps yields identity.

Note that (2.2) through (2.4) involve a non-stationary (i.e., time-dependent) velocity field v. This is a key assumption in the original LDDMM formulation [9]. To make the problem computational tractable it is, however, often assumed that v is stationary (*stationary velocity field* based registration) [3, 4, 40, 50, 51, 57].[5] The numerical framework we propose in this paper can efficiently handle both stationary and non-stationary velocities. As demonstrated in our numerical experiments in Sec. 4.2, the stationary model is less flexible in that we can only invert for a subset of the deformation maps living on the manifold of diffeomorphisms. Our experiments also suggest that stationary velocity fields are adequate for registration problems involving two topologically similar images, yielding little to no difference in the recovered deformation map [4, 40, 48]. However, using non-stationary velocity fields may become critical in applications involving large and highly nonlinear transformations and/or the registration of time series of images with large motion between time frames (e.g., typically seen in tracking or optical flow problems).

2.1. Regularization functionals

Due to the ill-posedness of the image registration problem, the literature on regularization in image registration is rich; see, e.g., [53, 27, 54, 63] for extensive overviews. It is established that the existence and regularity of a diffeomorphic map y depends on the smoothness of the velocity field v as well as the smoothness of the images $\mathcal{R}$ and $\mathcal{T}$ [23, 65, 50, 17]. Modeling the images as functions of bounded variation, H^3-regularity [17] is required (assuming that v is divergence free). For continuous images we can relax the H^3-regularity to an H^2-regularity [9] or—under additional assumptions on the divergence of v—even to an H^1-regularity [17].[6] Most implementations for LDDMM consider an H^2-norm for $\mathcal{S}$ in (2.1) (or an approximation based on a Gaussian kernel within a gradient descent scheme in the Sobolev space induced by the regularization norm); see, e.g., [5, 9, 40].

In our framework, we regard the regularizer as a modular component that can be replaced or extended. In practical applications we control the regularization parameter by monitoring the Jacobian in an attempt to generate transformations that are diffeomorphic (in a discrete setting) and yield high-fidelity (low mismatch) results. In our numerical examples, we consider H^1- and H^2-seminorms as regularization models. These type of regularization models are also referred to as *diffusive* or *curvature* regularizers in the context of traditional variational image registration formulations [26, 53, 54], respectively. Assuming that v is non-stationary, we have

[5]Another strategy to reduce the computational burden is to invert for an initial momentum [67] that encodes the trajectory of the diffeomorphism.

[6]We use interpolation and padding of the discrete image data to obtain continuously differentiable and compactly supported functions (see Sec. 3).

$$\mathscr{S}^{\mathrm{diff}}(v)=\frac{1}{2}\int_0^1\int_\Omega\sum_{k=1}^{d}\left|\nabla v^k(x,t)\right|^2\mathrm{d}x\mathrm{d}t$$

and

$$\mathscr{S}^{\mathrm{curv}}(v)=\frac{1}{2}\int_0^1\int_\Omega\sum_{k=1}^{d}\left|\Delta v^k(x,t)\right|^2\mathrm{d}x\mathrm{d}t,$$

respectively. Regularization of stationary velocity fields is along the same lines, by simply dropping the time integration in the above equations. Our formulation can also be extended to enforce smoothness in time, as also done in [11].

3. Numerical Methods

In this section, we describe a discretize-then-optimize approach for solving the variational problem (2.1). We eliminate the hyperbolic PDE constraints (2.2) and (2.3) using Lagrangian methods. We then describe the discretization of the objective functional itself. Following [54], we consider a multilevel Gauss–Newton method that allows us to efficiently solve the discrete optimization problem. Finally, we give some details about our implementation as an extension to the FAIR toolbox [54].

Assume, for simplicity, that the domain $\Omega = (0, 1)^d$ is divided into a regular mesh of m cells of edge length $h = 1/m$ along each coordinate direction. We use interpolation and padding of the discrete image data to obtain continuously differentiable and compactly supported functions. We approximate integrals in (2.1) by a midpoint quadrature rule, which requires evaluating the final state on the cell-centered points, $\mathbf{x}_c \in \mathbb{R}^{d \cdot m^d}$, of the grid. Without loss of generality, we assume that the velocity field v is discretized on the same mesh. However, the domain size and number of cells can be varied in practice. The discrete velocity field, denoted by $\mathbf{v} \in \mathbb{R}^n$, is discretized in time at the nodes of a regular grid with n_t cells and in space at cell-centered grid points. The total number of unknowns is $n = (n_t + 1) \cdot d \cdot (m^d)$.

3.1. Lagrangian Methods for Hyperbolic PDEs

Lagrangian methods exploit the fact that solutions to hyperbolic PDEs evolve along characteristic curves [46, 24]. These methods are Lagrangian in the sense that the transport of the density is referred to in the moving coordinate system. Lagrangian methods typically consist of two steps: First, the characteristics are computed numerically. Then, in a second step, the final image (or density) is computed. While the characteristic curves are identical for the advection and the continuity equation, the computation of the second part is not.

Step 1 (Computing the Characteristics)—Here, we describe our implementation for computing the characteristic curve passing through a given point. The velocity field $v : \Omega \times [0, 1] \to \mathbb{R}^d$ is known and represented by the coefficients $\mathbf{v} \in \mathbb{R}^n$. We solve (2.4) using an RK4 method with N equidistant time steps of size $\Delta t = \pm 1/N$; the sign of the time step depends on whether the characteristics are computed forward or backward in time. Note that the number of time steps N for the RK4 method and the number of cells n^t in the space-time grid do not necessarily have to be equal. The former is a parameter of the numerical solver and controls the accuracy of the characteristics. The latter is a modeling parameter and ultimately controls the search space for the transformation y.

Our choice of an RK4 scheme is motivated by accuracy considerations for computing the characteristics. For simplicity, we illustrate the concept of integrating v based on a first-order forward Euler scheme. The derivation of our RK4 scheme is along the same lines; it is outlined in Algorithm 1.

Let $\mathbf{x} \in \mathbb{R}^{d \cdot n_p}$ be the coordinates of the start (or end) points of the characteristics, e.g., the cell-centers of a regular mesh. Introducing the (time-dependent) transformation $y : \mathbb{R}^n \times \mathbb{R}^{d \cdot n_p} \times [0, 1]^2 \to \mathbb{R}^{d \cdot n_p}$, and imposing the initial condition $y(\mathbf{v}, \mathbf{x}, 0, 0) = \mathbf{x}$, we compute

$$y(\mathbf{v},\mathbf{x},0,t_{k+1})=y(\mathbf{v},\mathbf{x},0,t_k)+\Delta t\, I(\mathbf{v},y(\mathbf{v},\mathbf{x},0,t_k),t_k), \quad \forall k=0,1,\dots,N-1, \tag{3.1}$$

where $t_k = k\Delta t$ are the time points and I interpolates the velocity field $\mathbf{v}$ at the transformed points y. In our experiments, we use a bi- or trilinear interpolation model in space and a linear interpolation model in time, applied separately to each component of the velocity field $\mathbf{v}$. We found by experimentation that using low-order interpolation schemes for the (smoothness regularized) velocity fields is sufficiently accurate for our numerical scheme to yield high-fidelity results in practical applications. We note that the interpolation is a modular component; it can be replaced by (computationally more expensive) higher-order methods. Notice, that the positions at previous time steps, $y(\mathbf{v}, \mathbf{x}, 0, 0), \dots , y(\mathbf{v}, \mathbf{x}, 0, t_{k-1})$, do not enter the computation in (3.1); the memory requirements of our numerical scheme are independent of the number of time steps N. The end points of the characteristics are then $y(\mathbf{v}, \mathbf{x}, 0,1)$, which, if $\mathbf{x} = \mathbf{x}_c$, can also be interpreted and visualized as a deformed regular grid.

Step 2 (Solving the hyperbolic PDE)—While solutions to both the transport and the continuity equations evolve along the same characteristics, the steps for computing the transported quantity at $t = 1$ vary. Thus, we discuss both cases separately. An illustration of both schemes can be found in Fig. 3.1.

Transport equation: Considering the transport equation (2.2), we compute the intensities of the advected image $\mathscr{T}$ on the deformed cell-centered grid $\mathbf{x}_c$ by following the characteristics backwards in time. This yields

$$u(\mathbf{x}_c,1)=\mathscr{T}(y(\mathbf{v},\mathbf{x}_c,1,0)). \tag{3.2}$$

Algorithm 1

RK4 method for computing the characteristics $\mathbf{y}$ and the derivative $d_\mathbf{v}\mathbf{y}$.

```
Input: Discrete (non-stationary) velocity field $\mathbf{v} \in \mathbb{R}^n$, start points of characteristics $\mathbf{x} \in \mathbb{R}^{d \cdot n_p}$, number of time steps $N$
Set: $\mathbf{y} \leftarrow \mathbf{x}$, $d_\mathbf{v}\mathbf{y} \leftarrow 0$, $\Delta t \leftarrow 1/N$.
for $k = 0, 1, \ldots, N-1$ do
  compute $\mathbf{v}_1 \leftarrow I(\mathbf{v}, \mathbf{y}, 0, t_k)$ (spatio-temporal vector field interpolation) and set $\mathbf{y}_1 \leftarrow \mathbf{y} + (\Delta t/2)\mathbf{v}_1$
  compute $\mathbf{v}_2 \leftarrow I(\mathbf{v}, \mathbf{y}_1, 0, t_{k+1/2})$ and set $\mathbf{y}_2 \leftarrow \mathbf{y} + (\Delta t/2)\mathbf{v}_2$
  compute $\mathbf{v}_3 \leftarrow I(\mathbf{v}, \mathbf{y}_2, 0, t_{k+1/2})$ and set $\mathbf{y}_3 \leftarrow \mathbf{y} + (\Delta t/2)\mathbf{v}_3$
  compute $\mathbf{v}_4 \leftarrow I(\mathbf{v}, \mathbf{y}_3, 0, t_{k+1})$
  if derivative required then
    $\mathbf{D}_1 \leftarrow d_\mathbf{v}I(\mathbf{v}, \mathbf{y}, 0, t_k) + d_\mathbf{y}I(\mathbf{v}, \mathbf{y}, 0, t_k)\, d_\mathbf{v}\mathbf{y}$
    $\mathbf{D}_2 \leftarrow d_\mathbf{v}I(\mathbf{v}, \mathbf{y}_1, 0, t_{k+1/2}) + d_\mathbf{y}I(\mathbf{v}, \mathbf{y}_1, 0, t_{k+1/2})\,(d_\mathbf{v}\mathbf{y} + (\Delta t/2)\mathbf{D}_1)$
    $\mathbf{D}_3 \leftarrow d_\mathbf{v}I(\mathbf{v}, \mathbf{y}_2, 0, t_{k+1/2}) + d_\mathbf{y}I(\mathbf{v}, \mathbf{y}_2, 0, t_{k+1/2})\,(d_\mathbf{v}\mathbf{y} + (\Delta t/2)\mathbf{D}_2)$
    $\mathbf{D}_4 \leftarrow d_\mathbf{v}I(\mathbf{v}, \mathbf{y}_3, 0, t_{k+1}) + d_\mathbf{y}I(\mathbf{v}, \mathbf{y}_3, 0, t_{k+1})\,(d_\mathbf{v}\mathbf{y}+\Delta t\mathbf{D}_3)$
    $d_\mathbf{v}\mathbf{y} \leftarrow d_\mathbf{v}\mathbf{y} + (\Delta t/6)\,(\mathbf{D}_1 + 2\mathbf{D}_2 + 2\mathbf{D}_3 + \mathbf{D}_4)$
  end if
  $\mathbf{y} \leftarrow \mathbf{y} + (\Delta t/6)\,(\mathbf{v}_1 + 2\mathbf{v}_2 + 2\mathbf{v}_3 + \mathbf{v}_4)$
end for
Output: end of characteristics, $\mathbf{y} \in \mathbb{R}^{d \cdot n_p}$, and (if required) gradient, $d_\mathbf{v}\mathbf{y} \in \mathbb{R}^{n \times d \cdot n}$
```

In general, $y(\mathbf{v}, \mathbf{x}_c, 1, 0)$ does not coincide with a grid point; the intensity has to be computed by interpolation. In our numerical experiments we use a bi- or tri-linear interpolation model and regularized cubic approximation methods provided in FAIR to obtain the intensity values of the deformed image; see [54] for implementation details and other common choices.

Continuity equation: To solve the continuity equation (2.3) we consider the *Particle-In-Cell* (**PIC**) method that pushes mass along the characteristics forward in time; see, e.g., [18]. For a given grid point $y_0 \in \mathbb{R}^d$, we introduce a particle with its mass given by the intensity value $\mathscr{T}(y_0)$. Then, we follow the trajectory of the particle along the characteristic to its final point $y_1 := y(\mathbf{v}, y_0, 0, 1)$. In general, y_1 does not coincide with a grid point. We obtain the value of the final state in a given cell by integrating the mass of all particles whose support intersects the cell. Equivalently, we compute the final density by splitting the mass of each particle among the cells adjacent to its final location. Ideally, the particles are represented by Dirac delta functions. In practice, we consider bi- or tri-linear hat functions of a certain isotropic width $\delta > 0$ as proposed in [18].

Using the push-forward matrix $\mathbf{F}$, which has also been used in [29], this process can be written as

$$u(\mathbf{x}_c, 1) = \mathbf{F}(y(\mathbf{v}, \mathbf{x}_c, 0, 1))\mathscr{T}(\mathbf{x}_c). \quad (3.3)$$

In the following, we construct the push-forward matrix in a way that ensures mass-preservation at the discrete level for any choice of δ and h. For ease of presentation, we describe the procedure for the one dimensional case (d = 1). The derivation extends to tensor meshes in higher dimensions in a straightforward way under the assumption that the basis functions are piecewise polynomials in the coordinate directions. Let us assume that for j = 1, 2, … , m particles are located at the cell-centered points $\mathbf{x}_j = (j - \frac{1}{2})h$ and their respective mass is given by $\mathbf{u}_{0,j} = \mathscr{T}(\mathbf{x}_j)$. The particles are advected to the points $\mathbf{y}_j = y(\mathbf{v}, \mathbf{x}_j, 0, 1)$. For some $\delta > 0$ the particles are represented by the shifted basis functions

$$b^{\delta}(x, \mathbf{y}_j) = \begin{cases} 1 - (\mathbf{y}_j - x)/\delta & \text{for } \mathbf{y}_j - \delta \le x \le \mathbf{y}_j, \\ 1 + (x - \mathbf{y}_j)/\delta & \text{for } \mathbf{y}_j < x \le \mathbf{y}_j + \delta, \\ 0 & \text{else,} \end{cases}$$

for each $j = 1, 2, \dots, n_p$. The mass of $u(\cdot, 1)$ contained in the ith interval $[\mathbf{x}_i, \mathbf{x}_{i+1}]$ is given by

$$\mathbf{u}_i(\mathbf{v}, 1) = \sum_{j=1}^{n_p} \int_{\mathbf{x}_i}^{\mathbf{x}_{i+1}} \mathbf{u}_j b^{\delta}(x, \mathbf{y}_j)dx = \sum_{j=1}^{n_p} \mathbf{u}_j \left(B^{\delta}(\mathbf{x}_{i+1}, \mathbf{y}_j) - B^{\delta}(\mathbf{x}_i, \mathbf{y}_j)\right) = \sum_{j=1}^{n_p} \mathbf{u}_j \mathbf{F}_{ij}, \tag{3.4}$$

where B^{δ} denotes an anti-derivative of b^{δ} and is given by

$$B^{\delta}(x, \mathbf{y}_j) = \begin{cases} 0 & \text{for } x < \mathbf{y}_j - \delta, \\ x - \frac{1}{2\delta}(2\mathbf{y}_j x - x^2) & \text{for } \mathbf{y}_j - \delta \le x \le \mathbf{y}_j, \\ x + \frac{1}{2\delta}(x^2 - 2\mathbf{y}_j x) & \text{for } \mathbf{y}_j < x \le \mathbf{y}_j + \delta, \\ 1 & \text{for } \mathbf{y}_j + \delta < x. \end{cases}$$

Repeating the process outlined in (3.4) for all i = 1, 2, … , m yields the discrete transformed density, which is summarized in (3.3). Note that our scheme is *mass-preserving at the discrete level by design* since exact integration is performed. In other words: the columns in **F** sum to one regardless of the choices for δ and h. Also note that **F** is sparse. The level of sparsity depends on the ration between δ and h. If we choose $n_p = m$, $\delta = h$, and $\mathbf{x}_j = h(j + 1/2)$, it is easy to see that **F** is the transpose of the linear interpolation matrix [29]. Thus, the relation between (3.2) and (3.3) mirrors the adjoint relation between the continuity and the advection equation.

3.2. Optimization

Using the Lagrangian methods outlined above we parametrize the final state in terms of the velocities, which we denote by $u_1(\mathbf{v})$. We eliminate the state equation from the variational problem (2.1) and—upon discretization—obtain the finite-dimensional unconstrained problem min

$$\min_{\mathbf{v}} \{J(\mathbf{v}) := D(u_1(\mathbf{v}), \mathscr{R}) + \alpha S(\mathbf{v})\}, \quad (3.5)$$

where D and S are discrete versions of the distance measure $\mathscr{D}$ and regularizer $\mathscr{S}$ in (2.1). As an example, we consider the squared $\mathbf{L}_2$-distance functional. We note, that this is a modular building block of our formulation; for other choices we refer to [54]. Using a midpoint rule, the discrete distance measure—also known as the *sum-of-squared-differences* (**SSD**)—reads

$$D^{\mathrm{SSD}}(u_1(\mathbf{v}), \mathscr{R}) = \frac{h^d}{2} \mathrm{res}(\mathbf{v})^\top \mathrm{res}(\mathbf{v}), \quad \text{where} \quad \mathrm{res}(\mathbf{v}) := u_1(\mathbf{v}) - \mathscr{R}. \quad (3.6)$$

To enable a Gauss–Newton optimization, we compute the derivative of the objective function. Using the chain rule we obtain

$$d_{\mathbf{v}} J(\mathbf{v}) = d_{\mathbf{v}} u_1(\mathbf{v}) d_{u_1} D(u_1(\mathbf{v}), \mathscr{R}) + \alpha d_{\mathbf{v}} S,$$

where the derivative of the distance measure with respect to the final state u_1 is for (3.6) given by

$$d_{u_1} D^{\mathrm{SSD}}(u_1(\mathbf{v}), \mathscr{R}) = h^d \mathrm{res}(\mathbf{v}).$$

We again refer to [54] for the derivatives for other distance measures. The derivative of the regularizer can be written as

$$d_{\mathbf{v}} S(\mathbf{v}) = \Delta t h^d (\mathbf{B}^\top \mathbf{B}) \mathbf{v}.$$

Here, $\mathbf{B}$ is a discretization of the spatial or spatio-temporal derivative operator. Similarly, the approximated Hessian is given by

$$\mathbf{H}(\mathbf{v}) \approx d_2 J(\mathbf{v}) = d_{\mathbf{v}} u_1(\mathbf{v})\, d_2 D(u_1(\mathbf{v}), \mathscr{R})\, d_{\mathbf{v}} u_1(\mathbf{v})^\top + \alpha\, \Delta t h^d\, \mathbf{B}^\top \mathbf{B} + \gamma \mathbf{I}_n,$$

where $\mathbf{I}_n \in \mathbb{R}^{n \times n}$ denotes the identity matrix and $\gamma > 0$ is a small constant to ensure positive semi-definiteness. In our numerical experiments we use $\gamma = 0.01$. The Hessian of the distance measure is given by

$$d_2 D^{\mathrm{SSD}} = h^3 \mathbf{I}_n.$$

Next, we compute the derivative of the mapping $\mathbf{v} \mapsto u_1(\mathbf{v})$. We first consider the advection equation. Using the chain rule to differentiate (3.2) we obtain

$$d_{\mathbf{v}} u_1(\mathbf{v}) = d_{\mathbf{v}} \mathscr{T}(y(\mathbf{v}, \mathbf{x}_c, 1, 0)) = \nabla \mathscr{T}(y(\mathbf{v}, \mathbf{x}_c, 1, 0))\, d_{\mathbf{v}} y(\mathbf{v}, \mathbf{x}_c, 1, 0).$$

The first term in the product is an image gradient evaluated at the end points of the characteristic curves. It is computed by differentiating the interpolation model; see [54] for details. The second term is the derivative of the endpoint of the characteristic curve with respect to **v**. How we compute this derivative is explained below. For the continuity equation (3.3) we obtain

$$d_{\mathbf{v}} u_1(\mathbf{v}) = d_{\mathbf{v}} \left(\mathbf{F}(y(\mathbf{v}, \mathbf{x}_c, 0, 1)) \mathscr{T}(\mathbf{x}_c)\right) = d_y \left(\mathbf{F}(y(\mathbf{v}, \mathbf{x}_c, 0, 1)) \mathscr{T}(\mathbf{x}_c)\right) d_{\mathbf{v}} y(\mathbf{v}, \mathbf{x}_c, 0, 1).$$

The first term can be computed by differentiating the terms in (3.4), for which $\mathbf{F}_{ij} > 0$, with respect to the end points of the characteristic curves. Notice that this also implies that $d_y(\mathbf{F}(y(\mathbf{v}, \mathbf{x}_c, 0, 1)))$ is at least as sparse as the push-forward matrix.

We now present an efficient way for computing the derivative of the end point of the characteristics with respect to the velocity field. Since we use explicit time stepping schemes the derivative can be computed recursively alongside the computation of the characteristics. For example, if we use the forward Euler method in (3.1) we have $d_{\mathbf{v}} y(\mathbf{v}, \mathbf{x}_c, 1, 0) = 0$; we obtain

$$\begin{aligned} & d_{\mathbf{v}} y(\mathbf{v}, \mathbf{x}_c, 1, t_{k+1}) \\ & \quad = d_{\mathbf{v}} y(\mathbf{v}, \mathbf{x}_c, 1, t_k) \\ & \quad + \Delta t\, d_{\mathbf{v}} I(\mathbf{v}, y(\mathbf{v}, \mathbf{x}_c, 1, t_k), t_k) \\ & \quad + \ldots\ldots + \Delta t\, d_y I(\mathbf{v}, y(\mathbf{v}, \mathbf{x}_c, 1, t_k), t_k)\, d_{\mathbf{v}} y(\mathbf{v}, \mathbf{x}_c, 1, t_k), \end{aligned}$$

for all $k = 0, 1, \ldots, N - 1$. The derivatives of the interpolation scheme, $d_{\mathbf{v}} I$ and $d_y I$, are computed as described in [54]. Notice that we do not need $d_{\mathbf{v}} y(\mathbf{v}, \mathbf{x}, 1, t_k)$ in subsequent time steps. Thus, in practice, we update it directly. It is straightforward to extend this procedure to other explicit methods, such as the RK4 scheme used in our experiments; see Algorithm 1 for details. We emphasize that neither intermediate transformations nor intermediate state variables need to be stored to compute the derivative. Therefore, the storage requirement is essentially independent of the number of time steps N used to compute the characteristics. This is different to the methods described in [9, 48, 50, 51], which require storing at least one time-dependent scalar field to evaluate the gradient or Hessian operator.

We use a standard inexact Gauss–Newton–Krylov method for solving the finite-dimensional optimization problem. We use the implementation and stopping conditions described in [54]. As to be expected, the computationally most challenging task is the computation of the search direction. Let $\mathbf{v}^i$ denote the velocity field at the ith iteration. Given $\mathbf{v}^i$ we obtain the search direction $\delta\mathbf{v}$ by solving

$$\mathbf{H}(\mathbf{v}^i)\delta\mathbf{v} = -d_{\mathbf{v}}J(\mathbf{v}^i). \quad (3.7)$$

The next iterate $\mathbf{v}^{i+1}$ is computed via $\mathbf{v}^{i+1} = \mathbf{v}^i + \mu\delta\mathbf{v}$; an Armijo linesearch is performed to determine the step size μ (see, e.g., [55]).

We solve the symmetric and positive definite linear system in (3.7) via a Cholesky factorization or, for large-scale problems, via iterative methods such as the conjugate gradient (**CG**) method [42]. The convergence of iterative methods depends on the clustering of the eigenvalues of $\mathbf{H}$, which can be improved by appropriate preconditioning [62]; yielding a preconditioned CG (**PCG**) method. For the examples considered in this paper, the Hessian of the regularizer is block diagonal with d blocks corresponding to a discretized second- or fourth-order differential operator. Since the Hessian of the regularizer is of higher-order as compared to the Hessian of the distance term, we exploit the structure of $\mathbf{A} = \mathbf{B}^\top\mathbf{B}$ for preconditioning. Given that the velocity is discretized on a regular mesh in space and time, $\mathbf{A}$ is a structured matrix and can be written as a sum of Kronecker products of Toeplitz-plus-Hankel matrices. Thus, its pseudo-inverse can be computed efficiently using the *Discrete Cosine Transform* (**DCT**) [37]. In addition to the preconditioners available in FAIR (such as multigrid or Jacobi) we also provide an option to use $\mathbf{A} + \gamma\mathbf{I}_n$ as preconditioner; a common choice in PDE-constrained optimization problems [48, 2].

The optimization problem in (3.5) is known to be non-convex. To limit the risk of being trapped in a local minimum, we use a multilevel strategy similar to the one described in [54]. First, we solve (3.5) with a coarse discretization for the distance, regularizer, and velocities, and then refine the solution and use it as a starting guess for the optimization problem obtained on the next level. We continue this procedure until we reach a sufficiently fine discretization level, which depends on the application at hand. In addition to improving robustness, the scheme often leads to an overall reduction of computation time.

3.3. Implementation

We have implemented our method in MATLAB as an extension to the 2011 version of the FAIR toolbox described in [54]. This allows us to exploit all distance measures, interpolation kernels, and numerical schemes provided in FAIR. A pseudocode of the RK4 method used to compute the characteristics and the derivative of the end point with respect to the velocity field $\mathbf{v}$ is given in Algorithm 1. It can be seen that both the characteristics and the gradient are computed in one sweep over all time points. The characteristic and the gradient can be updated in each step. This makes the memory requirements independent of the number of time steps N used to compute the characteristics; the size of $d_{\mathbf{v}}u_1(\mathbf{v})$ is $n \times m^d$. The gradient matrix is sparse. Its columns will have non-zero entries only in rows associated with discrete velocities in close proximity to the characteristic curve; we will demonstrate this experimentally; see Fig. 4.2.

The reduced memory requirement is a significant improvement over existing Eulerian or Semi-Lagrangian methods. These require storing (or recomputing) the transported images or characteristics for each time step [9, 48, 50, 51]. The Lagrangian method proposed here, requires only the allocation of the transformed grid, which is independent of the number of time steps for the forward or adjoint solves. Further, it is possible to adapt the time step used for computing the characteristics, e.g., depending on the complexity of the trajectory.

4. Numerical Experiments

In this section, we demonstrate the potential of our solver based on two- and three-dimensional synthetic and real world problems of varying complexity. We compare our prototype implementation to tailored and highly optimized state-of-the-art packages for diffeomorphic image registration. For mass-preserving registration we consider the VAMPIRE package [30]. For large deformation diffeomorphic registration we consider the hyperelastic registration model originally described in [15, 60].

4.1. General Setup

As stopping criteria for the Gauss–Newton optimization, we use standard settings provided in FAIR. The maximum number of inner iterations for the PCG method is set to 50; the tolerance for the relative residual is set to 0.1. We use a spectral preconditioner.[7] The benchmark methods employ a hyperelastic regularization model, for which effective preconditioning is more challenging; see, e.g., [15, 60]. Here, we use a matrix-free implementation of a Jacobi-PCG solver. Problem-specific parameters, such as the number of time points to represent velocity fields, times steps in the RK4 method, the image domain, the number of multi-level steps, or the padding of the domain used to represent the velocity field, are described in the respective subsections. We perform all our experiments on a ×68 compute node with 40 Intel(R) Xeon(R) CPU E5-2660 processors running at 2.60GHz with a total of 256GB of memory.

We consider H^1 (diffusive) and/or H^2 (curvature) regularization models throughout our experiments. We emphasize that, as we have already pointed out in Sec. 2.1, theoretical considerations require imposing H^2-regularity on v in order to guarantee that v gives rise to a diffeomorphic map y (see, e.g., [9]). Our argument for also considering an H^1 regularization model is that, in practice, we can control the weight α by monitoring det ∇y to ensure that the discretized map y is indeed a diffeomorphism. We also note that the regularization is a modular block of our formulation. If theoretical requirements are of concern, one can switch to H^2 regularity.

4.2. 2D C-Shape

We consider the classical test case of registering a C-shaped object to a disc as initially proposed by Christensen [20]. We study registration quality and performance. We compare our results to the hyperelastic registration method described in [15, 60]. We also study the convergence for different types of preconditioners for this problem.

[7]Since the regularization operator corresponds to a block diagonal matrix whose $4 \cdot 2$ blocks are discretizations of a 2D Laplacian, its pseudo inverse can be computed efficiently using DCTs (see Sec. 3.2).

Experimental Setup—The test data is taken from FAIR and consists of two binary image data with 128 × 128 pixels on the image domain $\Omega = (0, 1)^2$. To build a continuously differentiable image model from the binary image data, we use the moments-regularized cubic B-spline interpolation with an experimentally tuned smoothing parameter of $\theta = 0.1$; see [54] for details. Since the image modality is comparable in both images, we use the SSD distance measure in (3.6) to assess image similarity.

- LDDMM: We model the velocity field on a padded domain $(-0.5, 1.5)^2$ to reduce boundary effects. We use the diffusion regularizer with an empirically determined weight of $\alpha = 400$; we set $\gamma = 0$. We compare results for a stationary velocity model to those obtained for a non-stationary velocity model with $n_t = 2$ time intervals. We use a three-step multilevel strategy and discretize the domain for the velocities using regular meshes with 32^2, 64^2, and 128^2 cells, respectively. The characteristics are computed using an RK4 method with $N = 3$ time steps. We assess registration quality and the impact of different preconditioning techniques (no preconditioning, Jacobi preconditioning, Symmetric Gauss Seidel, and spectral) on the convergence of the PCG method used to approximately solve (3.7). For the convergence study, we only consider the coarsest discretization level (32 × 32 cells). The structure of the Hessian depends on the current velocity estimate. We compare the convergence of the PCG method at the first and final Gauss–Newton iteration. We consider an H^1 regularization model. We report results for a stationary and a non-stationary velocity field ($n_t = 2$). In each case, we aim to solve the linear system up to a relative error of 10^{-10} and set the maximum number of iterations to 250.
- Hyperelastic Registration [15]: We use the default parameters provided in FAIR to solve this problem. The values for the regularization are empirically chosen and set to $\alpha_1 = 100$ for the length $\alpha_2 = 0$ for the area, and $\alpha_3 = 18$ for the volume regularizer. We employ a five-step multilevel strategy, where the transformation is discretized on meshes with 8^2, 16^2, 32^2, 64^2, and 128^2 cells.

Observations—For the proposed method with a stationary velocity model we require 16, 4, and 4 Gauss–Newton iterations per level with a total runtime of roughly 6 seconds. Using the non-stationary velocity model we require 25, 3, and 3 iterations per level and a runtime of about 12 seconds. For the hyperelastic registration approach 30, 17, 6, 7, and 3 iterations are performed on each resolution level. The total computational time is about 35 seconds.

We visualize the results in Fig. 4.1. As can be seen in Fig. 4.1, the proposed methods deliver transformed template images that are qualitatively similar to the reference image (small residual). Both methods result in diffeomorphic transformations as judged by the values of the determinant of the Jacobian. As to be expected, the range of the relative volume change is considerably larger for the proposed methods ($\det \nabla y(\mathbf{v}, \mathbf{x}, 1, 0) \in [0.05, 20.58]$ and $\det \nabla y(\mathbf{v}, \mathbf{x}, 1, 0) \in [0.16, 14.25]$ for the stationary and non-stationary field, respectively) as compared to the hyperelastic registration ($\det \nabla y \in [0.34, 5.88]$). This is due to the fact that the hyperelastic registration model explicitly controls and penalizes volume change. Comparing the estimated stationary and non-stationary velocity fields shows that for the latter the estimate changes considerably in time. Also, it should be noted that the registration quality is slightly better for the non-stationary approach (smaller range for the Jacobians and a reduction of the distance measure of 99.48% vs. 97.85%, respectively).

We show the results of the experimental evaluation of different preconditioning techniques in Fig. 4.2. The number of non-zero elements in the Hessian increases from the first to the final iteration for both regularizers. This is due to the fact that particles travel a longer distance through the domain. The performance of the preconditioner deteriorates in the final iteration for all preconditioners. We observe a similar behavior for the hyperelastic formulation (see [60]). We can observe that we need fewer iterations for the stationary case to reach the tolerance; we invert for fewer unknowns, which results in a smaller linear system that needs to be solved. The proposed spectral preconditioner displays the best rate of convergence amongst the considered schemes for preconditioning the Hessian; we use it for all our experiments. We note that we have performed the same study for the curvature regularization (results not included in this study). We observed a similar behavior.

4.3. 2D Mass-Preserving Registration

We consider an academic test problem for mass-preserving registration. We compare our method against the VAMPIRE toolbox for mass-preserving registration [30].

Experimental Setup—The test data is designed to mimic the contraction of a tissue containing a fixed amount of tracer. The data is obtained by subtracting two Gaussians with different standard deviations. The mass is exactly equal, but in the reference image it is concentrated in a smaller region so that the image overall appears brighter. The image domain is $(-5, 5)^2$ and the full resolution is 256 × 256. For all experiments we use a four-level multi-level strategy with resolutions 32^2, 64^2, 128^2, and 256^2, respectively. A continuous image model is built using bi-linear interpolation and the SSD distance measure is used to quantify image similarity.

- MP-LDDMM: As in the previous example the velocity field is modeled on a padded spatial domain $(-5.4, 5.4)^2$ to reduce boundary effects. We use $n_t = 1$ for the spatial discretization of the velocity. The characteristics are approximated using $N = 2$ time steps for the RK4 method. The push-forward matrices are build from bilinear basis functions whose width equals the cell size. We use the diffusion regularizer with weight $\alpha = 1000$ and $\gamma = 1\text{E}{-2}$. We compare results for a stationary and a non-stationary velocity field.
- VAMPIRE: We use the default parameters for the hyperelastic regularizer ($\alpha_1 = 10,000$ for the length-, $\alpha_2 = 0$ for the area, and $\alpha_3 = 100$ for the volume regularizer).

Observations—Registration results are visualized in Fig. 4.3. For the MP-LDDMM using a stationary velocity model, we perform 4, 2, 1, and 1 iterations per resolution level. The total computation time is about 4 seconds. Using the non-stationary velocity model we require 5, 2, 2, and 2 iterations and require a computation time of about 8 seconds. For

VAMPIRE we perform 5, 2, 2, and 1 iterations on the respective levels. The time-to-solution is approximately 12 seconds.

Both methods also yield comparable results in terms of data misfit as well as the final transformation. This is not only confirmed qualitatively by visual inspection of the transformed template image and the deformed grids, but also quantitatively: The volume change introduced by the transformation obtained using the proposed methods ($\det \nabla y(\mathbf{v}, \mathbf{x}, 1, 0) \in [0.89, 2.33]$ and $\det \nabla y(\mathbf{v}, \mathbf{x}, 1, 0) \in [0.67, 2.54]$ for a stationary and a non-stationary velocity model, respectively) is comparable to the one obtained using VAMPIRE ($\det \nabla y \in [0.76, 2.40]$). The largest improvement in image similarity (with respect to the SSD) is achieved for the MP-LDDMM method with a non-stationary velocity (distance reduction of 99.98% vs. 97.43%) although—in contrast to the previous experiment—it should be noted that the estimated velocities are very similar for both approaches.

4.4. 3D Cardiac PET

We consider a 3D mass-preserving registration problem of registering systolic and diastolic cardiac PET data of a mouse heart. The data is provided in FAIR.[8] The image domain is $\Omega = (0, 32)^3$ with a resolution of 40^3 grid points. The results are illustrated in Fig. 4.4. We use a three-level multi-level strategy with resolutions 10^3, 20^3, and 40^3, respectively, for all approaches. On the finest level, the number of unknowns is 384 000.

Experimental Setup

- MP-LDDMM: The velocity field is modeled on the same domain as the image data and the same number of cells is used for spatial discretization. We will only consider the non-stationary case, here. We use $n_t = 1$ time intervals for the velocity (which results in two discretization points for the velocity v). We use an RK4 method to compute the characteristics with $N = 2$ time steps. The push-forward matrix is build using tri-linear hat functions, with a width that corresponds to the voxel size of the image data. We use the diffusion regularizer with regularization weight $\alpha = 100$ and $\gamma = 1\mathrm{E}{-2}$.
- VAMPIRE: We use $\alpha_1 = 100$ for the length regularizer, $\alpha_2 = 10$ for the area regularizer, and $\alpha_3 = 100$ for the volume regularizer. We use the same number of multi-resolution levels.

Observations—For MP-LDDMM the optimization scheme performs 9, 3, and 3 iterations on the respective levels. The time-to-solution is about 36 seconds. For VAMPIRE we require 5, 4, and 3 iterations on the respective levels. The total runtime is about roughly 74 seconds. Both schemes yield qualitatively almost identical results. The residual differences between the transformed template image and the reference image is small. Overall, our current prototype implementation of a Gauss–Newton–Krylov method for LDDMM is competitive with VAMPIRE in terms of the runtime. We expect to be able to drastically reduce the runtime in near future. For the hyperelastic registration most time is spent on determining

[8]We thank the European Institute for Molecular Imaging (EIMI) and SFB 656, University of Münster, Germany for contributing the image data.

the search direction, which requires solving an ill-conditioned linear system; see also [60]; a reduction in runtime for this scheme is much more difficult.

4.5. 3D Brain Registration

Experimental Setup—The data is taken from the NIREP repository [19]. We consider the datasets `na02` (template image) and `na01` (reference image) for our experiments. The grid size for these images is 256 × 300 × 256. We downsample these images to a size of 128 × 150 × 128 voxels to make the problem computationally tractable for our prototype and the reference implementation. The image domain is defined to be $\Omega = (0, 20) \times (0, 23.4375) \times (0, 20)$. We use SSD as distance measure and a multi-level strategy with 3 resolution levels (32 × 38 × 32, 64 × 75 × 64, and 128 × 150 × 128). The number of unknowns is 7 372 800 for the finest level.

We evaluate registration performance based on overlap measures evaluated for the label maps associated with the images. The data comes with 32 labels for gray matter regions [19]. We simplify the presentation of our results by only considering the union of these labels to evaluate the performance of our method. We use the Dice coefficient as a measure for registration quality, which has an optimal value of 1. We use a nearest-neighbor interpolation model to transform the label maps with the computed y to avoid any additional thresholding.

We limit the evaluation of the determinant of the Jacobian to the foreground (i.e., the brain) in the reference image. We identify this foreground by thresholding; we consider intensities with a value of 0.05 and larger as foreground. We slightly extend this mask by smoothing it with a Gaussian kernel of width $2h$. A second thresholding step defines the final brain mask used for the evaluation of the Jacobians.

- Proposed (LDDMM): The velocity field is modeled on a slightly larger domain than the image domain to reduce boundary effects; we choose $\Omega^V = (-1, 21)\times(-1, 24.4375)\times(-1, 21)$. We consider stationary and non-stationary velocities v. We use $n_t = 1$ time intervals for the non-stationary case (which results in two discretization points for the velocity v). We use an RK4 method with $N = 5$ time steps to compute the characteristics. The push-forward matrix is build using tri-linear hat functions, with a width that corresponds to the voxel size of the image data. We consider the curvature (H^2) and the diffusive (H^1) regularization model. We study registration performance (data mismatch and extremal values of the Jacobians $\det \nabla y$) as a function of the regularization weight α. Once we have found the velocity v, we compute the transformation y we use to evaluate the performance of our method using $N = 20$ time steps. We experimentally found that a shift of $\gamma = 0$ and $\gamma = 1\text{E}{-2}$ yields the optimal rate of convergence for the diffusive and the curvature regularization model, respectively. We set the tolerance for the optimization to $\text{tol}_J = 5\text{E}{-2}$. We use a relative tolerance of 1E−1 for the PCG method; we limit the number of Krylov iterations to 50.
- Hyperelastic registration: We experimentally found that regularization weight of $\alpha_1 = 100$ (length regularizer), $\alpha_2 = 10$ (surface regularizer), and $\alpha_3 = 100$

(volume regularizer) yields high data fidelity (good mismatch) and well behaved Jacobians. We use this setting throughout our experiments. We set the tolerance for the optimization to $\mathrm{tol}_J = 1E - 3$.

Observations—We show exemplary results for the registration in Fig. 4.5. We report results for the quantitative evaluation in Table 4.1. An illustration of the velocities can be found in Fig. 4.6.

All methods yield high fidelity results with diffeomorphic transformations and well behaved Jacobians. We achieve the best Dice score for a diffusive regularization model for $\alpha = 300$ (run #13 in Table 4.1) and a non-stationary velocity field. This is the only run, for which we outperform the hyperelastic approach. The results for the curvature regularization model do not vary significantly when switching from a stationary to a non-stationary formulation; we obtain similar extremal values for the Jacobians and similar Dice values. This is different for the diffusive regularization model. We obtain slightly better values for the Dice coefficient with similar extremal values for the Jacobian.

If we consider a stationary velocity field we can reduce the time-to-solution by a factor of two compared to the hyperelastic approach. These findings are consistent for both regularization approaches (runs #2 to #8 in Table 4.1). If we turn to non-stationary velocity fields, our current implementation of the curvature regularization model is no longer competitive in terms of time-to-solution. For a diffusive regularization model we are, however, still slightly faster than the hyperelastic approach (runs #13 to #15 in Table 4.1) despite an increase of the number of unknowns by a factor of 2 (we use $n_t = 1$, which results in two discretization points for the velocity).

We need, for instance, 5, 3, and 3 iterations for the individual levels for the stationary case and a diffusive regularization model ($\alpha = 400$; run #6 in Table 4.1). For each iteration we require 22, 23, 24, and 24 PCG iterations (level 1), 22, 25, and 27 PCG iterations (level 2), and 27, 29, and 29 PCG iterations (level 3), respectively. The stationary case and a curvature regularization model ($\alpha = 50$; run #4 in Table 4.1) requires 5, 3, and 2 iterations, with 7, 11, 17, 16, and 16 PCG iterations (level 1), 20, 36, 50 PCG iterations (level 2), and 50, and 50 PCG iterations (level 3), respectively. The hyperelastic regularization approach converges after 7, 6, and 5 iterations per level.

The results in Fig. 4.5 suggest that all methods yield comparable residual differences after registration. However, we can, likewise to the former experiments, observe drastic differences in the Jacobians. The hyperelastic regularization allows us to better control the Jacobians (the values range from 4.51E−1 to 1.86). If this control is indeed of importance in practical applications remains to be seen. Notice, that we can either add hard constraints on the divergence of the velocity to our formulation [48, 50] or constraints on $\det \nabla y$ to enable such control.

The projections of the velocity fields in Fig. 4.6 show significant differences between the stationary and the non-stationary case for the curvature regularization model. We can also observe large differences in the appearance of the velocity fields in time for the non-stationary case. This is different for the diffusive regularization model. The stationary and non-stationary velocities do not differ significantly. The differences in time for the non-stationary case are also less pronounced. We can also observe that the energy for both components of the non-stationary velocity field is quite similar (as judged by the values for the ℓ^2-norm reported in Fig. 4.6). This is true for both regularization models. As to be expected we obtain much smoother velocities for the curvature regularization model.

5. Summary and Conclusion

In this paper, we propose efficient numerical algorithms based on Lagrangian hyperbolic PDE solvers to efficiently solve the reduced formulation of the PDE-constrained optimization problem arising in LDDMM. Our formulation can be used for classical, intensity-preserving, registration but also extends the LDDMM framework to mass-preserving registration problems. We consider an optimal control formulation and propose an efficient discretize-then-optimize approach amendable for standard Gauss–Newton methods. The key idea of our approach is to eliminate the hyperbolic PDE constraint using a Lagrangian method with an explicit time integration of the characteristics. Our formulation can handle both stationary and non-stationary velocity fields efficiently. We present economical schemes for analytically computing its derivatives. A main advantage of our method over existing solvers is that derivatives of the solution to the hyperbolic PDE with respect to the velocity field can be explicitly constructed. This leads to an overall memory requirement that is independent of the number of time steps used for solving the PDE. This is a significant advantage over most existing work, which in general require the storage (or re-computing) of spatio-temporal state and adjoint fields or the transformation.

We studied registration performance considering different synthetic and real-world problems. We made the following observations:

- Our results are competitive in terms of both time-to-solution and inversion quality (mismatch) compared to state-of-the-art packages for diffeomorphic image registration across a wide range of applications, which includes mass-preserving and intensity-preserving registration problems.
- Our spectral preconditioner yields a good performance. However, the rate of convergence deteriorates when switching from stationary to non-stationary velocities. Designing a more effective preconditioner for these cases is an item of future work.
- We could observe differences between the stationary and non-stationary formulation in terms of the reconstruction accuracy. This especially becomes apparent for the classical problem of registering a C-shaped object to a disc [20]. In this example a considerable improvement can be achieved using a small number of time discretization points for the velocity. In general, increasing the number of time points enriches the space of transformations, however, it also increases the complexity of the optimization problem.
- Since y appears explicitly in our formulation, we can control $\det \nabla y$ by adjusting the regularization weight α (additional comments can be found below). We have

considered H^1- and H^2-regularization norms. While theoretical considerations do require (more than) H^2-regularity on v to guarantee that a diffeomorphic map y exists [9], we could demonstrate our numerical scheme allows us to ensure that the final map y is diffeomorphic at a discrete level, even for H^1-regularity. However, we note that we consider the regularization as a modular building block. If theoretical requirements are of concern, one can switch to H^2-regularity.

In theory, solutions to the variational optimal control problem are guaranteed to be diffeomorphic (under the assumption of sufficient regularity of v). However, as compared to other diffeomorphic registration approaches that control and thus guarantee invertibility of the *discrete* transformation such as [32, 15], it is more difficult to ensure this for discrete solutions to the optimal control problems. An inaccurate approximation of the characteristics may cause characteristics to cross and thus lead to non-diffeomorphic transformations. This problem is also inherent in other numerical implementations of LDDMM. Therefore, we recommend monitoring volume changes induced by the transformation to adapt the number of time steps and/or smoothness parameter. In our method, the end points of the characteristics correspond to a deformed grid, which can be analyzed or even regularized using techniques described in [32, 15]. Monitoring volume changes via Jacobian determinants can also be done in Eulerian or SL methods, in which the transformation is generally not computed [48, 50].

In this paper, we optimize over the velocity field v instead of optimizing over the final transformation y, a strategy that has become predominantly used in many practical applications. Optimizing for the velocity allows us to use a fairly simple quadratic regularization model while still (in theory) ensuring invertibility of the resulting transformation. In the discrete setting, the grid might have foldings, depending on the regularization parameter and/or the accuracy of the time integration. This can be seen as a drawback compared to image registration methods that use invertibility constraints or nonlinear regularizers directly acting on the transformation. However, these regularizers are very challenging both in theory and in practice; see, e.g., [60]. Another feature of more complicated regularization models such as, e.g., the elastic regularization proposed in [28, 14, 22, 69, 15] is that they are motivated based on physical principles. The notion of plausibility of a transformation y is for these type of regularization models not only limited to the prerequisite that y is a diffeomorphism; it is based on bio-mechanical considerations. Thus, while achieving a very good similarity of the final images, the obtained transformation might not be plausible in all applications. However, a similar argument can be made for elasticity-based regularizers unless true material properties are known and incorporated into the regularization. Another approach to integrate bio-physical priors into diffeomorphic registration is to include more complicated state equations that model the bio-physics of a system under investigation; an example in the context of large deformation diffeomorphic image registration is the incorporation of incompressibility constraints [48, 50]. This, likewise to more sophisticated regularization norms, introduces additional parameters, and as such makes an automated calibration of the method to unseen data more difficult.

Some limitations of the current method will be addressed in future work. First, for mass-preserving registration, the width of the particle kernels may be adjusted locally depending on the spacing of particles after transformation [18]. Computing the distance to the closest neighbor is expensive, however, in our framework the Jacobian determinant is available and can be used to detect relative changes in the density of particles. Second, we will investigate locally adaptive time stepping schemes for computing the characteristics that account for the complexity of the velocity fields.

Acknowledgments

AM is supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program under Award Numbers DE-SC0010518 and DE-SC0009286; and by NIH grant 10042242. LR is supported in part by National Science Foundation (NSF) award DMS 1522599.

References

1. Akcelik, V., Biros, G., Ghattas, O. Parallel multiscale Gauss-Newton-Krylov methods for inverse wave propagation. Proc ACM/IEEE Conference on Supercomputing; 2002. p. 1-15.
2. Alexanderian A, Petra N, Stadler G, Ghattas O. A fast and scalable method for A-optimal design of experiments for infinite-dimensional Bayesian nonlinear inverse problems. SIAM Journal on Scientific Computing. 2016; 38(1):A243–A272.
3. Arsigny V, Commowick O, Pennec X, Ayache N. A log-Euclidean framework for statistics on diffeomorphisms. 2006; 9(Pt 1):924–931.
4. Ashburner J. A fast diffeomorphic image registration algorithm. NeuroImage. Oct; 2007 38(1):95–113. [PubMed: 17761438]
5. Ashburner J, Friston KJ. Diffeomorphic registration using geodesic shooting and Gauss–Newton optimisation. NeuroImage. Apr; 2011 55(3):954–967. [PubMed: 21216294]
6. Avants B, Schoenemann PT, Gee JC. Lagrangian frame diffeomorphic image registration: Morphometric comparison of human and chimpanzee cortex. Medical Image Analysis. 2006; 10:397–412. [PubMed: 15948659]
7. Avants BB, Epstein CL, Brossman M, Gee JC. Symmetric diffeomorphic image registration with cross-correlation: Evaluating automated labeling of elderly and neurodegenerative brain. Medical Image Analysis. 2008; 12(1):26–41. [PubMed: 17659998]
8. Avants BB, Tustison NJ, Song G, Cook PA, Klein A, Gee JC. A reproducible evaluation of ANTs similarity metric performance in brain image registration. NeuroImage. 2011; 54:2033–2044. [PubMed: 20851191]
9. Beg MF, Miller MI, Trouvé A, Younes L. Computing Large Deformation Metric Mappings via Geodesic Flows of Diffeomorphisms. International journal of computer vision. 2005; 61(2):139–157.
10. Benzi M, Haber E, Taralli L. A preconditioning technique for a class of PDE-constrained optimization problems. Advances in Computational Mathematics. Jul; 2011 35(2–4):149–173.
11. Borzi A, Ito K, Kunisch K. Optimal control formulation for determining optical flow. SIAM Journal on Scientific Computing. 2002; 24(3):818–847. (electronic).
12. Borzì A, Ito K, Kunish K. An optimal control approach to optical flow computation. International Journal for numerical methods in fluids. 2002; 40(1–2):231–240.
13. Borzì, A., Schulz, V. Computational optimization of systems governed by partial differential equations. SIAM; Philadelphia, Pennsylvania, US: 2012.
14. Broit, C. PhD thesis. University of Pennsylvania; 1981. Optimal registration of deformed images.
15. Burger M, Modersitzki J, Ruthotto L. A hyperelastic regularization energy for image registration. SIAM Journal on Scientific Computing. 2013; 35(1):B132–B148.
16. Chang H, Fitzpatrick JM. A technique for accurate magnetic-resonance-imaging in the presence of field inhomogeneities. Medical Imaging, IEEE Transactions on. Sep; 1992 11(3):319–329.
17. Chen K, Lorenz DA. Image Sequence Interpolation Using Optimal Control. Journal of Mathematical Imaging and Vision. Mar; 2011 41(3):222–238.

18. Chertock A, Kurganov A. On a practical implementation of particle methods. Applied Numerical Mathematics. 2006; 56(10–11):1418–1431.

19. Christensen GE, Geng X, Kuhl JG, Bruss J, Grabowski TJ, Pirwani IA, Vannier MW, Allen JS, Damasio H. Introduction to the non-rigid image registration evaluation project. Proc Biomedical Image Registration. 2006:128–135. volume LNCS 4057.

20. Christensen GE, Rabbitt RD, Miller MI. Deformable templates using large deformation kinematics. Image Processing, IEEE Transactions on. 1996; 5(10):1435–1447.

21. Dawood, M., Brune, C., Jiang, X., Büther, F., Burger, M., Schober, O., Schäfers, M., Schäfers, KP. Medical Imaging and Augmented Reality. Springer Berlin Heidelberg; Berlin, Heidelberg: Sep. 2010 A Continuity Equation Based Optical Flow Method for Cardiac Motion Correction in 3D PET Data; p. 88-97.

22. Droske M, Rumpf M. A Variational Approach to Nonrigid Morphological Image Registration. SIAM Journal on Applied Mathematics. Jan; 2004 64(2):668–687.

23. Dupuis P, Grenander U, Miller MI. Variational problems on flows of diffeomorphisms for image matching. Quarterly of applied mathematics. 1998

24. Evans, LC. Partial Differential Equations. American Mathematical Soc; 2010.

25. Ewing RE, Wang H. A summary of numerical methods for time-dependent advection-dominated partial differential equations. Journal of Computational and Applied Mathematics. 2001; 128(1–2): 423–445.

26. Fischer B, Modersitzki J. Curvature Based Image Registration. Journal of Mathematical Imaging and Vision. 2003; 18(1):81–85.

27. Fischer B, Modersitzki J. Ill-posed medicine—an introduction to image registration. Inverse Problems. 2008; 24(3):034008.

28. Fischler MA, Elschlager RA. The representation and matching of pictorial structures. Computers, IEEE Transactions on. 1973

29. Fohring J, Haber E, Ruthotto L. Geophysical Imaging of Fluid Flow in Porous Media. SIAM Journal on Scientific Computing. 2014; 36(5):S218–S236.

30. Gigengack F, Ruthotto L, Burger M, Wolters CH, Jiang X, Schafers KP. Motion Correction in Dual Gated Cardiac PET Using Mass-Preserving Image Registration. Medical Imaging, IEEE Transactions on. Mar; 2012 31(3):698–712.

31. Gunzburger, MD. Perspectives in flow control and optimization. SIAM; Philadelphia, Pennsylvania, US: 2003.

32. Haber E, Modersitzki J. Image Registration with Guaranteed Displacement Regularity. International journal of computer vision. Jul; 2006 71(3):361–372.

33. Haber E, Modersitzki J. A multilevel method for image registration. SIAM Journal on Scientific Computing. 2006; 27(5):1594–1607.

34. Haber E, Oldenburg D. A GCV based method for nonlinear ill-posed problems. Computational Geosciences. 2000; 4:41–63.

35. Haker S, Zhu L, Tannenbaum A, Angenent A. Optimal mass transport for registration and warping. International Journal of Computer Vision. 2004; 60(3):225–240.

36. Hansen, PC. SIAM Monographs on Mathematical Modeling and Computation. Society for Industrial and Applied Mathematics (SIAM); Philadelphia, PA: 1998. Rank-deficient and discrete ill-posed problems.

37. Hansen, PC., Nagy, JG., O'Leary, DP. Matrices, Spectra, and Filtering. Society for Industrial and Applied Mathematics (SIAM); Philadelphia, PA: 2006. Deblurring Images: Matrices, Spectra and Filtering.

38. Hart, GL., Zach, C., Niethammer, M. An optimal control approach for deformable registration. Proc IEEE Conference on Computer Vision and Pattern Recognition; 2009. p. 9-16.

39. Hernandez M. Gauss–Newton inspired preconditioned optimization in large deformation diffeomorphic metric mapping. Physics in Medicine and Biology. 2014; 59(20):6085–6115. [PubMed: 25254606]

40. Hernandez M, Bossa MN, Olmos S. Registration of anatomical images using paths of diffeomorphisms parameterized with stationary vector field flows. International Journal of Computer Vision. 2009; 85(3):291–306.

41. Herzog R, Kunisch K. Algorithms for PDE-constrained optimization. GAMM Mitteilungen. 2010; 33(2):163–176.

42. Hestenes MR, Stiefel E. Methods of Conjugate Gradients for Solving Linear Systems. Journal of Research of the National Bureau of Standards. 1952; 49(6):409–436.

43. Hinze, M., Pinnau, R., Ulbrich, M., Ulbrich, S. Optimization with PDE constraints. Springer; Berlin, DE: 2009.

44. Lee E, Gunzburger M. An optimal control formulation of an image registration problem. Journal of Mathematical Imaging and Vision. 2010; 36(1):69–80.

45. LeVeque, RJ. Numerical methods for conservation laws. 1992.

46. LeVeque, RJ. Finite Volume Methods for Hyperbolic Problems. Cambridge University Press; 2002.

47. Lorenzi M, Pennec X. Geodesics, parallel transport and one-parameter subgroups for diffeomorphic image registration. International Journal of Computer Vision. 2013; 105(2):111–127.

48. Mang A, Biros G. An inexact Newton–Krylov algorithm for constrained diffeomorphic image registration. SIAM Journal on Imaging Sciences. 2015; 8(2):1030–1069. [PubMed: 27617052]

49. Mang A, Biros G. A Semi-Lagrangian two-level preconditioned Newton-Krylov solver for constrained diffeomorphic image registration. Apr.2016

50. Mang A, Biros G. Constrained H^1-regularization schemes for diffeomorphic image registration. SIAM Journal on Imaging Sciences. 2016; 9(3):1154–1194. [PubMed: 29075361]

51. Mang, A., Gholami, A., Biros, G. Distributed-memory large-deformation diffeomorphic 3D image registration. Proc ACM/IEEE Conference on Supercomputing; 2016.

52. Miller MI, Younes L. Group actions, homeomorphism, and matching: A general framework. International Journal of Computer Vision. 2001; 41(1/2):61–81.

53. Modersitzki, J. Numerical methods for image registration. Oxford University Press on Demand; 2004.

54. Modersitzki, J. FAIR: Flexible Algorithms for Image Registration, volume 6 of Fundamentals of Algorithms. Society for Industrial and Applied Mathematics (SIAM); Philadelphia, PA: 2009.

55. Nocedal, J., Wright, SJ. Numerical Optimization. Springer; New York, New York, US: 2006.

56. Ou Y, Akbari H, Bilello M, Da X, Davatzikos C. Comparative evaluation of registration algorithms in different brain databases with varying difficulty: Results and insights. IEEE T Med Imaging. 2014; 33(10):2039–2065.

57. Pai A, Sommer S, Sorensen L, Darkner S, Sporring J, Nielsen M. Kernel bundle diffeomorphic image registration using stationary velocity fields and wendland basis functions. Medical Imaging, IEEE Transactions on. 2016; 35(6):1369–1380.

58. Polzin T, Niethammer M, Heinrich MP, Handels H, Modersitzki J. Memory efficient LDDMM for lung CT. Proc Medical Image Computing and Computer-Assisted Intervention. 2016:28–36. volume LNCS 9902.

59. Rehman, Tu, Haber, E., Pryor, G., Melonakos, J., Tannenbaum, A. 3D nonrigid registration via optimal mass transport on the GPU. Medical Image Analysis. Dec; 2009 13(6):931–940. [PubMed: 19135403]

60. Ruthotto L, Greif C, Modersitzki J. A Stabilized Multigrid Solver for Hyperelastic Image Registration. Jul.2016 :1–16. in revision at Numerical Linear Algebra With Application.

61. Ruthotto L, Kugel H, Olesch J, Fischer B, Modersitzki J, Burger M, Wolters CH. Diffeomorphic susceptibility artifact correction of diffusion-weighted magnetic resonance images. Physics in Medicine and Biology. Sep; 2012 57(18):5715–5731. [PubMed: 22941943]

62. Saad, Y. Iterative Methods for Sparse Linear Systems. 2. SIAM; Philadelphia: Apr. 2003

63. Sotiras A, Davatzikos C, Paragios N. Deformable medical image registration: A survey. Medical Imaging, IEEE Transactions on. 2013; 32(7):1153–1190.

64. Staniforth A, Côté J. Semi-Lagrangian integration schemes for atmospheric models—A review. Montly Weather Review. 1991; 119(9):2206–2223.

65. Trouvé A. Diffeomorphisms Groups and Pattern Matching in Image Analysis. International journal of computer vision. 1998; 28(3):213–221.

66. Vercauteren T, Pennec X, Perchant A, Ayache N. Diffeomorphic demons: Efficient non-parametric image registration. NeuroImage. 2009; 45(1):S61–S72. [PubMed: 19041946]

67. Vialard F-X, Risser L, Rueckert D, Cotter CJ. Diffeomorphic 3D Image Registration via Geodesic Shooting Using an Efficient Adjoint Calculation. International Journal of Computer Vision. Aug; 2011 97(2):229–241.

68. Vogel, CR. Computational Methods for Inverse Problems. SIAM; Philadelphia: 2002.

69. Yanovsky, I., Le Guyader, C., Leow, A., Toga, A., Thompson, P., Vese, L. Unbiased volumetric registration via nonlinear elastic regularization. 2nd MICCAI Workshop on Mathematical Foundations of Computational Anatomy; 2008.

70. Younes L. Jacobi fields in groups of diffeomorphisms and applications. Quarterly of Applied Mathematics. 2007; 650(1):113–134.

71. Younes L, Arrate F, Miller MI. Evolutions equations in computational anatomy. NeuroImage. 2009; 45:S40–S50. [PubMed: 19059343]

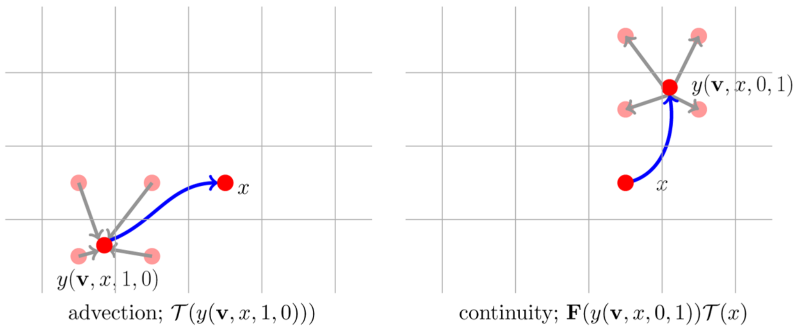
x
$y(\mathbf{v}, x, 1, 0)$
advection; $\mathcal{T}(y(\mathbf{v}, x, 1, 0)))$
$y(\mathbf{v}, x, 0, 1)$
x
continuity; $\mathbf{F}(y(\mathbf{v}, x, 0, 1))\mathcal{T}(x)$

Fig. 3.1.

Illustration of Lagrangian methods for solving linear hyperbolic PDEs. In both cases, the characteristic curves (indicated by a blue line) are computed starting from a grid point x. Left: The advection problem is solved by traveling along the characteristics backwards in time to the non-grid point y(**v**, x, 1, 0). The associated image intensity is computed by interpolating the intensities of the adjacent cells. Right: The continuity equation is solved by pushing the mass $\mathcal{T}$ (x) from x along the characteristics to the non-grid point y(**v**, x, 0, 1) and then distributing $\mathcal{T}$ (x) among the cells adjacent to y(**v**, x, 0, 1).

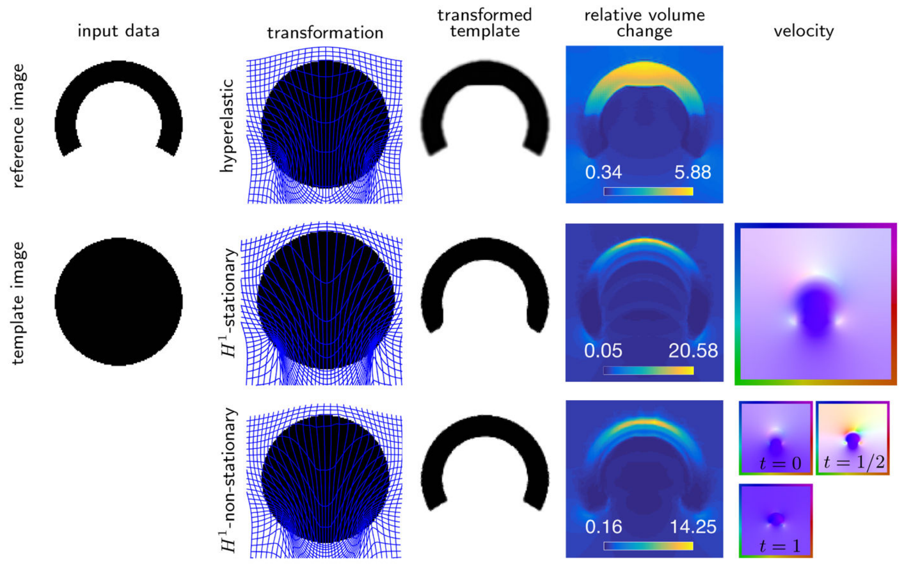

Fig. 4.1.

2D registration results for an academic benchmark problem also considered in [20]. First column visualizes test data and the remaining images visualize registration results for hyperelastic registration [15] (first row) and the proposed method with H^1-regularization and stationary (second row) and non-stationary (third row) velocity models. It can be seen that the proposed methods improves the similarity between the reference and the transformed template image without foldings of the grid. However, the ranges of the relative volume change is considerably larger. It is also evident that the non-stationary velocity model improves the registration result and comparing the estimated velocity fields (right column) shows substantial differences of the velocity estimates.

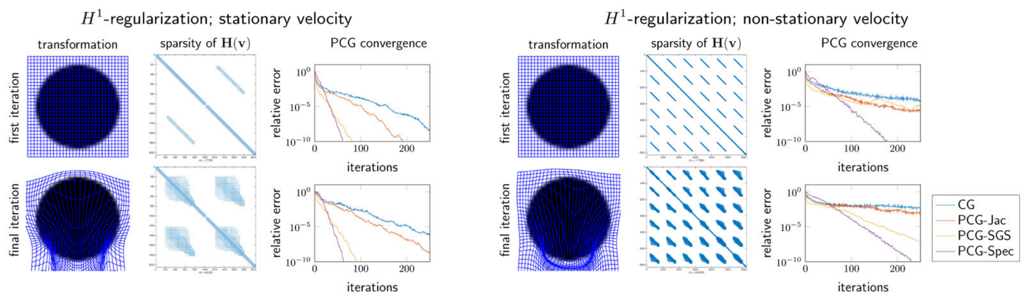

Fig. 4.2.

Sparsity pattern and PCG convergence plots at first and final Gauss–Newton iteration for the 2D test problem on a coarse mesh (m = [32, 32]). We compare the the stationary (left) and non-stationary (right) diffusion regularizer. In both cases the number of non-zero elements in the Hessian grows between the iterations since particles move farther through the domain. In all four cases we compare the convergence of different PCG schemes (no preconditioning, Jacobi, Symmetric Gauss Seidel, and spectral preconditioning). It can be seen that the problems at the final iteration are, in this example, more difficult to solve, however, the spectral preconditioner outperforms the other choices.

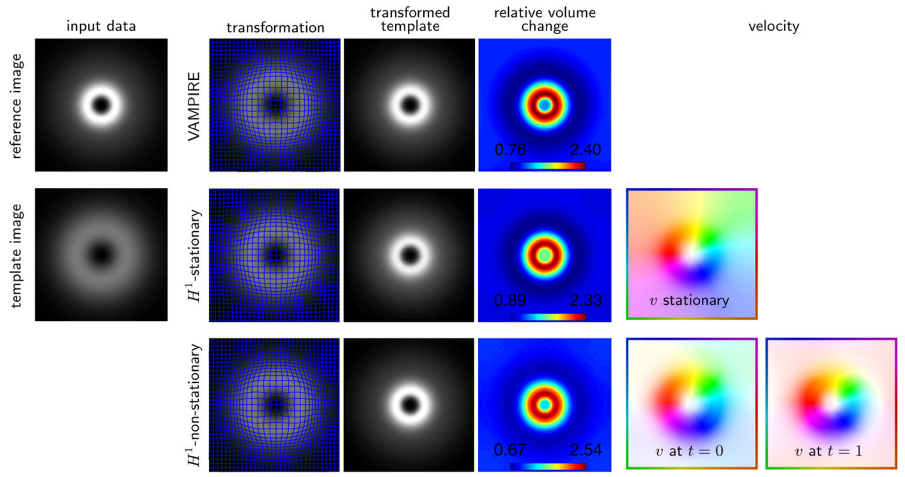

Fig. 4.3.
2D mass-preserving registration for an academic test problem. The image data is generated by subtracting two Gaussian kernels with different standard deviations; the data is designed to have equal mass. The reference image (top) and the template image (bottom) are shown in the left column. We compare the VAMPIRE method (first row) [30] to the proposed mass-preserving LDDMM with stationary (middle row) and non-stationary velocity model (bottom row). For all three methods, we visualize the transformation, the transformed template, and the relative volume change. For the LDDMM methods we also visualize the velocity. Comparing the results in the middle and bottom row, it can be seen that the underlying transformation is rather simple; it can be well represented using a stationary velocity field.

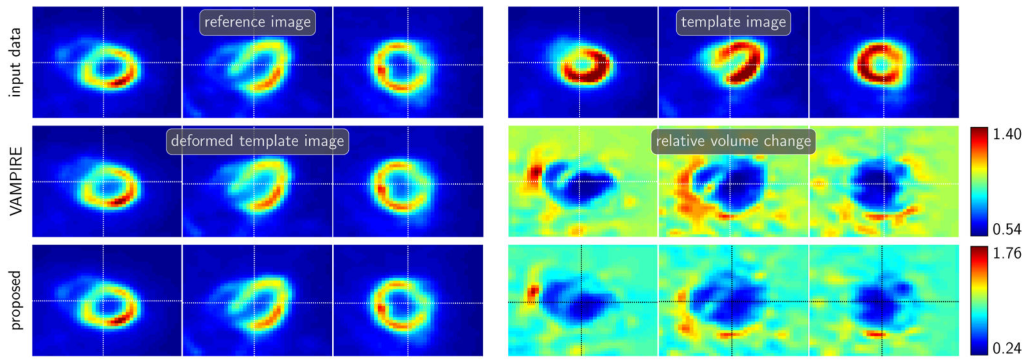

Fig. 4.4.
3D mass-preserving registration of diastolic and systolic PET images of a mouse heart. We display the input data in the first row (left: reference image; right: template image; from left to right: axial, coronal and sagittal view). The deformed template images are shown in the left column (middle row: VAMPIRE; bottom row: proposed method). The relative volume change is shown in the right column (middle row: VAMPIRE; bottom row: proposed method). The color bars to the right illustrate the color coding and provide the range of the Jacobian fields.

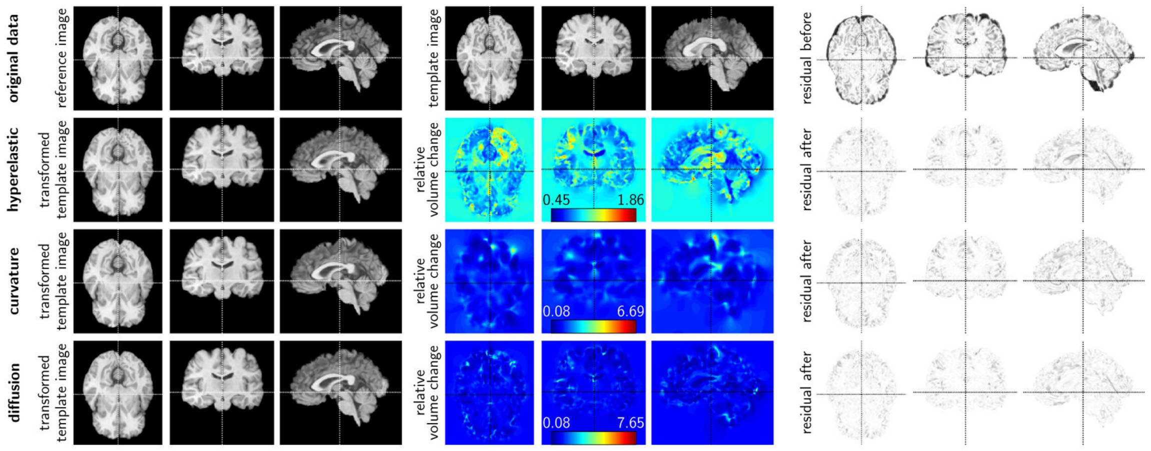

Fig. 4.5.

Exemplary results for a 3D intensity-preserving registration problem based on MRI datasets of the human brain. The data is taken from the NIREP repository. We show (from left to right) an axial, coronal, and sagittal view of the reference image (dataset `na01`), the template image (dataset `na02`), and the residual differences between these two images in the top row. The results correspond to those reported in Table 4.1. We report results for a map based approach with a hyperelastic regularization model (second row: run #1 in Table 4.1; $\alpha_1 = 100$ (length regularizer), $\alpha_2 = 10$ (surface regularizer), and $\alpha_3 = 100$ (volume regularizer)) [15], and for the proposed method for a non-stationary velocity field (third row: curvature regularization model; $\alpha = 10$; run #10 in Table 4.1; bottom row: diffusive regularization model; $\alpha = 300$; run #13 in Table 4.1). For each of these methods we show (from left to right) an axial, a coronal, and a sagittal view of the deformed template image, a map for the relative volume change, and the residual differences between the transformed template image and the reference image after registration. We also display the color bar and the maximal and minimal values for the maps for the relative volume change.

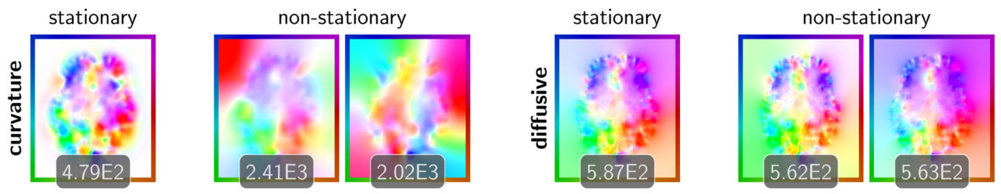

Fig. 4.6.

Illustration of the obtained velocity fields for the registration of 3D brain imaging data. We show the velocities for the curvature (left; $\alpha = 10$; runs #2 and #10 in Table 4.1) and the diffusive regularization model (right; $\alpha = 300$; runs #6 and #13 in Table 4.1). We report the ℓ^2-norm of the velocity field below each individual figure.

Table 1.1

Commonly used symbols and abbreviations.

DCT	discrete cosine transform
FAIR	Flexible Algorithms for Image Registration [54]
LDDMM	large deformation diffeomorphic metric mapping
MRI	magnetic resonance imaging
PDE	partial differential equation
PET	positron emission tomography
PIC	particle-in-cell (method)
SSD	sum-of-squared-differences
$\mathcal{R}(x)$	reference/fixed image
$\mathcal{T}(x)$	template image (image to be registered)
$\mathcal{C}$	PDE constraint
$\mathcal{D}$	distance or similarity measure
$\mathcal{S}$	regularization model (smoother)
α	regularization weight
x	spatial coordinate; $x \in \Omega$
Ω	spatial domain; $\Omega \subset \mathbb{R}_d$
$v(x, t)$	velocity field
$u(x, t)$	transported image intensities
$y(x)$	transformation/mapping
N	number of time steps for computing the characteristic
n	number of unknowns (i.e., the dimension of the discretized velocity field)
n_t	number of cells in space-time grid
I	interpolation operator
∇	gradient operator
$\nabla\cdot$	divergence operator
∂_t	time derivative

Table 4.1

Registration quality. We report registration results as a function of the regularization weight α for the first two datasets of the NIREP repository. We compare the proposed method considering stationary and non-stationary velocity fields. We report results for the curvature (H^2) regularization model (theoretically required to guarantee the existence of a diffeomorphic deformation map) and a diffusive (H^1) regularization model. We compare the proposed method to a formulation for diffeomorphic image registration based on a hyperelastic regularization method [15]. We use the experimentally determined regularization weights $\alpha_1 = 100$ (length regularizer), $\alpha_2 = 10$ (surface regularizer), and $\alpha_3 = 100$ (volume regularizer). We report values (from left to right) for the Dice coefficient after registration and the extremal values for the Jacobian (min and max). The initial value for the Dice coefficient for the considered datasets is 5.54E−1.

method	run	α	dice	min(det ∇y)	max(det ∇y)	time (speedup)	
hyperelastic	#1	100, 10, 100	7.93E−1	4.51E−1	1.86	3.71E+3	(1.00)
			stationary LDDMM				
curvature	#2	10	7.76E−1	8.88E−2	8.79	1.82E+3	(2.04)
	#3	25	7.55E−1	1.19E−1	5.77	1.86E+3	(2.00)
	#4	50	7.34E−1	1.79E−1	4.74	1.45E+3	(2.56)
	#5	100	7.14E−1	2.37E−1	3.13	1.47E+3	(2.51)
diffusion	#6	300	7.86E−1	7.79E−2	6.38	1.96E+3	(1.89)
	#7	400	7.75E−1	1.02E−1	6.26	1.94E+3	(1.91)
	#8	500	7.66E−1	1.18E−1	6.17	1.95E+3	(1.90)
			non-stationary LDDMM				
curvature	#9	5	7.60E−1	9.55E−2	6.36	7.64E+3	(0.49)
	#10	10	7.60E−1	8.48E−2	5.73	7.50E+3	(0.49)
	#11	25	7.47E−1	1.31E−1	5.68	6.18E+3	(0.60)
	#12	50	7.35E−1	2.12E−1	3.68	5.98E+3	(0.62)
diffusion	#13	300	8.05E−1	7.71E−2	6.48	3.29E+3	(1.13)
	#14	400	7.93E−1	1.08E−1	5.40	3.34E+3	(1.11)
	#15	500	7.67E−1	1.53E−1	6.50	2.59E+3	(1.43)