Journal of Mathematical Imaging and Vision (2026) 68:9
https://doi.org/10.1007/s10851-026-01284-y

Stationary Velocity Fields on Matrix Groups for Deformable Image Registration

Johannes Bostelmann[1] · Ole Gildemeister[1] · Jan Lellmann[1]

Received: 28 August 2024 / Accepted: 13 January 2026 / Published online: 17 February 2026

Abstract
The stationary velocity field (SVF) approach allows to build parametrizations of invertible deformation fields, which is often a desirable property in image registration. Its expressiveness is particularly attractive when used as a block following a machine learning-inspired network. However, it can struggle with large deformations. We extend the SVF approach to matrix groups, in particular SE(3). This moves Euclidean transformations into the low-frequency part, toward which network architectures are often naturally biased, so that larger motions can be recovered more easily. This requires an extension of the flow equation, for which we provide sufficient conditions for existence. We further prove a decomposition condition that allows us to apply a scaling-and-squaring approach for efficient numerical integration of the flow equation. We numerically validate the approach on inter-patient registration of 3D MRI images of the human brain

Keywords Deformable image registration · Implicit neural representation · Stationary velocity fields · Flow equation · Scaling and squaring

1 Introduction

Image registration describes the task of aligning two or more images by providing a reasonable transformation. This task has a wide range of applications, including atlas-based segmentation [1], tracking of changes in medical data such as tumor growth and fracture healing [2], Synchronous Localization and Mapping [3], and image stitching for panoramic or (satellite/drone) captures [4].
Typically, the sought transformation increases the image similarity while fulfilling additional criteria for a physically plausible deformation, such as its smoothness or invertibility. The computation of a useful alignment is often described by an optimization problem, in which an objective function is minimized over a specific space of parametrized deformations.

✉ Johannes Bostelmann
johannes.bostelmann@uni-luebeck.de

✉ Ole Gildemeister
o.gildemeister@uni-luebeck.de

✉ Jan Lellmann
jan.lellmann@uni-luebeck.de

[1] Institute of Mathematics and Image Computing, University of Lübeck, Maria-Goeppert-Str. 3, 23562 Lübeck, Germany

Given two scalar-valued images represented by functions $I_1, I_2 : \Omega \to \mathbb{R}$ on the image domain $\Omega \subseteq \mathbb{R}^d$, a typical approach for finding a suitable deformation $\phi : \Omega \to \mathbb{R}^d$ that maps corresponding points in I_1 to points in I_2 is to formulate a variational problem of the form

$$\min_{\phi \in \mathcal{D}} \mathcal{L}(\phi; I_1, I_2), \quad \mathcal{L}(\phi; I_1, I_2) := J(I_1, I_2 \circ \phi) + R(\phi). \tag{1}$$

The set of deformations $\mathcal{D}$ is often chosen to consist of all functions $\phi : x \mapsto x + u(x)$, where the displacement $u : \Omega \to \mathbb{R}^d$ is an element of a suitable topological vector space. The functional J is a similarity term, in which smaller values indicate a higher similarity between its arguments, such as an L^2 distance; the regularizer $R : \mathcal{D} \to \mathbb{R}$ shall provide a bias toward physically reasonable deformations and can be used to stabilize the problem.
Choosing suitable similarity terms and regularizers has been intensively investigated [5, 6] and is not the goal of this work. Instead, we focus on another important aspect: the choice and parametrization of the deformation space $\mathcal{D}$. In practice, expressiveness and convergence can depend heavily on the parametrization used. Moreover, this choice can influence the meaning and practicability of the regularizer.

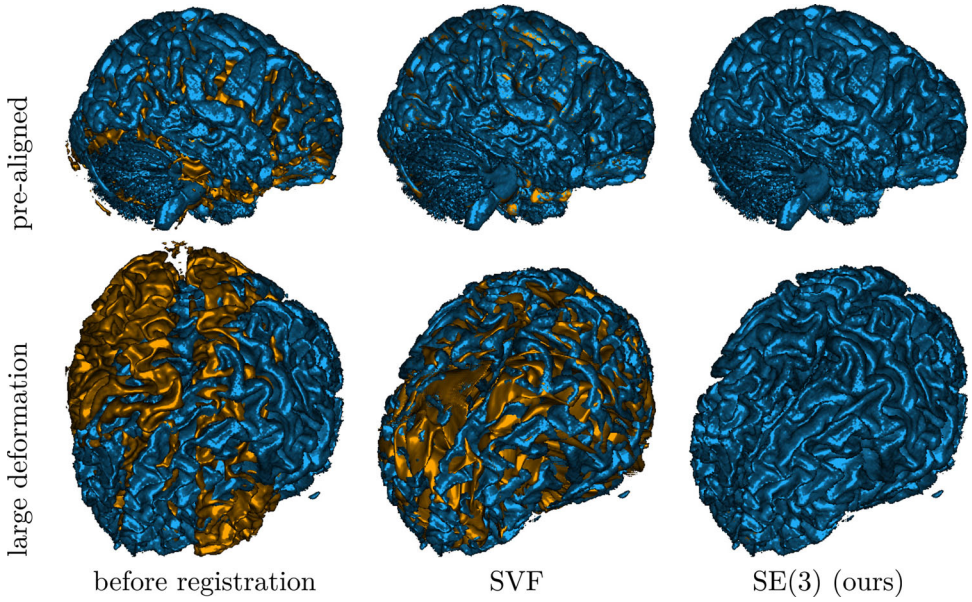


Fig. 1 Comparison of the stationary velocity field (SVF) approach and our proposed matrix group-valued approach using the SE(3) group on synthetically deformed human brain MRI data. While both methods manage to align the 3D volumes under small deformations (top row), the SVF approach struggles when the input images are not pre-aligned (bottom row). The resulting deformation field shows clear alignment issues (center), which are alleviated by the proposed matrix group-valued approach (right)

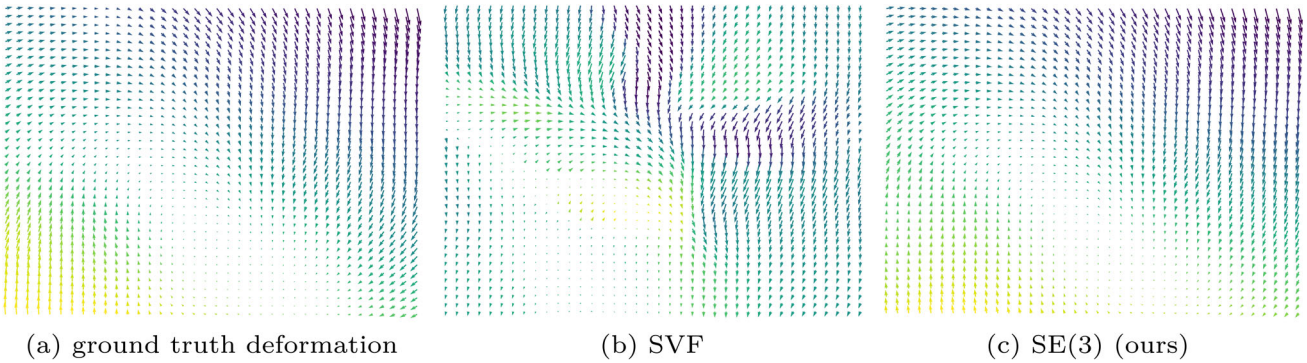


Fig. 2 2D slices of the 3D deformation fields from the bottom row of Fig. 1. The arrow colors indicate the displacement in the direction orthogonal to the slice. While SVF roughly aligns the images judging in terms of visual quality in Fig. 1, inspecting the deformation field (**b**) shows that it is far from the ground truth (**a**). The proposed parametrization with matrix groups (**c**) yields a result that closely matches the ground truth, (color figure online)

To motivate our work, consider the example in Fig. 1, in which we performed a 3D registration of two MRI brain scans from the OASIS [7] dataset using the popular *Stationary Velocity Field (SVF)* approach [8–10] and our proposed method (denoted SE(3)). In the first example, the sought deformation is comparably small before the registration process, and both approaches perform equally well. In the second example, the deformation includes a larger rotational component. When looking at the deformed images only, the SVF approach appears to generate an alignment that is clearly worse, but not catastrophically so. This is deceptive: inspecting the generated deformation fields (Fig. 2), it becomes clear that the SVF approach generates a deformation that—while it maps corresponding intensity values reasonably well between the images—is far from the ground truth.

The findings on these (synthetic) examples are compatible with our perception of the existing literature: In general, we found that SVF-based approaches mostly seem to be applied to pre-aligned images, and/or that the judgment of their accuracy is solely based on the similarity of the images after registration or on related proxies, such as the overlap of known segmented regions, that do not necessarily imply sensible deformation fields.

Nevertheless, the SVF approach was used successfully in the past and has many attractive properties, such as the possibility of generating diffeomorphic deformations and of easily obtaining the inverse deformation, for example, for bidirectional approaches. Therefore, in this work, we aim to derive an extension that preserves these benefits while allowing for a better handling of large deformations.

General approach In general, in order to solve (1) numerically, a parametrization of the deformation is required, which we denote abstractly by ϕ_θ, where $\theta \in \mathbb{R}^n$ is a set of real

parameters to be determined for each given image pair I_1 and I_2.
The registration problem (1) then becomes

$$\min_{\theta \in \mathbb{R}^n} \mathcal{L}(\phi_\theta; I_1, I_2). \tag{2}$$

Classically, ϕ_θ is either—in the finite-differences setting—represented by its values on a grid, taken directly from the parameter vector θ, or—in a finite elements approach—is assumed to be a linear combination of basis functions, where θ determines the coefficients in the linear combination. Typical choices include spline-based parametrizations [11] and the use of radial basis functions [12].
While comparably straightforward to apply, such linear parametrizations have disadvantages: Firstly, it is hard to enforce global regularity of the deformation, in particular invertibility, which is often desirable or even mandated by the physical requirement that the deformation should not contain any "folds", i.e., self-intersections. Secondly, due to the local nature of the classical basis functions, components in θ typically affect only small parts of the deformation, which makes the optimization process more complicated: In order to correctly identify global deformations, a large share of the parameters needs to be adjusted. Furthermore, random local similarities often create local minima or stationary points in the energy, which can trap iterative solvers.
The approach taken in this work builds on the approach used in [10]:

1. Rather than explicitly parametrizing the deformation, we assume it to be the solution of a *flow equation.* The behavior of the flow equation is governed by its right-hand side in the form of a *velocity field* function v. Under certain conditions, this setup guarantees that the deformation is sufficiently regular and that an inverse exists which can be computed efficiently.
2. For the parametrization of the velocity field v, we build on the recent success of coordinate-based networks and assume that the velocity field is described by a network/neural architecture, which is parametrized —in a nonlinear way—by θ.

From now on, by S we denote the (nonlinear) solution operator that maps such a velocity field to a deformation $\phi = S(v)$. The neural network can, in the most generic way, be thought of as a mapping G that takes a parameter vector θ and generates a velocity field $v_\theta = G(\theta)$.
Overall, this results in the model

$$\min_{\theta \in \mathbb{R}^n} \mathcal{L}(S(G(\theta)); I_1, I_2), \tag{3}$$

where the generator G turns the parameter vector θ into a velocity field, which is then mapped to a deformation by the solution operator S that solves a *flow equation*. Figure 3 visualizes the complete process of generating the deformation ϕ_θ from the parameter vector θ.
Flow equation In this work, we particularly focus on the flow equation that is used. In the classical setting also used in [10], one introduces an artificial time $t \in [0, 1]$ and prescribes a—possibly time-varying—velocity field v, for which at each time point t, the associated velocity field v_t is an element of a suitable function space V. The deformation is then induced through the ordinary differential equation

$$\begin{aligned} \frac{\partial \phi(x,t)}{\partial t} &= v_t(\phi(x,t)) \text{ for } t \in [0,1], \\ \phi(x,0) &= x, \end{aligned} \tag{4}$$

yielding a final deformation $\phi(\cdot, 1)$ at time $t = 1$.
In the LDDMM approach, this allows to construct a metric on the infinite-dimensional manifold of diffeomorphisms based on properties of v, which has also found use as a regularizer [13]. Finding that minimizing curves are geodesics with respect to a particular Riemannian metric, a shooting formulation with an initial momentum was subsequently developed [14]. Promising a lower computational complexity without significantly hurting expressivity in practical applications, stationary velocity fields also found use [9]. These flow-based approaches have been successful in medical image registration, as they enforce diffeomorphic deformations if the velocity field fulfills an integrability condition, which can be ensured by a given differentiability operator [13, 15].
Matrix-valued velocity fields A disadvantage of (4) is that the velocity fields that represent common affine deformations such as rotations are non-trivial and require the network to generate such vector fields from scratch (Fig. 4).
Therefore, we propose to extend the stationary flow equation (4) by lifting it to *matrix fields:* Firstly, the deformation is parametrized as $\phi(x) := M(x)\bar{x}$ where $\bar{x}$ is the extension $(x, 1)^\top$ of x to homogeneous coordinates, and the matrix field $M : \Omega \to G$ maps to a subgroup $G \subset \mathrm{GL}(\mathbb{R}, d+1)$ of matrices in $\mathbb{R}^{(d+1)\times(d+1)}$. This allows parametrizing common linear transformations, such as rotations, using *constant* matrix fields, which can potentially be learned more easily.
Secondly, in order to formulate the flow equation for these matrix fields, we replace the classical velocity field v by a stationary velocity field $\nu : \Omega \to \mathfrak{g}$ with values in $\mathfrak{g}$, the finite-dimensional vector space of right-invariant vector fields on G, that is, the Lie algebra of G.
The generalized flow equation then takes the form

$$\frac{\partial M}{\partial t}(x,t) = \nu(M(x,t)\bar{x})_{M(x,t)} \tag{5}$$

$$M(\cdot, 0) = id. \tag{6}$$

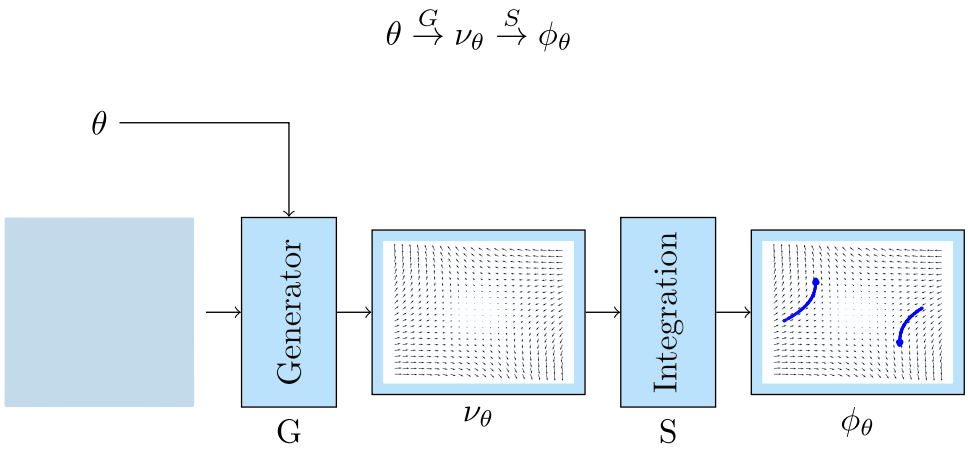


Fig. 3 Process of generating a deformation field ϕ_θ from the parameter vector θ. The network-based *generator* G transforms the parameter vector into a *velocity field* v_θ. The *solution operator* S solves an associated flow equation, resulting in a final deformation ϕ_θ. Classically, the velocity field and the flow equation are formulated in Euclidean space; we extend the approach to the matrix group setting

Approach	Implicit Neural Representation	Flow Equation	Matrix Groups
Balakrishnan et al. 19		✓	
Han et al. 23	✓	✓	
Park et al. 21	✓		✓
ours	✓	✓	✓

Fig. 4 Components used to parametrize deformations in recent approaches. In this work, we introduce a combination of matrix groups with a flow equation based on implicit neural representations

For the specific choice $G = T(\mathbb{R}, 3)$, the group of translations in $\mathbb{R}^3$, the formulation (5) reduces to the classical flow equation (4) in the stationary case.

In a similar spirit, the authors of [16] proposed a parametrization of the deformation using a manifold/SE(3)-valued field, which improved the learning of rotational deformations. This technique shifts the output field into a lower-frequency domain. However, their description is not flow-based and does not inherently guarantee diffeomorphisms (Fig. 4).

Contribution To the best of our knowledge, the differentiable structure of matrix groups has not yet been exploited for diffeomorphic, deformable (non-rigid) image registration tasks using neural architectures.

This article is structured as follows.

- We propose to extend the existing framework that relies on implicit neural representation and the flow equation by lifting the flow equation to the setting of matrix fields in Sect. 3. We prove the existence of a solution of the generalized flow equation in Thm. 1 (proof in Sect. 6.1).
- To obtain a network of manageable depth, in Sect. 3.2, we extend and employ the numerical scaling-and-squaring approach for integrating the flow equation to the matrix group-valued setting. Specifically, we prove that the lifted flow equation satisfies a decomposition proposition that is crucial for the scaling-and-squaring approach (Thm. 2, proof in Sect. 6.2).
- Finally, in Sect. 4, we confirm numerically that our approach generates invertible deformations, and we validate the accuracy of the generated dense displacement field on synthetic and real-world 3-D registration problems.

Related Work

Supervised learning frameworks for deformable image registration include [17–19] which are based on artificial deformations and label-driven approaches (also called semi-supervised) [20]. Ensuring that the artificial training data generalizes well to a specific task is a difficult problem known in the literature as reality gap [21].

Unsupervised learning-based methods for image registration are typically tuned to minimize a loss based on an image similarity term with possible additional regularization. A successful approach is to start from established image registration approaches that are based on finding a certain optimal set of parameters and replace the optimization process with a trained network: In [22], the network is trained to output a stationary velocity field as the basis for generating the deformation in a setting solely based on image similarity and regularization, as well as in a combined approach using additional segmentation data. In [23], a network is trained to generate initial moments for the shooting formulation of LDDMM based on precalculated moments using a classical approach. In [24], the networks directly predict parameters for B-spline-based registration. In [25], Laplacian Pyramid Networks are used which directly produce a deformation field.

A conceptional somewhat different approach uses trained neural architectures as an image similarity metric called "deep similarity metrics," to which classical optimization

methods are applied [17]. Combining neural architectures with LDDMM approaches, the authors of [26] proposed a ResNet architecture to model the time-variant flow equation, while [27] introduces an adversarial LDDMM learning method.

Architecturally more akin to classical variational approaches is the concept of implicit neural representations [28], in which the unknown function is parametrized nonlinearly using a network. There, the estimation of the network parameters is input-specific and generally requires an optimization procedure as in classical variational approaches. Such approaches work particularly well for novel view synthesis, for which the approach is termed Neural Radiance Fields [29]; see also [16] for an adaptation to deformable scenes. In the context of solving partial differential equations, these networks are known as physics-informed neural networks (PINNs) and have been applied to tasks such as reconstructing flows from 2D observations [30] and PDE-constrained optimization [31]. Other examples include data compression of images and videos [28, 32].

In the context of deformable image registration, implicit neural representations were studied in [10, 33] in a diffeomorphic setting with stationary velocity fields and, among others, in [6] for implicit regularization.

Our proposed approach is based on a specific representation of transformations in the form of matrix groups. Transformations of the Euclidean space that preserve specific structures have already been studied and categorized for a long time. Since the 19th century, continuous transformation groups were considered with a differentiable structure, also known as infinitesimal transformations. This effort culminated in a full-fledged theory of Lie groups and their corresponding Lie algebras [34]. Since then, the theory found use in applications including quantum theory [35], computer graphics [36, 37], odometry [38], and robotics [39, 40]. In the context of machine learning, matrix groups were often employed to ensure specific invariances—for example, in classification tasks [41]—or equivariances [42]. The authors of [43] introduced tangent space backpropagation for automatic differentiation on Lie groups including SO(3), SE(3), and SIM(3). Combining image registration with transformation groups, the authors of [16] proposed parametrizing a deformation pointwise by members of SE(3) to better represent rotational deformations.

2 Mathematical Preliminaries

A real *Lie group* is a group $(G, \cdot)$ that is also a smooth real *manifold* such that the following conditions hold [44]:

- The mapping $\cdot : G \times G \to G : (g, h) \mapsto g \cdot h$ is continuous.
- The mapping $()^{-1} : G \to G : g \mapsto g^{-1}$ is continuous.

Table 1 lists the Lie groups that are most relevant to this work, such as the special Euclidean group of rigid transformation of points in $\mathbb{R}^3$, SE(3), which can be thought of as a subset of real 4×4 matrices when using the usual homogeneous coordinate representation for the points in $\mathbb{R}^3$. Such groups that are representable by (invertible) matrices are also known as *matrix groups*.

All groups in Table 1 can thus be embedded in the 16-dimensional vector space $\mathbb{R}^{4\times4}$. In fact, Whitney's embedding theorem [45] guarantees that, under weak theoretical conditions, each d-dimensional smooth real manifold can be embedded into a $2d$-dimensional Euclidean space.

Therefore, the *tangent space* at a point $p \in M$ can be visualized as an affine subspace in $\mathbb{R}^{4\times4}$ that touches and linearly approximates the manifold at p; a tangent vector is then a vector attached to p that lies in this affine subspace.

More rigorously, the tangent space T_pG at a point $p \in G$ is defined here as a set of equivalence classes of smooth curves on the manifold $\gamma : I \subset \mathbb{R} \to G$, $\gamma(0) = p$ with the equivalence relation

$$\gamma_1 \sim \gamma_2 \Leftrightarrow (\phi \circ \gamma_1)'(0) = (\phi \circ \gamma_2)'(0) \tag{7}$$

for every chart $\phi : U_p \to \mathbb{R}^k$ defined on a neighborhood U_p of p. If the manifold can be embedded into a vector space $\mathbb{R}^l$, the charts can be replaced by embeddings $i : U_p \to \mathbb{R}^l$ [46].

The disjoint union of all tangent spaces $TG = \bigsqcup_{p\in G} T_pG$ is called a *tangent bundle*. A *vector field* $X : G \to TG$ with $Xp \in T_pG$ is a section through the tangent bundle in the sense that it assigns to each point in G a vector in its tangent space.

Given a differentiable map on the manifold $\chi : G \to G$, its *push-forward* or *differential* $\mathcal{D}\chi_p : T_p \to T_{\chi(p)}$ maps between corresponding tangent spaces. This operation can be defined at a point $p \in G$ by mapping a representative γ from T_p to equivalent curves of $\chi \circ \gamma$, i.e.,

$$\mathcal{D}(\chi)_p(\gamma) = \chi \circ \gamma. \tag{8}$$

A particular case of smooth vector fields is the class of *left/right invariant* (smooth) vector fields, which is of interest for the following chapters. It is reasonable to consider the smooth and bijective maps induced by right/left multiplication of group elements $l_g : G \to G$, $h \mapsto g \cdot h$ and $r_g : G \to G$, $h \mapsto h \cdot g$, respectively. Such invariant vector fields fulfill the criteria

$$\text{left invariance:} \quad \mathcal{D}_{l_g} X_p = X_{g\cdot p}$$
$$\text{right invariance:} \quad \mathcal{D}_{r_g} X_p = X_{p\cdot g}$$

Table 1 Common matrix groups acting on $\mathbb{R}^3$.

Matrix group	T($\mathbb{R}$, 3)	SE($\mathbb{R}$, 3)	SIM($\mathbb{R}$, 3)	Aff($\mathbb{R}$, 3)	PGL($\mathbb{R}$, 3)
Dimension	3	6	7	12	15
Transform	Translation	Rigid	Rigid, scaling	Affine	Perspective
Preserves	Orientation &	Area &	Angles &	Parallels,ratio of distances &	Collinearity, cross-ratio
Known closed form of exp; log	yes	yes	yes	no	no

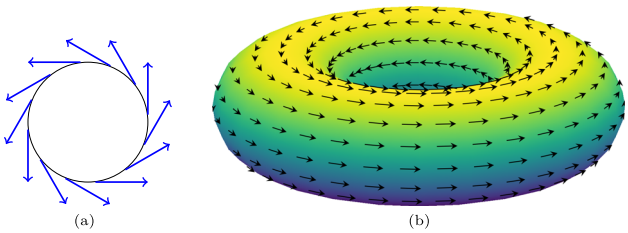


Fig. 5 **a** Visualization of a right-invariant vector field on $SO(\mathbb{R}, 2)$, homeomorphically identified with a circle. **b** Visualization of a right-invariant vector field on the torus corresponding to the product group $SO(\mathbb{R}, 2) \times SO(\mathbb{R}, 2)$. In both cases, right-invariant vector fields can be parametrized using a single scalar (circle) or two scalars (torus). This is, in general, not possible for non-invariant vector fields, which form a vector space of infinite dimension

for all $p, g \in G$, where D_{l_g} describes the push-forward/differential of the left or correspondingly right multiplication. In general, the space of vector fields is a function space, which makes finding a numerical representation difficult. The class of invariant vector fields, however, can be uniquely described by a single value: their value at the identity. For the finite-dimensional manifolds in this work, the space of such invariant vector fields is finite-dimensional. This greatly simplifies numerical treatment and allows for generating invariant vector fields using a coordinate-based network in which each output channel corresponds to one dimension of the matrix group.

An example of a right invariant vector field for the commutative one-dimensional group SO(2) of two-dimensional rotation matrices, which is topologically equivalent to a circle, is given in Fig. 5. The right invariance of the vector field translates to the condition of the tangential vectors having the same signed length.

Curves $\gamma : \mathbb{R} \to G$ solving the differential equation

$$\frac{\partial \gamma}{\partial t}(t) = X_{\gamma(t)} \tag{9}$$

$$\gamma(0) = g_0 \tag{10}$$

are called *integral curves* originating from g_0 [44].

In the case of embedded matrix groups, integral curves of right invariant vector fields X take the form

$$\gamma(t) = \exp(t \cdot X_{id}) g_0 \tag{11}$$

where "exp" denotes the matrix exponential

$$\exp(M) := \sum_{i \in \mathbb{N}} \frac{M^i}{i!}. \tag{12}$$

This can be shown by differentiating the curve (11) with respect to time:

$$\begin{aligned} \frac{\partial \gamma}{\partial t}(t) &= \exp(tX_{id}) X_{id} g_0 = X_{id} \exp(tX_{id}) g_0 \\ &= X_{\exp(tX_{id})g_0} = X_{\gamma(t)}. \end{aligned} \tag{13}$$

In Sect. 3.2, the matrix exponential will be used to solve the extended flow equation on matrix groups numerically.

3 Image Registration using Velocity Flow on Matrix Group Valued Fields

Before going into the analysis, we briefly summarize our approach as outlined in the introduction. We consider an overall model of the form

$$\min_{\theta \in \mathbb{R}^n} \mathcal{L}(S(G(\theta)); I_1, I_2). \tag{14}$$

The final deformation $\phi_\theta := S(G(\theta))$ is parametrized by the parameter vector θ. This parameter vector is first turned into a stationary velocity field $v_\theta := G(\theta)$ by the neural network G. The solution operator S then computes the final state $M(\cdot, 1)$ of the flow equation

$$\begin{aligned} \frac{\partial M}{\partial t}(x, t) &= v_\theta(M(x, t)\bar{x})_{M(x,t)} \\ M(\cdot, 0) &= id \end{aligned} \tag{15}$$

and returns the final deformation

$$\phi_\theta(x) := S(v_\theta)(x) := M(x,1)\begin{pmatrix} x \\ 1 \end{pmatrix}. \tag{16}$$

In the following, we discuss the individual parts of this approach in more detail:

- Section 3.1 covers the generalized flow equation and existence,
- Section 3.2 addresses the question on how to efficiently implement the solution operator S of the flow equation,
- Section 3.3 concerns the choice of similarity measure J and regularizer R that form the objective $\mathcal{L}$, and
- In Sect. 3.4, we discuss the network architecture G.

In Sect. 4, we discuss numerical results for synthetic deformations and for inter-patient registration tasks.

The idea behind the generalized flow equation is to obtain more freedom and control in tuning the hyperparameters to create a bias so that specific (large) deformations become easier to learn, while preserving the theoretical features of a flow-based approach, namely "nearly" diffeomorphic deformations and a simple method for calculating the inverse deformation.

3.1 Extended Flow Equation and Existence

Velocity fields and vector fields The starting point for including the flow equation (15) in our approach is the classical *stationary velocity field* approach, in which the flow equation takes the form

$$\begin{aligned} \frac{\phi(x,t)}{\partial t} &= v(\phi(x,t)) \\ \phi(x,0) &= x. \end{aligned} \tag{17}$$

In contrast to a time-dependent velocity field v_t as in (4), this formulation reduces the computational effort [47] notably. It is used in classical frameworks [47] as well as frameworks using a neural architecture [10, 22]. If the velocity field v is an element of $C_0^1(\Omega)$, the solution $\phi(\cdot,1)$ is continuously differentiable and has a continuously differentiable inverse, i.e., it belongs to the class of *diffeomorphisms*. This is a direct consequence of the results in [9], where the more general case of time-varying velocity fields was considered.

Diffeomorphic deformations are often desirable for registration tasks in which the deformation is caused by a physical process, as is commonly the case in biomedical data. This is due to the fact that non-diffeomorphic deformations, in particular self-intersections, typically are physically implausible. Unfortunately, not every diffeomorphism can be obtained as a solution of (4) with some stationary velocity field [48]; however, known exceptions seem rather implausible for anatomical deformations [48].

Our approach (15) is based on the idea that velocity fields that correspond to basic, expected deformations—in particular, translations and rotations—should be "simple" in the sense that they are low-frequency, ideally even constant, and therefore easy to generate. For the standard SVF model (17), constant velocity fields can only model homogeneous translations of the whole domain. By employing *matrices* $M(x,t)$ instead of vectors $v(x,t)$ as in (15) and constructing the final deformation as in (16), constant velocity fields can also model affine deformations.

In order to restrict the space of possible matrices, we require that all $M(x,t)$ are elements of a *matrix group* G, i.e., of a continuous subgroup of the general linear group $\mathrm{GL}(4,\mathbb{R})$. With the usual matrix multiplication as group operation, these groups are also Lie groups with a differential (manifold) structure.

Then, the time derivative $\frac{\partial}{\partial t}M(x,t)$ in the flow equation (5) is an element of the tangent space $T_{M(x,t)}G$. This requires the velocity field v_θ on the right-hand side to be an element of $T_{M(x,t)}G$, which is problematic, as our goal is to parametrize the velocity field by a neural network: For a general manifold, the parametrization of the tangent space depends on the location $M(x,t)$, which is not known when deciding on the structure of the network, and furthermore changes with time.

The key is to restrict the vector fields to the space $\mathfrak{g}$ of *right-invariant vector fields* on G. An element $v(x) =: v' \in \mathfrak{g}$, $v' : G \to TG$ of the space of right-invariant vector fields assigns to each point $p \in G$ a vector in the tangent space T_pG, denoted by v'_g. Most importantly, the right invariance assures that each such vector field v' is uniquely described by its value at the identity id. Therefore, the neural network can be constructed to map into the tangent space $T_{id}G$ of G at the identity.

As an example, consider $SO(2)$, the group of rotations around the origin in $\mathbb{R}^2$. Any such rotation is uniquely parametrized by a rotation angle α. Solving the flow equation (15) with a right-invariant vector field would take any such rotation and increase or decrease the rotation angle by the same amount, irrespective of the angle of the original rotation (Fig. 5).

As the velocity field v is stationary, but not necessarily constant, this formulation may also capture non-rigid deformations, even when the chosen matrix group only corresponds to (a subgroup) of rigid deformations. Setting G to the group of translations $T(\mathbb{R})$, this formulation reduces to the previous SVF setting of (4).

If G contains the subgroup of translations, then the space of possible deformations is at least as large as the classical SVF approach. However, choosing a more expressible matrix group may be seen as an over-parametrization of the resulting deformation, as different velocity fields may lead to the same

induced deformation. For example, in case of $G = \mathrm{SE}(2)$, a rotation around the origin can either be achieved by a constant field v in which only the angular velocity is nonzero, or by a spatially varying "swirl" of the translational components of $SE(2)$.

Existence and smoothness We first justify the use of the generalized SVF/flow equation (15) by showing the existence and smoothness of the solutions. Note that v_{id} maps from the image domain Ω into the space of right-invariant vector fields $\mathfrak{g}$ *evaluated at* id, so that $v_{id}(x) \in T_{id}G$.

We are interested in a 3D registration setting, so we now consider subgroups of the affine group of $\mathbb{R}^3$ for G. Its elements can be represented as 4×4 matrices.

Theorem 1 *Let Ω be compact, let $\mathfrak{g}$ denote the space of right-invariant vector fields on $G \subset \mathrm{GL}(\mathbb{R}, 4)$, and let $v : \Omega \to \mathfrak{g}$ be such that its evaluation at the identity v_{id} is Lipschitz continuous. Then, the solution M of (5) exists uniquely and remains bounded in $C(\Omega, \mathbb{R}^{4\times4})$ when viewed as a map $t \mapsto M(\cdot, t)$.*

Proof See Sect. 6.1. □

3.2 Numerical Integration by Scaling and Squaring

In the classical setting, stationary velocity fields allow the use of the *scaling-and-squaring* approach [8] to approximate the solution of the ODE (4) numerically. As can be seen from Alg. 1, the first step is essentially a forward Euler step with step size 2^{-n}. This is followed by the squaring steps $v_{j+1} \leftarrow v'_j + v_j$ which correspond for $k = n - j$ to the "decomposition condition"

$$\phi\left(x, 2^{-k+1}\right) = \phi\left(\phi(x, 2^{-k}), 2^{-k}\right). \tag{18}$$

Intuitively, the time resolution scales *exponentially* in n, making this approach computationally attractive.

As we aim at including the solution operator into a network-based optimization scheme, this idea is crucial for limiting the network depth. Therefore, we extend the technique to the setting of matrix groups in order to approximate the solution to (15) effectively in a discretized setting. The complete algorithm is shown in Alg. 2. (Fig. 6) The final deformation field at $t = 1$ is calculated iteratively, starting with a simple forward Euler step

$$M(x, 2^{-n}) = \exp\left(2^{-n} v_{id}(x)\right) \tag{19}$$

and iterating over n squaring steps. The algorithm is based on the "Exponential discretization" scheme (82), which is further discussed in Sect. 6.2. There, we also prove the convergence of the semi-discrete scheme to the continuous solution of (15).

For the matrix-valued deformations considered here, the "decomposition condition" (18) takes the form

$$M(x, 2^{-k+1}) = M\left(PM\left(x, 2^{-k}\right)\bar{x}, 2^{-k}\right) M\left(x, 2^{-k}\right), \tag{20}$$

where $P : \mathbb{R}^4 \to \mathbb{R}^3$ denotes the projection which removes the last component.

In order to successfully apply the scaling-and-squaring scheme, it is crucial that the solution of the flow equation satisfies the decomposition condition (20). This is guaranteed by the following theorem.

Theorem 2 *Let M be a solution of the flow equation (5). Assume that $\Omega \subset \mathbb{R}^3$ is compact and that the velocity term $v : \Omega \to \mathfrak{g}$ satisfies $v_{id} \in C^1(\Omega, \mathbb{R}^{4\times4})$. Then, M satisfies the decomposition condition*

$$M(x, 2T) = M(PM(x, T)\bar{x}, T)M(x, T) \quad \text{for all } T > 0. \tag{21}$$

Furthermore, when viewed as a map $t \mapsto M(\cdot, t)$, it holds that $M \in C^2([0, 1], C(\Omega, \mathbb{R}^{4\times4}))$. where we identify $M(x, t)$ with $M(t)(x)$ as required.

Proof See Sect. 6.2. □

When applying the scaling-and-squaring scheme for integrating the flow equation, there are some implementation details to consider:

- *Network depth* Each squaring operation doubles the effective resolution in the synthetic time variable. Setting $n = 0$ reduces the method to the direct/non-flow-based approach, where the neural network directly outputs the values for the displacement vector field. Larger values of n yield a better numerical approximation of the ODE, at the cost of network depth and additional computational effort. In this work, n is always set to 7, which yields an implicit time discretization of $\frac{1}{128}$.
- *Interpolation* The scaling-and-squaring approach relies on the efficient evaluation of the solution at the current time and at arbitrary points in space. When discretizing the matrix field on a spatial grid, this requires an interpolation between elements of the given matrix group. As the matrix group is generally curved and non-convex, multi-linear interpolation in the surrounding vector space does not ensure that the matrix-valued fields map to G. Instead, after interpolation, the matrix fields would, in general, map into $\mathbb{R}^{4\times4}$. Therefore, we resort to interpolation on the space of right invariant vector fields on G, which carries vector space structure. The use of the matrix logarithm, which can be calculated analytically

Algorithm 1 Scaling and Squaring

$v_0 \leftarrow 2^{-n} v$ ▷ scaling
for $j \in \{0, \ldots, n-1\}$ **do**
  $v'_j \leftarrow \mathrm{Int}(v_j, x + v_j(x))$
  $v_{j+1} \leftarrow v'_j + v_j$ ▷ squaring
end for
return v_{j+1}

Algorithm 2 Generalized Scaling and Sq.

$\nu_0 \leftarrow 2^{-n} \nu$ ▷ scaling
for $j \in \{0, \ldots, n-1\}$ **do**
  $M_j \leftarrow \exp(\nu_j)$
  $\nu'_j \leftarrow \mathrm{Int}(\nu_j, PM_j\bar{x})$
  $\nu_{j+1} \leftarrow \log(\exp(\nu'_j) \cdot M_j)$ ▷ squaring
end for
return $\exp(\nu_{j+1})$

Fig. 6 Left: Classical scaling-and-squaring approach for fast numerical integration of the flow equation (4) based on a given velocity field v. The interpolation operator $\mathrm{Int}(v, x)$ computes an approximation of $v(x)$. Right: Proposed generalization to the matrix-valued setting (15) for given velocity field v. Again the interpolation operator $\mathrm{Int}(v, x)$ approximates $v(x)$

for SE(3) and SIM(3), allows us to carry out the interpolation in the Lie algebra. The result is then mapped back to the matrix group using the matrix exponential. This step guarantees that the interpolated result is still an element of the given matrix group G.

- *Differentiability* The matrix logarithm and exponential can be implemented in a fully differentiable way; we rely on the implementations for SE(3) and SIM(3) in [43]. The necessary interpolation creates an additional numerical error depending on the smoothness of the velocity field. With our approach using matrix-valued fields, specific deformations, such as rigid deformations for $G = \mathrm{SE}(3)$, can be recreated without these numerical approximation errors, which might be advantageous.
- *Boundary conditions* Deviating from the usual diffeomorphic approach, we do *not* fix the boundary but constantly extend the velocity field by its boundary values outside of the domain with the value at the border. This allows to reconstruct translations or rigid deformations in the SE(3) setting exactly.

Inverse deformation and validation
Taking the interpolation into account, it is prudent to ask whether the proposed scaling-and-squaring approach generates reasonably accurate deformations in practice. To approach this question, note that if

$$\phi = S(\nu_\theta), \tag{22}$$

i.e., ϕ solves the flow equation for given velocity field ν_θ, then the inverse ϕ^{-1} solves the flow equation for $-\nu_\theta$:

$$\phi^{-1} = S(-\nu_\theta). \tag{23}$$

This suggests a method for verifying the consistency of the numerical (scaling-and-squaring) integration scheme: For a given velocity field ν_θ, compute both ϕ_{ν_θ} and $\phi_{-\nu_\theta}$ numerically and measure the residual *forward–backward error*

$$\int_\Omega |(\phi_{\nu_i} \circ \phi_{-\nu_i} - id)(x)| \mathrm{d}x. \tag{24}$$

Similarly, the backward–forward error can be computed from $\phi_{-\nu_i} \circ \phi_{\nu_i}$. If the discretization is perfectly self-consistent, both residuals should be close to zero.
Figure 7 shows the results depending on the number of integration steps n. Up to $n = 10$, the relative error decreases exponentially before stagnating, which we assume is caused by an accumulation of interpolation errors. For our common choice $n = 7$, the average forward–backward error is below 0.05 voxels for both the classical flow equation and our proposed matrix-valued approach on SE(3).
Perfect invertibility is neither expected nor required by the overall approach: After all, the primary goal of employing the flow equation is to formulate an expressive mapping from network outputs to a deformation field. However, the good results suggest that the scheme could be useful for a bidirectional registration approach in which both ϕ and $-\phi$ occur in the objective. We investigate this further in the following section.

3.3 Similarity Term and Regularizer

Regardless of the parametrization of the deformation, the approach (14) still requires choosing a distance J for the data term as well as a regularizer R.
Similarity We consider the global negative normalized cross-correlation NCC for the data term. In order to compare the deformed template T and reference images R using NCC, we define the mean value $\bar{T}$ as

$$\bar{T} = \frac{1}{|\Omega|} \int_\Omega T(x) \mathrm{d}x \tag{25}$$

and $\bar{R}$ analogously. The data term is then given by

$$J(R, T) := 1 - \mathrm{NCC}(R, T) := 1 - \frac{\langle T - \bar{T}, R - \bar{R} \rangle_{L^2}}{\sqrt{\|T - \bar{T}\|^2_{L^2} \|R - \bar{R}\|^2_{L^2} + \epsilon}} \in [0, 2]. \tag{26}$$

This corresponds to the negative cosine of the angle between the vectorized zero-mean images and is an established multimodal similarity term due to its invariance to affine intensity

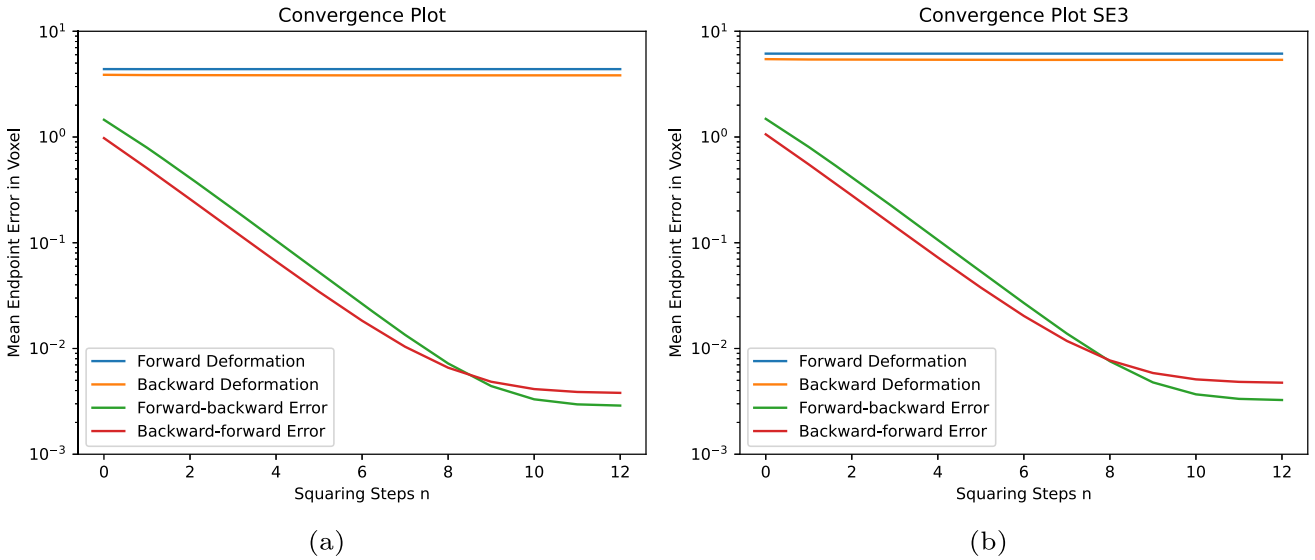


Fig. 7 Residual errors depending on the number of scaling and squaring steps for a given velocity field for (**a**) classical integration of the flow equation in Euclidean space and (**b**) integration on SE(3) using the modified flow equation

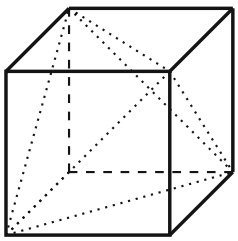


Fig. 8 Decomposition of a voxel into one inner and four outer tetrahedra. The edges of the tetrahedra are drawn with dotted lines. Evaluating the oriented volume of all tetrahedra after deformation allows to detect foldings

transformations [49]. We stabilize the division by adding $\epsilon = 10^{-5}$ to the square root in the denominator.

Regularizer To prevent folding caused by the discretization and/or interpolation, we include a regularization term that penalizes Jacobians with determinant smaller than a threshold $\epsilon > 0$ via [50]:

$$R_\epsilon(\phi) := \int_\Omega \max[-\det(D\phi(x)) + \epsilon, 0]\mathrm{d}x. \tag{27}$$

In our experiments, we set $\epsilon = 0.01$ in order to decrease the likelihood of the first-order optimization method accidentally passing into a region of zero or negative determinant due to the non-infinitesimal step size.

The determinants can be evaluated in the discretized setting on a (transformed) grid by dividing each voxel into five 3-simplices/tetrahedra (Fig. 8) and computing their oriented volume. The latter are then summed with weights of $\frac{1}{6}$ for the outer, and $\frac{1}{3}$ for the inner tetrahedra. This approach is a more isotropic variant of the one used in [51], where all tetrahedra are extended from a single vertex.

In addition to that, we also employ gradient or Hessian regularization to encourage smoothness. The respective measures are defined as

$$R_g(\phi) := \int_\Omega \sum_{i=1}^{n} \|\nabla\phi_i\|^2 dx, \quad R_h(\phi) := \int_\Omega \sum_{i=1}^{n} \|D^2\phi_i\|_F^2 \mathrm{d}x, \tag{28}$$

where $\|\cdot\|_F$ denotes the Frobenius norm, evaluated at each component of the deformation vector field $\phi : \Omega \to \mathbb{R}^n$. Translations lie in the kernel of the sublinear functional R_g and are not penalized. Analogously, affine transformations are not penalized by R_h.

Bidirectional approach The simplicity of obtaining both ϕ and ϕ^{-1} suggests a *bidirectional* approach: Instead of solving only

$$\min_{\theta\in\mathbb{R}^n} \mathcal{L}(\phi_\theta; I_1, I_2) \quad \text{with} \quad \phi_\theta = S(G(\theta)), \tag{29}$$

in the bidirectional setting, we minimize

$$\min_{\theta\in\mathbb{R}^n} \frac{1}{2}(\mathcal{L}(\phi_\theta; I_1, I_2) + \mathcal{L}(\phi_\theta^{-1}; I_2, I_1)),$$
$$\text{where} \quad \phi_\theta^{-1} := S(-G(\theta)). \tag{30}$$

An appealing feature of this approach is that the loss is invariant with respect to the order of I_1 and I_2, when the deformation is inverted analogously.

3.4 Network Architecture and Optimization

Network architecture The network architecture for generating the velocity field v_θ is visualized in Fig. 9. The cuboid image domain is uniformly scaled to fit into the unit cube,

$\Omega \subset [-1, 1]^3$. The network accepts coordinates as input and returns a right invariant vector field on G from the Lie algebra $\mathfrak{g}$, which is a k-dimensional vector space. The batched output forms a four-dimensional tensor with dimension $n_h \times n_w \times n_z \times k$. Following [28], the implicit neural representation is designed as a (fully connected) multi-layer perceptron with sinusoidal activation functions. We modified the proposed initialization scheme by initializing the weights of the last linear layer uniformly between $[-10^{-4}, 10^{-4}]$. This ensures that the initial deformation is close to the identity transformation.

The network consists of five layers of 512 neurons each. Additionally, residual connections were added between layers 1 to 5. The neural velocity field is evaluated on an equidistant, rectangular grid to perform its subsequent integration.

Overall, the network contains between $1.31 \cdot 10^7$ and $1.33 \cdot 10^7$ trainable weights, depending on the dimension of $\mathfrak{g}$.

Low-frequency bias Studies such as [52] show that neural networks exhibit a low-frequencies bias. To mitigate this effect, we scale the weights of the first layer with a preset scalar hyperparameter w_0 as is done in [28]:

$$h_j^1 = \sin\left(\sum_{i=1}^{3} w_0 \omega_{ji}^1 \cdot x_i + b_j^1\right). \tag{31}$$

Here, the variable x_i represents the i-th image coordinate, h_j^1 denotes the value of the j-th hidden neuron in the first layer, while ω^1 and b^1 are the weights and biases of the first layer, respectively.

The scaling increases the frequencies of the sinusoids in the first layer during initialization which in return increases the curvature of the loss with respect to the corresponding weight parameters. For $w_0 = 0$, the network output does not depend on the input coordinates, but only on the biases. This produces channel-wise constant fields, yielding a translation for the SVF and a rigid deformation for the $\mathrm{SE}(\mathbb{R}, 3)$ setting. Smaller values of w_0 generally yield smoother deformations, whereas larger values allow the reconstruction of finer deformations.

Optimization and post-scaling For the optimization, we used ADAM [53] in full-batched mode, so there was no stochastic uncertainty introduced by sampling. Nonetheless, the method is not fully deterministic, as the network weights are initialized randomly and a stochastic gradient estimation is performed for the interpolation while integrating the velocity fields. As we found that the network tends to yield relatively large initial velocity fields, we introduced a scalar hyperparameter for post-scaling the field from the output of the network. In our experiments, the factor was tuned for the specific application between 10^{-4} and 10^{-1}. In the SE(3) and SIM(3) settings, we used two different factors, one for the rotational part (angular velocities) and one for the translational part, to accommodate their inherently different range.

Hyperparameter tuning was conducted on the learning rate (lr), the post-scaling factor (sf), and w_0. Using [54], for each experiment and chosen matrix group, a tree-structured Parzen estimator was utilized and evaluated for 120 trials, each consisting of the registration of five randomly chosen volume pairs. The choice of subsequent evaluations was controlled by the estimation of the Parzen estimator. In the initial search space, w_0 was uniformly distributed over the interval $[0, 30]$, whereas the scaling factor and learning rate were both logarithmically distributed over $[10^{-5}, 10^{-1}]$. Results of the tuning process are summarized in Appendix A.

4 Numerical Results

To validate the proposed method and evaluate its performance, we conducted numerical experiments both on synthetic and real-world data.

The implementation for the stationary velocity fields is based on [10], but with differently chosen hyperparameters for the learning rate, scaling factor after integration, and frequency parameter ω_0, which yielded better results in our implementation. Hyperparameters used in the experiments are listed in Appendix A.

4.1 Datasets

Real-world data We benchmarked on the OASIS-1 dataset [7], which consists of 414 brain MRI scans of individual patients in different stages of dementia and a healthy control group. We used 5 pairs for hyperparameter tuning and the rest for evaluation. The image dimensions are $160 \times 192 \times 224$ voxels. Besides the intensity values, each dataset contains segmentations on 35 regions, conducted by experts. We used the skull-stripped, affinely pre-aligned volumes provided in the dataset.

Synthetic data To validate our method on data with a known ground truth deformation field, which is typically unavailable for real-world data, we generated an artificial dataset based on random nonlinear deformation fields.

As our method especially aims at recovering deformations with a large rigid motion, we chose $7^3 = 343$ equidistantly distributed control points in the image domain $\Omega \subset \mathbb{R}^3$ and applied a rotation around a random axis to these points. Following [55], the axis was sampled randomly from a uniform distribution over the unit sphere; the angle was drawn uniformly distributed over the interval $[-\frac{\pi}{4}, \frac{\pi}{4}]$, producing rotations up to 45 degrees. The rotation was then augmented by a random translation with its length chosen uniformly distributed over $[0, 0.1]$.

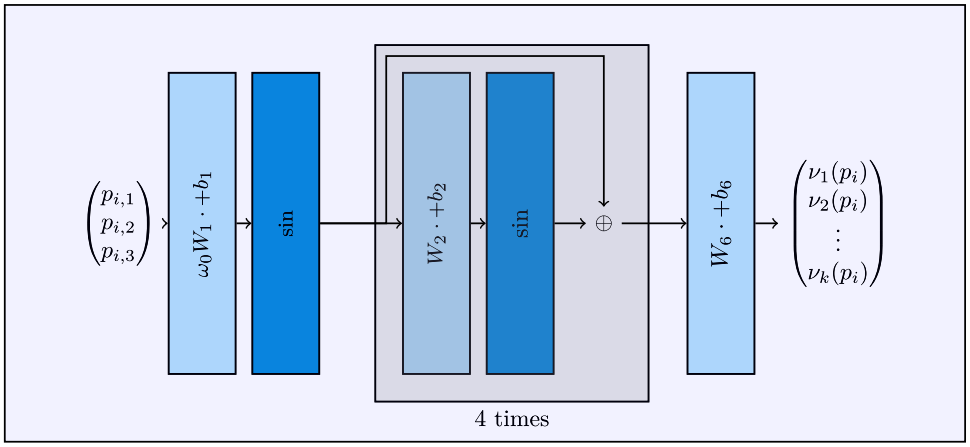


Fig. 9 Network architecture of G for generating the velocity field. The matrix-valued velocity field is described by a implicit neural representation that subsequently drives the evolution of a matrix field, generating a deformation

To this global rigid deformation, we added a non-rigid component by perturbing each control point by a random translation with length uniformly distributed over $[0, 0.05]$. The dense synthetic deformation vector field ϕ_{syn} was ultimately computed by interpolating between the perturbed control points using radial basis functions based on second-order polyharmonic splines [56].

To generate the image pairs, the deformation was applied to a zero-padded reference image from the OASIS dataset, to which Gaussian noise with mean zero and standard derivation 0.01 was added, resulting in a synthetic template image. Some exemplary deformations can be seen in Fig. 10.

4.2 Benchmark Metrics

For the evaluation of the results, we used different metrics:

- *Root-mean-square error.* For the synthetic test data with known ground truth, we computed the root-mean-square error (RMSE) between the recovered deformation field and the ground truth deformation field, measured in pixels:

$$\mathrm{RMSE}_{\Omega_M}(\phi_{\mathrm{syn}}, \phi_{\mathrm{reg}}) = \sqrt{\frac{1}{|\Omega_M|}\int_{\Omega_M} |\phi_{\mathrm{syn}} - \phi_{\mathrm{reg}}|^2} \tag{32}$$

  In this, we masked the background of the reference image by choosing the domain Ω_M such that the deformation field was compared to the ground truth only at points where image information was available.
- *Dice score.* For the real-world data, no dense ground truth was available. To still get a coarse indication of the registration quality, we followed an approach common in medical image registration when segmentation annotations are available. Given two segmented regions (i.e., subsets of Ω) S_1 and S_2 on the first and second image, we deformed S_2 using the estimated deformation ϕ and computed the distance to S_1 as the (forward) Dice score

$$\mathrm{DS}(S_1, \phi^{-1}(S_2)), \tag{33}$$

  where

$$\mathrm{DS}(A, B) = \frac{2|A \cap B|}{|A| + |B|}. \tag{34}$$

  As this metric depends on the transformation direction, we also list the backward Dice score $\mathrm{DS}(\phi(S_1), S_2)$.
- *SSIM.* As an additional metric, we computed the Structural Similarity

$$\mathrm{SSIM}(I_1, I_2 \circ \phi) \tag{35}$$

  between reference and deformed template image, as this is closer to human perception than simpler norm-based metrics [57]. Note that this metric is purely based on intensity values and that there are generally many deformations producing the same deformed image.

Lastly, we counted the percentage of sub-pixel simplices of the deformed grid at which the diffeomorphism property $\det D\phi(x) < 0$ was violated; for details, see Sect. 3.3.

4.3 Results

All benchmarks were implemented in PyTorch 2.3.0 and performed on a 24-core AMD EPYC 74F3 system with 256GB of RAM, 3x NVIDIA A100 and CUDA 12.0.

Fitting deformations We first validated the effectiveness of the proposed approach for expressing arbitrary deformations with large rigid components by reconstructing synthetic deformations generated as described in Sect. 4.1. As optimization loss, we solely used the squared RMSE. Note that despite being convex in the resultant deformation, the loss term and thereby the optimization is non-convex in the network weights θ.

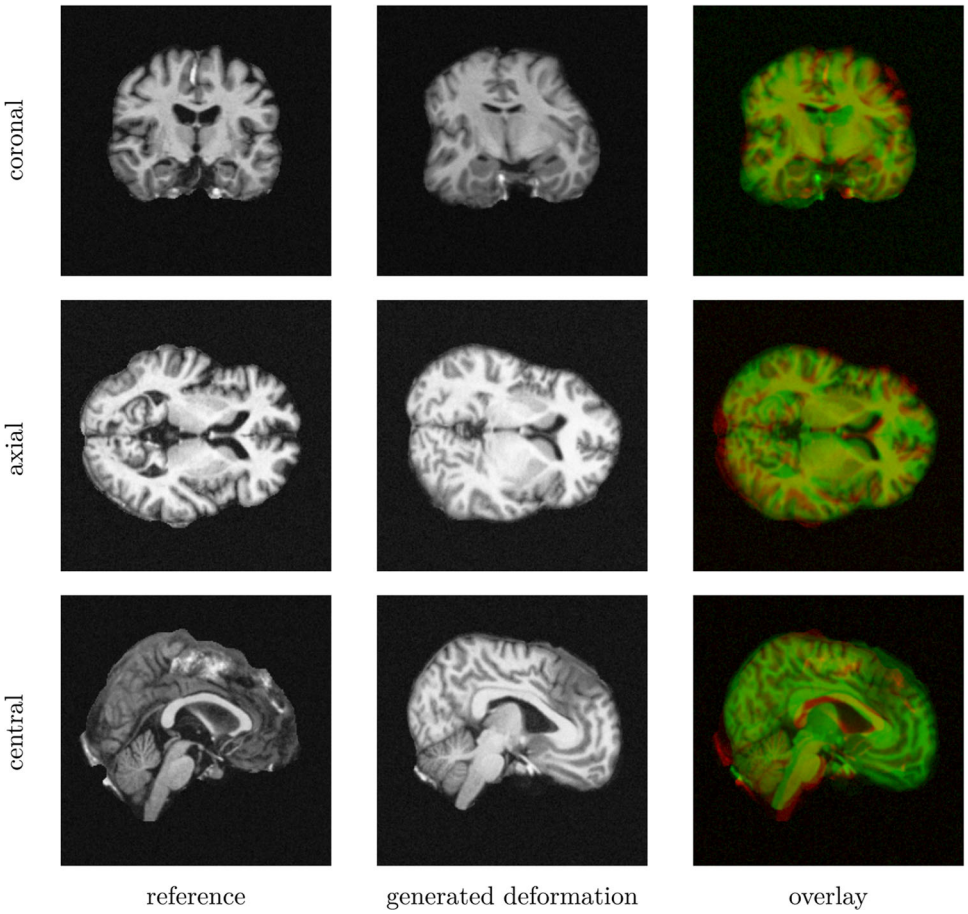
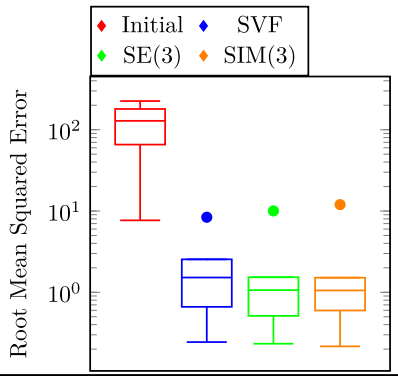


Fig. 10 Exemplary slices from the 3D OASIS benchmark dataset (left). In order to accurately evaluate the full registration process, we generated synthetic ground truth deformations and deformed the images (center). The color overlay (right) highlights the differences (reference image in red, deformed image in green). While this example exhibits only a small roto-translational component, the full synthetic dataset contains rotations up to 45 degrees, (color figure online)

The results in Fig. 11, evaluated for 409 synthetically generated deformations, show that the matrix field-based approaches both using SE(3) and SIM(3) allow to approximate the deformation field better than with a classical SVF approach. As initial error, we measured the difference between the synthetic deformation and the identity mapping. *Registering synthetic deformations* Fig. 12 shows the results for the pairwise image registration on a set of 409 image pairs with synthetic deformations. While the SVF approach greatly struggled to reconstruct the deformations here, presumably due to their large rigid component, the proposed use of matrix fields allowed us to robustly approximate the correct global deformation.

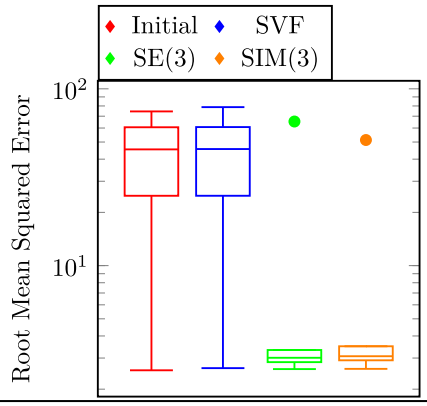
This behavior is clearly visible in Figs. 1 and 2 shown in the introduction, in which the resultant images as well as an exemplary slice of the deformation field are depicted under small and large deformations. It illustrates how the matrix group approach is able to capture even large rotational deformations, whereas the use of an SVF tends to align the intensity values by local deformations instead of a global rotation, resulting in the large RMSE observed in Fig. 12.

To further examine at which point the SVF approach breaks down, we repeated the experiment by varying the magnitude of the synthetically generated deformations and measured



method	RMSE (mean) (↓)	RMSE (median) (↓)
Initial	123.84	128.53
SVF [10]	1.73	1.52
SE(3) (ours)	<u>1.33</u>	<u>1.06</u>
SIM(3) (ours)	**1.26**	**1.05**

Fig. 11 RMSE of fitting large synthetic deformations using the classical SVF approach and the proposed approach with two different matrix fields SE(3) and SIM(3). The method is evaluated for 409 random deformations, and the RMSE is computed after 120 iterations each. The proposed parametrization aids the optimization process in finding a more precise representation of the deformation, resulting in a lower RMSE. Best scores are typeset in bold; second-best scores are underlined



method	RMSE (mean) (↓)	RMSE (median) (↓)
Initial	43.35	45.43
SVF [10]	43.47	45.64
SE(3) (ours)	6.25	**3.01**
SIM(3) (ours)	**4.87**	3.07

Fig. 12 Registration of synthetically generated deformations using the classical SVF approach and the proposed approach with two different matrix fields SE(3) and SIM(3). The RMSE in the deformation after 120 iterations is shown, computed for 409 random deformations. In this particular setting, SVF seems to fail at capturing the rotations, while the parametrization with matrix groups produces deformations close to the ground truth in most cases

the RMSE of the recovered deformations. The experiments were performed both with and without a global rotation, and the results are presented in Fig. 13.

To this end, we generated 10 rotation-free volumes by fixing the rotation angle to zero and linearly scaling the magnitude of the global translation as well as the local non-rigid deformation, such that the global translation was scaled between 0% (volume pair 1) and 10% (volume pair 10) of the domain width. We then generated 10 more volumes, in which a rotation was added, and the angle was linearly increased from 0° (volume pair 1) to 90° (volume pair 10). All 20 volumes were generated separately, that is, with different random components.

Figure 13a shows that the SVF, as well as both matrix-based methods, performed reasonably well even under large translations. In the presence of additional global rotations, however, the picture changed considerably. As can be seen in Fig. 13b, the SVF-based approach broke down completely once the rotation angle exceeded 30°, ultimately yielding deformations that hardly had a lower RMSE than the identity. The matrix-valued approaches, in contrast, returned stable results close to the ground truth even under larger rotations, and only started deteriorating at rotations with an angle greater than 70°.

We conclude that at least in these experiments, the matrix group approach achieved the goal of improving the network's ability to find deformations with large rotational components and clearly outperformed the approach solely based on SVFs.

In all experiments, we used NCC as similarity term and the Hessian regularization R_h, scaled with factor $2 \cdot 10^{-5}$.

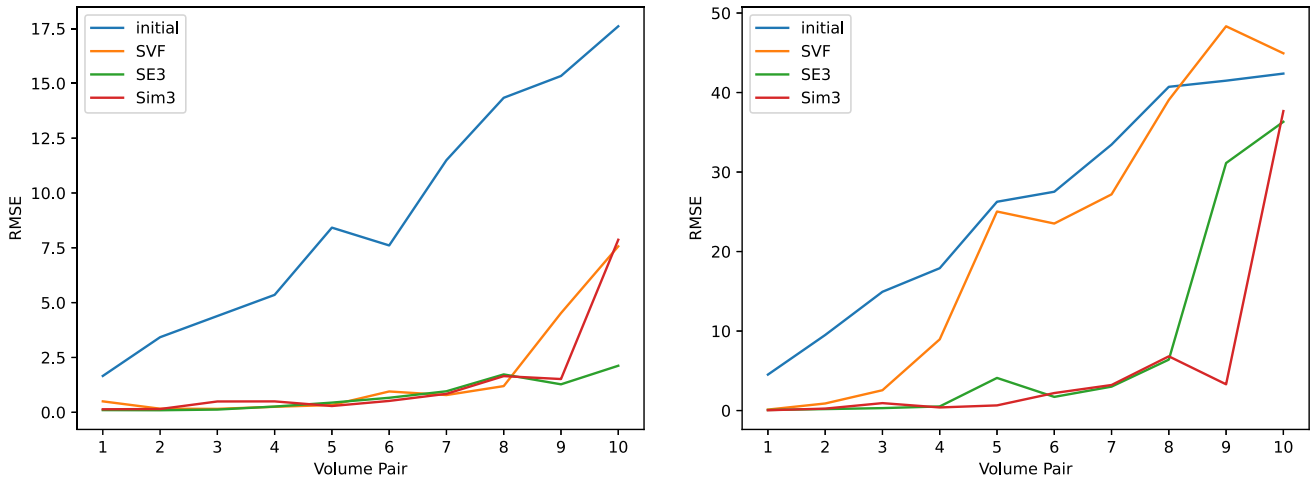
Registering real-world OASIS data We tested our approach on the task of inter-patient registration with real-world data. We used NCC as image similarity metric, gradient regularization R_g on the final displacement field, scaled with a factor of 0.1, and the folding prevention term R_ϵ with a factor of 200. We compared the classical SVF method with the matrix field-based methods for SE(3) and SIM(3), each in forward and bidirectional mode (30). Furthermore, we included the SyN method [58] with setting (100, 100, 25) for the maximal number of optimization steps in the pyramid in the comparison. The bidirectional approach improves the backward Dice score and the mean Dice score in the parametrizations with an SVF and the matrix field approaches, as expected.

For the experiments, we registered 150 randomly selected image pairs from the OASIS-1 dataset. Additionally, we benchmarked on a task with larger deformations starting from the unaligned brain scans from [7]. We used FreeSurfer [59] for skull stripping the unprocessed volumes. As there are no ground truth deformation fields available, we used the Dice score as a proxy. For the second task, we furthermore list the SSIM metric.

The results of the benchmark are presented in Tables 2 and 3. They show that, in general, the matrix-group approach performed similarly well as the classical SVF approach, both for the pre-aligned images and for the unaligned brain scans. In the unaligned case, that is, for larger deformations, both approaches also improve the performance on the Dice score compared to SyN (Table 3).

These results stand in contrast to the ones on the synthetic examples presented in Figs. 12 and 13, in which the matrix-based approaches SE(3) and SIM(3) performed significantly better than SVF. The reason for this performance difference likely lies in the nature of the underlying deformations: For the pre-aligned images, there are only marginal global deformations sought, and even in the unaligned case, the global rotational component of the deformation is relatively small. Furthermore, in the synthetic examples, we were able to compare the recovered and ground truth deformation fields directly. On the OASIS data set (and other real-world image registration data), no dense ground truth deformation is available, which is the reason we resort to comparing Dice scores and SSIM. As these metrics, however, are only based on the segments and intensity values and thereby not explicitly on the deformation itself, they ultimately provide only limited insight into the quality of the actual deformation field. Therefore, even though SVF generated good results in terms of the deformed images, it might be obscured that the deformation is far from an actually sought "truth", see Figs. 2 and 13.

Comparison to learning-based methods We further compared our method to TransMorph [60], a state-of-the-art deep learning-based image registration technique. The Dice scores



(a) without global rotation (b) with global rotation

Fig. 13 Registration results of OASIS image pairs with synthetically generated nonlinear deformations of increasing magnitude. The RMSE between the recovered deformation field and the ground truth is shown. In the left figure, only a global translation and local perturbances are performed. The global translation increases from 0–10% of the domain width from left to right. In the right figure, a global rotational component is further added whose angle increases from $0 - 90°$. Without a global rotation, all methods work reasonably well even for larger translations. In the presence of global rotations, however, the classical SVF breaks down already at smaller angles, whereas even large rotations up to 70° only have a small impact on the RMSE when using the proposed parametrization with matrix-valued fields (SE(3), SIM(3))

Table 2 Inter-patient 3D registration of pre-aligned and skull-stripped volumes of the OASIS-1 dataset

method	Dice score ± std. dev. (↑)	Dice score backwards (↑)	mean Dice score (↑)	$\det(J_\phi) \leq 0$ (↓)
initial	0.5838 ± 0.0610	0.5838 ± 0.0610	0.5838 ± 0.0610	–
SVF [10]	0.8153 ± 0.0231	0.8002 ± 0.0233	0.8077 ± 0.0223	$2.35 \cdot 10^{-6}$
bidir. SVF	0.8149 ± 0.0235	0.8160 ± 0.0210	0.8154 ± 0.02197	$3.58 \cdot 10^{-6}$
SE(3) [ours]	**0.8159 ± 0.0231**	0.8001 ± 0.0233	0.8080 ± 0.0229	$1.85 \cdot 10^{-6}$
bidir. SE(3) [ours]	0.8148 ± 0.0238	0.8159 ± 0.0216	0.8153 ± 0.0224	$2.90 \cdot 10^{-6}$
SIM(3) [ours]	0.8157 ± 0.0233	0.8009 ± 0.0233	0.8083 ± 0.0231	$1.95 \cdot 10^{-6}$
bidir. SIM(3) [ours]	0.8151 ± 0.0236	**0.8162 ± 0.0212**	**0.8156 ± 0.0221**	$3.20 \cdot 10^{-6}$
SyN [58]	0.8062 ± 0.0248	0.8069 ± 0.0235	0.8066 ± 0.0241	**0**

The best value is typeset in bold, and the second best is underlined
The classical (SVF) and the proposed matrix-valued approaches (SE(3) and SIM(3)) all yield similar Dice scores after registration. A bidirectional approach improves the backward Dice score and the mean of both scores

Table 3 Unidirectional registration of unaligned OASIS brain scans

method	Dice score, forward (↑)	SSIM, forward (↑)	fraction with $\det(J_\phi) \leq 0$ (↓)
initial	0.1178 ± 0.1001	0.7987 ± 0.011	–
SVF [10]	**0.7278 ± 0.0628**	0.9444 ± 0.0170	$\mathbf{4.04 \cdot 10^{-7}}$
SE(3)	0.7223 ± 0.1012	0.9479 ± 0.0158	$7.65 \cdot 10^{-7}$
SIM(3)	0.6199 ± 0.2416	0.9367 ± 0.0187	$4.75 \cdot 10^{-7}$
SyN [58]	0.4987 ± 0.3081	**0.9572 ± 0.0266**	$4.10 \cdot 10^{-6}$

The best value is typeset in bold, and the second best is underlined.
SVF and SE(3) work comparably well in terms of Dice score, beating the SyN method by a wide margin. Note that Dice score and SSIM provide only limited insight into the quality of the actual deformation field, compare Figs. 2 and 13

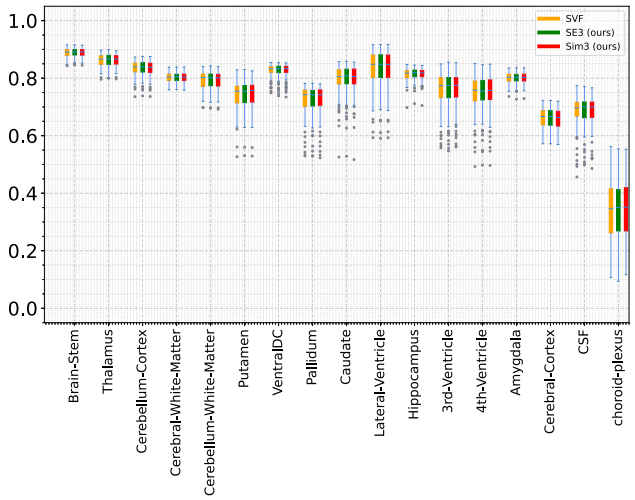


Fig. 14 Dice scores for different brain structures evaluated for the atlas-based registration task on the test fold of the IXI dataset [60] with one atlas and 114 reference images. Viewed over the entire dataset, all three parametrizations produce similar results. The code to create the plots had been taken from the TransMorph repository [60]

presented in Fig. 14 and Table 4 show that our method is on par with TransMorph, which is encouraging considering that TransMorph was trained specifically on this dataset and atlas. We evaluated all variants on the task of atlas-based image registration on the IXI dataset consisting of MR brain scans from the original publication [60]. Here, we used the same hyperparameters for the SVF case as well as our methods that we previously tuned on the OASIS Dataset 6 for the SVF architecture. The loss was composed of a data term based on windowed NCC and gradient regularization R_g on the deformation field, weighted with factor 1. This is the loss that had been used to train TransMorph—in an unsupervised way—in the original paper [60].

In comparison with learning-based methods such as TransMorph, a potential advantage of optimization-based methods like ours is that they tend to be comparably robust with varying image data and loss functions. To this extent, we repeated the experiments on the OASIS-1 dataset, now using the energy functional from TransMorph [60] mentioned above. While the results obtained in this way are slightly worse in terms of Dice score (Table 5) than with the previous loss term (Table 3), they indicate the robustness of our optimization-based approach with respect to varying data and loss functions.

5 Conclusion

In this paper, we proposed, discussed and analyzed a novel approach to deformable image registration which extends the concept of integrating stationary velocity fields to matrix groups. We proved the unique existence of a solution to the extended flow equation in Thm. 1 and derived a scaling-and-squaring algorithm to efficiently approximate this solution, see Alg. 2.

Fairly evaluating nonlinear image registration methods on real data is notoriously hard, as there is typically no ground truth for the deformation field available. As we observed, popular methods such as SVF can perform well in terms of intensity value-based metrics as well as proxy metrics such as the Dice score on segmentations, while the generated deformation field might be far from the ground truth or a desired deformation.

Our experiments indicate that moving the velocity representation and flow equation from a purely translational approach (as in SVF) to the matrix group setting can ameliorate this issue. This is especially the case when the "true" deformation field comprises larger affine deformations such as a global rotation.

The matrix-based formulation allows the specification of the velocity field in a chosen matrix group which naturally covers a wider range of deformations. In our approach, we parametrize the matrix group by implicit neural representations, allowing for optimization techniques from machine learning. In future work, it will be interesting to adapt this idea to other network architectures, such as UNet- or Transformer-based approaches.

The formulation with a matrix group gives a fairly general method to choose between different parametrization schemes. Further paths to explore include using the affine group Aff(3) as parametrization for the velocity field. The lack of a closed form for the matrix exponential and logarithm in this setting, however, would require a different interpolation strategy or the use of iterative methods, making the optimization more difficult and time-consuming. Other suitable matrix groups might also be considered.

Another interesting direction of research concerns the extension of matrix-valued fields to approaches with a non-stationary flow such as the LDDMM framework. However, this would require a different numeric integration method than scaling-and-squaring, which was central to our approach.

Overall, we hope that the proposed matrix group approach provides a building block for creating robust registration methods that can cope with large deformations not only in terms of image similarity, but also in terms of the quality of the deformation fields.

6 Proofs

6.1 Proof of Theorem 1

Proof Embedding the manifold of a matrix group in a surrounding vector field space (here $\mathbb{R}^{4\times4}$ with the operator

Table 4 Comparison of our method to the learning-based TransMorph [60] scheme on the tasks of atlas-to-patient registration on the IXI dataset

method	Dice score ± std. dev. (↑)	$\det(J_\phi) \leq 0$ (↓)
SVF	0.7589 ± 0.1292	$1.51 \cdot 10^{-6}$
SE(3) [ours]	$\underline{0.7591 \pm 0.1287}$	$1.25 \cdot 10^{-6}$
SIM(3) [ours]	0.7589 ± 0.1290	$\underline{1.15 \cdot 10^{-6}}$
TransMorph	0.753 ± 0.123	$1.579 \cdot 10^{-2}$
TransMorph-Bayes	0.754 ± 0.124	$1.560 \cdot 10^{-2}$
TransMorph-bspl	$\mathbf{0.761 \pm 0.122}$	$< \mathbf{10^{-6}}$

The best value is typeset in bold, and the second best is underlined.
The transformer architecture used in the comparison was specifically trained on the IXI dataset. Our methods do not require training data but are slower in the "inference step." All methods yield comparable results

Table 5 Inter-patient 3D registration on pre-aligned and skull-stripped volumes of the OASIS-1 dataset

method	Dice score ± std. dev. (↑)	Dice score backwards ± std. dev.(↑)	mean Dice score (↑)	$\det(J_\phi) \leq 0$ (↓)
initial	0.5838 ± 0.0610	0.5838 ± 0.0610	0.5838	–
SVF bidir	0.8037 ± 0.0270	$\underline{0.8038 \pm 0.0264}$	$\underline{0.8038}$	$\underline{4.50 \cdot 10^{-8}}$
SVF [10]	$\underline{0.8057 \pm 0.0275}$	0.7968 ± 0.0282	0.8013	$6.92 \cdot 10^{-6}$
SE(3) bidir [ours]	0.8049 ± 0.0268	$\mathbf{0.8050 \pm 0.0264}$	**0.8050**	$7.038 \cdot 10^{-8}$
SE(3) [ours]	$\mathbf{0.8071 \pm 0.0276}$	0.7980 ± 0.0281	0.8026	$1.00 \cdot 10^{-5}$
SIM(3) bidir [ours]	0.8000 ± 0.0276	0.8001 ± 0.0275	0.8001	$\mathbf{6.34 \cdot 10^{-9}}$
SIM(3) [ours]	0.8013 ± 0.0286	0.7930 ± 0.0292	0.7972	$7.32 \cdot 10^{-7}$

The best value is typeset in bold, and the second best is underlined.
In contrast to the results in Table 2, here we have used the generic loss function from TransMorph [60] with windowed NCC and regularization weight $\lambda = 1$, and without orientation penalty. Although this loss had originally been proposed for the IXI dataset, with our method, the results on the OASIS-1 dataset are only slightly worse than when using the specifically tuned loss (Table 2). Bidirectional registration eliminates the difference in forward and backward Dice scores and preserves the orientation better, as it penalizes high gradients also on the inverse displacement vector field. In this setting, the parameterization with the SE(3) matrix group has yielded the best results, slightly improving the classical setting with stationary velocity fields

norm) allows the use of classical results of ODE theory in this setting. For the matrix subgroup $G \subset GL(\mathbb{R}, 4)$, the space of continuous matrix fields $C(\Omega, G)$ can be embedded in the unitary function Banach algebra

$$\mathcal{B} := C(\Omega, \mathbb{R}^{4\times 4}) \tag{36}$$

with the corresponding pointwise supremum norm on the compact domain $\Omega \subset \mathbb{R}^k$,

$$\|B\|_{\mathcal{B}} := \sup_{x\in\Omega} |B(x)|_F \quad \text{for } B \in \mathcal{B}, \tag{37}$$

and pointwise matrix multiplication.
Crucially, this norm is sub-multiplicative with regard to the pointwise multiplication of matrix fields. Consider for now M as a univariate function of time $M : [0, 1] \to \mathcal{B}$. The time-invariant case of Equation (5) can then be written compactly as

$$\frac{\partial M}{\partial t} = f(M) \tag{38}$$

$$M(0) = id \tag{39}$$

with

$$f : \mathcal{B} \to \mathcal{B}, \tag{40}$$

$$f(B) := \left(x \mapsto i\,(v(PB(x)\bar{x}))_{B(x)}\right), \tag{41}$$

for the canonical embedding i which extends the right invariant vector fields on G to the vector fields on $\mathbb{R}^{4\times 4}$, and the projection $P : \mathbb{R}^4 \to \mathbb{R}^3$ which removes the last component. For the sake of readability, we will omit this embedding in the following.
As $v : \mathbb{R}^3 \to \mathfrak{g}$ maps each point to a right-invariant vector field on the manifold G, this vector field is uniquely described by its value at the identity of the matrix group:

$$v(PB(x)\bar{x})_{B(x)} = v_{id}(PB(x)\bar{x})B(x). \tag{42}$$

By sub-multiplicativity of the matrix norm and the compactness of Ω (by which v_{id}) is bounded on Ω), it follows that

$$\begin{aligned}\|f(B)\|_{\mathcal{B}} &= \sup_x |v_{id}(PB(x)\bar{x})B(x)| \\ &\leq \sup_x |v_{id}(PB(x)\bar{x})| \sup_x |B(x)| = C_v\|B\|_{\mathcal{B}},\end{aligned} \tag{43}$$

yielding that f is linearly bounded. By Gronwall's inequality [61] it holds

$$\|M(t)\|_{\mathcal{B}} \leq \|M(0)\|_{\mathcal{B}} e^{C_v t} = \|id\|_{\mathcal{B}} e^{C_v t}, \tag{44}$$

hence, every solution $M(t)$ is bounded for every finite time interval.
Next, we show that f is locally Lipschitz continuous. For $S, T \in \mathcal{B}$, it holds

$$\|f(S) - f(T)\|_{\mathcal{B}} = \sup_{x\in\Omega} |v(PS(x)\bar{x})_{S(x)} - v(PT(x)\bar{x})_{T(x)}|_F. \tag{45}$$

Using (42) and adding zero, we obtain

$$\begin{aligned} &= \sup_{x\in\Omega} |\left(v_{id}(PS(x)x) - v_{id}(PT(x)\bar{x})\right) S(x) \\ &- v_{id}(PT(x)\bar{x})(T(x) - S(x))|_F. \end{aligned} \tag{46}$$

By the triangle inequality and the boundedness of v_{id}, we can estimate

$$\begin{aligned} &\leq \sup_{x\in\Omega} |\left(v_{id}(PS(x)\bar{x}) - v_{id}(PT(x)\bar{x})\right) S(x)|_F \\ &+ \sup_{x\in\Omega} C_v |T(x) - S(x)|_F. \end{aligned} \tag{47}$$

Using sub-multiplicativity,

$$\begin{aligned} &\leq \sup_{x\in\Omega} |v_{id}(PS(x)\bar{x}) - v_{id}(PT(x)\bar{x})|_F \sup_{x'\in\Omega} |S(x')|_F \\ &+ C_v \|T - S\|_{\mathcal{B}}, \end{aligned} \tag{48}$$

and then Lipschitz continuity of v_{id}, we arrive at

$$\leq L_v \sup_{x\in\Omega} |PS(x)\bar{x} - PT(x)\bar{x}| \, \|S\|_{\mathcal{B}} + C_v \|T - S\|_{\mathcal{B}}. \tag{49}$$

Finally, using sub-multiplicativity, we conclude

$$\|f(S) - f(T)\|_{\mathcal{B}} \leq L_v \|S - T\|_{\mathcal{B}} \sup_{x\in\Omega} |\bar{x}| \, \|S\|_{\mathcal{B}} + C_v \|T - S\|_{\mathcal{B}} \tag{50}$$

$$= (L_v \sup_{x\in\Omega} |\bar{x}| \, \|S\|_{\mathcal{B}} + C_v) \|T - S\|_{\mathcal{B}}. \tag{51}$$

Hence, as $M(t)$ is bounded for a bounded time interval, f is Lipschitz on some open neighborhood of the image of M over this time. By Picard's theorem for Banach space-valued functions [62], there exists a unique solution $M : [0, 1] \to \mathcal{B}$ of Equation (5). In fact, as the vector field $v(x$ is tangential on G everywhere, it holds, that $M : [0, 1] \to C(\Omega, G)$. By (44), this solution remains finite for finite time, in particular also on the unit interval $[0, 1]$, which is important for the extended flow equation. □

6.2 Proof of Theorem 2

Proof The proof consists of four steps. Throughout the proof, we will associate $M(t)(x)$ with $M(x, t)$:

1. We first show that $M \in \mathcal{C}^2([0, 1], \mathcal{B})$.
2. We then prove that a semi-discrete forward Euler scheme in time, formulated in the embedding space $\mathcal{B}$, converges to the solution of the flow equation (5).
3. We afterward construct an exponential discretization scheme on the manifold and show convergence toward the Euler scheme. This discretization satisfies the decomposition condition by construction. This scheme also builds the foundation of the scaling-and-squaring approach of Alg. 2.
4. Finally, we prove that the decomposition condition is preserved under convergence and therefore transfers to the fully continuous solution of the flow equation.

Calculating M'' We aim at calculating the second derivative M'' of the solution of the extended flow equation with respect to the time. To this end, we first calculate the derivative of

$$f_{id} : \mathcal{B} \to \mathcal{B}, \tag{52}$$

$$B \mapsto v_{id}(PB(\cdot)\bar{\cdot}) \in \mathcal{B}. \tag{53}$$

We denote the normed function space of continuous vector-valued functions as $C(\Omega, \mathbb{R}^3)$ with the corresponding supremum norm and apply the decomposition $f_{id} = f_1 \circ f_2$ with $f_2 : \mathcal{B} \to C(\Omega, \mathbb{R}^3), B \mapsto PB(\cdot)\bar{\cdot}$ and $f_1 : C(\Omega, \mathbb{R}^3) \to \mathcal{B}$, $V \mapsto v_{id}(V(\cdot))$. We show that the Fréchet derivative of f_1 at V is the left-multiplication

$$Df_1(V)(h) = (Dv_{id})(V(\cdot))h(\cdot). \tag{54}$$

This follows from

$$\sup_{x\in\Omega} |v_{id}\left((V + h)(x)\right) - v_{id}(V(x)) - (Dv_{id})(V(x))h(x)|_F \tag{55}$$

$$\begin{aligned} &= \sup_{x\in\Omega} |v_{id}\left(V(x)\right) + (Dv_{id})(V(x))h(x) \\ &+ o(|h(x)|) - v_{id}(V(x)) - (Dv_{id})(V(x))h(x)|_F \end{aligned} \tag{56}$$

$$= \sup_{x\in\Omega} |o(|h(x)|)|. \tag{57}$$

As $v_{id} \in C^1(\Omega, \mathbb{R}^{4\times4})$, it has a uniformly continuous derivative on the compact domain Ω and is therefore uniformly differentiable on Ω. By definition, this means that [63]

$$\forall \epsilon > 0 \, \exists \delta > 0 : \, \forall x \in \Omega \, \exists L_x \in L(\mathbb{R}^3, \mathbb{R}^{4\times4}) :$$

$$\forall h : 0 < |h| \leq \delta : \frac{|v_{id}(x+h) - v_{id}(x) - L_x h|}{|h|} \leq \epsilon. \tag{58}$$

Hence, the convergence of the o-term in Eq. (57) holds uniformly for all $x \in \Omega$, and we can conclude

$$\|v_{id}((V+h)(\cdot)) - v_{id}(V(\cdot)) - (Dv_{id})(V(\cdot))h(\cdot)\|_{\mathcal{B}} = o(\|h\|_{\mathcal{B}}), \tag{59}$$

which shows the claim in (54).
The mapping f_2 is linear and therefore coincides with its Fréchet derivative:

$$(Df_2(B))(h) = f_2(h) = Ph(\cdot)^{\cdot}. \tag{60}$$

Collecting the results, one can conclude from the chain rule for the Fréchet derivative:

$$\begin{aligned}(Df_{id}(B))(h) &= Df_1(f_2(B))(Df_2(B)(h)) &(61)\\ &= Dv_{id}(PB(\cdot)^{\cdot})Ph(\cdot)^{\cdot}. &(62)\end{aligned}$$

We return to calculating M'' using $M'(t) = f(M(t))$, see (38):

$$M''(t) = (f \circ M)'(t) = Df(M(t)) \circ M'(t). \tag{63}$$

We can further expand on this by using $M'(t) = f(M(t))$ again:

$$\ldots = Df(M(t)) \circ f(M(t)). \tag{64}$$

Equation (42) and inserting the definition of f yield for the multiplication with the identity operator $e : \mathcal{B} \to \mathcal{B}, e(h) = h$ that

$$\ldots = D(f_{id}e)(M(t)) \circ f(M(t)). \tag{65}$$

Using the product rule for bilinear pointwise function multiplication in $\mathcal{B}$ for the left term and defining the left- and right-multiplication operator as

$$\begin{aligned}L : \mathcal{B} \to \mathcal{L}(\mathcal{B}, \mathcal{B}),\ (L(g))(h) &= g \cdot h &(66)\\ R : \mathcal{B} \to \mathcal{L}(\mathcal{B}, \mathcal{B}),\ (R(g))(h) &= h \cdot g, &(67)\end{aligned}$$

we get

$$\ldots = (L(f_{id}(M(t))) + R(M(t))(Df_{id}(M(t)))) \circ f(M(t)). \tag{68}$$

Substituting the chain rule calculated in (62), we obtain

$$\ldots = L(f_{id}(M(t)))f(M(t)) + R(M(t))(Dv_{id}(PM(\cdot)^{\cdot})Pf(M(t))(\cdot)^{\cdot}) \tag{69}$$

which, as a combination of continuous functions, is continuous and, accordingly, ensures $M \in \mathcal{C}^2([0,1], \mathcal{B})$. Additionally, as $[0,1]$ is compact, the range $M''([0,1])$ is compact and therefore bounded.
Euler discretization For some fixed end time $T > 0$, consider an equidistant time discretization $\{t_0 = 0 < t_1 < \ldots < t_n = T\}$ into n intervals of length $\delta_t := \frac{T}{n}$ and the associated forward-Euler discretization of the embedded flow equation (38)

$$\begin{aligned}M_{\text{eu}}(t_{k+1}) &:= M_{\text{eu}}(t_k) + f(M_{\text{eu}}(t_k))(t_{k+1} - t_k)\\ M_{\text{eu}}(t_0) &:= id.\end{aligned} \tag{70}$$

Note that due to the embedding, each $M_{\text{eu}}(t_k)$ maps from Ω into $\mathbb{R}^{4\times 4}$.
Our goal is to show convergence of $M_{\text{eu}}(t_n)$ to $M(t_n)$, following the proof for the scalar-valued case in [64, Sect. 2.2, Thm. 2.4]. This requires an estimate of the truncation error. For this, we first use the fundamental theorem of calculus for Bochner integrals [65] twice and rearrange:

$$\begin{aligned}M(t_{k+1}) &= M(t_k) + \int_{t_k}^{t_{k+1}} M'(s)\mathrm{d}s &(71)\\ &= M(t_k) + \int_{t_k}^{t_{k+1}} \left(M'(t_k) + \int_{t_k}^{s} M''(t)\mathrm{d}t\right) ds &(72)\\ &= M(t_k) + (t_{k+1} - t_k)M'(t_k) + \int_{t_k}^{t_{k+1}} \int_{t_k}^{s} M''(t)\,\mathrm{d}t\,\mathrm{d}s. &(73)\end{aligned}$$

This is equivalent to

$$M(t_{k+1}) - M(t_k) - (t_{k+1} - t_k)M'(t_k) = \int_{t_k}^{t_{k+1}} \left(\int_{t_k}^{s} M''(t)\mathrm{d}t\right) \mathrm{d}s. \tag{74}$$

Taking the norm on both sides and using the Bochner inequality [65] (recall that $\|\cdot\|_{\mathcal{B}}$ is the supremum norm as defined in (37)), we obtain

$$\begin{aligned}&\|M(t_{k+1}) - M(t_k) - (t_{k+1} - t_k)M'(t_k)\|_{\mathcal{B}}\\ &\leq \int_{t_k}^{t_{k+1}} \int_{t_k}^{s} \|M''(t)\|_{\mathcal{B}}\,\mathrm{d}t\,\mathrm{d}s &(75)\\ &\leq \frac{(t_{k+1} - t_k)^2}{2} \max_{t\in[t_k, t_{k+1}]} \|M''(t)\|_{\mathcal{B}}. &(76)\end{aligned}$$

Noting that $M'(t_k) = f(M(t_k))$, this provides a bound for the one-step truncation error of the forward Euler method depending on the norm of the second derivatives M''.
This extends the first step of the convergence proof of the forward Euler method for the scalar-valued case in [64, Sect. 2.2, Thm. 2.4] to the Banach space-valued case; the remainder of the proof is identical if one replaces absolute values with

$\|\cdot\|_{\mathcal{B}}$. This ensures

$$\|M(t) - M_{\text{eu}}(t)\|_{\mathcal{B}} \leq \frac{e^{tL_v} - 1}{L_v} \frac{\delta_t}{2} \max_{s\in[0,t]} \|M''(s)\|_{\mathcal{B}}, \tag{77}$$

where L'_v is the Lipschitz constant of f. Lipschitz continuity of f on some open neighborhood of the image of M was shown in Eq. (51) during the proof of Thm. 1. It requires Lipschitz continuity of v_{id}, which is ensured here by the C^1 assumption on v_{id} and compactness of Ω.
Overall, (77) proves convergence of the (final-time) solution of the forward Euler approximation $M_{\text{eu}}(T)$ to the solution $M(T)$ of the time-continuous flow equation, i.e.,

$$M_{\text{eu}}(T) \to M(T) \quad \text{in } O(\delta_t). \tag{78}$$

Exponential discretization For the second step, in order to motivate the construction of the exponential discretization scheme, consider the matrix flow equation (15) with spatially constant velocity $\mu \in \mathfrak{g}$, starting at $g_0 \in G$:

$$\frac{\partial M}{\partial t}(x,t) = \mu_{M(x,t)} \tag{79}$$

$$M(\cdot, 0) = g_0. \tag{80}$$

Due to the right invariance of μ, the solution takes the analytic form (compare (11)):

$$M(x,t) = \exp(t\mu_{id})g_0, \tag{81}$$

where exp denotes the matrix exponential. Applying this idea for each time interval $[t_k, t_{k+1}]$ and to each point x separately, and setting $\mu := v(PM(t_k)\bar{x})$ and $g_0 := M(x,t_k)$, we obtain the time-discrete exponential scheme

$$\begin{aligned} M_{\text{ex}}(t_{k+1}) &:= \exp((t_{k+1} - t_k)v_{id}(PM_{\text{ex}}(t_k)\bar{\cdot}))\, M_{\text{ex}}(t_k) \\ M_{\text{ex}}(t_0) &:= id. \end{aligned} \tag{82}$$

Fixing $x \in \Omega$ and rewriting the exponential scheme evaluated at the endpoint T as a product of matrices results in

$$M_{\text{ex}}(x,T) = \coprod_{k=0}^{n-1} \exp\left(\delta_t v_{id}\left(PM_{\text{ex}}\left(x,t_k\right)\bar{x}\right)\right), \tag{83}$$

where $\coprod$ denotes the matrix product evaluated from right to left, i.e.,

$$\coprod_{k=0}^{n-1} A_k := A_{n-1}A_{n-2}\cdots A_1 A_0. \tag{84}$$

We will later need a bound on the norm of M_{ex}. We use the fact that $|\exp(M)| \leq e^{|M|}$ to derive

$$|M_{\text{ex}}(x,t_k)| \leq \prod_{j=0}^{k-1} |\exp\left(\delta_t v_{id}\left(PM_{\text{ex}}\left(x,t_j\right)\bar{x}\right)\right)| \tag{85}$$

$$\leq \prod_{j=0}^{k-1} e^{|\delta_t v_{id}(PM_{\text{ex}}(x,t_j)\bar{x})|} \tag{86}$$

$$= e^{\sum_{j=0}^{k-1}|\delta_t v_{id}(PM_{\text{ex}}(x,t_j)\bar{x})|}. \tag{87}$$

As $v_{id} \in C^1(\Omega, \mathbb{R}^{4\times 4})$ is bounded on the compact set Ω by some $C_v > 0$, we continue

$$|M_{\text{ex}}(x,t_k)| \leq e^{\sum_{j=0}^{k-1}(\delta_t C_v)} \tag{88}$$

$$= e^{t_k C_v} \tag{89}$$

$$\leq e^{TC_v} =: C'_{v,T}. \tag{90}$$

Importantly, this bound depends neither on the time step δ_t nor on the spatial position x.
In order to bound the difference between the forward Euler iteration in the previous section and the exponential approach defined by (82), we define the error

$$e_k(x) := M_{\text{eu}}(x,t_k) - M_{\text{ex}}(x,t_k). \tag{91}$$

The proof for bounding the error closely follows the strategy for proving convergence of the classical forward Euler method [64]. By definition of M_{eu} and M_{ex},

$$\begin{aligned} |e_{k+1}(x)| = |&M_{\text{eu}}(x,t_k) + \delta_t v(PM_{\text{eu}}(x,t_k)\bar{x})_{M_{\text{eu}}(x,t)} \\ &- \exp\left(\delta_t v_{id}\left(PM_{\text{ex}}\left(x,t_k\right)\bar{x}\right)\right) M_{\text{ex}}(x,t_k)|. \end{aligned} \tag{92}$$

We rewrite the matrix exponential by its Taylor series:

$$\begin{aligned} |e_{k+1}(x)| \leq \Bigg| &M_{\text{eu}}(x,t_k) + \delta_t v_{id}(PM_{\text{eu}}(x,t_k)\bar{x})\, M_{\text{eu}}(x,t_k) \\ &- \sum_{l=0}^{\infty} \frac{(\delta_t)^l v_{id}\left(PM_{\text{ex}}\left(x,t_k\right)\bar{x}\right)^l}{l!} M_{\text{ex}}(x,t_k)\Bigg|. \end{aligned} \tag{93}$$

Reordering the terms and using the triangle inequality, we obtain

$$\begin{aligned} |e_{k+1}(x)| &= |M_{\text{eu}}(x,t_k) - M_{\text{ex}}(x,t_k)| \\ &+ \delta_t |v_{id}(PM_{\text{eu}}(x,t_k)\bar{x})M_{\text{eu}}(x,t_k) \\ &- v_{id}(PM_{\text{ex}}(x,t_k)\bar{x})M_{\text{ex}}(x,t_k)| \\ &+ \left|\sum_{l=2}^{\infty} \frac{(\delta_t)^l v_{id}\left(PM_{\text{ex}}\left(x,t_k\right)\bar{x}\right)^l}{l!} M_{\text{ex}}(x,t_k)\right| \end{aligned} \tag{94}$$

$$
\begin{aligned}
&= |e_k(x)| \\
&\quad + \delta_t |f(M_{\text{eu}}(t_k))(x) - f(M_{\text{ex}}(t_k))(x)| \\
&\quad + \left| \sum_{l=2}^{\infty} \frac{(\delta_t)^l \nu_{id} \left(P M_{\text{ex}}\left(x, t_k\right) \bar{x}\right)^l}{l!} M_{\text{ex}}(x, t_k) \right|. \qquad (95)
\end{aligned}
$$

We rewrite the second term in the sum using the Lipschitz estimate (51):

$$
\begin{aligned}
&\delta_t |f(M_{\text{ex}}(t_k))(x) - f(M_{\text{eu}}(t_k))(x)| \qquad (96) \\
&\leq \delta_t (L_\nu (\sup_{x \in \Omega} |\bar{x}| \|M_{\text{ex}}(t_k)\|_{\mathcal{B}} + C_\nu) \|M_{\text{eu}}(t_k) - M_{\text{ex}}(t_k)\|_{\mathcal{B}} \qquad (97) \\
&\overset{(90)}{\leq} \delta_t (L_\nu (\sup_{x \in \Omega} |\bar{x}| C'_{\nu,T} + C_\nu) \|e_k\|_{\mathcal{B}} \qquad (98) \\
&= \delta_t C''_{\nu,T} \|e_k\|_{\mathcal{B}} \qquad (99)
\end{aligned}
$$

for some constant $C''_{\nu,T}$.
We continue bounding the third term in (95):

$$
\begin{aligned}
&\left| \sum_{l=2}^{\infty} \frac{(\delta_t)^l \nu_{id} \left(P M_{\text{ex}}\left(x, t_k\right) \bar{x}\right)^l}{l!} M_{\text{ex}}(x, t_k) \right| \qquad (100) \\
&= (\delta_t)^2 \left| \nu_{id} \left(P M_{\text{ex}}\left(x, t_k\right) \bar{x}\right)^2 \sum_{l=0}^{\infty} \frac{(\delta_t)^l \nu_{id} \left(P M_{\text{ex}}\left(x, t_k\right) \bar{x}\right)^l}{(l+2)!} M_{\text{ex}}(x, t_k) \right| \qquad (101) \\
&\leq (\delta_t)^2 \left| \nu_{id} \left(P M_{\text{ex}}\left(x, t_k\right) \bar{x}\right) \right|^2 \sum_{l=0}^{\infty} \frac{\left| (\delta_t)^l \nu_{id} \left(P M_{\text{ex}}\left(x, t_k\right) \bar{x}\right)^l \right|}{l!} |M_{\text{ex}}(x, t_k)| \qquad (102) \\
&\overset{(90)}{\leq} (\delta_t)^2 (C_\nu)^2 e^{\delta_t C_\nu} C'_{\nu,T} \qquad (103) \\
&=: (\delta_t)^2 C'''_{\nu,T}. \qquad (104)
\end{aligned}
$$

Inserting both bounds into (95) leads to

$$
|e_{k+1}(x)| \leq |e_k(x)| + \delta_t C''_{\nu,T} \|e_k\|_{\mathcal{B}} + (\delta_t)^2 C'''_{\nu,T}. \qquad (105)
$$

Taking the supremum on both sides yields:

$$
\|e_{k+1}\|_{\mathcal{B}} \leq (1 + \delta_t C''_{\nu,T}) \|e_k\|_{\mathcal{B}} + (\delta_t)^2 C'''_{\nu,T}. \qquad (106)
$$

Iterating this estimate leads to an estimate for the final error e_n:

$$
\begin{aligned}
\|e_n\|_{\mathcal{B}} &\leq (1 + \delta_t C''_{\nu,T})^n \|e_0\|_{\mathcal{B}} \\
&\quad + (1 + (1 + \delta_t C''_{\nu,T}) + (1 + \delta_t C''_{\nu,T})^2 \\
&\quad + \ldots + (1 + \delta_t C''_{\nu,T})^{n-1}) (\delta_t)^2 C'''_{\nu,T}. \qquad (107)
\end{aligned}
$$

By the *finite* geometric series,

$$
\|e_n\|_{\mathcal{B}} \leq (1 + \delta_t C''_{\nu,T})^n \|e_0\|_{\mathcal{B}} - \frac{1 - (1 + \delta_t C''_{\nu,T})^n}{\delta_t C''_{\nu,T}} (\delta_t)^2 C'''_{\nu,T} \qquad (108)
$$

$$
\begin{aligned}
&= (1 + \delta_t C''_{\nu,T})^n \|e_0\|_{\mathcal{B}} + \frac{(1 + \delta_t C''_{\nu,T})^n - 1}{C''_{\nu,T}} \delta_t C'''_{\nu,T} \qquad (109) \\
&\leq (1 + \delta_t C''_{\nu,T})^n \|e_0\|_{\mathcal{B}} + \frac{e^{n \delta_t C''_{\nu,T}} - 1}{C''_{\nu,T}} \delta_t C'''_{\nu,T} \qquad (110) \\
&= (1 + \delta_t C''_{\nu,T})^n \|e_0\|_{\mathcal{B}} + \frac{e^{T C''_{\nu,T}} - 1}{C''_{\nu,T}} \delta_t C'''_{\nu,T}. \qquad (111)
\end{aligned}
$$

As the evolution of the matrix fields as well as of both schemes starts in the identity, the initial error $\|e_0\|_{\mathcal{B}}$ is 0. Thus,

$$
\|M_{\text{ex}}(T) - M_{\text{eu}}(T)\|_{\mathcal{B}} = \|e_n\|_{\mathcal{B}} \leq \delta_t C^{(4)}_{\nu,T}, \qquad (112)
$$

showing that the solution of the exponential discretization converges to the solution of the Euler scheme for $\delta_t \to 0$. Combining this with the convergence of the Euler scheme (78) gives

$$
M_{\text{ex}}(T) \to M(T) \quad \text{in } O(\delta_t) \qquad (113)
$$

for arbitrary $T > 0$. Thus, the solution of the exponential discretization converges to the continuous solution.
Decomposition property Now, we can infer the decomposition property by defining the difference between the exponential scheme and the analytical solution as

$$
e_{\delta_t, T, \nu_{id}}(x) := M(x, T) - M_{\text{ex}}(x, T). \qquad (114)
$$

and calculate

$$
M(x, 2T) = M_{\text{ex}}(x, 2T) + e_{\delta_t, 2T, \nu_{id}}(x). \qquad (115)
$$

Inserting the definition of M_{ex} in (83), we obtain

$$
\begin{aligned}
\ldots &= \coprod_{k=0}^{2n-1} \exp\left(\delta_t \nu_{id} \left(P M_{\text{ex}}\left(x, t_k\right) \bar{x}\right)\right) + e_{\delta_t, 2T, \nu_{id}}(x) \qquad (116) \\
&= \coprod_{k=n}^{2n-1} \exp\left(\delta_t \nu_{id} \left(P M_{\text{ex}}\left(x, t_k\right) \bar{x}\right)\right) \\
&\quad \coprod_{k=0}^{n-1} \exp\left(\delta_t \nu_{id} \left(P M_{\text{ex}}\left(x, t_k\right) \bar{x}\right)\right) \\
&\quad + e_{\delta_t, 2T, \nu_{id}}(x) \qquad (117) \\
&\overset{(*)}{=} M_{\text{ex}}(P M_{\text{ex}}(x, T) \bar{x}, T) M_{\text{ex}}(x, T) + e_{\delta_t, 2T, \nu_{id}}(x) \qquad (118) \\
&= \left(M(P M_{\text{ex}}(x, T) \bar{x}, T) - e_{\delta_t, T, \nu_{id}}(P M_{\text{ex}}(x, T) \bar{x})\right) \\
&\quad \left(M(x, T) - e_{\delta_t, T, \nu_{id}}(x)\right) \\
&\quad + e_{\delta_t, 2T, \nu_{id}}(x) \qquad (119) \\
&= M\left(P M_{\text{ex}}(x, T) \bar{x}, T\right) M\left(x, T\right)
\end{aligned}
$$

$$
\begin{aligned}
&- e_{\delta_t,T,v_{id}}(PM_{\text{ex}}(x,T)\bar{x})M(x,T)\\
&- M\left(PM_{\text{ex}}(x,T)\bar{x},T\right)e_{\delta_t,T,v_{id}}(x)\\
&+ e_{\delta_t,T,v_{id}}(PM_{\text{ex}}(x,T)\bar{x})e_{\delta_t,T,v_{id}}(x)\\
&+ e_{\delta_t,2T,v_{id}}(x). \qquad (120)
\end{aligned}
$$

The step in $(*)$ will be further explained below. As this equation for $M(x,2T)$ holds for all $\delta_t > 0$, the error terms vanish for $\delta_t \to 0$. As further $M_{\text{ex}}(T)$ is bounded for arbitrarily small $\delta_t > 0$, we obtain the claimed decomposition property:

$$
\begin{aligned}
M(x,2T) &= \lim_{\delta_t\to 0} M\left(PM_{\text{ex}}(x,T)\bar{x},T\right)M(x,T) \qquad (121)\\
&= M\left(PM(x,T)\bar{x},T\right)M(x,T), \qquad (122)
\end{aligned}
$$

where the last equality follows from $\lim_{\delta_t\to 0} M_{\text{ex}}(x,T) = M(x,T)$ and the continuity of M. This concludes the proof of the theorem.

$(*)$: We further explain the steps in this equality. The equality

$$
M_{\text{ex}}(x,T) = \coprod_{k=0}^{n-1} \exp\left(\delta_t v_{id}\left(PM_{\text{ex}}\left(x,t_k\right)\bar{x}\right)\right) \qquad (123)
$$

holds by definition of the exponential scheme, see (83). We now show the equality

$$
M_{\text{ex}}(PM_{\text{ex}}(x,T)\bar{x},T) = \coprod_{k=n}^{2n-1} \exp\left(\delta_t v_{id}\left(PM_{\text{ex}}\left(x,t_k\right)\bar{x}\right)\right). \qquad (124)
$$

From (83) with $x = PM_{\text{ex}}(x,T)\bar{x}$, it follows

$$
\begin{aligned}
&M_{\text{ex}}(PM_{\text{ex}}(x,T)\bar{x},T) = \coprod_{k=0}^{n-1} \exp\\
&\left(\delta_t v_{id}\left(PM_{\text{ex}}\left(PM_{\text{ex}}(x,T)\bar{x},t_k\right)\overline{PM_{\text{ex}}(x,T)\bar{x}}\right)\right). \qquad (125)
\end{aligned}
$$

The projection operator P and mapping $\bar{\cdot}$ to homogeneous coordinates cancel:

$$
\ldots = \coprod_{k=0}^{n-1} \exp\left(\delta_t v_{id}\left(PM_{\text{ex}}\left(PM_{\text{ex}}(x,T)\bar{x},t_k\right)M_{\text{ex}}(x,T)\bar{x}\right)\right). \qquad (126)
$$

Splitting the right-most factor from the matrix product, we obtain

$$
\begin{aligned}
\ldots = &\coprod_{k=1}^{n-1} \exp\left(\delta_t v_{id}\left(PM_{\text{ex}}\left(PM_{\text{ex}}(x,T)\bar{x},t_k\right)M_{\text{ex}}(x,T)\bar{x}\right)\right)\\
&\cdot \exp\left(\delta_t v_{id}\left(PM_{\text{ex}}\left(PM_{\text{ex}}(x,T)\bar{x},t_0\right)M_{\text{ex}}(x,T)\bar{x}\right)\right). \qquad (127)
\end{aligned}
$$

From the definition of the exponential scheme, it holds that $M_{\text{ex}}(\cdot,t_0) = id$, see (82), and the expression simplifies to

$$
\begin{aligned}
&M_{\text{ex}}(PM_{\text{ex}}(x,T)\bar{x},T) = \coprod_{k=1}^{n-1} \exp\\
&\left(\delta_t v_{id}\left(PM_{\text{ex}}\left(PM_{\text{ex}}(x,T)\bar{x},t_k\right)M_{\text{ex}}(x,T)\bar{x}\right)\right)\\
&\cdot \exp\left(\delta_t v_{id}\left(PM_{\text{ex}}(x,T)\bar{x}\right)\right). \qquad (128)
\end{aligned}
$$

As $t_n = T$, it holds

$$
\exp\left(\delta_t v_{id}\left(PM_{\text{ex}}(x,T)\bar{x}\right)\right) = \exp\left(\delta_t v_{id}\left(PM_{\text{ex}}(x,t_n)\bar{x}\right)\right), \qquad (129)
$$

which is precisely the right-most factor in (124). This shows that the right-most factors in the products of (124) and (125) coincide. Iteratively applying the fact that each factor in the exponential scheme (82) depends only on the (temporally) previous one, we arrive at the claimed equality. □

Appendix A Hyperparameter Tuning Results

We used the Optuna framework [54] for choosing the hyperparameters for the experiments in Sect. 4. The results of the parameter tuning are presented in Table 6. Additional information is presented in Figures 15, 16, 17. The bar graphs on the left show Optuna's estimate of how strong the specific hyperparameter impacts the score on the set of test image pairs. On the right are 2D projections of the score functions with respect to different pairs of hyperparameters.

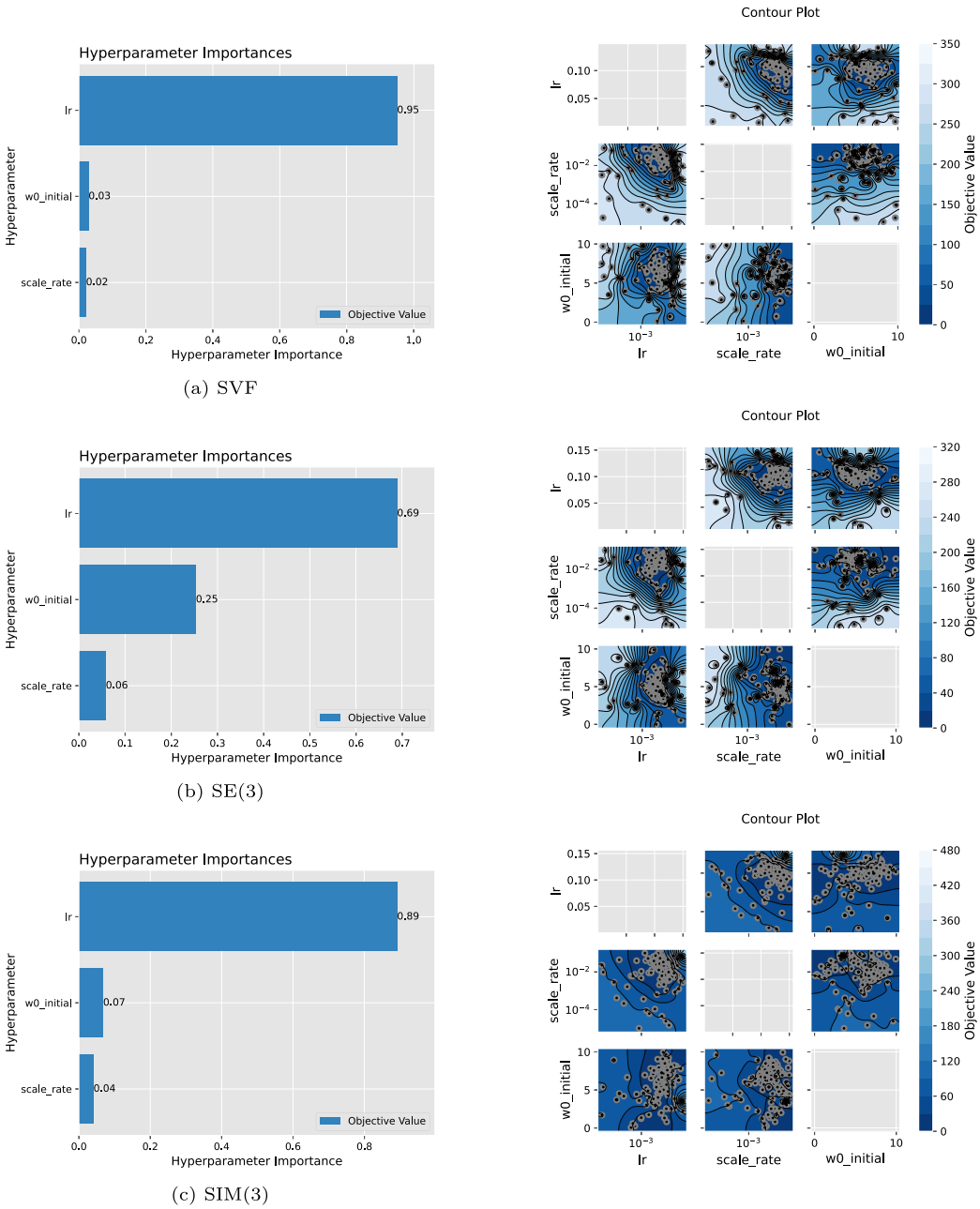


Fig. 15 Hyperparameter for fitting 5 synthetically generated deformation fields (Fig. 11). Parameters were optimized to minimize the RMSE between the given and generated deformation fields

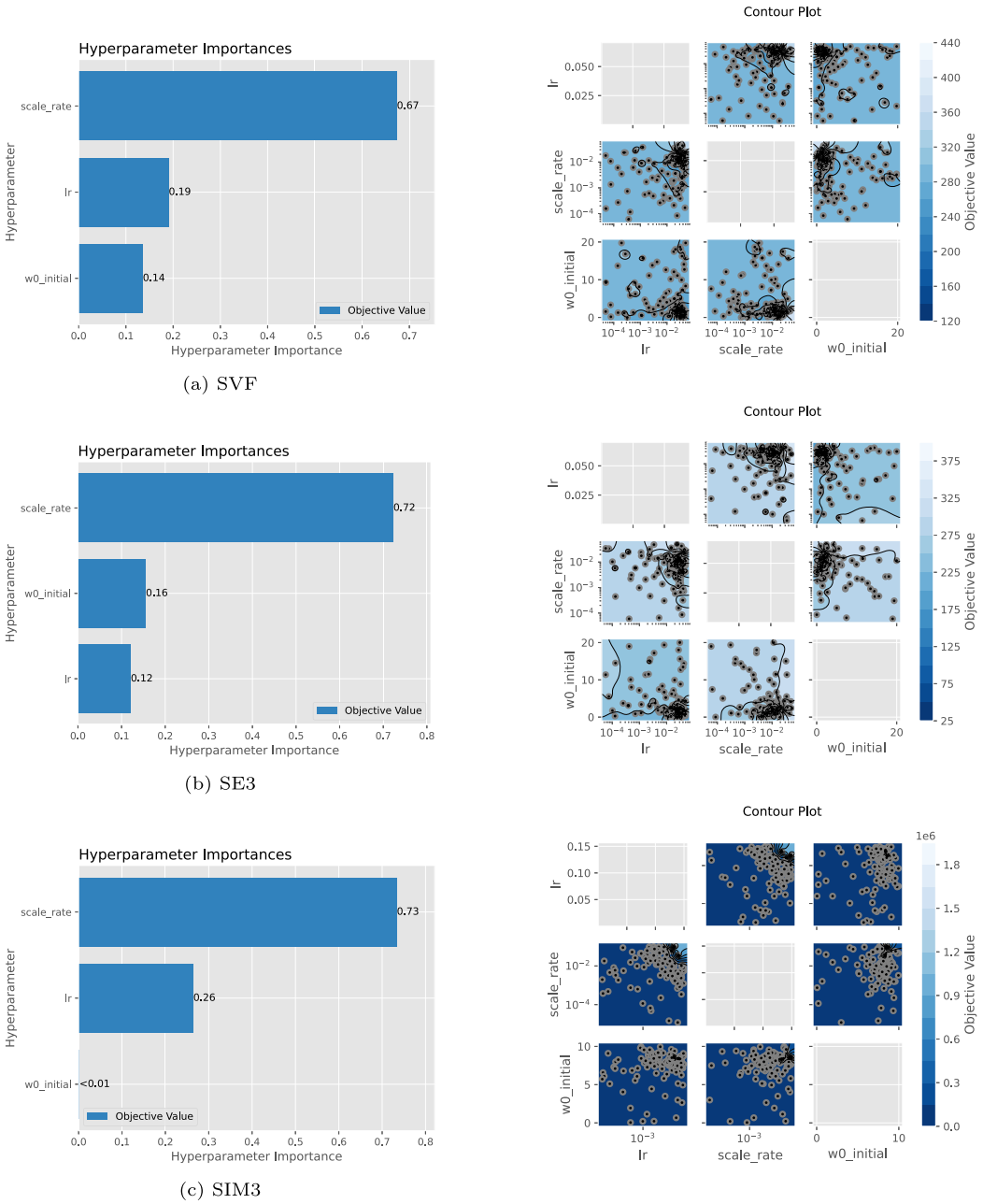
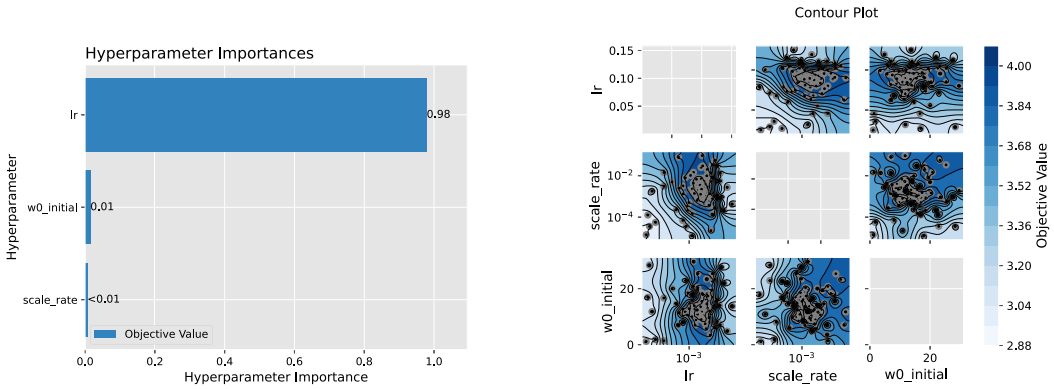


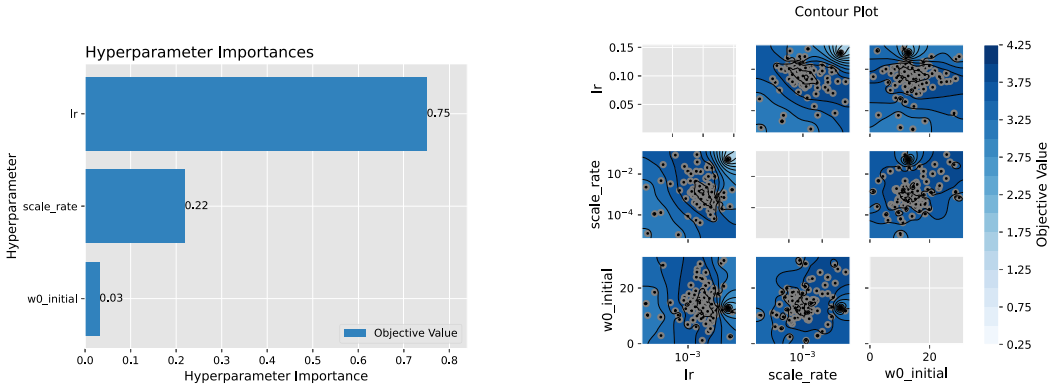
Fig. 16 Hyperparameter Tuning for registration with synthetic deformations (Fig. 12). For SIM(3), hyperparameters were ultimately chosen the same as for SE(3), as the automated search did not yield satisfactory results

Table 6 Hyperparameters after tuning with the Optuna framework: SIREN initial frequency scaling (w_0), learning rate (lr), and scaling factor (sf); see Sect. 3.4

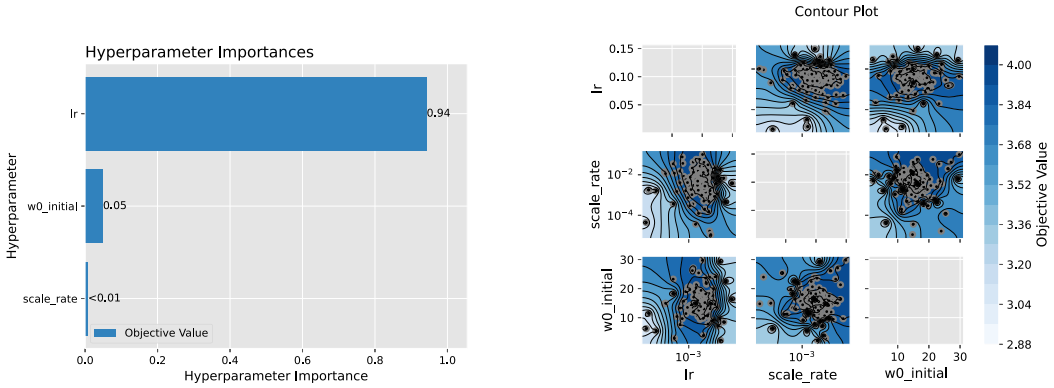
	vector field fitting			synthetic deformations			inter-patient registration		
method	w_0	lr	sf	w_0	lr	sf	w_0	lr	sf
SVF [10]	4.85	0.017	0.009	1.1e−4	3.8e−5	0.053	15.18	0.004	0.006
SE(3) (ours)	4.12	0.007	0.006	2.8e−3	0.001	0.058	13.11	0.005	0.002
SIM(3) (ours)	4.89	0.009	0.017	2.8e−3	0.001	0.058	13.92	0.008	0.018



(a) SVF



(b) SE(3)



(c) SIM(3)

Fig. 17 Hyperparameter tuning on 5 selected scull stripped volume pairs from the OASIS dataset. Parameters were optimized to maximize the sum of the Dice scores (Table 2)

Author Contributions J.B. conceived and implemented the approach. J.B. and J.L. developed the proofs and wrote the manuscript. All authors reviewed the manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data Availability No datasets were generated or analyzed during the current study.

Declarations

Conflict of interests The authors declare no Conflict of interest.

References

1. Wyawahare, M.V., Patil, P.M., Abhyankar, H.K., et al.: Image registration techniques: an overview. International Journal of Signal Processing, Image Processing and Pattern Recognition **2**(3), 11–28 (2009)
2. Sotiras, A., Davatzikos, C., Paragios, N.: Deformable medical image registration: A survey. IEEE Trans. Med. Imaging **32**(7), 1153–1190 (2013)
3. Pfingsthorn, M., Birk, A., Schwertfeger, S., Bülow, H., Pathak, K.: Maximum likelihood mapping with spectral image registration. In: 2010 IEEE International Conference on Robotics and Automation, pp. 4282–4287 (2010). IEEE
4. Sakharkar, V.S., Gupta, S.: Image stitching techniques-an overview. Int. J. Comput. Sci. Appl. **6**(2), 324–330 (2013)
5. Burger, M., Modersitzki, J., Ruthotto, L.: A hyperelastic regularization energy for image registration. SIAM J. Sci. Comput. **35**(1), 132–148 (2013)
6. Wolterink, J.M., Zwienenberg, J.C., Brune, C.: Implicit neural representations for deformable image registration. In: International Conference on Medical Imaging with Deep Learning, pp. 1349–1359 (2022). PMLR
7. Marcus, D.S., Wang, T.H., Parker, J., Csernansky, J.G., Morris, J.C., Buckner, R.L.: Open access series of imaging studies (OASIS): cross-sectional MRI data in young, middle aged, nondemented, and demented older adults. J. Cogn. Neurosci. **19**(9), 1498–1507 (2007)
8. Arsigny, V., Commowick, O., Pennec, X., Ayache, N.: A log-euclidean framework for statistics on diffeomorphisms. In: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2006: 9th International Conference, Copenhagen, Denmark, October 1-6, 2006. Proceedings, Part I 9, pp. 924–931 (2006). Springer
9. Arguillere, S., Trélat, E., Trouvé, A., Younes, L.: Shape deformation analysis from the optimal control viewpoint. Journal de Mathématiques Pures et Appliquées **104**(1), 139–178 (2015)
10. Han, K., Sun, S., Yan, X., You, C., Tang, H., Naushad, J., Ma, H., Kong, D., Xie, X.: Diffeomorphic image registration with neural velocity field. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 1869–1879 (2023)
11. Szeliski, R., Coughlan, J.: Spline-based image registration. Int. J. Comput. Vision **22**, 199–218 (1997)
12. Fornefett, M., Rohr, K., Stiehl, H.S.: Radial basis functions with compact support for elastic registration of medical images. Image Vis. Comput. **19**(1–2), 87–96 (2001)
13. Beg, M.F., Miller, M.I., Trouvé, A., Younes, L.: Computing large deformation metric mappings via geodesic flows of diffeomorphisms. Int. J. Comput. Vision **61**, 139–157 (2005)
14. Singh, N., Hinkle, J., Joshi, S., Fletcher, P.T.: A vector momenta formulation of diffeomorphisms for improved geodesic regression and atlas construction. In: 2013 IEEE 10th International Symposium on Biomedical Imaging, pp. 1219–1222 (2013)
15. Dupuis, P., Grenander, U., Miller, M.I.: Variational problems on flows of diffeomorphisms for image matching. Q. Appl. Math. **56**, 587–600 (1998)
16. Park, K., Sinha, U., Barron, J.T., Bouaziz, S., Goldman, D.B., Seitz, S.M., Martin-Brualla, R.: Nerfies: Deformable neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5865–5874 (2021)
17. Haskins, G., Kruger, U., Yan, P.: Deep learning in medical image registration: a survey. Mach. Vis. Appl. **31**, 1–18 (2020)
18. Fu, Y., Lei, Y., Wang, T., Curran, W.J., Liu, T., Yang, X.: Deep learning in medical image registration: a review. Physics in Medicine & Biology **65**(20), 20TR01 (2020)
19. Sokooti, H., Vos, B., Berendsen, F., Ghafoorian, M., Yousefi, S., Lelieveldt, B.P., Isgum, I., Staring, M.: 3d convolutional neural networks image registration based on efficient supervised learning from artificial deformations (2019). arXiv preprint arXiv:1908.10235
20. Hu, Y., Modat, M., Gibson, E., Ghavami, N., Bonmati, E., Moore, C.M., Emberton, M., Noble, J.A., Barratt, D.C., Vercauteren, T.: Label-driven weakly-supervised learning for multimodal deformable image registration. In: 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018), pp. 1070–1074 (2018). IEEE
21. Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T., Cameracci, E., Boochoon, S., Birchfield, S.: Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 969–977 (2018)
22. Balakrishnan, G., Zhao, A., Sabuncu, M.R., Guttag, J., Dalca, A.V.: Voxelmorph: a learning framework for deformable medical image registration. IEEE Trans. Med. Imaging **38**(8), 1788–1800 (2019)
23. Yang, X., Kwitt, R., Styner, M., Niethammer, M.: Quicksilver: Fast predictive image registration-a deep learning approach. Neuroimage **158**, 378–396 (2017)
24. De Vos, B.D., Berendsen, F.F., Viergever, M.A., Sokooti, H., Staring, M., Išgum, I.: A deep learning framework for unsupervised affine and deformable image registration. Med. Image Anal. **52**, 128–143 (2019)
25. Mok, T.C., Chung, A.C.: Large deformation diffeomorphic image registration with laplacian pyramid networks. In: Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part III 23, pp. 211–221 (2020). Springer
26. Amor, B.B., Arguillère, S., Shao, L.: Resnet-lddmm: advancing the lddmm framework using deep residual networks. IEEE Trans. Pattern Anal. Mach. Intell. **45**(3), 3707–3720 (2022)
27. Ramon, U., Hernandez, M., Mayordomo, E.: Lddmm meets gans: Generative adversarial networks for diffeomorphic registration. In: International Workshop on Biomedical Image Registration, pp. 18–28 (2022). Springer
28. Sitzmann, V., Martel, J., Bergman, A., Lindell, D., Wetzstein, G.: Implicit neural representations with periodic activation functions. Adv. Neural. Inf. Process. Syst. **33**, 7462–7473 (2020)
29. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Commun. ACM **65**(1), 99–106 (2021)
30. Cai, S., Mao, Z., Wang, Z., Yin, M., Karniadakis, G.E.: Physics-informed neural networks (pinns) for fluid mechanics: a review. Acta. Mech. Sin. **37**(12), 1727–1738 (2021)
31. Lu, L., Pestourie, R., Yao, W., Wang, Z., Verdugo, F., Johnson, S.G.: Physics-informed neural networks with hard constraints for inverse design. SIAM J. Sci. Comput. **43**(6), 1105–1132 (2021)
32. Strümpler, Y., Postels, J., Yang, R., Gool, L.V., Tombari, F.: Implicit neural representations for image compression. In: European Conference on Computer Vision, pp. 74–91 (2022). Springer
33. Wu, Y., Jiahao, T.Z., Wang, J., Yushkevich, P.A., Hsieh, M.A., Gee, J.C.: Nodeo: A neural ordinary differential equation based

optimization framework for deformable image registration. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 20804–20813 (2022)
34. Hawkins, T.: Emergence of the Theory of Lie Groups: An Essay in the History of Mathematics 1869–1926, (2012). Springer
35. Iachello, F.: Lie Algebras and Applications, vol. 12. Springer, London (2006)
36. Eade, E.: Lie groups for computer vision. Cambridge Univ., Cambridge, UK, Tech. Rep **2** (2014)
37. Xu, Q., Ma, D.: Applications of Lie groups and Lie algebra to computer vision: A brief survey. In: 2012 International Conference on Systems and Informatics (ICSAI2012), pp. 2024–2029 (2012). IEEE
38. Loianno, G., Watterson, M., Kumar, V.: Visual inertial odometry for quadrotors on se (3). In: 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 1544–1551 (2016). IEEE
39. Park, F.C., Bobrow, J.E., Ploen, S.R.: A Lie group formulation of robot dynamics. The International Journal of Robotics Research **14**(6), 609–618 (1995)
40. Selig, J.M.: Geometrical Methods in Robotics. Springer Science & Business Media, Heidelberg (2013)
41. Moskalev, A., Sepliarskaia, A., Sosnovik, I., Smeulders, A.: Liegg: Studying learned Lie group generators. Adv. Neural. Inf. Process. Syst. **35**, 25212–25223 (2022)
42. Finzi, M., Stanton, S., Izmailov, P., Wilson, A.G.: Generalizing convolutional neural networks for equivariance to Lie groups on arbitrary continuous data. In: International Conference on Machine Learning, pp. 3165–3176 (2020). PMLR
43. Teed, Z., Deng, J.: Tangent space backpropagation for 3d transformation groups. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10338–10347 (2021)
44. Lee, J.M., Lee, J.M.: Smooth Manifolds. Springer, London (2012)
45. Mukherjee, A.: Approximation Theorems and Whitney's Embedding. Differential Topology, pp. 43–67. Springer, Cham (2015)
46. Boumal, N.: An Introduction to Optimization on Smooth Manifolds. Cambridge University Press, Cambridge (2023)
47. Yang, X., Li, Y., Reutens, D., Jiang, T.: Diffeomorphic metric landmark mapping using stationary velocity field parameterization. Int. J. Comput. Vision **115**, 69–86 (2015)
48. Lorenzi, M., Pennec, X.: Geodesics, parallel transport & one-parameter subgroups for diffeomorphic image registration. Int. J. Comput. Vision **105**(2), 111–127 (2013)
49. Lewis, J.P.: Fast normalized cross-correlation. In: Vision. Interface **10**, 120–123 (1995)
50. Mok, T.C., Chung, A.: Fast symmetric diffeomorphic image registration with convolutional neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4644–4653 (2020)
51. Haber, E., Modersitzki, J.: Numerical methods for volume preserving image registration. Inverse Prob. **20**(5), 1621 (2004)
52. Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., Courville, A.: On the spectral bias of neural networks. In: International Conference on Machine Learning, pp. 5301–5310 (2019). PMLR
53. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
54. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2019)
55. Muller, M.E.: A note on a method for generating points uniformly on n-dimensional spheres. Commun. ACM **2**(4), 19–20 (1959). https://doi.org/10.1145/377939.377946
56. Forti, D., Rozza, G.: Efficient geometrical parametrisation techniques of interfaces for reduced-order modelling: application to fluid-structure interaction coupling problems. International Journal of Computational Fluid Dynamics **28**(3–4), 158–169 (2014)
57. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004)
58. Avants, B.B., Epstein, C.L., Grossman, M., Gee, J.C.: Symmetric diffeomorphic image registration with cross-correlation: evaluating automated labeling of elderly and neurodegenerative brain. Med. Image Anal. **12**(1), 26–41 (2008)
59. Segonne, F., Pacheco, J., Fischl, B.: Geometrically accurate topology-correction of cortical surfaces using nonseparating loops. IEEE Trans. Med. Imaging **26**, 518–529 (2007)
60. Chen, J., Frey, E.C., He, Y., Segars, W.P., Li, Y., Du, Y.: Transmorph: Transformer for unsupervised medical image registration. Med. Image Anal. (2022). https://doi.org/10.1016/j.media.2022.102615
61. Schmidt, J.: W. Walter. Gewöhnliche Differentialgleichungen. Eine Einführung. XI+ 229 S. Berlin/Heidelberg/New York 1972. Springer-Verlag. Preis brosch. DM 14, 80. Wiley Online Library (1974)
62. Brezis, H., Brézis, H.: Functional Analysis, Sobolev Spaces and Partial Differential Equations, vol. 2. Springer, London (2011)
63. Bartle, R.G., Sherbert, D.R.: Introduction to Real Analysis, vol. 2. Wiley, New York (2000)
64. Atkinson, K., Han, W., Stewart, D.E.: Numerical Solution of Ordinary Differential Equations, vol. 81. John Wiley & Sons, New Jersey (2009)
65. Liu, W., Röckner, M.: Stochastic Partial Differential Equations: an Introduction. Springer, London (2015)

Johannes Bostelmann is a PhD student at the Institute of Mathematics and Image Computing, University of Lübeck. He received his M.Sc. degree in Industrial Mathematics at the University of Hamburg in 2023.

Ole Gildemeister is a PhD student at the Institute of Mathematics and Image Computing, University of Lübeck. He received his M.Sc. degree in Mathematics in medicine and life sciences at the University of Lübeck in 2024.

Jan Lellmann is a professor in Applied Mathematics at the Institute of Mathematics and Image Computing, University of Lübeck. He obtained his doctoral degree from University of Heidelberg in 2011. After spending four years as a post-doctoral researcher and Leverhulme Early Career Fellow in the Cambridge Image Analysis group, University of Cambridge, he became full professor at the University of Lübeck in 2015. His work focuses on mathematical approaches to image processing, with an emphasis on models and optimization strategies that allow to obtain globally optimal solutions, fusing approaches from energy-based image processing, optimization, machine learning, and quantum computing. Application areas include processing of images, videos, and other multidimensional data for clinical research, biology, and earth sciences.