`Quicksilver`: Fast Predictive Image Registration – a Deep Learning Approach

Xiao Yang[†], Roland Kwitt[+], Martin Styner[†,$] and Marc Niethammer[†,*]

[†] *University of North Carolina at Chapel Hill, Chapel Hill, USA*
[*] *Biomedical Research Imaging Center (BRIC), Chapel Hill, USA*
[$] *Department of Psychiatry, UNC Chapel Hill, USA*
[+] *Department of Computer Science, University of Salzburg, Austria*

Abstract

This paper introduces `Quicksilver`, a fast deformable image registration method. `Quicksilver` registration for image-pairs works by patch-wise prediction of a deformation model based *directly* on image appearance. A deep encoder-decoder network is used as the prediction model. While the prediction strategy is general, we focus on predictions for the Large Deformation Diffeomorphic Metric Mapping (LDDMM) model. Specifically, we predict the momentum-parameterization of LDDMM, which facilitates a patch-wise prediction strategy while maintaining the theoretical properties of LDDMM, such as guaranteed diffeomorphic mappings for sufficiently strong regularization. We also provide a probabilistic version of our prediction network which can be sampled during the testing time to calculate uncertainties in the predicted deformations. Finally, we introduce a new correction network which greatly increases the prediction accuracy of an already existing prediction network. We show experimental results for uni-modal atlas-to-image as well as uni- / multi-modal image-to-image registrations. These experiments demonstrate that our method accurately predicts registrations obtained by numerical optimization, is very fast, achieves state-of-the-art registration results on four standard validation datasets, and can jointly learn an image similarity measure. `Quicksilver` is freely available as an open-source software.

Keywords: Image registration, deep learning, brain imaging

1. Introduction

Image registration is a key component for medical image analysis to provide spatial correspondences. Image registration is typically formulated as an optimization problem [1], optimizing the parameters of a transformation model. The goal is to achieve the best possible agreement between a transformed source and a target image, subject to transformation constraints. Apart from simple, low-dimensional parametric models (e.g., rigid or affine transformations), more complex, high-dimensional parametric or non-parametric registration models are able to capture subtle, localized image deformations. However, these methods, in particular, the non-parametric approaches, have a very large numbers of parameters. Therefore, numerical optimization to solve the registration problems becomes computationally costly, even with acceleration by graphics processing units (GPUs).

While computation time may not be overly critical for imaging studies of moderate size, rapid registration approaches are needed to (i) allow for interactive analysis, to (ii) allow their use as building blocks for more advanced image analysis algorithms; and to (iii) time- and cost-efficiently analyze very large imaging studies. As a case in point, sample sizes of neuroimaging studies are rapidly increasing. While, only two decades ago, neuroimaging studies with few tens of subjects were not unusual, we are now witnessing the emergence of truly large-scale imaging studies. For example, the UK Biobank study is, at the moment, the world's largest health imaging study and will image "the brain, bones, heart, carotid arteries and abdominal fat of 100,000 participants" using magnetic resonance (MR) imaging within the next few years [2]. *Furthermore, image sizes are increasing drastically.* While, a decade ago, structural MR images of human brains with voxel sizes of $2 \times 2 \times 2$ mm^3 were typical for state-of-the-art MR acquisitions, today we have voxel sizes smaller than $1 \times 1 \times 1$ mm^3 as, for example, acquired by the human connectome project [3]. This increase in image resolution increases the data size by an order of magnitude. Even more dramatically, the microscopy field now routinely generates gigabytes of high-resolution imaging data, for example, by 3D imaging via tissue clearing [4]. Hence, fast, memory-efficient, and parallelizable image analysis approaches are critically needed. In particular, such approaches are needed for deformable image registration, which is a key component of many medical image analysis systems.

Attempts at speeding-up deformable image registration have primarily focused on GPU implementations [5], with impressive speed-ups over their CPU-based counter-

parts. However, these approaches are still relatively slow. Runtimes in the tens of minutes are the norm for popular deformable image registration solutions. For example, a GPU-based registration of a $128 \times 128 \times 128$ image volume using LDDMM will take about 10 minutes on a current GPU (e.g., a Nvidia TitanX). This is much too slow to allow for large-scale processing, the processing of large datasets, or close to interactive registration tasks. Hence, improved algorithmic approaches are desirable. Recent work has focused on *better numerical methods* and *approximate* approaches. For example, Ashburner and Friston [6] use a Gauss–Newton method to accelerate convergence for LDDMM and Zhang et al. [7] propose a finite-dimensional approximation of LDDMM, achieving a roughly 25× speed-up over a standard LDDMM optimization-based solution.

An alternative approach to improve registration speed is to *predict* deformation parameters, or deformation parameter update steps in the optimization via a regression model, instead of directly minimizing a registration energy [8, 9, 10]. The resulting predicted deformation fields can either be used directly, or as an initialization of a subsequent optimization-based registration. However, the high dimensionality of the deformation parameters as well as the non-linear relationship between the images and the parameters pose a significant challenge. Among these methods, Chou et al. [10] propose a multi-scale linear regressor which only applies to affine deformations and low-rank approximations of non-linear deformations. Wang et al. [11] predict deformations by key-point matching using sparse learning followed by dense deformation field generation with radial basis function interpolation. The performance of the method heavily depends on the accuracy of the key point selection. Cao et al. [12] use a semi-coupled dictionary learning method to directly model the relationship between the image appearance and the deformation parameters of the LDDMM model [13]. However, only a linear relationship is assumed between image appearance and the deformation parameters. Lastly, Gutierrez et al. [9] use a regression forest and gradient boosted trees [8] based on hand-crafted features to learn update steps for a rigid and a B-spline registration model.

In this work, we propose a deep regression model to predict deformation parameters using image appearances in a time-efficient manner. Deep learning has been used for optical flow estimation [14, 15] and deformation parameter prediction for affine transformations [16]. We investigate a non-parametric image registration approach, where we predict voxel-wise deformation parameters from image patches. Specifically, we focus on the initial momentum LDDMM shooting model [17], as it has many desirable properties:

- It is based on Riemannian geometry, and hence induces a distance metric on the space of images.
- It can capture large deformations.
- It results in highly desirable diffeomorphic spatial transformations (if regularized sufficiently). I.e., transformations which are smooth, one-to-one and have a smooth inverse.
- It uses the initial momentum as the registration parameter, which does not need to be spatially smooth, and hence can be predicted patch-by-patch, and from which the whole geodesic path can be computed.

The LDDMM shooting model in of itself is important for various image analysis tasks such as principal component analysis [18] and image regression [19, 20].

Our *contributions* are as follows:

- *Convenient parameterization:* Diffeomorphic transformations are desirable in medical image analysis applications to smoothly map between fixed and moving images, or to and from an atlas image. Methods, such as LDDMM, with strong theoretical guarantees exist, but are typically computationally very demanding. On the other hand, direct prediction, e.g., of optical flow [14, 15], is fast, but the regularity of the obtained solution is unclear as it is not considered within the regression formulation. We demonstrate that the momentum-parameterization for LDDMM shooting [17] is a convenient representation for regression approaches as (i) the momentum is typically compactly supported around image edges and (ii) there are no smoothness requirements on the momentum itself. Instead, smooth velocity fields are obtained in LDDMM from the momentum representation by *subsequent* smoothing. Hence, by predicting the momentum, we retain all the convenient mathematical properties of LDDMM and, at the same time, are able to predict diffeomorphic transformations *fast*. As the momentum has compact support around image edges, no ambiguities arise within uniform image areas (in which predicting a velocity or deformation field would be difficult).
- *Fast computation:* We use a sliding window to locally predict the LDDMM momentum from image patches. We experimentally show that by using patch pruning and a large sliding window stride, our method achieves dramatic speedups compared to the optimization approach, while maintaining good registration accuracy.
- *Uncertainty quantification:* We extend our network to a Bayesian model which is able to determine the uncertainty of the registration parameters and, as a result, the uncertainty of the deformation field. This uncertainty information could be used, e.g., for uncertainty-based smoothing [21], or for surgical treatment planning, or could be directly visualized for qualitative analyses.

- *Correction network:* Furthermore, we propose a correction network to increase the accuracy of the prediction network. Given a trained prediction network, the correction network predicts the difference between the ground truth momentum and the predicted result. The difference is used as a correction to the predicted momentum to increase prediction accuracy. Experiments show that the correction network improves registration results to the point where optimization-based and predicted registrations achieve a similar level of registration accuracy on registration validation experiments.

- *Multi-modal registration:* We also explore the use of our framework for multi-modal image registration prediction. The goal of multi-modal image registration is to establish spatial correspondences between images acquired by different modalities. Multi-modal image registration is, in general, significantly more difficult than uni-modal image registration since image appearance can change drastically between different modalities. General approaches address multi-modal image registration by either performing image synthesis [22, 23] to change the problem to an uni-modal image registration task, or by proposing complex, hand-crafted [24, 25, 26, 27] or learned [28, 29, 30, 31, 32] multi-modal image similarity measures. In contrast, we demonstrate that our framework can *simultaneously* predict registrations and learn a multi-modal image similarity measure. Our experiments show that our approach also predicts accurate deformations for multi-modal registration.

- *Extensive validation:* We extensively validate our predictive image registration approach for uni-modal image registration on the four validation datasets of Klein et al. [33] and demonstrate registration accuracies on these datasets on par with the state-of-the-art. Of note, these registration results are achieved using a model that was trained on an entirely different dataset (images from the OASIS dataset). Furthermore, we validate our model trained for multi-modal image registration using the IBIS 3D dataset [34]. Overall, our results are based on more than 2,400 image registration pairs.

The registration method described here, which we name `Quicksilver`, is an extension of the preliminary ideas we presented in a recent workshop paper [35] and in a conference paper [36]. This paper offers more details of our proposed approaches, introduces the idea of improving registration accuracy via a correction network, and includes a comprehensive set of experiments for image-to-image registration.

Organization. The remainder of the paper is organized as follows. Sec. 2.1 reviews the registration parameterization of the shooting-based LDDMM registration algorithm. Sec. 2.2 introduces our deep network architecture for deformation parameter prediction, the Bayesian formulation of our network, as well as our strategy for speeding up the deformation prediction. Sec. 2.3 discusses the *correction network* and the reason why it improves the registration prediction accuracy over an existing prediction network. Sec. 3 presents experimental results for *atlas-to-image* and *image-to-image* registration. Finally, Sec. 4 discusses potential extensions and applications of our method.

2. Materials and Methods

2.1. LDDMM Shooting

Given a moving (source) image M and a target image T, the goal of image registration is to find a deformation map $\Phi : \mathbb{R}^d \rightarrow \mathbb{R}^d$, which maps the moving image to the target image in such a way that the deformed moving image is similar to the target image, i.e., $M \circ \Phi^{-1}(x) \approx T(x)$. Here, d denotes the spatial dimension and x is the spatial coordinate of the fixed target image T. Due to the importance of image registration, a large number of different approaches have been proposed [1, 37, 38, 39]. Typically, these approaches are formulated as optimization problems, where one seeks to minimize an energy of the form

$$E(\Phi) = \text{Reg}[\Phi] + \frac{1}{\sigma^2}\text{Sim}[I_0 \circ \Phi^{-1}, I_1], \tag{1}$$

where $\sigma > 0$ is a balancing constant, $\text{Reg}[\cdot]$ regularizes the spatial transformation, Φ, by penalizing spatially irregular (for example non-smooth) spatial transformations, and $\text{Sim}[\cdot,\cdot]$ is an image dissimilarity measure, which becomes small if images are similar to each other. Image dissimilarity is commonly measured by computing the sum of squared differences (SSD) between the warped source image ($I_0 \circ \Phi^{-1}$) and the target image (I_1), or via (normalized) cross-correlation, or mutual information [26, 1]. For simplicity, we use SSD in what follows, but other similarity measures could also be used. The regularizer $\text{Reg}[\cdot]$ encodes what should be considered a plausible spatial transformation[1]. The form of the regularizer depends on how a transformation is represented. In general, one distinguishes between parametric and non-parametric transformation models [1]. Parametric transformation models make use of a relatively low-dimensional parameterization of the transformation. Examples are rigid, similarity, and affine transformations. But also the highly popular B-spline models [40] are examples of parametric transformation models. Non-parametric approaches on the other hand parameterize a transformation locally, with a parameter (or parameter vector) for each voxel. The most direct non-parametric approach is to represent voxel displacements,

[1] A regularizer is not necessarily required for simple, low-dimensional transformation models, such as rigid or affine transformations.

$u(x) = \Phi(x) - x$. Regularization then amounts to penalizing norms involving the spatial derivatives of the displacement vectors. Regularization is necessary for non-parametric approaches to avoid ill-posedness of the optimization problem. Optical flow approaches, such as the classical Horn and Schunck optical flow [41], the more recent total variation approaches [42], or methods based on linear elasticity theory [1] are examples for displacement-based registration formulations. Displacement-based approaches typically penalize large displacements strongly and hence have difficulty capturing large image deformations. Furthermore, they typically also only offer limited control over spatial regularity. Both shortcomings can be circumvented. The first by applying greedy optimization strategies (for example, by repeating registration and image warping steps) and the second, for example, by explicitly enforcing image regularity by constraining the determinant of the Jacobian of the transformation [43]. An alternative approach to allow for large deformations, while assuring diffeomorphic transformations, is to parameterize transformations via static or time-dependent velocity fields [44, 13]. In these approaches, the transformation Φ is obtained via time integration. For sufficiently regular velocity fields, diffeomorphic transformations can be obtained. As the regularizer operates on the velocity field(s) rather than the displacement field, large deformations are no longer strongly penalized and hence can be captured.

LDDMM is a non-parametric registration method which represents the transformation via spatio-temporal velocity fields. In particular, the sought-for mapping, Φ, is obtained via an integration of a spatio-temporal velocity field $v(x,t)$ for unit time, where t indicates time and $t \in [0,1]$, such that $\Phi_t(x,t) = v(\Phi(x,t),t)$ and the sought-for mapping is $\Phi(x,1)$. To single-out desirable velocity-fields, non-spatial-smoothness at any given time t is penalized by the regularizer Reg[$\cdot$], which is applied to the velocity field instead of the transform Φ directly. Specifically, LDDMM aims at minimizing the energy[2] [13]

$$E(v) = \int_0^1 \|v\|_L^2 \, dt + \frac{1}{\sigma^2}\|M \circ \Phi^{-1}(1) - T\|^2, \\ \text{s.t.} \quad \Phi_t(x,t) = v(\Phi(x,t),t), \;\; \Phi(x,0) = \text{id} \qquad (2)$$

where $\sigma > 0$, $\|v\|_L^2 = \langle Lv, v\rangle$, L is a self-adjoint differential operator[3], id is the identity map, and the differential equation constraint for Φ can be written in Eulerian coordinates as $\Phi_t^{-1} + D\Phi^{-1}v = 0$, where $\Phi_t(x,t)$ is the derivative of Φ with respect to time t, and D is the Jacobian matrix. In this LDDMM formulation (termed the *relaxation* formulation as a geodesic path – the optimal solution – is only obtained at optimality) the registration is parameterized by the full spatio-temporal velocity field $v(x,t)$. From the perspective of an individual particle, the transformation is simply obtained by following the velocity field over time. To optimize over the spatio-temporal velocity field one solves the associated adjoint system backward in time, where the final conditions of the adjoint system are determined by the current image mismatch as measured by the chosen similarity measure [13]. This adjoint system can easily be determined via a constrained optimization approach [45] (see [46] for the case of optical flow). From the solution of the adjoint system one can compute the gradient of the LDDMM energy with respect to the velocity field at any point in time[4] and use it to numerically solve the optimization problem, for example, by a line-search [49]. At convergence, the optimal solution will fulfill the optimality conditions of the constrained LDDMM energy of Eq. (2). These optimality conditions can be interpreted as the continuous equivalent of the Karush-Kuhn-Tucker conditions of constrained optimization [49]. On an intuitive level, if one were to find the shortest path between two points, one would (in Euclidean space) obtain the straight line connecting these two points. This straight line is the geodesic path in Euclidean space. For LDDMM, one instead tries to find the shortest path between two images based on the minimizer of the inexact matching problem of Eq. (2). The optimization via the adjoint equations corresponds to starting with a possible path and then successively improving it, until the optimal path is found. Again, going back to the example of matching points, one would start with any possible path connecting the two points and then successively improve it. The result at convergence is the optimal straight line path.

Convergence to the shortest path immediately suggests an alternative optimization formulation. To continue the point matching example: if one knows that the optimal solution needs to be a straight line (i.e., a geodesic) one can consider optimizing only over the space of straight lines instead of all possible paths connecting the two points. This dramatically reduces the parameter space for optimization as one now only needs to optimize over the y-intercept and the slope of the straight line. LDDMM can also be formulated in such a way. One obtains the *shooting* formulation [17, 19], which parameterizes the deformation via the initial momentum vector field $m_0 = m(0)$ and the initial map $\Phi^{-1}(0)$, from which the map Φ can be computed for any point in time. The initial momentum corresponds to the slope of the line and the initial map corresponds to

[2] When clear from the context, we suppress spatial dependencies for clarity of notation and only specify the time variable. E.g., we write $\Phi^{-1}(1)$ to mean $\Phi^{-1}(x,1)$.

[3] Note that we define $\|v\|_L^2$ here as $\langle Lv, v\rangle$ instead of $\langle Lv, Lv\rangle = \langle L^\dagger Lv, v\rangle$ as for example in Beg et al. [13].

[4] This approach is directly related to what is termed error back-propagation in the neural networks community [47] as well as the reverse mode in automatic differentiation [48]. The layers in neural networks are analogous to discretized time-steps for LDDMM. The weights which parameterize a neural network are analogous to the velocity fields for LDDMM. Error-backpropagation via the chain rule in neural networks corresponds to the adjoint system in LDDMM, which is a partial differential equation when written in the Eulerian form in the continuum.

the y-intercept. The geodesic equations correspond to the line equation. The geodesic equations, in turn, correspond to the optimality conditions of Eq. (2). Essentially, the shooting formulation enforces these optimality conditions of Eq. (2) as a constraint. In effect, one then searches only over geodesic paths, as these optimality conditions are geodesic equations. They can be written in terms of the momentum m alone. In particular, the momentum is the dual of the velocity v, which is an element in the reproducing kernel Hilbert space V; m and v are connected by a positive-definite, self-adjoint differential smoothing operator K by $v = Km$ and $m = Lv$, where L is the inverse of K. Given m_0, the complete spatio-temporal deformation $\Phi(x,t)$ is determined.

Specifically, the energy to be minimized for the shooting formulation of LDDMM is [50]

$$E(m_0) = \langle m_0, Km_0 \rangle + \frac{1}{\sigma^2}||M \circ \Phi^{-1}(1) - T||^2, \quad \text{s.t.} \tag{3}$$

$$\begin{aligned} m_t + \mathrm{ad}_v^* m &= 0, \\ m(0) &= m_0, \\ \Phi_t^{-1} + D\Phi^{-1} v &= 0, \\ \Phi^{-1}(0) &= \mathrm{id}, \\ m - Lv &= 0 \ , \end{aligned} \tag{4}$$

where id is the identity map, and the operator ad^* is the dual of the negative Jacobi-Lie bracket of vector fields, i.e., $\mathrm{ad}_v w = -[v,w] = Dvw - Dwv$. The optimization approach is similar to the one for the relaxation formulation. I.e., one determines the adjoint equations for the shooting formulation and uses them to compute the gradient with respect to the unknown initial momentum m_0 [50, 17]. Based on this gradient an optimal solution can, for example, be found via a line-search or by a simple gradient descent scheme.

A natural approach for deformation prediction would be to use the entire 3D moving and target images as input, and to directly predict the 3D displacement field. However, this is not feasible in our formulation (for large images) because of the limited memory in modern GPUs. We circumvent this problem by extracting image patches from the moving image and target image at the same location, and by then predicting deformation *parameters* for the patch. The entire 3D image prediction is then accomplished patch-by-patch via a sliding window approach. Specifically, in our framework, we predict the initial momentum m_0 given the moving and target images patch-by-patch. Using the initial momentum for patch-based prediction is a convenient parameterization because (i) the initial momentum is generally not smooth, but is compactly supported at image edges and (ii) the initial velocity is generated by applying a smoothing kernel K to the initial momentum. Therefore, the smoothness of the deformation does not need to be specifically considered during the parameter prediction step, but is imposed *after* the prediction. Since K governs the theoretical properties or LDDMM, a strong K assures diffeomorphic transformations[5], making predicting the initial momentum an ideal choice. However, predicting alternative parameterizations such as the initial velocity or directly the displacement field would make it difficult to obtain diffeomorphic transformations. Furthermore, it is hard to predict initial velocity or displacement for homogeneous image regions, as these regions locally provide no information from which to predict the spatial transformation. In these regions the deformations are purely driven by regularization. This is not a problem for the initial momentum parameterization, since the initial momentum in these areas, for image-based LDDMM, is zero. This can be seen as for image-based LDDMM [17, 19, 45] the momentum can be written as $m(x,t) = \lambda(x,t)\nabla I(x,t)$, where λ is a scalar field and ∇I is the spatial gradient of the image. Hence, for homogeneous areas, $\nabla I = 0$ and consequentially $m = 0$. Fig. 1 illustrates this graphically. In summary, the initial momentum parameterization is ideal for our patch-based prediction method. Note that since the initial momentum can be written as $m = \lambda \nabla I$ one can alternatively optimize LDDMM over the scalar-valued momentum λ. This is the approach that has historically been taken for LDDMM [13, 45, 17]. However, optimizing over the vector-valued momentum, m, instead is numerically better behaved [50], which is why we focus on it for our predictions. While we are not exploring the prediction of the scalar-valued momentum λ here, it would be interesting to see how scalar-valued and vector-valued momentum predictions compare. In particular, since the prediction of the scalar-valued momentum would allow for simpler prediction approaches (see details in Sec. 2.2).

2.2. Deep network for LDDMM prediction

The overall training strategy for our prediction models is as follows: We assume that we already have a set of LDDMM parameters which result in good registration results. We obtain these registration results by numerically optimizing the shooting formulation of LDMMM. These numerical optimizations can be based on images alone or could, of course, also make use of additional information available at training time, for example, object labels. For simplicity we only use image information here, but note that using additional information during training may result in increased prediction performance. The resulting initial momenta serve as training data. The goal is then to train a model to locally *predict* initial momenta from image patches of the moving and the target images. These predicted momenta should be good approximations of the initial momenta obtained via numerical optimization. In short, *we train our deep learning framework to predict the initial momenta from image patches based on training data obtained from numerical optimization of the LDDMM shooting formulation.* During testing, we predict

[5] See [13, 51] for the required regularity conditions.

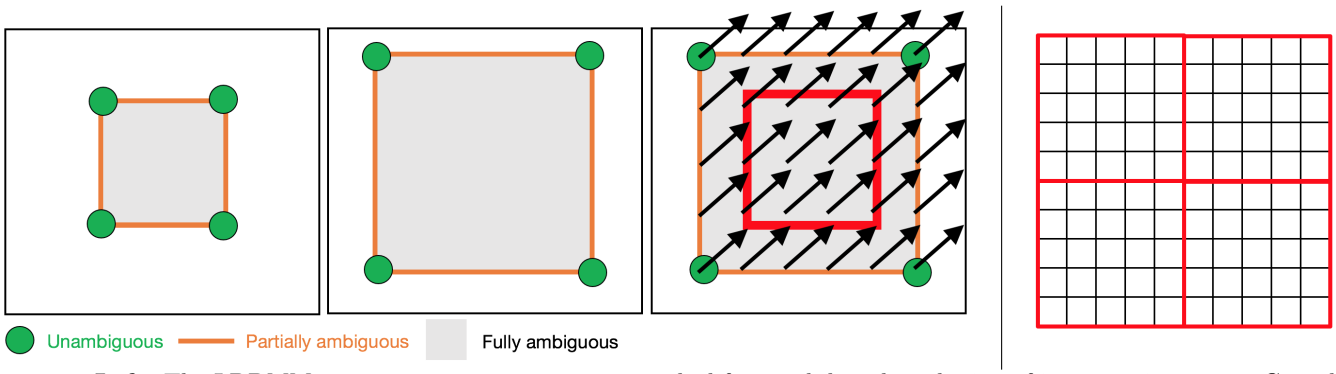


Figure 1: Left: The LDDMM momentum parameterization is ideal for patch-based prediction of image registrations. Consider registering a small square (left) to a large square (middle) with uniform intensity. Only the corner points suggest clear spatial correspondences. Edges also suggest spatial correspondences, however, correspondences between *individual* points on edges remain ambiguous. Lastly, points interior to the squares have ambiguous spatial correspondences, which are established purely based on regularization. Hence, predicting velocity or displacement fields (which are spatially dense) from patches is challenging in these interior areas (right), in the absence of sufficient spatial context. Predicting a displacement field as illustrated in the right image from an interior patch (illustrated by the red square) would be impossible if both the target and the source image patches are uniform in intensity. In this scenario, the patch information would not provide sufficient spatial context to capture aspects of the deformation. On the other hand, we know from LDDMM theory that the optimal momentum, m, to match images can be written as $m(x,t) = \lambda(x,t)\nabla I(x,t)$, where $\lambda(x,t) \mapsto \mathbb{R}$ is a spatio-temporal scalar field and $I(x,t)$ is the image at time t [45, 19, 17]. Hence, in spatially uniform areas (where correspondences are ambiguous) $\nabla I = 0$ and consequentially $m(x,t) = 0$. This is highly beneficial for prediction as the momentum only needs to be predicted at image edges. **Right:** Furthermore, as the momentum is not spatially smooth, the regression approach does not need to account for spatial smoothness, which allows predictions with non-overlapping or hardly-overlapping patches as illustrated in the figure by the red squares. This is not easily possible for the prediction of displacement or velocity fields since these are expected to be spatially dense and smooth, which would need to be considered in the prediction. Consequentially, predictions of velocity or displacement fields will inevitably result in discontinuities across patch boundaries (i.e., across the red square boundaries shown in the figure) if they are predicted independently of each other.

the initial momenta for the test image pairs, and generate the predicted deformation result simply by performing LDDMM shooting.

Fig. 2 shows the structure of the initial momentum prediction network. We first discuss the deterministic version of the network without dropout layers. We then introduce the Bayesian version of our network where dropout layers are used to convert the architecture into a probabilistic deep network. Finally, we discuss our strategy for patch pruning to reduce the number of patches needed for whole image prediction.

2.2.1. Deterministic network

Our goal is to learn a *prediction function* that takes two input patches, extracted at the same location[6] from the *moving* and *target image*, and predicts a desired initial vector-valued momentum patch, separated into the x, y and z dimensions, respectively. This prediction function should be learned from a set of training sample patches. These initial vector-valued momentum patches are obtained by numerical optimization of the LDDMM shooting formulation. More formally, given a 3D patch of size $p \times p \times p$ voxels, we want to learn a function $f : \mathbb{R}^{3p} \times \mathbb{R}^{3p} \rightarrow \mathbb{R}^{9p}$.

[6]The locations of these patches are the same locations with respect to image grid coordinates, as the images are still unregistered at this point.

In our formulation, f is implemented by a deep neural network. Ideally, for two 3D image patches $(\mathbf{u}, \mathbf{v}) = \mathbf{x}'$, with $\mathbf{u}, \mathbf{v} \in \mathbb{R}^{3p}$, we want $\mathbf{y}' = f(\mathbf{x}')$ to be as close as possible to the desired LDDMM optimization momentum patch $\mathbf{y}$ with respect to an appropriate loss function (e.g., the 1-norm). Our proposed architecture (for f) consists of two parts: an *encoder* and a *decoder* which we describe next.

Encoder. The **Encoder** consists of two parallel encoders which learn features from the moving/target image patches independently. Each encoder contains two blocks of three $3 \times 3 \times 3$ 3D convolution layers and PReLU [52] activation layers, followed by another $2 \times 2 \times 2$ convolution+PReLU with a stride of two, cf. Fig. 2. The convolution layers with a stride of two reduce the size of the output patch, and essentially perform pooling operations. PReLU is an extension of the ReLU activation [53], given as

$$\mathrm{PReLU}(x) = \begin{cases} x, & \text{if } x > 0 \\ ax, & \text{otherwise} \end{cases},$$

where a is a parameter that is learned when training the network. In contrast to ReLU, PReLU avoids a zero gradient for negative inputs, effectively improving the network performance. The number of features in the first block is 64 and increases to 128 in the second block. The learned features from the two encoders are then concatenated and sent to three parallel decoders (one per dimension x, y, z).

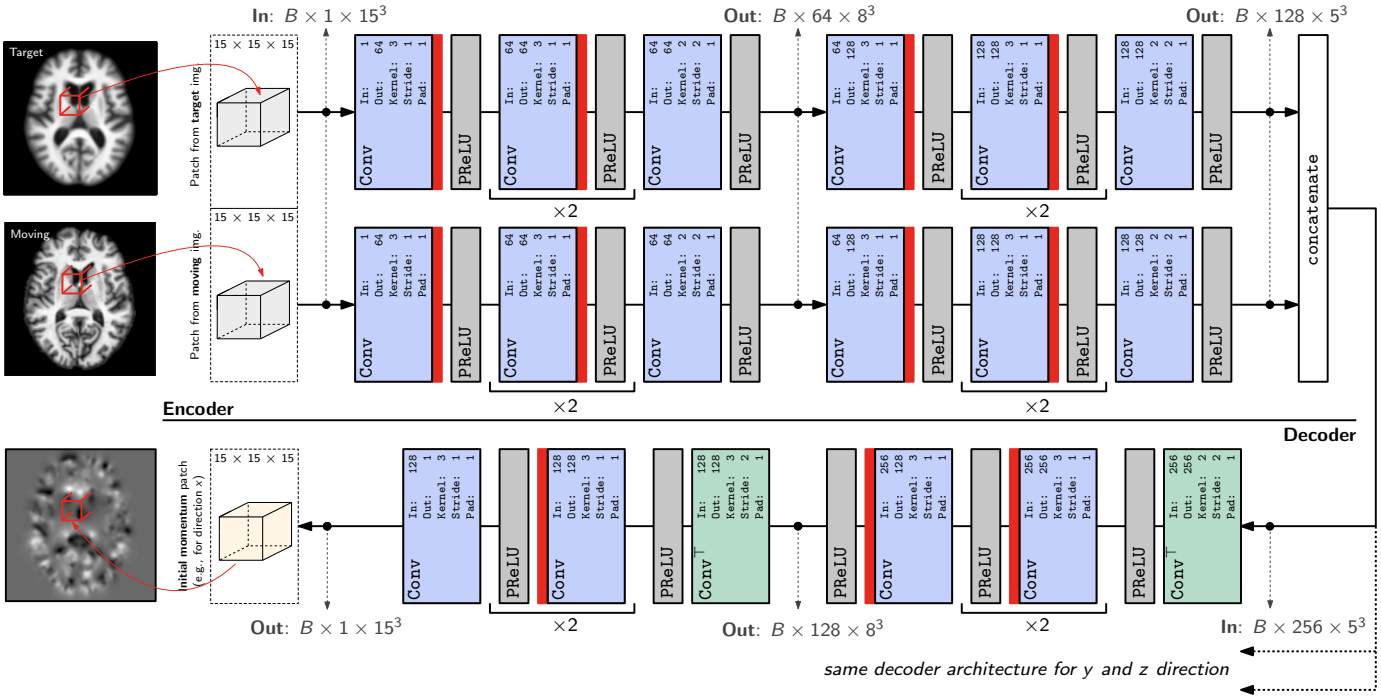


Figure 2: 3D (probabilistic) network architecture. The network takes two 3D patches from the moving and target image as the input, and outputs 3 3D initial momentum patches (one for each of the x, y and z dimensions respectively; for readability, only one decoder branch is shown in the figure). In case of the deterministic network, see Sec. 2.2.1, the dropout layers, illustrated by ■, are removed. `Conv`: 3D convolution layer. `Conv`$^\intercal$: 3D transposed convolution layer. Parameters for the `Conv` and `Conv`$^\intercal$ layers: `In`: input channel. `Out`: output channel. `Kernel`: 3D filter kernel size in each dimension. `Stride`: stride for the 3D convolution. `Pad`: zero-padding added to the boundaries of the input patch. Note that in this illustration B denotes the batch size.

Decoder. Each decoder's structure is the inverse of the encoder, except that the number of features is doubled (256 in the first block and 128 in the second block) as the decoder's input is obtained from the *two* encoder branches. We use 3D transposed convolution layers [54] with a stride of 2, which are shown as the cyan layers in Fig. 2 and can be regarded as the backward propagation of 3D convolution operations, to perform "unpooling". We also omit the non-linearity after the final convolution layer, cf. Fig. 2.

The idea of using convolution and transpose of convolution to learn the pooling/unpooling operation is motivated by [55], and it is especially suited for our network as the two encoders perform pooling independently which prevents us from using the pooling index for unpooling in the decoder. During training, we use the *1-norm* between the predicted and the desired momentum to measure the prediction error. We chose the 1-norm instead of the 2-norm as our loss function to be able to tolerate outliers and to generate sharper momentum predictions. Ultimately, we are interested in predicting the deformation map and not the patch-wise momentum. However, this would require forming the entire momentum image from a collection of patches followed by shooting as part of the network training. Instead, predicting the momentum itself patch-wise significantly simplifies the network training procedure. Also note that, while we predict the momentum patch-by-patch, smoothing is performed over the full momentum image (reassembled from the patches) based on the smoothing kernel, K, of LDDMM. Specifically, when predicting the deformation parameters for the whole image, we follow a sliding window strategy to predict the initial momentum in a patch-by-patch manner and then average the overlapping areas of the patches to obtain the final prediction result.

The number of 3D filters used in the network is 975,360. The overall number of parameters is 21,826,344. While this is a large number of parameters, we also have a very large number of training patches. For example, in our image-to-image registration experiments (see Sec. 3), the total number of $15 \times 15 \times 15$ 3D training patches to train the prediction network is 1,002,404. This amounts to approximately 3.4 billion voxels and is much larger than the total number of parameters in the network. Moreover, recent research [56] suggests that the degrees of freedom for a deep network can be significantly smaller than the number of its parameters.

One question that naturally arises is why to use independent encoders/decoders in the prediction network. For the decoder part, we observed that an independent decoder structure is much easier to train than a network with one large decoder (3 times the number of features of a single decoder in our network) to predict the initial momentum in all dimensions simultaneously. In our exper-

iments, such a combined network easily got stuck in poor local minima. As to the encoders, experiments do not show an obvious difference in prediction accuracy between using two independent encoders and one single large encoder. However, such a two-encoder strategy is beneficial when extending the approach to multi-modal image registration [36]. Hence, using a two-encoder strategy here will make the approach easily retrainable for multi-modal image registration. In short, our network structure can be viewed as a multi-input multi-task network, where each encoder learns features for one patch source, and each decoder uses the shared image features from the encoders to predict one spatial dimension of the initial momenta. We remark that, if one were to predict the scalar-valued momentum, λ, instead of the vector-valued momentum, m, the network architecture could remain largely unchanged. The main difference would be that only one decoder would be required. Due to the simpler network architecture such an approach could potentially speed-up predictions. However, it remains to be investigated how such a network would perform in practice as the vector-valued momentum has been found to numerically better behave for LDDMM optimizations [50].

2.2.2. Probabilistic network

We extend our architecture to a probabilistic network using dropout [57], which can be viewed as (Bernoulli) approximate inference in Bayesian neural networks [58, 59]. In the following, we briefly review the basic concepts, but refer the interested reader to the corresponding references for further technical details.

In our problem setting, we are given training patch tuples $\mathbf{x}_i = (\mathbf{u}_i, \mathbf{v}_i)$ with associated desired initial momentum patches $\mathbf{y}_i$. We denote the collection of this training data by $\mathbf{X}$ and $\mathbf{Y}$. In the standard, non-probabilistic setting, we aim for predictions of the form $\mathbf{y}' = f(\mathbf{x}')$, given a new input patch $\mathbf{x}'$, where f is implemented by the proposed encoder-decoder network. In the *probabilistic* setting, however, the goal is to make predictions of the form $p(\mathbf{y}'|\mathbf{x}', \mathbf{X}, \mathbf{Y})$. As this predictive distribution is intractable for most underlying models (as it would require integrating over all possible models, and neural networks in particular), the idea is to condition the model on a set of random variables $\mathbf{w}$. In case of (convolutional) neural networks with N layers, these random variables are the weight matrices, i.e., $\mathbf{w} = (\mathbf{W}_i)_{i=1}^N$. However, evaluation of the predictive distribution $p(\mathbf{y}'|\mathbf{x}', \mathbf{X}, \mathbf{Y})$ then requires the posterior over the weights $p(\mathbf{w}|\mathbf{X}, \mathbf{Y})$ which can (usually) not be evaluated analytically. Therefore, in variational inference, $p(\mathbf{w}|\mathbf{X}, \mathbf{Y})$ is replaced by a tractable variational distribution $q(\mathbf{w})$ and one minimizes the Kullback-Leibler divergence between $q(\mathbf{w})$ and $p(\mathbf{w}|\mathbf{X}, Y)$ with respect to the variational parameters $\mathbf{w}$. This turns out to be equivalent to maximization of the *log evidence lower bound (ELBO)*. When the variational distribution is defined as

$$q(\mathbf{W}_i) = \mathbf{M}_i \cdot \text{diag}([z_{i,j}]_{j=1}^{K_i}), \quad z_{i,j} \sim \text{Bernoulli}(d) \ , \quad (5)$$

where $\mathbf{M}_i$ is the convolutional weight, $i = 1, \ldots, N$, d is the probability that $z_{i,j} = 0$ and K_i is chosen appropriately to match the dimensionality of $\mathbf{M}_i$, Gal et al. [58] show that ELBO maximization is achieved by training with dropout [57]. In the case of convolutional neural networks, dropout is applied after each convolution layer (with dropout probability d)[7]. In Eq. (5), $\mathbf{M}_i$ is the variational parameter which is optimized during training. Evaluation of the predictive distribution $p(\mathbf{y}'|\mathbf{x}', \mathbf{X}, \mathbf{Y})$ can then be approximated via Monte-Carlo integration, i.e.,

$$p(\mathbf{y}'|\mathbf{x}', \mathbf{X}, \mathbf{Y}) \approx \frac{1}{T} \sum_{t=1}^{T} \hat{f}(\mathbf{x}', \hat{\mathbf{w}}) \ . \quad (6)$$

In detail, this corresponds to averaging the output of T forward passes through the network with dropout *enabled*. Note that $\hat{f}$ and $\hat{\mathbf{w}}$ now correspond to random variables, as dropout means that we sample, in each forward pass, which connections are dropped. In our implementation, we add dropout layers after all convolutional layers except for those used as pooling/unpooling layers (which are considered non-linearities applied to the weight matrices [58]), as well as the final convolution layer in the decoder, which generates the predicted momentum. We train the network using stochastic gradient descent (SGD).

Network evaluation. For testing, we keep the dropout layers enabled to maintain the probabilistic property of the network, and sample the network to obtain multiple momentum predictions for one moving/target image pair. We then choose the sample mean as the prediction result, see Eq. (6), and perform LDDMM shooting using all the samples to generate multiple deformation fields. The local variance of these deformation fields can then be used as an uncertainty estimate of the predicted deformation field. When selecting the dropout probability, d, a probability of 0.5 would provide the largest variance, but may also enforce too much regularity for a convolutional network, especially in our case where dropout layers are added after every convolution layer. In our experiments, we use a dropout probability of 0.2 (for all dropout units) as a balanced choice.

2.2.3. Patch pruning

As discussed in Sec. 2.2.1, we use a sliding-window approach to predict the deformation parameters (the momenta for `Quicksilver`) patch-by-patch for a whole image. Thus, computation time is proportional to the number of the patches we need to predict. When using a 1-voxel sliding window stride, the number of patches to predict for a

[7] with additional l_2 regularization on the weight matrices of each layer.

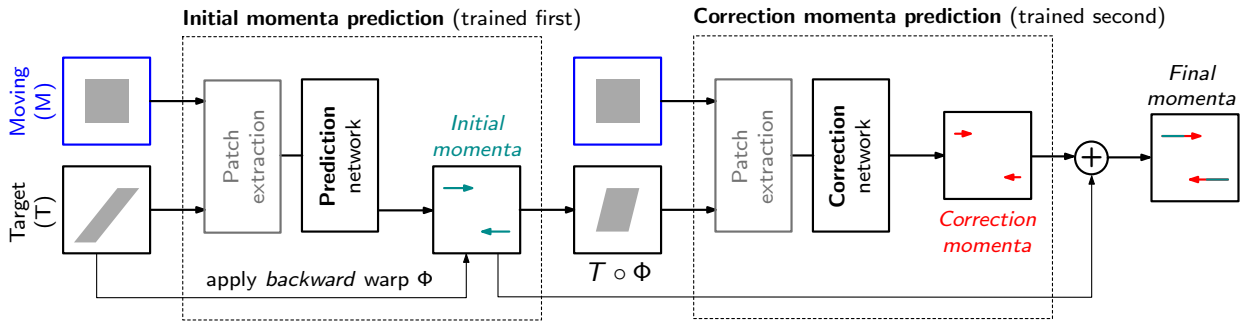


Figure 3: The full prediction + correction architecture for LDDMM momenta. First, a rough prediction of the initial momentum, m_{LP}, is obtained by the prediction network (LP) based on the patches from the unaligned moving image, M and target image, T, respectively. The resulting deformation maps Φ^{-1} and Φ are computed by shooting. Φ is then applied to the target image to warp it to the space of the moving image. A second correction network is then applied to patches from the moving image M and the warped target image $T \circ \Phi$ to predict a correction of the initial momentum, m_C in the space of the moving image, M. The final momentum is then simply the sum of the predicted momenta, $m = m_{LP} + m_C$, which parameterizes a geodesic between the moving image and the target image.

whole image could be substantial. For a typical 3D image of size $128 \times 128 \times 128$ using a $15 \times 15 \times 15$ patch for prediction will require more than 1.4 million patch predictions. Hence, we use two techniques to drastically reduce the number of patches needed for deformation prediction. First, we perform patch pruning by ignoring all patches that belong to the background of both the moving image and the target image. This is justified, because according to LDDMM theory the initial momentum in constant image regions, and hence also in the image background, should be zero. Second, we use a large voxel stride (e.g., 14 for $15 \times 15 \times 15$ patches) for the sliding window operations. This is reasonable for our initial momentum parameterization because of the compact support (at edges) of the initial momentum and the spatial shift invariance we obtain via the pooling/unpooling operations. By using these two techniques, we can reduce the number of predicted patches for one single image dramatically. For example, by 99.995% for 3D brain images of dimension $229 \times 193 \times 193$.

2.3. Correction network

There are two main shortcomings of the deformation prediction network. (i) The complete iterative numerical approach typically used for LDDMM registration is replaced by a *single* prediction step. Hence, it is not possible to recover from any prediction errors. (ii) To facilitate training a network with a small number of images, to make predictions easily parallelizable, and to be able to perform predictions for large 3D image volumes, the prediction network predicts the initial momentum *patch-by-patch*. However, since patches are extracted at the same spatial grid locations from the moving and target images, large deformations may result in drastic appearance changes between a source and a target patch. In the extreme case, corresponding image information may no longer be found for a given source and target patch pair. This may happen, for example, when a small patch-size encounters a large deformation. While using larger patches would be an option (in the extreme case the entire image would be represented by one patch), this would require a network with substantially larger capacity (to store the information for larger image patches and all meaningful deformations) and would also likely require much larger training datasets[8].

To address these shortcomings, we propose a two-step prediction approach to improve overall prediction accuracy. The first step is our already described prediction network. We refer to the second step as the *correction network*. The task of the correction network is to compensate for prediction errors of the first prediction step. The idea is grounded in two observations: The first observation is that patch-based prediction is accurate when the deformation inside the patch is small. This is sensible as the initial momentum is concentrated along the edges, small deformations are commonly seen in training images, and less deformation results in less drastic momentum values. Hence, more accurate predictions are expected for smaller deformations. Our second observation is that, given the initial momentum, we are able to generate the whole geodesic path using the geodesic shooting equations. Hence, we can generate two deformation maps: the forward warp Φ^{-1} that maps the moving image to the coordinates of the target image, and the backward warp Φ mapping the target image back to the coordinates of the moving image. Hence, after the first prediction step using our prediction network, we can warp the target image back to the moving image M via $T \circ \Phi$. We can then train the *correction network* based on the difference between the moving image M and the warped-back target image $T \circ \Phi$,

[8] In fact, we have successfully trained prediction models with as little as ten images using all combinations of pair-wise registrations to create training data [36]. This is possible, because even in such a case of severely limited training data the number of *patches* that can be used for training is very large.

such that it makes adjustments to the initial momentum predicted in the first step by our prediction network. Because M and $T \circ \Phi$ are in the same coordinate system, the differences between these two images are small as long as the predicted deformation is reasonable, and more accurate predictions can be expected. Furthermore, the correction for the initial momentum is then performed in the original coordinate space (of the moving image) which allows us to obtain an overall corrected initial momentum, m_0. This is for example a useful property when the goal is to do statistics with respect to a fixed coordinate system, for example, an atlas coordinate system.

Fig. 3 shows a graphical illustration of the resulting two-step prediction framework. In the framework, the correction network has the same structure as the prediction network, and the only difference is the input of the networks and the output they produce. Training the overall framework is done sequentially:

1. Train the prediction network using training images and the ground truth initial momentum obtained by numerical optimization of the LDDMM registration model.

2. Use the *predicted* momentum from the prediction network to generate deformation fields to warp the target images in the training dataset back to the space of the moving images.

3. Use the moving images and the warped-back target images to train the correction network. The correction network learns to predict the *difference* between the ground truth momentum and the predicted momentum from the prediction network.

Using the framework during testing is similar to the training procedure, except here the outputs from the prediction network (using moving and target images as input) and the correction network (using moving and warped-back target images as input) are summed up to obtain the final predicted initial momentum. This summation is justified from the LDDMM theory as it is performed in a fixed coordinate system (a fixed tangent space), which is the coordinate system of the moving image. Experiments show that our prediction+correction approach results in lower training and testing error compared with only using a prediction network, as shown in Sec. 2.4 and Sec. 3.

2.4. Datasets / Setup

We evaluate our method using three 3D brain image registration experiments. The first experiment is designed to assess *atlas-to-image* registration. In this experiment, the moving image is always the atlas image. The second experiment addresses general *image-to-image* registration. The final experiment explores *multi-modal image* registration; specifically, the registration of T1-weighted (T1w) and T2-weighted (T2w) magnetic resonance images.

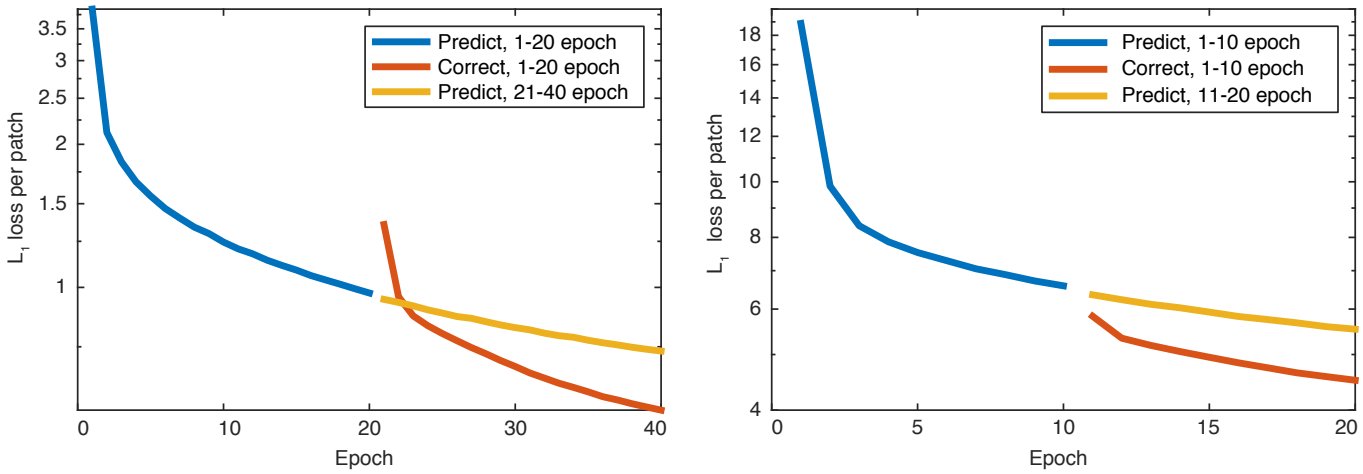
For the *atlas-to-image* registration experiment, we use 3D image volumes from the OASIS longitudinal dataset [60]. Specifically, we use the first scan of all subjects, resulting in 150 brain images. We select the first 100 images as our training target images and the remaining 50 as our test target images. We create an unbiased atlas [61] from all training data using `PyCA`[9] [50, 62], and use the atlas as the moving image. We use the LDDMM shooting algorithm to register the atlas image to all 150 OASIS images. The obtained initial momenta from the training data are used to train our network; the remaining momenta are used for validation.

For the *image-to-image* registration experiment, we use all 373 images from the OASIS longitudinal dataset as the training data, and randomly select target images from different subjects for every image, creating 373 registrations for the training of our prediction and correction networks. For testing, we choose the four datasets (`LPBA40`, `IBSR18`, `MGH10`, `CUMC12`) evaluated in [33]. We perform LDDMM shooting for all training registrations, and follow the evaluation procedure described in [33] to perform pairwise registrations within all datasets, resulting in a total of 2168 registration (1560 from `LPBA40`, 306 from `IBSR18`, 90 from `MGH10`, 132 from `CUMC12`) test cases.

For the *multi-modal* registration experiment, we use the IBIS 3D Autism Brain image dataset [34]. This dataset contains 375 T1w/T2w brain images from 2 years old subjects. We select 359 of the images for training and use the remaining 16 images for testing. For training, we randomly select T1w-T1w image pairs and perform LDDMM shooting to generate the optimization momenta. *We then train the prediction and correction networks to predict the momenta obtained from LDDMM T1w-T1w optimization using the image patches from the corresponding T1w moving image and T2w target image as network inputs.* For testing, we perform pair-wise T1w-T2w registrations for all 16 test images, resulting in 250 test cases. For comparison, we also train a T1w-T1w prediction+correction network that performs prediction on the T1w-T1w test cases. This network acts as the "upper-bound" of the potential performance of our multi-modal networks as it addresses the uni-modal registration case and hence operates on image pairs which have very similar appearance. Furthermore, to test prediction performance when using very limited training data, we also train a multi-modal prediction network and a multi-modal prediction+correction network using only 10 of the 365 training images which are randomly chosen for training. In particular, we perform pair-wise T1w-T1w registration on the 10 images, resulting in 90 registration pairs. We then use these 90 registration cases to train the multi-modal prediction networks.

For skull stripping, we use FreeSurfer [63] for the OASIS dataset and AutoSeg [64] for the IBIS dataset. The

[9] https://bitbucket.org/scicompanat/pyca



(a) `Atlas-to-Image` (b) `Image-to-Image`

Figure 4: Log_{10} plot of l_1 training loss per patch. The loss is averaged across all iterations for every epoch for both the Atlas-to-Image case and the Image-to-Image case. The combined prediction + correction networks obtain a lower loss per patch than the loss obtained by simply training the prediction networks for more epochs.

4 evaluation datasets for image-to-image experiment are already skull stripped as described in [33]. All images used in our experiments are first affinely registered to the ICBM MNI152 nonlinear atlas [65] using `NiftyReg`[10] and intensity normalized via histogram equalization prior to atlas building and LDDMM registration. All 3D volumes are of size $229 \times 193 \times 193$ except for the LPBA dataset ($229 \times 193 \times 229$), where we add additional blank image voxels for the atlas to keep the cerebellum structure. LDDMM registration is done using `PyCA`[11] [50] with SSD as the image similarity measure. We set the parameters for the regularizer of LDDMM[12] to $L = -a\nabla^2 - b\nabla(\nabla\cdot) + c$ as $[a, b, c] = [0.01, 0.01, 0.001]$, and σ in Eqn. 3 to 0.2. We use a $15 \times 15 \times 15$ patch size for deformation prediction in all cases, and use a sliding window with step-size 14 to extract patches for training. The only exception is for the multi-modal network which is trained using only 10 images, where we choose a step-size of 10 to generate more training patches. Note that using a stride of 14 during training means that we are in fact discarding available training patches to allow for reasonable network training times. However, we still retain a very large number of patches for training. To check that our number of patches for training is sufficient, we performed additional experiments for the image-to-image registration task using smaller strides when selecting training patches. Specifically, we doubled and tripled the training size for the prediction network. These experiments indicated that increasing the training data size further only results in marginal improvements, which are clearly outperformed by a combined prediction + correction strategy. Exploring alternative network structures, which may be able to utilize larger training datasets, is beyond the scope of this paper, but would be an interesting topic for future research.

The network is implemented in `PyTorch`[13], and optimized using `Adam` [67]. We set the learning rate to 0.0001 and keep the remaining parameters at their default values. We train the prediction network for 10 epochs for the image-to-image registration experiment and the multi-modal image registration experiment, and 20 epochs for the atlas-to-image experiment. The correction networks are trained using the same number of epochs as their corresponding prediction networks. Fig. 4 shows the l_1 training loss per patch averaged for every epoch for the atlas-to-image and the image-to-image experiments. For both, using a correction network in conjunction with a prediction network results in lower training error compared with training the prediction network for more epochs.

3. Results

3.1. Atlas-to-Image registration

For the atlas-to-image registration experiment, we test two different sliding window strides for our patch-based prediction method: stride = 5 and stride = 14. We trained additional prediction networks predicting the initial velocity $v_0 = Km_0$ and the displacement field $\Phi(1) - \text{id}$ of

[10] https://cmiclab.cs.ucl.ac.uk/mmodat/niftyreg

[11] https://bitbucket.org/scicompanat/pyca

[12] This regularizer is too weak to assure a diffeomorphic transformation based on the *sufficient* regularity conditions discussed in [13]. For these conditions to hold in 3D, L would need to be at least a differential operator of order 6. However, as long as the obtained velocity fields v are finite over the unit interval, i.e., $\int_0^1 \|v\|_L^2 \, dt < \infty$ for an L of at least order 6, we will obtain a diffeomorphic transform [51]. In the discrete setting, this condition will be fulfilled for finite velocity fields. To side-step this issue, models based on Gaussian or multi-Gaussian kernels [66] could also be used instead.

[13] https://github.com/pytorch/pytorch

	Deformation Error for each voxel [mm]							$\det J > 0$
Data percentile for all voxels	0.3%	5%	25%	50%	75%	95%	99.7%	
`Affine`	0.0613	0.2520	0.6896	1.1911	1.8743	3.1413	5.3661	N/A
`D`, velocity, stride 5	0.0237	0.0709	0.1601	0.2626	0.4117	0.7336	1.5166	100%
`D`, velocity, stride 14	0.0254	0.075	0.1675	0.2703	0.415	0.743	1.5598	100%
`D`, deformation, stride 5	0.0223	0.0665	0.1549	0.2614	0.4119	0.7388	1.5845	56%
`D`, deformation, stride 14	0.0242	0.0721	0.1671	0.2772	0.4337	0.7932	1.6805	0%
`P`, momentum, stride 14, 50 samples	0.0166	0.0479	0.1054	0.1678	0.2546	0.4537	1.1049	100%
`D`, momentum, stride 5	0.0129	0.0376	0.0884	0.1534	0.2506	0.4716	1.1095	100%
`D`, momentum, stride 14	0.013	0.0372	0.0834	0.1359	0.2112	0.3902	0.9433	100%
`D`, momentum, stride 14, 40 epochs	0.0119	0.0351	0.0793	0.1309	0.2070	0.3924	0.9542	100%
`D`, momentum, stride 14 + correction	**0.0104**	**0.0309**	**0.0704**	**0.1167**	**0.185**	**0.3478**	**0.841**	100%

Table 1: Test result for *atlas-to-image* registration. The table shows the distribution of the 2-norm of the deformation error of the predicted deformation with respect to the deformation obtained by numerical optimization. Percentiles of the displacement errors are shown to provide a complete picture of the error distribution over just reporting the mean or median errors over all voxels within the brain mask in the dataset. `D`: deterministic network; `P`: probabilistic network; stride: stride length of the sliding window for whole image prediction; velocity: predicting initial velocity; deformation: predicting the deformation field; momentum: predicting the initial momentum; correction: using the correction network. The $\det J > 0$ column shows the ratio of test cases with only positive-definite determinants of the Jacobian of the deformation map to the overall number of registrations (100% indicates that all registration results were diffeomorphic). Our initial momentum networks are highlighted in **bold**. The best results are also highlighted in **bold**.

LDDMM to show the effect of different deformation parameterizations on deformation prediction accuracy. We generate the predicted deformation map by integrating the shooting equation 4 for the initial momentum and the initial velocity parameterization respectively. For the displacement parameterization we can directly read-off the map from the network output. We quantify the deformation errors per voxel using the voxel-wise two-norm of the deformation error with respect to the result obtained via numerical optimization for LDDMM using `PyCA`. Table 1 shows the error percentiles over all voxels and test cases.

We observe that the initial momentum network has better prediction accuracy compared to the results obtained via the initial velocity and displacement parameterization in both the 5-stride and 14-stride cases. This validates our hypothesis that momentum-based LDDMM is better suited for patch-wise deformation prediction. We also observe that the momentum prediction result using a smaller sliding window stride is slightly worse than the one using a stride of 14. This is likely the case, because in the atlas-to-image setting, the number of atlas patches that extract features from the atlas image is very limited, and using a stride of 14 during the training phase further reduces the available data from the atlas image. Thus, during testing, the encoder will perform very well for the 14-stride test cases since it has already seen all the input atlas patches during training. For a stride of 5 however, unseen atlas patches will be input to the network, resulting in reduced registration accuracy[14]. In contrast, the velocity and the displacement parameterizations result in slightly better predictions for smaller sliding window strides. That this is not the case for the momentum parameterization suggests that it is easier for the network to learn to predict the momentum, as it indeed has become more specialized to the training data which was obtained with a stride of 14. One of the important properties of LDDMM shooting is its ability to generate diffeomorphic deformations. To assess this property, we calculate the local Jacobians of the resulting deformation maps. Assuming no flips of the entire coordinate system, a diffeomorphic deformation map should have positive Jacobian determinants everywhere, otherwise foldings occur in the deformation maps. We calculate the ratio of test cases with positive Jacobian determinants of the deformation maps to all test cases, shown as $\det J > 0$ in Table 1. We observe that the initial momentum and the initial velocity networks indeed generate diffeomorphic deformations in all scenarios. However, the deformation accuracy is significantly worse for the initial velocity network. Predicting the displacement directly cannot guarantee diffeomorphic deformations even for a small stride. This is unsurprising as, similar to existing optical flow approaches [14, 15], directly predicting displacements does not encode deformation smoothness. Hence, the initial momentum parameterization is the preferred choice among our three tested parameterizations as it achieves the best prediction accuracy and guarantees diffeomorphic deformations. Furthermore, the initial momentum prediction including the correction network with a stride of 14 achieves the best registration accuracy overall among the tested methods, even outperforming the prediction network alone trained with more training iterations (**`D`, stride 14, 40 epochs**). This demonstrates that the correction network is capable of improving the initial momentum prediction beyond the capabilities of the original prediction network.

[14]This behavior could likely be avoided by randomly sampling patch locations during training instead of using a regular grid. However, since we aim at reducing the number of predicted patches we did not explore this option and instead maintained the regular grid sampling.

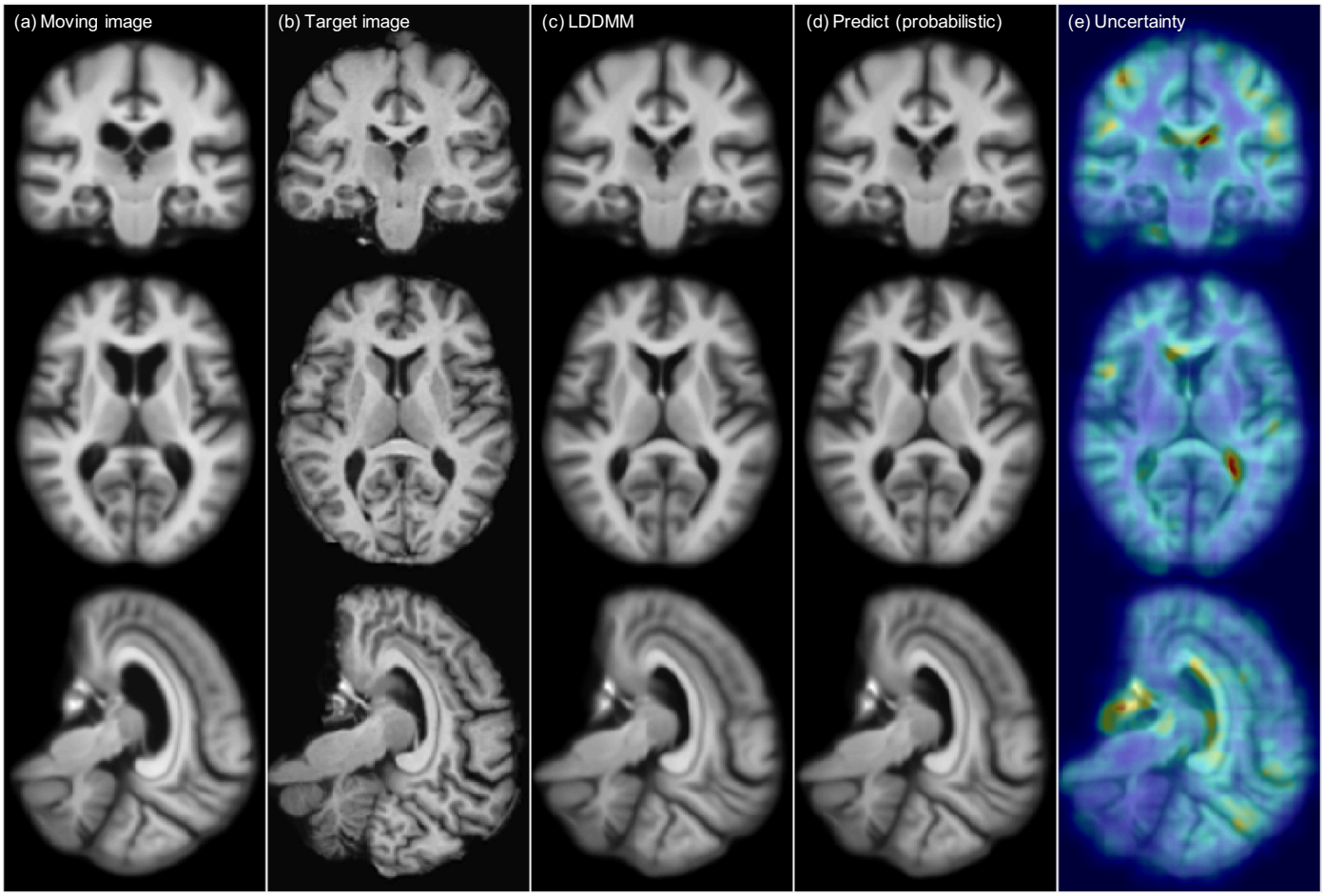


Figure 5: Atlas-to-image registration example. From *left* to *right*: (a): moving (atlas) image; (b): target image; (c): deformation from optimizing LDDMM energy; (d): deformation from using the mean of 50 samples from the probabilistic network with stride=14 and patch pruning; (e): the uncertainty map as square root of the sum of the variances of the deformation in x, y, and z directions mapped onto the predicted deformation result. The coloring indicates the level of uncertainty, with red = high uncertainty and blue = low uncertainty. Best-viewed in color.

Fig. 5 shows one example atlas-to-image registration case. The predicted deformation result is very similar to the deformation from LDDMM optimization. We compute the square root of the sum of the variance of the deformation in the x, y and z directions to quantify deformation uncertainty, and visualize it on the rightmost column of the figure. The uncertainty map shows high uncertainty along the ventricle areas where drastic deformations occur, as shown in the moving and target images.

3.2. Image-to-Image registration

In this experiment, we use a sliding window stride of 14 for both the prediction network and the correction network during evaluation. We mainly compare the following three LDDMM-based -methods: (i) the numerical LDDMM optimization approach (`LO`) as implemented in `PyCA`, which acts as an upper bound on the performance of our prediction methods; and two flavors of `Quicksilver`: (ii) only the prediction network (`LP`) and (iii) the prediction+correction network (`LPC`). Example registration cases are shown in Fig. 9.

3.2.1. LDDMM energy

To test the ability of our prediction networks to replace numerical optimization, we compare the LDDMM energies obtained using optimization from `LO` with the energies corresponding to the predicted momenta from `LP` and `LPC`. Low energies for the predicted momenta, which are comparable to the energies obtained by numerical optimization (`LO`), would suggest that our prediction models can indeed act as replacements for numerical optimization. However, note that, in general, a low energy will only imply a good registration result if the registration model is fully appropriate for the registration task. Ultimately, registration quality should be assessed based on a particular task: most directly by measuring landmark errors or (slightly more indirectly) by measuring overlaps of corresponding regions as done in Section 3.2.2. Note also that our networks for image-to-image registration are trained on the OASIS dataset. Hence, improved results may be

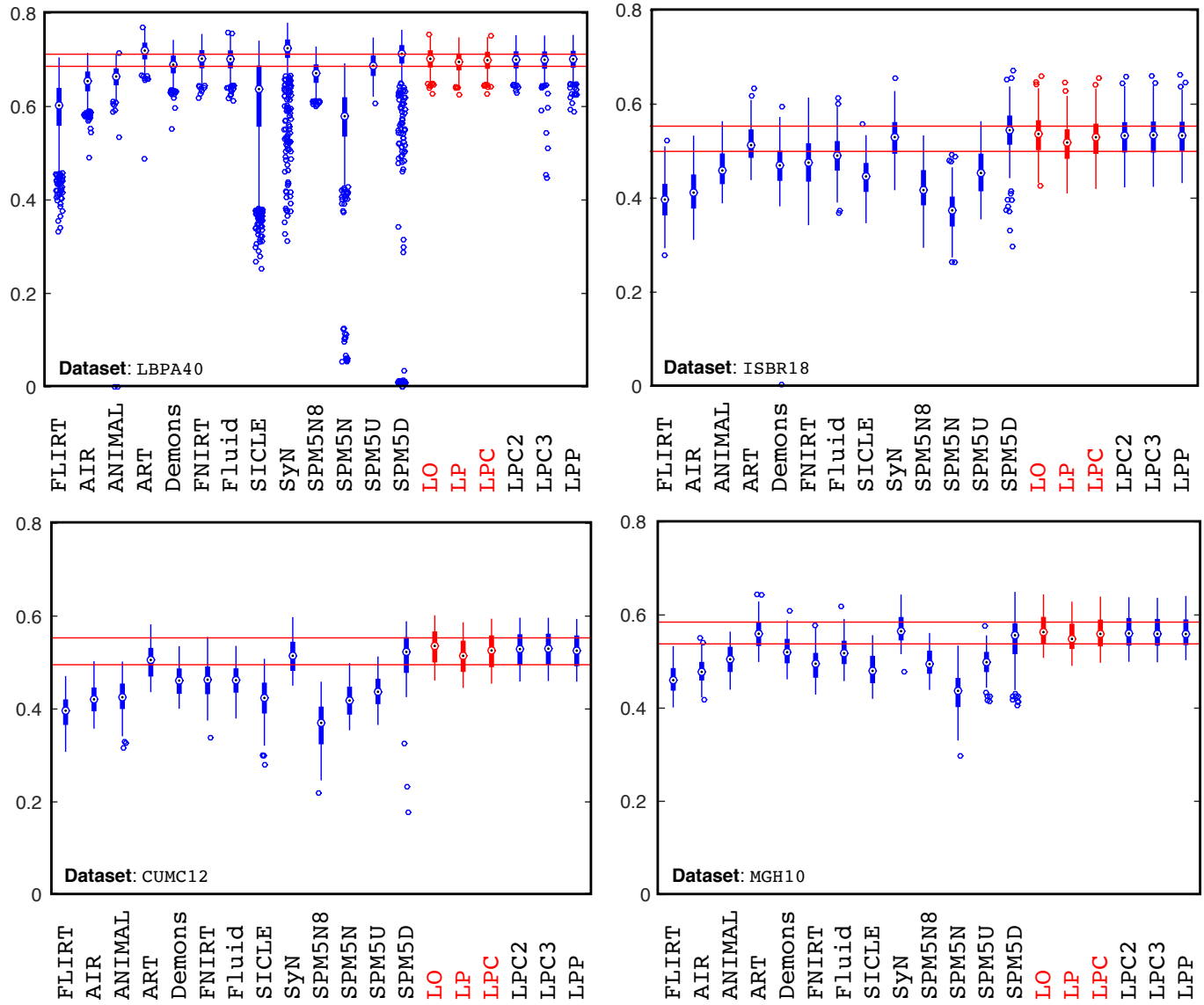


Figure 6: Overlap by registration method for the *image-to-image* registration case. The boxplots illustrate the mean target overlap measures averaged over all subjects in each label set, where mean target overlap is the average of the fraction of the target region overlapping with the registered moving region over all labels. The proposed LDDMM-based methods in this paper are highlighted in red. `LO` = LDDMM optimization; `LP` = prediction network; `LPC` = prediction network + correction network. `LPP`: prediction network + using the prediction network for correction. `LPC2/LPC3`: prediction network + iteratively using the correction network 2/3 times. Horizontal red lines show the `LPC` performance in the lower quartile to upper quartile (best-viewed in color). The medians of the overlapping scores for [`LPBA40`, `IBSR18`, `CUMC12`, `MGH10`] for `LO`, `LP` and `LPC` are: `LO`: [0.702, 0.537, 0.536, 0.563]; `LP`: [0.696, 0.518, 0.515, 0.549]; `LPC`: [0.702, 0.533, 0.526, 0.559]. Best-viewed in color.

achievable by training dataset specific models. Table 2 shows the results for the four test datasets. Compared with the initial LDDMM energy based on affine registration to the atlas space in the `initial` column, both `LP` and `LPC` have drastically lower LDDMM energy values; further, these values are only slightly higher than those for `LO`. Furthermore, compared with `LP`, `LPC` generates LDDMM energy values that are closer to `LO`, which indicates that using the prediction+correction approach results in momenta which are closer to the optimal solution than the ones obtained by using the prediction network only.

3.2.2. Label overlap

For image-to-image registration we follow the approach in [33] and calculate the target overlap (TO) of labeled brain regions after registration: $TO = \frac{|l_m \cap l_t|}{|l_t|}$, where l_m and l_t indicate the corresponding labels for the moving image (after registration) and the target image. We then evaluate the mean of the target overlap averaged first across all labels for every registration case. The evaluation results for other methods tested in [33] are available online. We compare our registration approaches to these results. An interesting question is if the prediction network and the

LDDMM energy for image-to-image test datasets			
LPBA40			
initial	LO	LP	LPC
0.120 ± 0.013	0.027 ± 0.004	0.036 ± 0.005	0.030 ± 0.005
IBSR18			
initial	LO	LP	LPC
0.214 ± 0.032	0.037 ± 0.008	0.058 ± 0.013	0.047 ± 0.011
CUMC12			
initial	LO	LP	LPC
0.246 ± 0.015	0.044 ± 0.003	0.071 ± 0.004	0.056 ± 0.004
MGH10			
initial	LO	LP	LPC
0.217 ± 0.012	0.039 ± 0.003	0.062 ± 0.004	0.049 ± 0.003

Table 2: Mean and standard deviation of the LDDMM energy for four image-to-image test datasets. `initial`: the initial LDDMM energy between the original moving image and the target image after affine registration to the atlas space, i.e. the original image matching energy. `LO`: LDDMM optimization. `LP`: prediction network. `LPC`: prediction+correction network.

correction network are identical, and whether the prediction network can be used in the correction step. Another question is if the correction network can be applied multiple times in the correction step to further improve results. Thus, to test the usefulness of the correction network in greater depth, we also create three additional formulations of our prediction framework: (i) prediction network + using the same prediction network to replace the correction network in the correction step (`LPP`); (ii) applying the correction network twice (`LPC2`) and (iii) applying the correction network three times (`LPC3`).

Fig. 6 shows the evaluation results. Several points should be noted: first, the LDDMM optimization performance is on par with `SyN` [68], `ART` [69] and the SPM5 DARTEL Toolbox (`SPM5D`) [70]. This is reasonable as these methods are all non-parametric diffeomorphic or homeomorphic registration methods, allowing the modeling of large deformations between image pairs. Second, using only the prediction network results in a slight performance drop compared to the numerical optimization results (`LO`), but the result is still competitive with the top-performing registration methods. Furthermore, also using the correction network boosts the deformation accuracy nearly to the same level as the LDDMM optimization approach (`LO`). The red horizontal lines in Fig. 6 show the lower and upper quartiles of the target overlap score of the prediction+correction method. Compared with other methods, our prediction+correction network achieves top-tier performance for label matching accuracy at a small fraction of the computational cost. Lastly, in contrast to many of the other methods `Quicksilver` produces virtually no outliers. One can speculate that this may be the benefit of learning to predict deformations from a large *population* of data, which may result in a prediction model which conservatively rejects unusual deformations. Note that such a population-based approach is very different from most existing registration methods which constrain deformations based on a regularizer chosen for a mathematical registration model. Ultimately, a deformation model for image registration should model what deformations are expected. Our population-based approach is a step in this direction, but, of course, still depends on a chosen regularizer to generate training data. Ideally, this regularizer itself should be learned from data.

An interesting discovery is that `LPP`, `LPC2` and `LPC3` produce label overlapping scores that are on-par with `LPC`. However, as we will show in Sec. 3.2.3, `LPP`, `LPC2` and `LPC3` deviate from our goal of predicting deformations that are similar to the LDDMM optimization result (`LO`). In fact, they produce more drastic deformations that can lead to worse label overlap and even numerical stability problems. These problems can be observed in the `LPBA40` results shown in Fig. 6, which show more outliers with low overlapping scores for `LPP` and `LPC3`. In fact, there are 12 cases for `LPP` where the predicted momentum cannot generate deformation fields via LDDMM shooting using `PyCA`, due to problems related to numerical integration. These cases are therefore not included in Fig. 6. `PyCA` uses an explicit Runge-Kutta method (RK4) for time-integration. Hence, numerical instability is likely due to the use of a fixed step size for this time-integration which is small enough for the deformations expected to occur for these brain registration tasks, but which may be too large for the more extreme momenta `LPP` and `LPC3` create for some of these cases. Using a smaller step-size would regain numerical stability in this case.

To study the differences among registration algorithms *statistically*, we performed paired t-tests[15] with respect to the target overlap scores between our LDDMM variants (`LO`, `LP`, `LPC`) and the methods in [33]. Our null-hypothesis is that the methods show the same target overlap scores. We use a significance level of $\alpha = 0.05/204$ for rejection of this null-hypothesis. We also computed the mean and the standard deviation of pair-wise differences between our LDDMM variants and these other methods. Table 3 shows the results. We observe that direct numerical optimization of the shooting LDDMM formulation via `PyCA` (`LO`) is a highly competitive registration method and shows better target overlap scores than most of the other registration algorithms for all four datasets (`LPBA40`, `IBSR18`, `CUMC12`, and `MGH10`). Notable exceptions are `ART` (on `LPBA40`), `SyN` (on `LBPA40`), and `SPM5D` (on `IBSR18`). However, performance decreases are generally very small: -0.017, -0.013, and -0.009 mean decrease in target overlap ratio for the three aforementioned exceptions, respectively. Specifically,

[15] To safe-guard against overly optimistic results due to multiple comparisons, we used Bonferroni correction for all statistical tests in the paper (paired t-tests and paired TOST) by dividing the significance level α by the total number (204) of statistical tests we performed. This resulted in an effective significance level $\alpha = 0.05/204 \approx 0.00025$. The Bonferroni correction is likely overly strict for our experiments as the different registration results will be highly correlated, because they are based on the same input data.

Dataset: LPBA40								
	FLIRT	AIR	ANIMAL	ART	Demons	FNIRT	Fluid	SLICE
LO	0.108 ± 0.054	0.049 ± 0.021	0.039 ± 0.029	−0.017±0.013	0.012 ± 0.014	0.001 ± 0.014	0.001 ± 0.013	0.097 ± 0.1
LP	0.102 ± 0.054	0.043 ± 0.02	0.033 ± 0.029	−0.024±0.013	0.006 ± 0.014	−0.006±0.014	−0.005±0.013	0.091 ± 0.1
LPC	0.106 ± 0.054	0.046 ± 0.021	0.037 ± 0.029	−0.02 ± 0.013	0.009 ± 0.014	−0.002±0.014	−0.002±0.013	0.095 ± 0.1
	SyN	SPM5N8	SPM5N	SPM5U	SPM5D	LO	LP	LPC
LO	−0.013 ± 0.05	0.032 ± 0.018	0.13 ± 0.07	0.015 ± 0.017	0.03 ± 0.16	N/A	0.006 ± 0.003	0.003 ± 0.002
LP	−0.02 ± 0.05	0.025 ± 0.018	0.124 ± 0.07	0.009 ± 0.017	0.023 ± 0.16	−0.006±0.003	N/A	−0.004±0.002
LPC	−0.016 ± 0.05	0.029 ± 0.018	0.127 ± 0.07	0.012 ± 0.017	0.027 ± 0.16	−0.003±0.002	0.004 ± 0.002	N/A

Dataset: IBSR18								
	FLIRT	AIR	ANIMAL	ART	Demons	FNIRT	Fluid	SLICE
LO	0.136 ± 0.025	0.119 ± 0.03	0.07 ± 0.027	0.018 ± 0.022	0.064 ± 0.034	0.057 ± 0.026	0.044 ± 0.019	0.088 ± 0.029
LP	0.118 ± 0.022	0.101 ± 0.028	0.052 ± 0.025	0 ± 0.021	0.047 ± 0.032	0.039 ± 0.023	0.026 ± 0.018	0.07 ± 0.027
LPC	0.129 ± 0.024	0.112 ± 0.03	0.063 ± 0.027	0.01 ± 0.022	0.058 ± 0.033	0.049 ± 0.025	0.036 ± 0.019	0.08 ± 0.029
	SyN	SPM5N8	SPM5N	SPM5U	SPM5D	LO	LP	LPC
LO	0.005 ± 0.024	0.112 ± 0.034	0.161 ± 0.042	0.08 ± 0.030	−0.009±0.035	N/A	0.018 ± 0.007	0.007 ± 0.004
LP	−0.013±0.024	0.094 ± 0.032	0.144 ± 0.042	0.062 ± 0.027	−0.026±0.035	−0.018±0.007	N/A	−0.01 ± 0.004
LPC	−0.002±0.024	0.105 ± 0.034	0.154 ± 0.043	0.073 ± 0.029	−0.016±0.035	−0.007±0.004	0.01 ± 0.004	N/A

Dataset: CUMC12								
	FLIRT	AIR	ANIMAL	ART	Demons	FNIRT	Fluid	SLICE
LO	0.14 ± 0.02	0.111 ± 0.019	0.108 ± 0.031	0.031 ± 0.01	0.072 ± 0.012	0.071 ± 0.019	0.073 ± 0.017	0.115 ± 0.03
LP	0.12 ± 0.017	0.092 ± 0.017	0.089 ± 0.031	0.012 ± 0.01	0.052 ± 0.01	0.052 ± 0.017	0.053 ± 0.015	0.096 ± 0.031
LPC	0.131 ± 0.018	0.102 ± 0.018	0.1 ± 0.031	0.023 ± 0.01	0.063 ± 0.011	0.062 ± 0.018	0.064 ± 0.016	0.107 ± 0.031
	SyN	SPM5N8	SPM5N	SPM5U	SPM5D	LO	LP	LPC
LO	0.020 ± 0.011	0.169 ± 0.029	0.114 ± 0.019	0.1 ± 0.015	0.022 ± 0.049	N/A	0.02 ± 0.004	0.009 ± 0.002
LP	0.001 ± 0.011	0.149 ± 0.028	0.095 ± 0.017	0.076 ± 0.013	0.003 ± 0.048	−0.02 ± 0.004	N/A	−0.011±0.003
LPC	0.012 ± 0.011	0.16 ± 0.028	0.106 ± 0.018	0.087 ± 0.013	0.013 ± 0.048	0.009 ± 0.002	0.011 ± 0.003	N/A

Dataset: MGH10								
	FLIRT	AIR	ANIMAL	ART	Demons	FNIRT	Fluid	SLICE
LO	0.104 ± 0.016	0.087 ± 0.015	0.062 ± 0.022	0.005 ± 0.016	0.044 ± 0.013	0.071 ± 0.018	0.043 ± 0.016	0.083 ± 0.017
LP	0.091 ± 0.016	0.073 ± 0.016	0.049 ± 0.023	−0.008±0.017	0.03 ± 0.013	0.058 ± 0.018	0.03 ± 0.015	0.07 ± 0.017
LPC	0.098 ± 0.016	0.081 ± 0.015	0.057 ± 0.022	0 ± 0.016	0.038 ± 0.013	0.065 ± 0.018	0.037 ± 0.016	0.077 ± 0.017
	SyN	SPM5N8	SPM5N	SPM5U	SPM5D	LO	LP	LPC
LO	−0.002±0.015	0.069 ± 0.02	0.135 ± 0.041	0.07 ± 0.024	0.023 ± 0.047	N/A	0.013 ± 0.004	0.006 ± 0.002
LP	−0.015±0.016	0.055 ± 0.02	0.121 ± 0.041	0.057 ± 0.025	0.01 ± 0.048	−0.013±0.004	N/A	−0.008±0.003
LPC	−0.008±0.015	0.063 ± 0.02	0.129 ± 0.041	0.065 ± 0.024	0.018 ± 0.047	−0.006±0.002	0.008 ± 0.003	N/A

Table 3: Mean and standard deviation of the difference of target overlap score between LDDMM variants (LDDMM optimization (LO), the proposed prediction network (LP) and prediction+correction network (LPC)) and all other methods for the *image-to-image* experiments. The cell coloring indicates significant differences calculated from a pair-wise t-test: green indicates that the row-method is statistically significantly *better* than the column-method; red indicates that the row-method is statistically significantly *worse* than the column-method, and blue indicates the difference is not statistically significant (best-viewed in color). We use Bonferroni correction to safe-guard against spurious results due to multiple comparisons by dividing the significance level α by 204 (the total number of statistical tests). The significance level for rejection of the null-hypothesis is $\alpha = 0.05/204$. Best-viewed in color.

Dataset: LPBA40					
	ART	SyN	LO	LP	LPC
LO			N/A	✓	✓
LP			✓	N/A	✓
LPC			✓	✓	N/A

Dataset: CUMC12					
	ART	SyN	LO	LP	LPC
LO			N/A		✓
LP		✓		N/A	
LPC			✓		N/A

Dataset: IBSR18					
	ART	SyN	LO	LP	LPC
LO		✓	N/A		✓
LP	✓			N/A	
LPC		✓	✓		N/A

Dataset: MGH10					
	ART	SyN	LO	LP	LPC
LO		✓	N/A		✓
LP				N/A	✓
LPC	✓		✓	✓	N/A

Table 4: Pairwise TOST, where we test the null-hypothesis that for the target overlap score for each row-method, t_{row}, and the target overlap score for each column-method, t_{column}, $\frac{t_{\text{row}}}{t_{\text{column}}} < 0.98$, or $\frac{t_{\text{row}}}{t_{\text{column}}} > 1.02$. Rejecting the null-hypothesis indicates that the row-method and column-method are statistically equivalent. Equivalence is marked as ✓s in the table. We use Bonferroni correction to safe-guard against spurious results due to multiple comparisons by dividing the significance level α by 204 (the total number of statistical tests). The significance level for rejection of the null-hypothesis is $\alpha = 0.05/204$.

a similar performance of LO to SyN, for example, is expected as SyN (as used in [33]) is based on a *relaxation* formulation of LDDMM, whereas LO is based on the shooting formulation of LDDMM. Performance differences may be due to differences in the used regularizer and the image similarity measure. In particular, where SyN was used with Gaussian smoothing and cross-correlation, we used SSD as the image similarity measure and a regularizer involving up to second order spatial derivatives.

LO is the algorithm that our predictive registration approaches (LP and LPC) are based on. Hence, LP and LPC are not expected to show improved performance with respect

to LO. However, similar performance for LP and LPC would indicate high quality predictions. Indeed, Table 3 shows that our prediction+correction approach (LPC) performs similar (with respect to the other registration methods) to LO. A slight performance drop with respect to LO can be observed for LPC and a slightly bigger performance drop for LP, which only uses the prediction model, but no correction model.

To assess statistical equivalence of the top performing registration algorithms we performed paired two one-sided tests (paired TOST) [71] with a relative threshold difference of 2%. In other words, our null-hypothesis is that methods show a relative difference of larger than 2%. Rejection of this null-hypothesis at a significance level of $\alpha = 0.05/204$ then indicates evidence for statistical equivalence. Table 4 shows the paired TOST results. For a relative threshold difference of 2% LPC can be considered statistically equivalent to LO for all four datasets and to many of the other top methods (e.g., LPC *vs.* SyN on MGH10 and IBSR18).

Overall, these statistical tests confirm that our prediction models, in particular LPC, are highly competitive registration algorithms. Computational cost, however, is very small. This is discussed in detail in Sec. 3.4.

3.2.3. Choosing the correct "correction step"

As shown in Sec. 3.2.2, LPP, LPC2 and LPC3 all result in label overlapping scores which are similar to the label overlapping scores obtained via LPC. This raises the question which method should be preferred for the correction step. Note that among these methods, only LPC is specifically trained to match the LDDMM optimization results and in particular to predict *corrections* to the initial momentum obtained by the prediction model (LP) in the tangent space of the moving image. In contrast, LPP, LPC2 and LPC3 lack this theoretical motivation. Hence, it is unclear for these methods what the overall optimization goal is. To show what this means in practice, we computed the determinant of the Jacobian of the deformation maps (Φ^{-1}) for all voxels for all four registration cases of [33] inside the brain mask and calculated the histogram of the computed values. Our goal is to check the similarity (in distribution) between deformations generated by the prediction models (LP, LPC, LPP, LPC2, LPC3) in comparison to the results obtained via numerical LDDMM optimization (LO).

As an example, Fig. 7 shows the result for the LPBA40 dataset. The other three datasets show similar results. Fig. 7(left) shows the histogram of the logarithmically transformed determinant of the Jacobian ($\log_{10}\det J$) for all the methods. A value of 0 on the x-axis indicates no deformation or a volume preserving deformation, > 0 indicates volumetric shrinkage and < 0 indicates volumetric expansion. We can see that LPC is closest to LO. LP generates smoother deformations compared with LO, which is sensible as one-step predictions will likely not be highly accurate and, in particular, may result in predicted momenta which are slightly smoother than the ones obtained by numerical optimization. Hence, in effect, the predictions may result in a more strongly spatially regularized deformation. LPP, LPC2 and LPC3 generate more drastic deformations (i.e., more spread out histograms indicating areas of stronger expansions and contractions). Fig. 7(right) shows this effect more clearly; it shows the differences between the histogram of the prediction models and the registration result obtained by numerical optimization (LO). Hence, a method which is similar to LO in distribution will show a curve close to $y = 0$.

This assessment also demonstrates that the correction network (of LPC) is different from the prediction network (LP): the correction network is trained specifically to correct *minor errors* in the predicted momenta of the prediction network with respect to the desired momenta obtained by numerical optimization (LO), while the prediction network is not. Thus, LPC is the only model among the prediction models (apart from LP) that has the explicit goal of predicting the behavior of the LDDMM optimization result (LO). When we use the prediction network in the correction step, the high label overlapping scores are due to more drastic deformations compared with LP, but there is no clear theoretical justification of LPP. In fact, it is more reminiscent of a greedy solution strategy, albeit still results in geodesic paths as the predicted momenta are added in the tangent space of the undeformed moving image. Similar arguments hold for LPC2 and LPC3: using the correction network multiple times (iteratively) in the correction step also results in increasingly drastic deformations, as illustrated by the curves for LPC, LPC2 and LPC3 in Fig. 7. Compared to the label overlapping accuracy boost from LP to LPC, LPC2 and LPC3 do not greatly improve the registration accuracy, and may even generate worse results (e.g., LPC3 on LPBA40). Furthermore, the additional computation cost for more iterations of the correction network + LDDMM shooting makes LPC2 and LPC3 less favorable, in comparison to LPC.

3.2.4. Predicting various ranges of deformations

Table 5 shows the range of deformations and associated percentiles for the deformation fields generated by LDDMM optimization for the four *image-to-image* test datasets. All computations were restricted to locations inside the brain mask. Table 5 also shows the means and standard deviations of the differences of deformations between the results for the prediction models and the results obtained by numerical optimization (LO). As shown in the table, the largest deformations that LDDMM optimization generates are 23.393 mm for LPBA40, 36.263 mm for IBSR18, 18.753 mm for CUMC12 and 18.727 mm for MGH10.

Among the prediction models, LPC improves the prediction accuracy compared with LP, and generally achieves the highest deformation prediction accuracy for up to 80% of the voxels. It is also on-par with other prediction models for up to 99% of the voxels, where the largest deformations are in the range between 7.317 mm-9.026 mm for the four

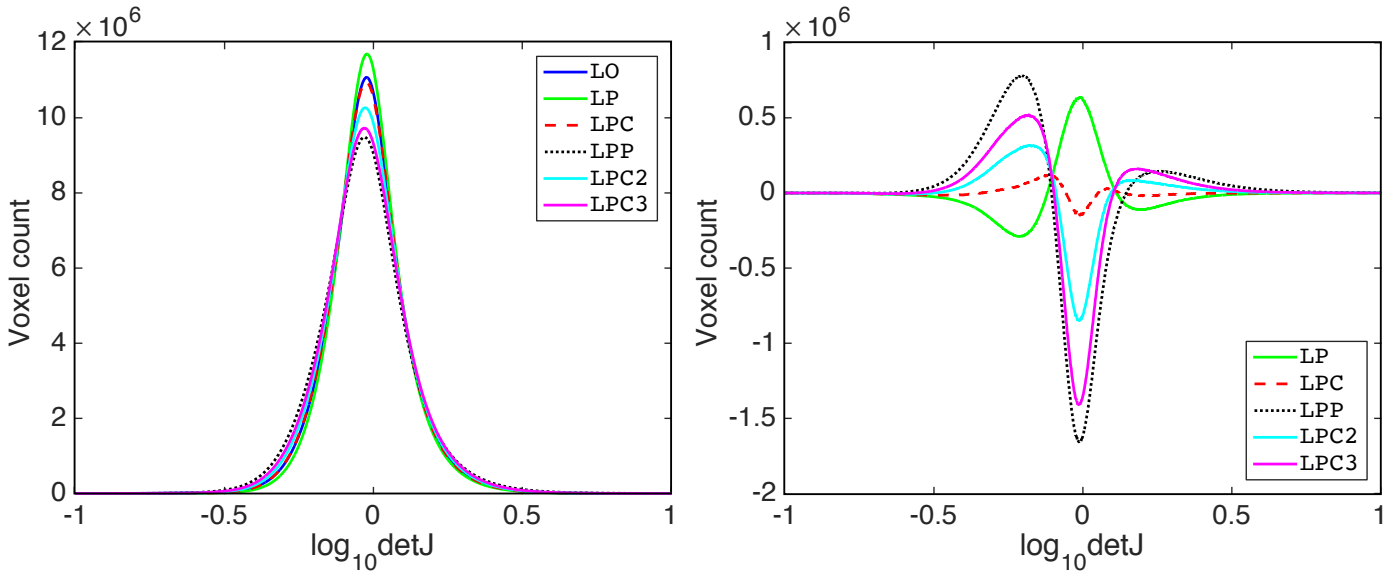


Figure 7: Distribution of the determinant of Jacobian of the deformations for `LPBA40` dataset registrations. *Left*: histograms of the log-transformed determinant of Jacobian for the deformation maps ($\log_{10}\det J$) for all registration cases. *Right*: difference of the histograms of $\log_{10}\det J$ between prediction models (`LP`, `LPC`, `LPP`, `LPC2`, `LPC3`) and `LO`. For the right figure, the closer a curve is to $y = 0$, the more similar the corresponding method is to `LO`. A value of 0 on the x-axis indicates no deformation, or a volume-preserving deformation, > 0 indicates shrinkage and < 0 indicates expansion. Best-viewed in color.

datasets. For very large deformations that occur for 1% of the total voxels, `LPC` does not drastically reduce the deformation error. This is due to the following three reasons: *First*, the input patch size of the deep learning framework is $15 \times 15 \times 15$, which means that the receptive field for the network input is limited to $15 \times 15 \times 15\text{mm}^3$. This constrains the network's ability to predict very large deformations, and can potentially be solved by implementing a multi-scale input network for prediction. *Second*, the deformations in the OASIS training images have a median of 2.609 mm, which is similar to the median observed in the four testing datasets. However, only 0.2% of the voxels in the OASIS training dataset have deformations larger than 10 mm. Such a small number of training patches containing very large deformations makes it difficult to train the network to accurately predict these very large deformations in the test data. If capturing these very large deformations is desired, a possible solution could be to provide a larger number of training examples for large deformations or to weight samples based on their importance. *Third*, outliers in the dataset whose appearances are very different from the other images in the dataset can cause very large deformations. For example, in the `IBSR18` dataset, only three distinct images are needed as moving or target images to cover the 49 registration cases that generate deformations larger than 20 mm. These large deformations created by numerical LDDMM optimization are not always desirable; and consequentially registration errors of the prediction models with respect to the numerical optimization result are in fact sometimes preferred. As a case in point, Fig. 8 shows a registration failure case from the `IBSR18` dataset for LDDMM optimization and the corresponding prediction result. In this example, the brain extraction did not extract consistent anatomy for the moving image and the target image. Specifically, only inconsistent parts of the cerebellum remain between the moving and the target images. As optimization-based LDDMM does not know about this inconsistency, it attempts to match the images as well as possible and thereby creates a very extreme deformation. Our prediction result, however, still generates reasonable deformations (where plausibility is based on the deformations that were observed during training) while matching the brain structures as much as possible. This can be regarded as an *advantage* of our network, where the conservative nature of patch-wise momentum prediction is more likely to generate reasonable deformations.

3.3. Multi-modal image registration

In this task, a sliding window stride of 14 is used for the test cases. Table 6 shows the prediction results compared to the deformation results obtained by T1w-T1w LDDMM optimization. The multi-modal networks (T1w-T2w, `LP`/`LPC`) significantly reduce deformation errors compared to affine registration, and only suffer a slight loss in accuracy compared to their T1w-T1w counterparts. This demonstrates the capability of our network architecture to implicitly learn the complex similarity measure between two modalities. Furthermore, for the networks trained using only 10 images, the performance only decreases slightly in comparison with the T1w-T2w multi-modal networks trained with 359 images. Hence, even when using very limited image data, we can still successfully train our prediction networks when a sufficient number of patches is

Dataset: LPBA40								
Voxel%	0%-10%	10%-20%	20%-30%	30%-40%	40%-50%	50%-60%	60%-70%	70%-80%
# of Test Cases	1560(100%)	1560(100%)	1560(100%)	1560(100%)	1560(100%)	1560(100%)	1560(100%)	1560(100%)
I_{main}	40	40	40	40	40	40	40	40
Deform (mm)	0.001-0.999	0.999-1.400	1.400-1.751	1.751-2.093	2.093-2.450	2.450-2.840	2.840-3.295	3.295-3.873
LP	0.478 ± 0.323	0.564 ± 0.361	0.623 ± 0.393	0.675 ± 0.425	0.728 ± 0.461	0.786 ± 0.503	0.853 ± 0.556	0.941 ± 0.630
LPC	0.415 ± 0.287	0.469 ± 0.327	0.509 ± 0.359	0.546 ± 0.390	0.584 ± 0.424	0.626 ± 0.464	0.676 ± 0.514	0.742 ± 0.583
LPP	0.543 ± 0.554	0.605 ± 0.613	0.650 ± 0.684	0.691 ± 0.823	0.733 ± 0.783	0.778 ± 0.852	0.830 ± 0.922	0.898 ± 1.055
LPC2	0.510 ± 0.339	0.551 ± 0.378	0.582 ± 0.409	0.611 ± 0.439	0.642 ± 0.471	0.676 ± 0.508	0.716 ± 0.553	0.770 ± 0.617
LPC3	0.637 ± 0.410	0.674 ± 0.451	0.703 ± 0.486	0.732 ± 0.516	0.762 ± 0.550	0.795 ± 0.588	0.834 ± 0.638	0.886 ± 0.713
Voxel%	80%-90%	90%-99%	99%-100%	99.9%-100%	99.99%-100%	99.999%-100%	99.9999%-100%	99.99999%-100%
# of Test Cases	1560(100%)	1560(100%)	1560(100%)	1474(94.5%)	417(26.7%)	72(4.6%)	30(1.9%)	8(0.5%)
I_{main}	40	40	40	38	31	8	2	1
Deform (mm)	3.873-4.757	4.757-7.317	7.317-23.393	9.866-23.393	12.435-23.393	14.734-23.393	16.835-23.393	19.090-23.393
LP	1.079 ± 0.752	1.418 ± 1.056	2.579 ± 1.869	4.395 ± 2.667	6.863 ± 3.657	9.220 ± 4.829	11.568 ± 6.340	14.475 ± 7.145
LPC	0.847 ± 0.696	1.101 ± 0.976	1.961 ± 1.761	3.431 ± 2.656	5.855 ± 3.711	8.408 ± 4.780	10.484 ± 6.148	14.041 ± 6.879
LPP	1.001 ± 1.396	1.229 ± 1.558	1.760 ± 2.965	2.514 ± 5.845	4.127 ± 7.988	7.097 ± 3.566	10.509 ± 5.093	11.436 ± 5.688
LPC2	0.856 ± 0.721	1.064 ± 0.979	1.765 ± 1.726	3.012 ± 2.644	5.351 ± 3.698	8.362 ± 4.765	11.310 ± 6.117	15.957 ± 6.458
LPC3	0.968 ± 0.835	1.166 ± 1.176	1.829 ± 2.880	3.019 ± 4.688	5.680 ± 10.450	11.434 ± 23.727	18.939 ± 33.953	24.152 ± 38.466

Dataset: IBSR18								
Voxel%	0%-10%	10%-20%	20%-30%	30%-40%	40%-50%	50%-60%	60%-70%	70%-80%
# of Test Cases	306(100%)	306(100%)	306(100%)	306(100%)	306(100%)	306(100%)	306(100%)	306(100%)
I_{main}	18	18	18	18	18	18	18	18
Deform (mm)	0.003-1.221	1.221-1.691	1.691-2.101	2.101-2.501	2.501-2.915	2.915-3.370	3.370-3.901	3.901-4.580
LP	0.573 ± 0.401	0.670 ± 0.448	0.742 ± 0.489	0.810 ± 0.532	0.879 ± 0.580	0.957 ± 0.637	1.053 ± 0.711	1.182 ± 0.815
LPC	0.450 ± 0.343	0.521 ± 0.393	0.577 ± 0.435	0.631 ± 0.479	0.687 ± 0.526	0.751 ± 0.582	0.829 ± 0.655	0.936 ± 0.756
LPP	0.624 ± 0.553	0.698 ± 0.623	0.755 ± 0.678	0.809 ± 0.729	0.863 ± 0.782	0.923 ± 0.843	0.996 ± 0.918	1.094 ± 1.022
LPC2	0.505 ± 0.383	0.566 ± 0.437	0.614 ± 0.480	0.660 ± 0.522	0.707 ± 0.568	0.761 ± 0.622	0.827 ± 0.691	0.917 ± 0.788
LPC3	0.606 ± 0.446	0.666 ± 0.504	0.713 ± 0.549	0.757 ± 0.594	0.803 ± 0.640	0.854 ± 0.694	0.915 ± 0.763	1.001 ± 0.860
Voxel%	80%-90%	90%-99%	99%-100%	99.9%-100%	99.99%-100%	99.999%-100%	99.9999%-100%	99.99999%-100%
# of Test Cases	306(100%)	306(100%)	306(100%)	125(40.8%)	46(15.0%)	12(3.9%)	3(1.0%)	3(1.0%)
I_{main}	18	18	18	10	3	2	1	1
Deform (mm)	4.580-5.629	5.629-9.026	9.026-36.263	14.306-36.263	19.527-36.263	23.725-36.263	27.533-36.263	29.154-36.263
LP	1.397 ± 0.988	2.007 ± 1.451	5.343 ± 3.868	13.103 ± 4.171	20.302 ± 2.287	24.666 ± 1.688	28.081 ± 1.190	30.436 ± 1.807
LPC	1.114 ± 0.924	1.609 ± 1.367	4.485 ± 3.862	11.928 ± 4.800	19.939 ± 2.549	24.457 ± 1.877	27.983 ± 1.218	30.000 ± 1.796
LPP	1.249 ± 1.184	1.621 ± 1.572	2.894 ± 2.798	5.054 ± 4.387	9.631 ± 6.406	13.541 ± 7.744	17.600 ± 7.317	15.553 ± 6.339
LPC2	1.068 ± 0.951	1.488 ± 1.375	3.917 ± 3.730	10.437 ± 5.395	19.345 ± 3.061	24.154 ± 2.173	27.834 ± 1.403	29.654 ± 1.909
LPC3	1.141 ± 1.022	1.521 ± 1.432	3.596 ± 3.555	8.962 ± 5.690	18.252 ± 4.077	23.673 ± 2.583	27.681 ± 1.641	29.460 ± 2.227

Dataset: CUMC12								
Voxel%	0%-10%	10%-20%	20%-30%	30%-40%	40%-50%	50%-60%	60%-70%	70%-80%
# of Test Cases	132(100%)	132(100%)	132(100%)	132(100%)	132(100%)	132(100%)	132(100%)	132(100%)
I_{main}	12	12	12	12	12	12	12	12
Deform (mm)	0.004-1.169	1.169-1.602	1.602-1.977	1.977-2.341	2.341-2.717	2.717-3.126	3.126-3.597	3.597-4.189
LP	0.617 ± 0.433	0.709 ± 0.480	0.784 ± 0.524	0.856 ± 0.570	0.929 ± 0.621	1.010 ± 0.680	1.109 ± 0.758	1.239 ± 0.868
LPC	0.525 ± 0.391	0.587 ± 0.441	0.640 ± 0.486	0.694 ± 0.534	0.750 ± 0.586	0.813 ± 0.646	0.890 ± 0.724	0.995 ± 0.834
LPP	0.653 ± 0.538	0.717 ± 0.607	0.772 ± 0.667	0.829 ± 0.727	0.888 ± 0.789	0.953 ± 0.859	1.032 ± 0.948	1.138 ± 1.068
LPC2	0.605 ± 0.444	0.653 ± 0.496	0.696 ± 0.543	0.739 ± 0.591	0.787 ± 0.645	0.839 ± 0.704	0.905 ± 0.782	0.996 ± 0.891
LPC3	0.730 ± 0.519	0.775 ± 0.575	0.815 ± 0.625	0.857 ± 0.675	0.903 ± 0.732	0.954 ± 0.793	1.019 ± 0.872	1.107 ± 0.984
Voxel%	80%-90%	90%-99%	99%-100%	99.9%-100%	99.99%-100%	99.999%-100%	99.9999%-100%	99.99999%-100%
# of Test Cases	132(100%)	132(100%)	132(100%)	132(100%)	75(56.8%)	18(13.6%)	1(0.8%)	1(0.8%)
I_{main}	12	12	12	12	9	6	1	1
Deform (mm)	4.189-5.070	5.070-7.443	7.443-18.753	9.581-18.753	12.115-18.753	14.383-18.753	16.651-18.753	18.297-18.753
LP	1.448 ± 1.050	1.955 ± 1.490	3.340 ± 2.412	4.882 ± 3.106	7.281 ± 3.378	9.978 ± 3.211	14.113 ± 1.211	15.868 ± 0.315
LPC	1.163 ± 1.017	1.571 ± 1.451	2.676 ± 2.386	3.930 ± 3.125	5.976 ± 3.646	8.527 ± 3.889	13.009 ± 1.584	14.908 ± 0.437
LPP	1.305 ± 1.258	1.683 ± 1.686	2.484 ± 2.503	3.047 ± 3.048	3.511 ± 3.287	2.999 ± 3.150	1.196 ± 0.327	1.277 ± 0.178
LPC2	1.142 ± 1.072	1.494 ± 1.494	2.405 ± 2.388	3.392 ± 3.055	5.003 ± 3.702	7.176 ± 4.218	11.155 ± 2.470	13.456 ± 0.812
LPC3	1.248 ± 1.165	1.581 ± 1.581	2.383 ± 2.434	3.165 ± 3.015	4.422 ± 3.629	6.283 ± 4.370	9.607 ± 3.120	11.757 ± 1.492

Dataset: MGH10								
Voxel%	0%-10%	10%-20%	20%-30%	30%-40%	40%-50%	50%-60%	60%-70%	70%-80%
# of Test Cases	90(100%)	90(100%)	90(100%)	90(100%)	90(100%)	90(100%)	90(100%)	90(100%)
I_{main}	10	10	10	10	10	10	10	10
Deform (mm)	0.003-1.122	1.122-1.553	1.553-1.929	1.929-2.294	2.294-2.674	2.674-3.089	3.089-3.567	3.567-4.163
LP	0.578 ± 0.422	0.680 ± 0.471	0.757 ± 0.514	0.829 ± 0.559	0.904 ± 0.610	0.986 ± 0.671	1.082 ± 0.748	1.207 ± 0.858
LPC	0.486 ± 0.382	0.558 ± 0.436	0.615 ± 0.481	0.669 ± 0.528	0.726 ± 0.580	0.790 ± 0.640	0.865 ± 0.715	0.963 ± 0.820
LPP	0.624 ± 0.556	0.701 ± 0.624	0.761 ± 0.680	0.819 ± 0.737	0.881 ± 0.796	0.948 ± 0.865	1.026 ± 0.948	1.127 ± 1.061
LPC2	0.562 ± 0.436	0.623 ± 0.493	0.670 ± 0.541	0.716 ± 0.589	0.766 ± 0.643	0.821 ± 0.703	0.886 ± 0.778	0.971 ± 0.881
LPC3	0.684 ± 0.512	0.745 ± 0.574	0.792 ± 0.625	0.838 ± 0.677	0.887 ± 0.734	0.942 ± 0.797	1.007 ± 0.874	1.091 ± 0.979
Voxel%	80%-90%	90%-99%	99%-100%	99.9%-100%	99.99%-100%	99.999%-100%	99.9999%-100%	99.99999%-100%
# of Test Cases	90(100%)	90(100%)	90(100%)	89(98.9%)	37(41.1%)	7(7.8%)	3(3.3%)	2(2.2%)
I_{main}	10	10	10	9	6	3	1	1
Deform (mm)	4.163-5.047	5.047-7.462	7.462-18.727	9.833-18.727	12.607-18.727	15.564-18.727	17.684-18.727	18.534-18.727
LP	1.408 ± 1.041	1.924 ± 1.499	3.348 ± 2.341	4.873 ± 2.704	7.299 ± 3.256	10.503 ± 4.049	11.764 ± 3.005	13.041 ± 1.691
LPC	1.120 ± 0.998	1.526 ± 1.444	2.627 ± 2.275	3.674 ± 2.621	5.283 ± 3.272	8.492 ± 4.088	9.336 ± 3.738	10.499 ± 2.692
LPP	1.289 ± 1.250	1.676 ± 1.705	2.430 ± 2.415	2.707 ± 2.403	2.998 ± 2.214	2.695 ± 1.419	2.377 ± 1.063	2.226 ± 0.365
LPC2	1.109 ± 1.055	1.455 ± 1.482	2.361 ± 2.270	3.138 ± 2.585	4.267 ± 3.133	6.603 ± 3.760	7.444 ± 3.657	8.415 ± 3.053
LPC3	1.225 ± 1.153	1.551 ± 1.570	2.358 ± 2.306	3.023 ± 2.588	3.755 ± 2.957	5.214 ± 3.486	6.240 ± 3.357	7.569 ± 3.283

Table 5: Deformation ranges and mean+standard deviation of the deformation errors between the prediction models (LP, LPC, LPP, LPC2, LPC3) and the optimization model (LO) for the *image-to-image* registration case. All measures are evaluated within the brain mask only. **All deformation values and deformation errors are evaluated in millimeters (mm)**. Voxel%: percentile range of voxels that fall in a particular deformation range based on the optimization model (LO). # of Test cases: The number of registration cases that contain voxels within a given percentile range. I_{main}: Minimum number of distinct images required to cover all registration test cases in a particular deformation range either as the moving or the target image. This measure is meant to quantify the influence of *few* images on very large deformations. For example, for the four registration cases A-B, B-C, B-D and E-A, $I_{\text{main}} = 2$ as it is sufficient to select images A and B to cover all four registrations. The results show that a comparatively small subset of images is responsible for most of the very large deformations. Of course, all images of a particular dataset are involved in the small deformation ranges. Deform: range of deformations within a given percentile range. The cells with the lowest mean deformation errors for every deformation range are highlighted. Best-viewed in color.

	Deformation Error w.r.t LDDMM optimization on T1w-T1w data [mm]						
Data percentile for all voxels	0.3%	5%	25%	50%	75%	95%	99.7%
Affine (Baseline)	0.1664	0.46	0.9376	1.4329	2.0952	3.5037	6.2576
T1w-T1w `LP`	0.0348	0.0933	0.1824	0.2726	0.3968	0.6779	1.3614
T1w-T1w `LPC`	0.0289	0.0777	0.1536	0.2318	0.3398	0.5803	1.1584
T1w-T2w `LP`	0.0544	0.1457	0.2847	0.4226	0.6057	1.0111	2.0402
T1w-T2w `LPC`	0.0520	0.1396	0.2735	0.4074	0.5855	0.9701	1.9322
T1w-T2w `LP`, 10 images	0.0660	0.1780	0.3511	0.5259	0.7598	1.2522	2.3496
T1w-T2w `LPC`, 10 images	0.0634	0.1707	0.3356	0.5021	0.7257	1.1999	2.2697

Table 6: Evaluation result for *multi-modal image-to-image* tests. Deformation error (2-norm) per voxel between predicted deformation and optimization deformation. Percentiles over all deformation errors are shown to illustrate the error distribution. `LP`: prediction network. `LPC`: prediction+correction network. 10 images: network is trained using 10 images (90 registrations as training cases).

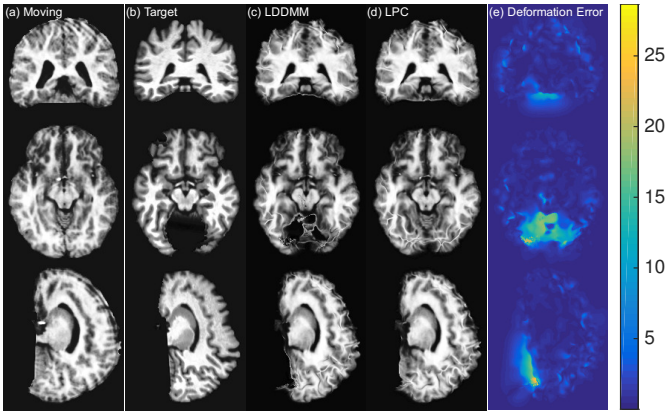


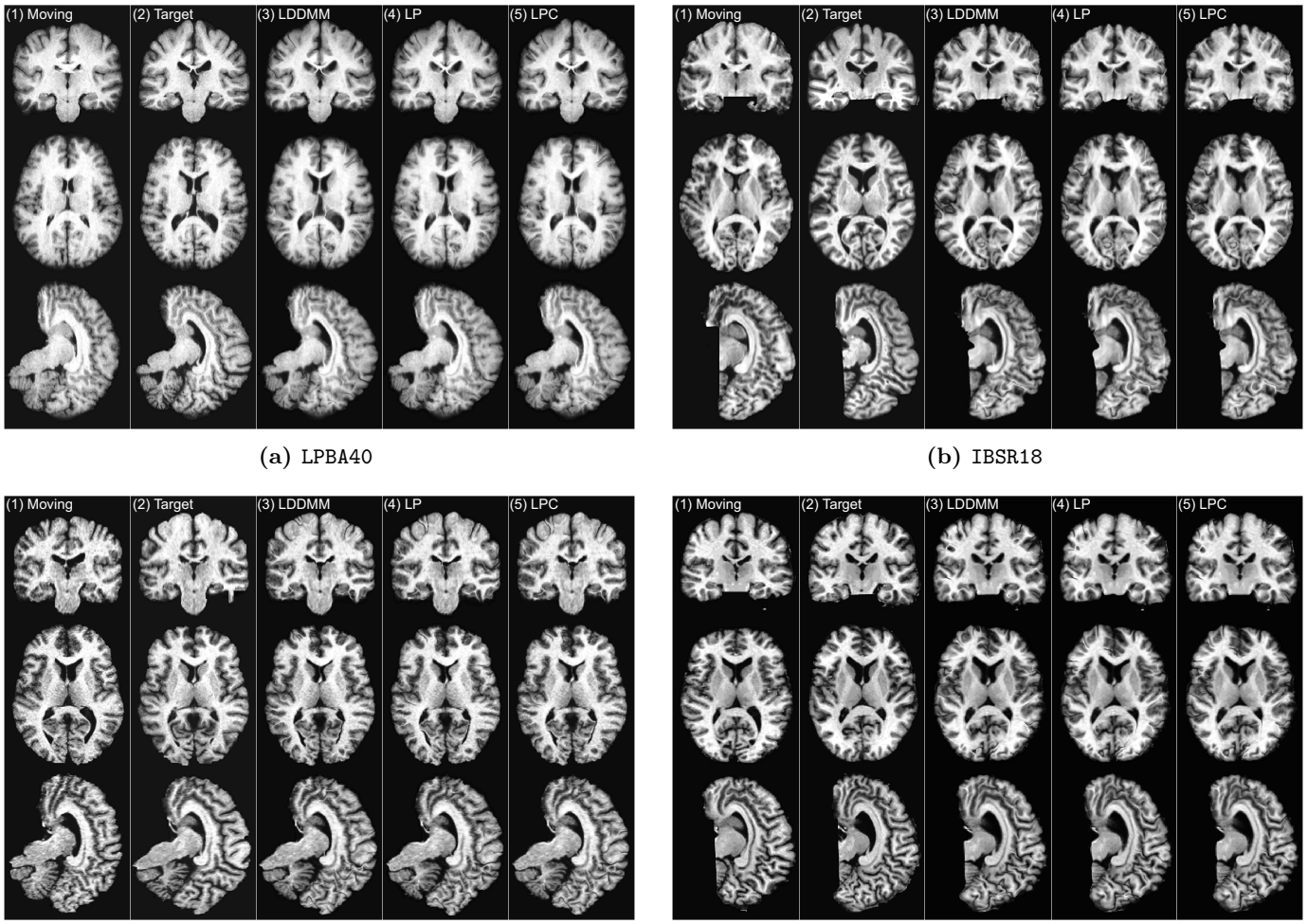
Figure 8: Failure case for `IBSR18` dataset where LDDMM optimization generated very extreme deformations. From *left* to *right*: (a): moving image; (b): target image; (c): LDDMM optimization result; (d): prediction+correction result (`LPC`); (e): heatmap showing the differences between the optimization deformation and predicted deformation in millimeters. Most registration errors occur in the area of the cerebellum, which has been inconsistently preserved in the moving and the target images during brain extraction. Hence, not all the retained brain regions in the moving image have correspondences in the target image. Best-viewed in color.

available. Again, using a correction network improves the prediction accuracy in all cases. Fig. 10 shows one multi-modal registration example. All three networks (T1w-T1w, T1w-T2w, T1w-T2w using 10 training images) generate warped images that are similar to the LDDMM optimization result.

3.4. Runtime study

We assess the runtime of `Quicksilver` on a single Nvidia TitanX (Pascal) GPU. Performing LDDMM optimization using the GPU-based implementation of `PyCA` for a $229 \times 193 \times 193$ 3D brain image takes approximately 10.8 minutes. Using our prediction network with a sliding window stride of 14, the initial momentum prediction time is, on average, 7.63 seconds. Subsequent geodesic shooting to generate the deformation field takes 8.9 seconds, resulting in a total runtime of 18.43 seconds. Compared to the LDDMM optimization approach, our method achieves a $35\times$ speed up. Using the correction network together with the prediction network doubles the computation time, but the overall runtime is still an order of magnitude faster than direct LDDMM optimization. Note that, at a stride of 1, computational cost increases about 3000-fold in 3D, resulting in runtimes of about $5^1/_2$ hours for 3D image registration (eleven hours when the correction network is also used). Hence the initial momentum parameterization, which can tolerate large sliding window strides, is essential for fast deformation prediction with high accuracy while guaranteeing diffeomorphic deformations.

Since we predict the whole image initial momentum in a patch-wise manner, it is natural to extend our approach to a multi-GPU implementation by distributing patches across multiple GPUs. We assess the runtime of this parallelization strategy on a cluster with multiple Nvidia GTX 1080 GPUs; the initial momentum prediction result is shown in Fig. 11. As we can see, by increasing the number of GPUs, the initial momentum prediction time decreases from 11.23 seconds (using 1 GPU) to 2.41 seconds using 7 GPUs. However, as the number of GPUs increases, the communication overhead between GPUs becomes larger which explains why computation time does not equal to 11.23/number of GPUs seconds. Also, when we increase the number of GPUs to 8, the prediction time



(a) LPBA40 **(b)** IBSR18

(c) CUMC12 **(d)** MGH10

Figure 9: Example test cases for the *image-to-image* registration. For every figure from *left* to *right*: (1): moving image; (2): target image; (3): registration result from optimizing LDDMM energy; (4): registration result from prediction network (LP); (5): registration result from prediction+correction network (LPC).

slightly increases to 2.48s. This can be attributed to the fact that PyTorch is still in Beta-stage and, according to the documentation, better performance for large numbers of GPUs (8+) is being actively developed[16]. Hence, we expect faster prediction times using a large number of GPUs in the future. Impressively, by using multiple GPUs, the runtime can be improved by two orders of magnitude over a direct (GPU-based) LDDMM optimization. Thus, our method can readily be used in a GPU-cluster environment for ultra-fast deformation prediction.

4. Discussion

We proposed a fast registration approach based on the patch-wise prediction of the initial momentum parameterization of the LDDMM shooting formulation. The proposed approach allows taking large strides for patch-wise prediction, without a substantial decrease in registration accuracy, resulting in fast and accurate deformation prediction. The proposed correction network is a step towards highly accurate deformation prediction, while only decreasing the computation speed by a factor of 2. Our method retains all theoretical properties of LDDMM and results in diffeomorphic transformations if appropriately regularized, but computes these transformations an order of magnitude faster than a GPU-based optimization for the LDDMM model. Moreover, the patch-wise prediction approach of our methods enables a multi-GPU implementation, further increasing the prediction speed. In effect, our Quicksilver registration approach converts a notoriously slow and memory-hungry registration approach to a fast method, while retaining all of its appealing mathematical properties.

Our framework is very general and can be directly applied to many other registration techniques. For non-parametric registration methods with pixel/voxel wise reg-

[16] http://pytorch.org/docs/master/notes/cuda.html#use-nn-dataparallel-instead-of-multiprocessing

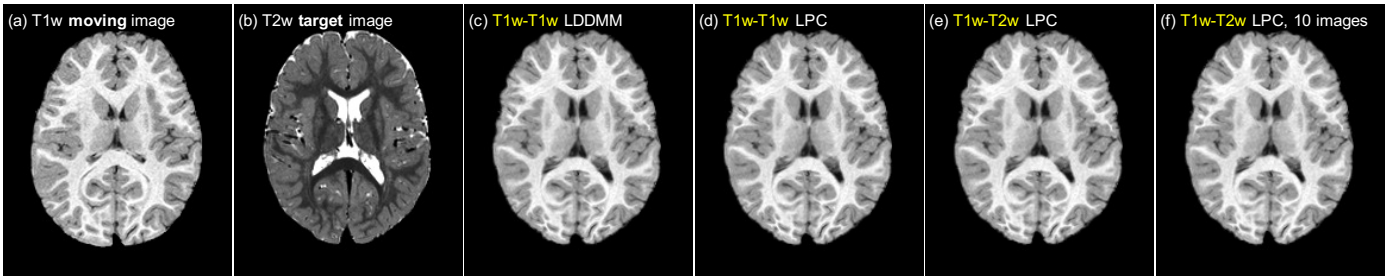


Figure 10: Example test case for *multi-modal image-to-image* tests. (a): T1w moving image; (b): T2w target image; (c): T1w-T1w LDDMM optimization (`LO`) result; (d)-(f): deformation prediction+correction (`LPC`) result using (d) T1w-T1w data; (e) T1w-T2w data; (f) T1w-T2w data using only 10 images as training data.

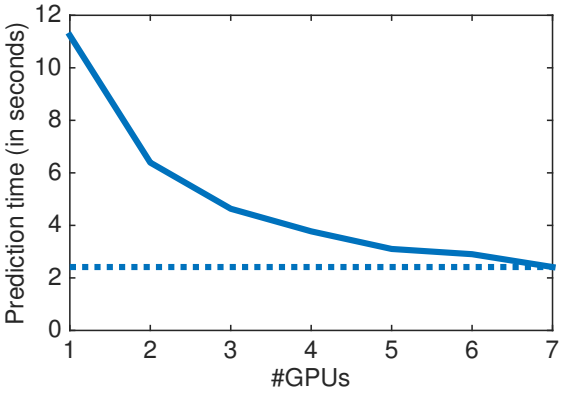


Figure 11: Average initial momentum prediction time (in seconds) for a single $229 \times 193 \times 193$ 3D brain image case using various number of GPUs.

istration parameters (e.g., elastic registration [1], or stationary velocity field [44] registration approaches), our approach can be directly applied for parameter prediction. For parametric registration methods with local control such as B-splines, we could attach fully connected layers to the decoder to reduce the network output dimension, thereby predicting low-dimensional registration parameters for a patch. Of course, the patch pruning techniques may not be applicable for these methods if the parameter locality cannot be guaranteed.

In summary, the presented deformation prediction approach is the first step towards more complex tasks where fast, deformable, predictive image registration techniques are required. It opens up possibilities for various extensions and applications. Exciting possibilities are, for example, to use `Quicksilver` as the registration approach for fast multi-atlas segmentation, fast image geodesic regression, fast atlas construction, or fast user-interactive registration refinements (where only a few patches need to be updated based on local changes). Furthermore, extending the deformation prediction network to more complex registration tasks could also be beneficial; e.g., to further explore the behavior of the prediction models for multi-modal image registration [36]. Other potential areas include joint image-label registration for better label-matching accuracy; multi-scale-patch networks for very large deformation prediction; deformation prediction for registration models with anisotropic regularizations; and end-to-end optical flow prediction via initial momentum parameterization. Other correction methods could also be explored, by using different network structures, or by recursively updating the deformation parameter prediction using the correction approach (e.g., with a sequence of correction networks where each network corrects the momenta predicted from the previous one). Finally, since our uncertainty quantification approach indicates high uncertainty for areas with large deformation or appearance changes, utilizing the uncertainty map to detect pathological areas could also be an interesting research direction.

Source code. To make the approach readily available to the community, we open-sourced `Quicksilver` at `https://github.com/rkwitt/quicksilver`. Our long-term goal is to make our framework the basis for different variants of predictive image registration; e.g., to provide `Quicksilver` variants for various organs and imaging types, as well as for different types of spatial regularization.

Acknowledgments. This work is supported by NIH 1 R41 NS091792-01, NIH HDO55741, NSF ECCS-1148870, and EECS-1711776. This work also made use of the XStream computational resource, supported by the National Science Foundation Major Research Instrumentation program (ACI-1429830). We also thank Nvidia for the donation of a TitanX GPU.

References

[1] J. Modersitzki, Numerical methods for image registration, Oxford University Press on Demand, 2004.
[2] Biobank website: `www.ukbiobank.ac.uk`.
[3] D. C. Van Essen, S. M. Smith, D. M. Barch, T. E. Behrens, E. Yacoub, K. Ugurbil, WU-Minn HCP Consortium, The WU-Minn human connectome project: an overview, NeuroImage 80 (2013) 62–79.
[4] K. Chung, K. Deisseroth, CLARITY for mapping the nervous system, Nature methods 10 (6) (2013) 508–513.
[5] R. Shams, P. Sadeghi, R. A. Kennedy, R. I. Hartley, A survey of medical image registration on multicore and the GPU, IEEE Signal Processing Magazine 27 (2) (2010) 50–60.

[6] J. Ashburner, K. J. Friston, Diffeomorphic registration using geodesic shooting and Gauss–Newton optimisation, NeuroImage 55 (3) (2011) 954–967.
[7] M. Zhang, P. Fletcher, Finite-dimensional Lie algebras for fast diffeomorphic image registration, in: IPMI, 2015, pp. 249–260.
[8] B. Gutierrez-Becker, D. Mateus, L. Peter, N. Navab, Guiding multimodal registration with learned optimization updates, MedIA.
[9] B. Gutiérrez-Becker, D. Mateus, L. Peter, N. Navab, Learning optimization updates for multimodal registration, in: MICCAI, 2016, pp. 19–27.
[10] C.-R. Chou, B. Frederick, G. Mageras, S. Chang, S. Pizer, 2D/3D image registration using regression learning, CVIU 117 (9) (2013) 1095–1106.
[11] Q. Wang, M. Kim, Y. Shi, G. Wu, D. Shen, Predict brain MR image registration via sparse learning of appearance & transformation, MedIA 20 (1) (2015) 61–75.
[12] T. Cao, N. Singh, V. Jojic, M. Niethammer, Semi-coupled dictionary learning for deformation prediction, in: ISBI, 2015, pp. 691–694.
[13] M. F. Beg, M. Miller, A. Trouvé, L. Younes, Computing large deformation metric mappings via geodesic flows of diffeomorphisms, IJCV 61 (2) (2005) 139–157.
[14] P. Weinzaepfel, J. Revaud, Z. Harchaoui, C. Schmid, DeepFlow: Large displacement optical flow with deep matching, in: ICCV, 2013, pp. 1385–1392.
[15] A. Dosovitskiy, P. Fischery, E. Ilg, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, T. Brox, Flownet: Learning optical flow with convolutional networks, in: ICCV, 2015, pp. 2758–2766.
[16] S. Miao, Z. J. Wang, R. Liao, A CNN regression approach for real-time 2D/3D registration, IEEE TMI 35 (5) (2016) 1352–1363.
[17] F.-X. Vialard, L. Risser, D. Rueckert, C. J. Cotter, Diffeomorphic 3D image registration via geodesic shooting using an efficient adjoint calculation, IJCV 97 (2) (2012) 229–241.
[18] M. Vaillant, M. Miller, L. Younes, A. Trouvé, Statistics on diffeomorphisms via tangent space representations, NeuroImage 23, Supplement 1 (2004) 161–169.
[19] M. Niethammer, Y. Huang, F.-X. Vialard, Geodesic regression for image time-series, in: MICCAI, 2011, pp. 655–662.
[20] Y. Hong, Y. Shi, M. Styner, M. Sanchez, M. Niethammer, Simple geodesic regression for image time-series, in: WBIR, 2012, pp. 11–20.
[21] I. Simpson, M. Woolrich, A. Groves, J. Schnabel, Longitudinal brain MRI analysis with uncertain registration, in: MICCAI, 2011, pp. 647–654.
[22] T. Cao, C. Zach, S. Modla, D. Powell, K. Czymmek, M. Niethammer, Multi-modal registration for correlative microscopy using image analogies, MedIA 18 (6) (2014) 914–926.
[23] W. Wein, S. Brunke, A. Khamene, M. R. Callstrom, N. Navab, Automatic CT-ultrasound registration for diagnostic imaging and image-guided intervention, MedIA 12 (5) (2008) 577–585.
[24] P. Viola, W. M. W. III, Alignment by maximization of mutual information, IJCV 24 (2) (1997) 137–154.
[25] C. Meyer, J. Boes, B. Kim, P. Bland, Evaluation of control point selection in automatic, mutual information driven, 3D warping, in: MICCAI, 1998, pp. 944–951.
[26] G. Hermosillo, C. Chefd'Hotel, O. Faugeras, Variational methods for multimodal image matching, IJCV 50 (3) (2002) 329–343.
[27] P. Lorenzen, M. Prastawa, B. Davis, G. Gerig, E. Bullitt, S. Joshi, Multi-modal image set registration and atlas formation, MedIA 10 (3) (2006) 440–451.
[28] C. Guetter, C. Xu, F. Sauer, J. Hornegger, Learning based non-rigid multi-modal image registration using Kullback-Leibler divergence, in: MICCAI, 2005, pp. 255–262.
[29] D. Lee, M. Hofmann, F. Steinke, Y. Altun, N. Cahill, B. Schölkopf, Learning similarity measure for multi-modal 3D image registration, in: CVPR, 2009, pp. 186–193.
[30] F. Michel, M. Bronstein, A. M. Bronstein, N. Paragios, Boosted metric learning for 3D multi-modal deformable registration, in: ISBI, 2011, pp. 1209–1214.
[31] X. Cheng, L. Z. Y. Zheng, Deep similarity learning for multimodal medical images, Comput. Methods Biomech. Biomed. Eng. Imaging Vis. (2015) 1–5.
[32] M. Simonovsky, B. Gutiérrez-Becker, D. Mateus, N. Navab, N. Komodakis, A deep metric for multimodal registration, MICCAI (2016) 10–18.
[33] A. Klein, J. Andersson, B. A. Ardekani, J. Ashburner, B. Avants, M.-C. Chiang, G. E. Christensen, D. L. Collins, J. Gee, P. Hellier, J. H. Song, M. Jenkinson, C. Lepage, D. Rueckert, P. Thompson, T. Vercauteren, R. P. Woods, J. J. Mann, R. V. Parsey, Evaluation of 14 nonlinear deformation algorithms applied to human brain MRI registration, NeuroImage 46 (3) (2009) 786 – 802.
[34] H. C. Hazlett, H. Gu, B. C. Munsell, S. H. Kim, M. Styner, J. J. Wolff, J. T. Elison, M. R. Swanson, H. Zhu, K. N. Botteron, et al., Early brain development in infants at high risk for autism spectrum disorder, Nature 542 (7641) (2017) 348–351.
[35] X. Yang, R. Kwitt, M. Niethammer, Fast predictive image registration, in: DLMIA/MICCAI, 2016, pp. 48–57.
[36] X. Yang, R. Kwitt, M. Styner, M. Niethammer, Fast predictive multimodal image registration, in: ISBI, 2017, pp. 858–862.
[37] D. L. Hill, P. G. Batchelor, M. Holden, D. J. Hawkes, Medical image registration, Physics in medicine and biology 46 (3) (2001) R1.
[38] A. Sotiras, C. Davatzikos, N. Paragios, Deformable medical image registration: A survey, IEEE transactions on medical imaging 32 (7) (2013) 1153–1190.
[39] F. P. Oliveira, J. M. R. Tavares, Medical image registration: a review, Computer methods in biomechanics and biomedical engineering 17 (2) (2014) 73–93.
[40] D. Rueckert, L. I. Sonoda, C. Hayes, D. L. Hill, M. O. Leach, D. J. Hawkes, Nonrigid registration using free-form deformations: application to breast MR images, IEEE transactions on medical imaging 18 (8) (1999) 712–721.
[41] B. K. Horn, B. G. Schunck, Determining optical flow, Artificial intelligence 17 (1-3) (1981) 185–203.
[42] C. Zach, T. Pock, H. Bischof, A duality based approach for realtime TV-L1 optical flow, Pattern Recognition (2007) 214–223.
[43] E. Haber, J. Modersitzki, Image registration with guaranteed displacement regularity, International Journal of Computer Vision 71 (3) (2007) 361–372.
[44] T. Vercauteren, X. Pennec, A. Perchant, N. Ayache, Diffeomorphic demons: Efficient non-parametric image registration, NeuroImage 45 (1) (2009) S61–S72.
[45] G. L. Hart, C. Zach, M. Niethammer, An optimal control approach for deformable registration, in: CVPR, IEEE, 2009, pp. 9–16.
[46] A. Borzi, K. Ito, K. Kunisch, Optimal control formulation for determining optical flow, SIAM journal on scientific computing 24 (3) (2003) 818–847.
[47] Y. LeCun, A theoretical framework for back-propagation, in: Proceedings of the 1988 connectionist models summer school, 1988, pp. 21–28.
[48] A. Griewank, A. Walther, Evaluating derivatives: principles and techniques of algorithmic differentiation, SIAM, 2008.
[49] J. Nocedal, S. J. Wright, Numerical optimization 2nd, Springer, 2006.
[50] N. Singh, J. Hinkle, S. Joshi, P. Fletcher, A vector momenta formulation of diffeomorphisms for improved geodesic regression and atlas construction, in: ISBI, 2013, pp. 1219–1222.
[51] P. Dupuis, U. Grenander, M. I. Miller, Variational problems on flows of diffeomorphisms for image matching, Quarterly of applied mathematics (1998) 587–600.
[52] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification, CoRR abs/1502.01852 (2015).
[53] V. Nair, G. E. Hinton, Rectified linear units improve restricted Boltzmann machines, in: J. Frnkranz, T. Joachims (Eds.),

Proceedings of the 27th International Conference on Machine Learning (ICML-10), Omnipress, 2010, pp. 807–814.
[54] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: CVPR, 2015, pp. 3431–3440.
[55] J. T. Springenberg, A. Dosovitskiy, T. Brox, M. A. Riedmiller, Striving for simplicity: The all convolutional net, CoRR abs/1412.6806 (2014).
[56] T. Gao, V. Jojic, Degrees of freedom in deep neural networks, in: UAI, 2016, pp. 232–241.
[57] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, JMLR 15 (2014) 1929–1958.
[58] Y. Gal, Z. Ghahramani, Bayesian convolutional neural networks with Bernoulli approximate variational inference, arXiv:1506.02158 (2015).
[59] Y. Gal, Z. Ghahramani, Dropout as a Bayesian approximation: Representing model uncertainty in deep learning, in: ICML, 2016, pp. 1050–1059.
[60] D. S. Marcus, T. H. Wang, J. Parker, J. G. Csernansky, J. C. Morris, R. L. Buckner, Open access series of imaging studies (OASIS): Cross-sectional MRI data in young, middle aged, nondemented, and demented older adults, J. Cognitive Neurosci. 19 (9) (2007) 1498–1507.
[61] S. Joshi, B. Davis, M. Jomier, G. Gerig, Unbiased diffeomorphic atlas construction for computational anatomy, NeuroImage 23 (2004) 151–160.
[62] N. Singh, J. Hinkle, S. Joshi, P. Fletcher, A hierarchical geodesic model for diffeomorphic longitudinal shape analysis, in: IPMI, 2013, pp. 560–571.
[63] M. Reuter, N. J. Schmansky, H. D. Rosas, B. Fischl, Within-subject template estimation for unbiased longitudinal image analysis, NeuroImage 61 (4) (2012) 1402 – 1418.
[64] J. Wang, C. Vachet, A. Rumple, S. Gouttard, C. Ouziel, E. Perrot, G. Du, X. Huang, G. Gerig, M. Styner, Multi-atlas segmentation of subcortical brain structures via the autoseg software pipeline, Frontiers in Neuroinformatics 8 (2014) 7.
[65] G. Grabner, A. L. Janke, M. M. Budge, D. Smith, J. Pruessner, D. L. Collins, Symmetric atlasing and model based segmentation: An application to the hippocampus in older adults, in: R. Larsen, M. Nielsen, J. Sporring (Eds.), MICCAI, 2006, pp. 58–66.
[66] M. Bruveris, L. Risser, F.-X. Vialard, Mixture of kernels and iterated semidirect product of diffeomorphisms groups, Multiscale Modeling & Simulation 10 (4) (2012) 1344–1368.
[67] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: ICLR, 2014.
[68] B. Avants, C. Epstein, M. Grossman, J. Gee, Symmetric diffeomorphic image registration with cross-correlation: Evaluating automated labeling of elderly and neurodegenerative brain, MedIA 12 (1) (2008) 26 – 41, special Issue on The Third International Workshop on Biomedical Image Registration WBIR 2006.
[69] B. A. Ardekani, S. Guckemus, A. Bachman, M. J. Hoptman, M. Wojtaszek, J. Nierenberg, Quantitative comparison of algorithms for inter-subject registration of 3D volumetric brain MRI scans, Journal of Neuroscience Methods 142 (1) (2005) 67 – 76.
[70] J. Ashburner, A fast diffeomorphic image registration algorithm, NeuroImage 38 (1) (2007) 95 – 113.
[71] S. Wellek, Testing statistical hypotheses of equivalence, CRC Press, 2010.