Published in final edited form as:
IEEE Trans Med Imaging. 2010 March ; 29(3): 650–668. doi:10.1109/TMI.2009.2030797.

Spherical Demons: Fast Diffeomorphic Landmark-Free Surface Registration

B.T. Thomas Yeo[*],
Computer Science and Artificial Intelligence Laboratory, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, USA

Mert R. Sabuncu[*],
Computer Science and Artificial Intelligence Laboratory, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, USA

Tom Vercauteren,
Mauna Kea Technologies, Paris, France

Nicholas Ayache,
Asclepios Group, INRIA, Sophia Antipolis, France

Bruce Fischl, and
Computer Science and Artificial Intelligence Laboratory, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, USA; Department of Radiology, Harvard Medical School, Charlestown, USA and the Divison of Health Sciences and Technology, Massachusetts Institute of Technology, Cambridge, USA

Polina Golland
Computer Science and Artificial Intelligence Laboratory, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, USA

B.T. Thomas Yeo: ythomas@csail.mit.edu; Mert R. Sabuncu: msabuncu@csail.mit.edu; Tom Vercauteren: tom.vercauteren@maunakeatech.com; Nicholas Ayache: nicholas.ayache@sophia.inria.fr; Bruce Fischl: fischl@nmr.mgh.harvard.edu; Polina Golland: polina@csail.mit.edu

Abstract

We present the Spherical Demons algorithm for registering two spherical images. By exploiting spherical vector spline interpolation theory, we show that a large class of regularizors for the modified Demons objective function can be efficiently approximated on the sphere using iterative smoothing. Based on one parameter subgroups of diffeomorphisms, the resulting registration is diffeomorphic and fast. The Spherical Demons algorithm can also be modified to register a given spherical image to a probabilistic atlas. We demonstrate two variants of the algorithm corresponding to warping the atlas or warping the subject. Registration of a cortical surface mesh to an atlas mesh, both with more than 160k nodes requires less than 5 minutes when warping the atlas and less than 3 minutes when warping the subject on a Xeon 3.2GHz single processor machine. This is comparable to the fastest non-diffeomorphic landmark-free surface registration algorithms. Furthermore, the accuracy of our method compares favorably to the popular FreeSurfer registration algorithm. We validate the technique in two different applications that use registration to transfer segmentation labels onto a new image: (1) parcellation of in-vivo cortical surfaces and (2) Brodmann area localization in ex-vivo cortical surfaces.

[*]B.T. Thomas Yeo and Mert R. Sabuncu contributed equally to this paper.

Index Terms

Surface Registration; Spherical Registration; Cortical Registration; Vector Field Interpolation; Demons; Diffeomorphism

I. Introduction

Motivated by many successful applications of the spherical representation of the cerebral cortex, this paper addresses the problem of registering two spherical images. Cortical folding patterns have been shown to correlate with both cytoarchitectural [25], [68] and functional regions [64], [27]. In group studies of cortical structure and function, determining corresponding folds across subjects is therefore important. There has been much effort focused on registering cortical surfaces in 3D [22], [23], [30], [58]. Since cortical areas – both structure and function – are arranged in a mosaic across the cortical surface, an alternative approach is to model the surface as a 2D closed manifold in 3D and to warp the underlying spherical coordinate system [27], [50], [59], [60], [64], [67]. Warping the spherical coordinate system establishes correspondences across the surfaces *without* actually deforming the surfaces in 3D.

Deformation Model

There is frequently a need for invertible deformations that preserve the topology of structural or functional regions across subjects. Unfortunately, this causes many spherical warping algorithms to be computationally expensive. Previously demonstrated methods for cortical registration [27], [60], [67] rely on soft regularization constraints to encourage invertibility. These require unfolding the mesh triangles, or limit the size of optimization steps to achieve invertibility [27], [67]. Elegant regularization penalties that guarantee invertibility exist [5], [46] but they explicitly rely on special properties of the Euclidean image space that do not hold for the sphere.

An alternative approach to achieving invertibility is to work in the group of diffeomorphisms [4], [7], [9], [22], [31], [43], [66]. In this case, the underlying theory of flows of vector fields can be extended to manifolds [11], [44], [47]. The Large Deformation Diffeomorphic Metric Mapping (LDDMM) [7], [9], [22], [31], [43] is a popular framework that seeks a time-varying velocity field representation of a diffeomorphism. Because LDDMM optimizes over the entire path of the diffeomorphism, the resulting method is slow and memory intensive. By contrast, Ashburner [4] and Hernandez *et al.* [33] consider diffeomorphic transformations parameterized by a single stationary velocity field. While restricting the space of solutions reduces the memory needs relative to LDDMM, these algorithms still have to consider the entire trajectory of the deformation induced by the velocity field when computing the gradients of the objective function, leading to a long run time. We note that recent algorithmic advances [34], [43] promise to improve the speed and relieve the memory requirements of both LDDMM and the stationary velocity approach.

In this work, we adopt the approach of the Diffeomorphic Demons algorithm [66], demonstrated in the Euclidean image space, which constructs the deformation space that contains compositions of diffeomorphisms, each of which is parameterized by a stationary velocity field. Unlike the Euclidean Diffeomorphic Demons, the Spherical Demons algorithm utilizes velocity vectors tangent to the sphere and not arbitrary 3D vectors. This constraint need not be taken care of explicitly in our algorithm since we directly work with the tangent spaces. In each iteration, the method greedily seeks the locally optimal diffeomorphism to be composed with the current transformation. As a result, the approach is

much faster than LDDMM [7], [9], [22], [31] and its simplifications [4], [33]. A drawback is that the path of deformation is no longer a geodesic in the group of diffeomorphisms.

Image Similarity *vs.* Regularization Tradeoffs

Another challenge in registration is the tradeoff between the image similarity measure and the regularization in the objective function. Since most types of regularization favor smooth deformations, the gradient computation is complicated by the need to take into account the deformation in neighboring regions. For Euclidean images, the popular Demons algorithm [57] can be interpreted as optimizing an objective function with two regularization terms [14], [66]. The special form of the objective function facilitates a fast two-step optimization where the second step handles the warp regularization via a single convolution with a smoothing filter.

Using spherical vector spline interpolation theory [31] and other differential geometric tools, we show that the two-stage optimization procedure of Demons can be efficiently approximated on the sphere. We note that the problem is not trivial since tangent vectors at different points on the sphere are not directly comparable. We also emphasize that this decoupling of the image similarity and the warp regularization could also be accomplished with a different space of admissible warps, e.g., spherical thin plate splines [72].

Interpolation

Yet another reason why spherical image registration is slow is because of the difficulty in performing interpolation on a spherical grid, unlike a regular Euclidean grid. In this paper, we use existing methods for interpolation, requiring about one second to interpolate data from a spherical mesh of 160k vertices onto another spherical mesh of 160k vertices. Recent work on using different coordinate charts of the sphere [63] promises to further speed up our implementation of the Spherical Demons algorithm.

While most discussion in this paper concentrates on pairwise registration of spherical images, the proposed Spherical Demons algorithm can be modified to incorporate a probabilistic atlas. We derive and implement two variants of the algorithm for registration to an atlas corresponding to whether we warp the atlas or the subject. On a Xeon 3.2GHz single processor machine, registration of a cortical surface mesh to an atlas mesh, both with more than 160k nodes, requires less than 5 minutes when warping the atlas and less than 3 minutes when warping the subject. Note that the registration runtime reported includes registration components dealing with rotation, which takes up one quarter of the total runtime. The total runtime is comparable to other nonlinear landmark-free cortical surface registration algorithms whose runtime ranges from minutes [23], [60] to more than an hour [27], [67]. However, the other fast algorithms suffer from folding spherical triangles [60] and intersecting triangles in 3D [23] since only soft constraints are used. No runtime comparison can be made with spherical registration algorithm of the LDDMM type because to the best of our knowledge, no landmark-free LDDMM spherical registration algorithm that handles cortical surfaces has been developed yet.

Unlike landmark-based methods for surface registration [8], [22], [31], [50], [58], [64], we do not assume the existence of corresponding landmarks. Landmark-free methods have the advantage of allowing for a fully automatic processing and analysis of medical images. Unfortunately, landmark-free registration is also more challenging, because no information about correspondences are provided. The difficulty is exacerbated for the cerebral cortex since different sulci and gyri appear locally similar. Nevertheless, we demonstrate that our algorithm is accurate in both cortical parcellation and cytoarchitectonic localization applications.

The Spherical Demons algorithm for registering cortical surfaces presented here does not take into account the metric properties of the original cortical surface. FreeSurfer [27] uses a regularization that penalizes deformation of the spherical coordinate system based on the distortion computed on the original cortical surface. Thompson *et al.* [59] suggest the use of Christoffel symbols [39] to correct for the metric distortion of the initial spherical coordinate system during the registration process. However, it is unclear whether correcting for the metric properties of the cortex is important in practice, since we demonstrate that the accuracy of the Spherical Demons algorithm compares favorably to that of FreeSurfer. A possible reason is that we initialize the registration with a spherical parametrization that minimizes metric distortion between the spherical representation and the original cortical surface [27].

This paper is organized as follows. In the next section, we discuss the classical Demons algorithm [57] and its diffeomorphic variant [66]. In Section III, we present the extension of the Diffeomorphic Demons algorithm to the sphere. We conclude with experiments in Section IV and further discussion in Section V. The appendices provide technical and implementation details of the Spherical Demons algorithm and the extension to probabilistic atlases. This paper extends a previously presented conference article [69] and contains detailed derivations and discussions that were left out in the conference version. We note that our Spherical Demons code is freely available[1]. To summarize,

1. We demonstrate that the Demons algorithm can be efficiently extended to the sphere.
2. We demonstrate that the use of a limited class of diffeomorphisms combined with the Demons algorithm yields a speed gain of more than an order of magnitude compared with other landmark-free invertible spherical registration methods, such as [27], [67].
3. We validate our algorithm by demonstrating an accuracy comparable to that of the popular FreeSurfer algorithm [27] in two different applications.

II. Background - Demons Algorithm

We choose to work with the modified Demons objective function [14], [66]. Let F be the fixed image, M be the moving image and Γ be the desired transformation that deforms the moving image M to match the fixed image F. Throughout this paper, we assume that F and M are scalar images, even though it is easy to extend the results to vector images [70]. We introduce a hidden transformation Υ and seek

Algorithm 1

Demons Algorithm

Data: A fixed image F and moving image M.

Result: Transformation Γ so that $M \circ \Gamma$ is "close" to F.

Set Υ^0 = identity transformation (or some a-priori transformation, e.g., from a previous registration)

repeat

Step 1. Given $\Upsilon^{(t)}$,

Minimize the first two terms of Eq. (3)

[1]There are two versions of the code (matlab and ITK) available at http://sites.google.com/site/yeoyeo02/software/sphericaldemonsrelease. The matlab code is used in the experiments of this paper. The preliminary ITK code [35], [36], [37] can also be found at http://hdl.handle.net/10380/3117.

$$u^{(t)} = \underset{u}{\operatorname{argmin}} \left\| \sum^{-1} \left(F - M \circ \{ \Upsilon^{(t)} \circ u \} \right) \right\|^2 + \frac{1}{\sigma_x^2} \operatorname{dist}\left(\Upsilon^{(t)}, \{ \Upsilon^{(t)} \circ u \} \right), \quad (1)$$

where u is any admissible transformation. Compute $\Gamma^{(t)} = \Upsilon^{(t)} \circ u^{(t)}$.

Step 2. Given $\Gamma^{(t)}$,

Minimize the last two terms of Eq. (3):

$$\Upsilon^{(t+1)} = \underset{\Upsilon}{\operatorname{argmin}} \frac{1}{\sigma_x^2} \operatorname{dist}(\Upsilon, \Gamma^{(t)}) + \frac{1}{\sigma_T^2} \operatorname{Reg}(\Upsilon). \quad (2)$$

until *convergence*;

$$(\Upsilon^*, \Gamma^*) = \underset{\Upsilon,\Gamma}{\operatorname{argmin}} \| \sum^{-1} (F - M \circ \Gamma) \|^2 + \frac{1}{\sigma_x^2} \operatorname{dist}(\Upsilon, \Gamma) + \frac{1}{\sigma_T^2} \operatorname{Reg}(\Upsilon). \quad (3)$$

In this case, the fixed image F and warped moving image $M \circ \Gamma$ are treated as $N \times 1$ vectors. Typically, $\operatorname{dist}(\Upsilon, \Gamma) = \|\Upsilon - \Gamma\|^2$, encouraging the resulting transformation Γ to be close to the hidden transformation Υ and $\operatorname{Reg}(\Upsilon) = \| \nabla(\Upsilon - \mathrm{Id})\|^2$, i.e., the regularization penalizes the gradient magnitude of the displacement field $\Upsilon - \mathrm{Id}$ of the hidden transformation Υ. σ_x and σ_t provide a tradeoff among the different terms of the objective function. Σ is typically a diagonal matrix that models the variability of a feature at a particular voxel. It can be set manually or estimated during the construction of an atlas.

This formulation facilitates a two-step optimization procedure that alternately optimizes the first two (first and second) and last two (second and third) terms of Eq. (3). Starting from an initial displacement field Υ^0, the Demons algorithm iteratively seeks an update transformation to be composed with the current estimate, as summarized in Algorithm 1.

In the original Demons algorithm [57], the space of admissible warps includes all 3D displacement fields, and the composition operator $\circ$ corresponds to the addition of displacement fields. The resulting transformation might therefore be not invertible. In the Diffeormorphic Demons algorithm [66], the update u is a diffeormorphism from $\mathbb{R}^3$ to $\mathbb{R}^3$ parameterized by a stationary velocity field $\vec{v}$. Note that $\vec{v}$ is a function that associates a tangent vector with each point in $\mathbb{R}^3$. Under certain mild smoothness conditions, a stationary velocity field $\vec{v}$ is related to a diffeomorphism through the exponential mapping $u = \exp(\vec{v})$. In this case, the stationary ODE $\partial x(t)/\partial t = \vec{v}(x(t))$ with the initial condition $x(0) \in \mathbb{R}^3$ yields $\exp(\vec{v})$ as a solution at time 1, i.e., $x(1) = \exp(\vec{v})(x(0)) \in \mathbb{R}^3$. In this case, $\exp(\vec{v})(x(0))$ maps point $x(0)$ to point $x(1)$.

The Demons algorithm and its variants are fast because for certain forms of $\operatorname{dist}(\Upsilon, \Gamma)$ and $\operatorname{Reg}(\Upsilon)$, Step 1 reduces to a non-linear least-squares problem that can be efficiently minimized via Gauss-Newton optimization and Step 2 can be solved by a single convolution of the displacement field Γ with a smoothing kernel. The proof of the reduction of Step 2 to a smoothing operation is illuminating and holds for $\operatorname{dist}(\Upsilon, \Gamma) = \|\Upsilon - \Gamma\|^2$ and any Sobolev norm $\operatorname{Reg}(\Upsilon) = \Sigma_i \sigma_i \| \nabla^i(\Upsilon - \mathrm{Id})\|^2$ [14], [45]. In practice, a Gaussian filter is used without consideration of the actual induced norm [14], [66]. The proof uses Fourier transforms and is therefore specific to the Euclidean domain. Due to differences between the geometry of the sphere and Euclidean spaces, we will see in Section III-D that the reduction of Step 2 to a smoothing operation is only an approximation on the sphere.

III. Spherical Demons

In this section, we demonstrate suitable choices of dist(Υ, Γ) and Reg(Υ) that lead to efficient optimization of the modified Demons objective function in Eq. (3) on the unit sphere $\mathcal{S}^2$. We construct updates u as diffeomorphisms from $\mathcal{S}^2$ to $\mathcal{S}^2$ parameterized by a stationary velocity field v. We emphasize that unlike Diffeomorphic Demons [66], v is a tangent vector field on the sphere and not an arbitrary 3D vector field. A glossary of common terms used throughout the paper is found in Table I.

A. Choice of dist(Υ, Γ)

Suppose the transformations Γ and Υ map a point $x_n \in \mathcal{S}^2$ to two different points $\Gamma(x_n) \in \mathcal{S}^2$ and $\Upsilon(x_n) \in \mathcal{S}^2$ respectively. An intuitive notion of distance between $\Gamma(x_n)$ and $\Upsilon(x_n)$ would be the geodesic distance between $\Gamma(x_n)$ and $\Upsilon(x_n)$. Therefore, we could define $\mathrm{dist}(\Upsilon, \Gamma) = \sum_{n=1}^{N} \mathrm{geodesic}(\Upsilon(x_n), \Gamma(x_n))$. For reasons that will become clear in Section III-D, we prefer to define dist(Υ, Γ) in terms of a tangent vector representation of the transformations Γ and Υ, illustrated in Fig. 1, where the length of the tangent vector encodes the amount of deformation.

Let $T_{x_n}\mathcal{S}^2$ be the tangent space at x_n. We define $\vec{\Gamma}_n \in T_{x_n}\mathcal{S}^2$ to be the tangent vector at x_n pointing along the great circle connecting x_n to $\Gamma(x_n)$. In this work, we set the length of $\vec{\Gamma}_n$ to be equal to the sine of the angle between x_n and $\Gamma(x_n)$. With this particular choice of length, there is a one-to-one correspondence between $\Gamma(x_n)$ and $\vec{\Gamma}_n$, assuming the angle between x_n and $\Gamma(x_n)$ is less than $\pi/2$, which is a reasonable assumption even for relatively large deformations. The choice of this length leads to a compact representation of $\vec{\Gamma}_n$ via vector products. We define G_n to be the 3 × 3 skew-symmetric matrix representing the cross-product of x_n with any vector:

$$G_n = \begin{pmatrix} 0 & -x_n(3) & x_n(2) \\ x_n(3) & 0 & -x_n(1) \\ -x_n(2) & x_n(1) & 0 \end{pmatrix}, \quad (4)$$

where $x_n(i)$ is the *i-th* coordinate of x_n. Thus, $x_n \times \Gamma(x_n) = G_n\Gamma(x_n)$. Then on a unit sphere, we obtain

$$\vec{\Gamma}_n = -x_n \times (x_n \times \Gamma(x_n)) = -G_n^2\Gamma(x_n). \quad (5)$$

A more intuitive choice for the length of $\vec{\Gamma}_n$ might be the geodesic distance between x_n and $\Gamma(x_n)$. If we restrict $\vec{\Gamma}_n$ to be at most length π, there is a one-to-one mapping between this choice of the tangent vector $\vec{\Gamma}_n$ and the resulting transformation $\Gamma(x_n)$. Indeed, such a choice of a tangent vector corresponds to an exponential map of $\mathcal{S}^2$ [39]. The resulting expression for $\vec{\Gamma}_n = \frac{-2G_n^2\Gamma(x_n)}{\|G_n^2\Gamma(x_n)\|}\sin^{-1}\left(\frac{\|\Gamma(x_n) - x_n\|}{2}\right)$ is feasible, but more complicated than Eq. (5). In this paper, for simplicity, we follow the definition in Eq. (5).

Given N vertices $\{x_n\}_{n=1}^N$, the set of transformed points $\{\Gamma(x_n)\}_{n=1}^N$ – or equivalently the tangent vectors $\{\vec{\Gamma}_n\}_{n=1}^N$ – together with a choice of an interpolation function define the transformation Γ everywhere on $\mathcal{S}^2$. Similarly, we can define the transformation Υ or the equivalent tangent vector field $\vec{\Upsilon}$ through a set of N tangent vectors $\{\vec{\Upsilon}_n\}_{n=1}^N$. We emphasize

that these tangent vector fields are simply a convenient representation of the transformations Υ and Γ and should not be confused with the stationary velocity field $\vec{v}$ that will be used later on. We now set

$$\mathrm{dist}(\Upsilon,\Gamma)=\sum_{n=1}^{N}\|\vec{\Upsilon}_n-\vec{\Gamma}_n\|^2, \quad (6)$$

which is well-defined since both $\vec{\Gamma}_n$ and $\vec{\Upsilon}_n$ belong to $T_{x_n}S^2$ for each $n = 1,\ldots, N$.

B. Spherical Demons Step 1

In this section, we show that the update for Step 1 of the Spherical Demons algorithm can be computed independently for each vertex. With our choice of dist(Υ, Γ), step 1 of the algorithm becomes a minimization with respect to the velocity field $\vec{v}\triangleq\{\triangleq_n\in T_{x_n}S^2\}_{n=1}^N$. By substituting $u=\exp(\vec{v})$ and $\mathrm{dist}\left(\Upsilon,\Gamma\right)=\sum_{n=1}^{N}\|\vec{\Upsilon}_n-\vec{\Gamma}_n\|^2$ into Eq. (1), we obtain

$$\vec{v}^{(t)}=\underset{\vec{v}}{\mathrm{argmin}}\, f(\vec{v}) \quad (7)$$

$$=\underset{\vec{v}}{\mathrm{argmin}}\|\Sigma^{-1}\left(F-M\circ\{\Upsilon^{(t)}\circ\exp(\vec{v})\}\right)\|^2 + \frac{1}{\sigma_x^2}\mathrm{dist}\left(\Upsilon^{(t)},\{\Upsilon^{(t)}\circ\exp(\vec{v})\}\right) \quad (8)$$

$$=\underset{\vec{v}}{\mathrm{argmin}}\sum_{n=1}^{N}\frac{1}{\sigma_n^2}\left(F(x_n)-M\circ\{\Upsilon^{(t)}\circ\exp(\vec{v})\}(x_n)\right)^2 + \frac{1}{\sigma_x^2}\sum_{n=1}^{N}\|\vec{\Upsilon}_n^{(t)}+G_n^2\{\Upsilon^{(t)}\circ\exp(\vec{v})\}(x_n)\|^2, \quad (9)$$

where σ_n^2 is the n-th diagonal entry of Σ and $\circ$ denotes warp composition.

Defining Coordinate Charts on the Sphere

The cost function in Eq. (9) is a mapping from the tangent bundle TS^2 to the real numbers $\mathbb{R}$. We can think of each tangent vector v_n as a 3×1 vector in $\mathbb{R}^3$ tangent to the sphere at x_n. Therefore v_n has 2 degrees of freedom and Eq. (9) represents a constrained optimization problem. Instead of dealing with the constraints, we employ coordinate charts that are diffeomorphisms (smooth and invertible mappings) between open sets in R^2 and open sets on S^2. The differential of the coordinate chart establishes correspondences between the tangent bundles $T\mathbb{R}^2$ and TS^2 [39], [44], so we can reparameterize the constrained optimization problem into an unconstrained one in terms of $T\mathbb{R}^2$ (see Fig. 2).

It is a well-known fact in differential geometry that covering S^2 requires at least two coordinate charts. Since the tools of differential geometry are coordinate-free [39], [44], our results are independent of the choice of the coordinate charts. Let $\vec{e}^{n1}$, $\vec{e}^{n2}$ be any two orthonormal 3×1 vectors tangent to the sphere at x_n, where orthonormality is defined via the usual Euclidean inner product in $3D$. In this work, for each mesh vertex x_n, we define a local coordinate chart Ψ_n: $\mathbb{R}^2 \mapsto S^2$,

$$\Psi_n(x') = \frac{x_n + E_n x'}{\|x_n + E_n x'\|}, \text{ where } E_n = [\vec{e}^{n1}\ \vec{e}^{n2}]. \quad (10)$$

As illustrated in Fig. 2, $\Psi_n(0) = x_n$. Let $\vec{z}_n$ be a 2 × 1 tangent vector at the origin of $\mathbb{R}^2$. With the choice of the coordinate chart above, the corresponding tangent vector at x_n is given by the differential of the mapping $D\Psi_n(\cdot)$ evaluated at $x' = 0$:

$$\vec{v}_n = D\Psi_n(0)\vec{z}_n \quad (11)$$

$$= \frac{I_{3\times 3} - \Psi_n(0)\Psi_n^T(0)}{\|\Psi_n(0)\|} E_n \vec{z}_n \quad (12)$$

$$= \frac{I_{3\times 3} - x_n x_n^T}{\|x_n\|} E_n \vec{z}_n \quad (13)$$

$$= E_n \vec{z}_n = [\vec{e}^{n1}\ \vec{e}^{n2}]\vec{z}_n. \quad (14)$$

The above equation defines the mapping of a tangent vector $\vec{z}_n$ at the origin of $\mathbb{R}^2$ to the tangent vector $\vec{v}_n$ at x_n via the differential of the coordinate chart $D\Psi_n$ at $x' = 0$. We note that for a tangent vector at an arbitrary point in $\mathbb{R}^2$, the expression for the corresponding tangent vector on the sphere is more complicated. This motivates our definition of a separate chart for each mesh vertex, to simplify the derivations.

Gauss-Newton Step of Spherical Demons

From Eq. (14), we obtain $\exp(\vec{v}) = \exp(\{\vec{v}_n\}) = \exp(\{E_n\vec{z}_n\})$ and rewrite Eq. (9) as an unconstrained optimization problem:

$$\begin{aligned} \{\vec{z}_n^{(t)}\} = \underset{\{\vec{z}_n\}}{\text{argmin}} & \sum_{n=1}^{N} \frac{1}{\sigma_n^2}\left(F(x_n) - M \circ \{\Upsilon^{(t)} \circ \exp\left(\{E_n \vec{z}_n\}\right)\}(x_n)\right)^2 \\ & + \frac{1}{\sigma_x^2} \sum_{n=1}^{N} \|\vec{\Upsilon}_n^{(t)} + G_n^2 \{\Upsilon^{(t)} \circ \exp\left(\{E_n \vec{z}_n\}\right)\}(x_n)\|^2, \end{aligned} \quad (15)$$

$$\underset{=}{\Delta} \underset{\{\vec{z}_n\}}{\text{argmin}} \sum_{n=1}^{N} \frac{1}{\sigma_n^2} f_n^2(\vec{z}) + \frac{1}{\sigma_x^2} \sum_{n=1}^{N} \|g_n\|^2(\vec{z}) \quad (16)$$

This non-linear least-squares form can be optimized efficiently with the Gauss-Newton method, which requires finding the gradient of both terms with respect to $\{\vec{z}_n\}$ at $\{\vec{z}_n = 0\}$ and solving a linearized least-squares problem.

We let $\vec{m}_n^T$ be the 1 × 3 spatial gradient of the warped moving image $M \circ \Upsilon^{(t)}(\cdot)$ at x_n and note that $\vec{m}_n^T$ is tangent to the sphere at x_n. The computation of $\vec{m}_n^T$ is discussed in Appendix A-A. Defining $u_n \triangleq \exp(\{E_n\vec{z}_n\})(x_n)$, we differentiate the first term of the cost function $f_n(\vec{z})$ in Eq. (15) using the chain rule, resulting in the 1 × 2 vector:

$$\begin{aligned}&\frac{\partial}{\partial \vec{z}_k}\left[F(x_n) - M \circ \{\Upsilon^{(t)} \circ \exp\left(\{E_n \vec{z}_n\}\right)\}(x_n)\right]_{\vec{z}=0} \\ &= -\frac{\partial}{\partial \vec{z}_k} M \circ \{\Upsilon^{(t)} \circ \exp\left(\{E_n \vec{z}_n\}\right)\}(x_n)\Big|_{\vec{z}=0}\end{aligned} \quad (17)$$

$$\begin{aligned}&= -\frac{\partial M \circ \{\Upsilon^{(t)} \circ \exp\left(\{E_n \vec{z}_n\}\right)\}(x_n)}{\partial \exp\left(\{E_n \vec{z}_n\}\right)(x_n)} \times \\ &\quad \times \left[\frac{\partial \exp\left(\{E_n \vec{z}_n\}\right)(x_n)}{\partial \vec{z}_k}\right]\Big|_{\vec{z}=0}\end{aligned} \quad (18)$$

$$= -\frac{\partial M \circ \Upsilon^{(t)}(u_n)}{\partial u_n}\Big|_{u_n=x_n}\left[\frac{\partial \exp\left(\{E_n \vec{z}_n\}\right)(x_n)}{\partial E_k \vec{z}_k}\frac{\partial E_k \vec{z}_k}{\partial \vec{z}_k}\right]\Big|_{\vec{z}=0} \quad (19)$$

$$= -\vec{m}_n^T E_n \delta(k,n), \quad (20)$$

where $\delta(k, n) = 1$ if $k = n$ and 0 otherwise. Eq. (20) uses the fact that the differential of $\exp(v)$ at $v = 0$ is the identity [47], i.e, $[D\exp(0)]\, v = v$. In other words, a change in velocity v_k at vertex x_k does not affect $\exp(v)(x_n)$ for $n \neq k$ up to the first order derivatives.

Similarly, we define S_n^T to be the 3 × 3 Jacobian of $\Upsilon^{(t)}(\cdot)$ at x_n. The computation of S_n^T is discussed in Appendix A-B. Differentiating the second term of the cost function $g_n(\vec{z})$ in Eq. (15) using the chain rule, we get the 3 × 2 matrix:

$$\begin{aligned}&\frac{\partial}{\partial \vec{z}_k}\left[\vec{\Upsilon}_n^{(t)} + G_n^2\{\Upsilon^{(t)} \circ \exp\left(\{E_n \vec{z}_n\}\right)\}(x_n)\right]_{\vec{z}=0} \\ &= G_n^2 S_n^T E_n \delta(k,n),\end{aligned} \quad (21)$$

where G_n is the skew-symmetric matrix defined in Eq. (4).

Once the derivatives are known, we can compute the corresponding gradients based on our choice of the metric of vector fields on S^2. In this work, we assume an l_2 inner product, so that the inner product of vector fields is equal to the sum of the inner products of the individual vectors. The inner product of individual vectors is in turn specified by the choice of the Riemannian metric on S^2. Assuming the canonical metric, so that the inner product of two tangent vectors is the usual inner product in the Euclidean space [39], the gradients are equal to the transpose of the derivatives Eqs. (20), (21) (see Appendix A-C). We can then rewrite Eq. (15) as a linearized least-squares objective function:

$$\begin{aligned}&\{\vec{z}_n^{(t)}\} \\ &\approx \underset{\{\vec{z}_n\}}{\operatorname{argmin}} \sum_{n=1}^{N} \frac{1}{\sigma_n^2}\left(f_n\left(\vec{z}=0\right) + \nabla_{l_2} f_n \vec{z}\right)^2 \\ &\quad + \frac{1}{\sigma_x^2}\sum_{n=1}^{N} \|g_n(\vec{z}=0) + \nabla_{l_2} g_n \vec{z}\|^2\end{aligned} \quad (22)$$

$$= \underset{\{\vec{z}_n\}}{\operatorname{argmin}} \sum_{n=1}^{N} \frac{1}{\sigma_n^2} \left(\left(F(x_n) - M \circ \Upsilon^{(t)}(x_n) \right) - \vec{m}_n^T E_n \vec{z}_n \right)^2 + \frac{1}{\sigma_x^2} \sum_{n=1}^{N} \| G_n^2 \mathrm{S}_n^T E_n \vec{z}_n \|^2 \quad (23)$$

$$= \underset{\{\vec{z}_n\}}{\operatorname{argmin}} \sum_{n=1}^{N} \left\| \begin{pmatrix} \frac{1}{\sigma_n} \left(F(x_n) - M \circ \Upsilon^{(t)}(x_n) \right) \\ 0 \end{pmatrix} + \begin{pmatrix} -\frac{1}{\sigma_n} \vec{m}_n^T \\ \frac{1}{\sigma_x} G_n^2 \mathrm{S}_n^T \end{pmatrix} E_n \vec{z}_n \right\|^2 . \quad (24)$$

Because of the delta function $\delta(k, n)$ in the derivatives in Eqs. (20), (21), $\vec{z}_n$ only appears in the n-th term of the cost function Eq. (24). The solution of Eq. (24) can therefore be computed independently for each $\vec{z}_n$. Solving this linear least-squares equation yields an update rule for $\vec{z}_n$:

$$\vec{z}_n^{(t)} = \frac{F(x_n) - M \circ \Upsilon^{(t)}(x_n)}{\sigma_n^2} \left(\mathrm{E}_n^T \left[\frac{1}{\sigma_n^2} \vec{m}_n \vec{m}_n^T + \frac{1}{\sigma_x^2} S_n (G_n^2)^T G_n^2 \mathrm{S}_n^T \right] E_n \right)^{-1} \mathrm{E}_n^T \vec{m}_n . \quad (25)$$

For each vertex, we only need to perform matrix-vector multiplication of up to 3 × 3 matrices and matrix inversion of 2 × 2 matrices. This implies the update rule for $\vec{v}_n$:

$$\vec{v}_n^{(t)} = E_n \vec{z}_n^{(t)} \quad (26)$$

$$= \frac{F(x_n) - M \circ \Upsilon^{(t)}(x_n)}{\sigma_n^2} E_n \left(\mathrm{E}_n^T \left[\frac{1}{\sigma_n^2} \vec{m}_n \vec{m}_n^T + \frac{1}{\sigma_x^2} S_n (G_n^2)^T G_n^2 \mathrm{S}_n^T \right] E_n \right)^{-1} \mathrm{E}_n^T \vec{m}_n . \quad (27)$$

In practice, we use the Levenberg-Marquardt modification of Gauss-Newton optimization [49] to ensure matrix invertibility:

$$\vec{v}_n^{(t)} = \frac{F(x_n) - M \circ \Upsilon^{(t)}(x_n)}{\sigma_n^2} E_n \left(\mathrm{E}_n^T \left[\frac{1}{\sigma_n^2} \vec{m}_n \vec{m}_n^T + \frac{1}{\sigma_x^2} S_n (G_n^2)^T G_n^2 \mathrm{S}_n^T \right] E_n + \varepsilon I_{2\times 2} \right)^{-1} \mathrm{E}_n^T \vec{m}_n . \quad (28)$$

where ε is a regularization constant. We note that in the classical Euclidean Demons [57], [14], the term $\mathrm{E}_n^T S_n \left(G_n^2 \right)^T G_n^2 \mathrm{S}_n^T E_n$ turns out to be the identity, so it can also be seen as utilizing Levenberg-Marquardt optimization. Once again, we emphasize that a different choice of the coordinate charts will lead to the same update.

Given $\{\vec{v}_n^{(t)}\}_{n=1}^N$, we use "scaling and squaring" to compute $\exp(\vec{v}^{(t)})$ [3], which is then composed with the current transformation estimate $\Upsilon^{(t)}$ to form $\Gamma^{(t)} = \Upsilon^{(t)} \circ \exp(\vec{v}^{(t)})$. Appendix D discusses implementation details of extending the "scaling and squaring" procedure in Euclidean spaces to S^2.

C. Choice of Reg(ϒ)

We now define the Reg(ϒ) term using the corresponding tangent vector field representation $\vec{\Upsilon}$. Following the work of [31], [61], we let H be the Hilbert space of square integrable vector fields on the sphere defined by the inner product:

$$\left\langle \vec{u}_1, \vec{u}_2 \right\rangle_H = \int_{S^2} \left\langle \vec{u}_1(x), \vec{u}_2(x) \right\rangle_R dS^2, \quad (29)$$

where $\vec{u}_1, \vec{u}_2 \in H$ and $\langle \cdot, \cdot \rangle_R$ refers to the canonical metric. Because vector fields from H are not necessarily smooth, we restrict the deformation $\vec{\Upsilon}$ to belong to the Hilbert space $V \subset H$ of vector fields obtained by the closure of the space of smooth vector fields on S^2 via a choice of the so-called energetic inner product denoted by

$$\left\langle \vec{u}, \vec{v} \right\rangle_V = \left\langle L\vec{u}, \vec{v} \right\rangle_H, \quad (30)$$

where L could for example be the Laplacian operator on smooth vector fields on S^2 [31], [61].

We define $\mathrm{Reg}(\Upsilon) \triangleq \|\vec{\Upsilon}\|_V$. With a proper choice of the energetic inner product (e.g., Laplacian), a smaller value of $\|\vec{\Upsilon}\|_V$ corresponds to a smoother vector field and thus smoother transformation Υ. As we will see later in this section, the exact choice of the inner product is unimportant in our implementation.

D. Optimizing Step 2 of Spherical Demons

With our choice of dist(ϒ, Γ) in Section III-A and Reg(ϒ) in Section III-C, the optimization in Step 2 of the Spherical Demons algorithm

$$\vec{\Upsilon}^{(t+1)} = \underset{\vec{\Upsilon}}{\operatorname{argmin}} \frac{1}{\sigma_x^2} \sum_{n=1}^{N} \| \vec{\Upsilon}_n - \vec{\Gamma}_n^{(t)} \|^2 + \frac{1}{\sigma_T^2} \|\vec{\Upsilon}\|_V \quad (31)$$

seeks a smooth vector field $\vec{\Upsilon} \in V$ that approximates the tangent vectors $\{\vec{\Gamma}_n^{(t)}\}_{n=1}^N$. This problem corresponds to the inexact vector spline interpolation problem solved in [31], motivating our use of tangent vectors in the definition of dist(ϒ, Γ) in Section III-A, instead of the more intuitive choice of geodesic distance.

We can express the tangent vectors $\vec{\Gamma}_n$ and $\vec{\Upsilon}_n$ as $E_n\Gamma_n$ and $E_n\Upsilon_n$ respectively. Essentially, this represents Γ_n and Υ_n in terms of the tangent space basis E_n at x_n, where Γ_n and Υ_n are the components of the tangent vectors with respect to this basis. $\hat{\Gamma}$ and $\hat{\Upsilon}$ be $2N \times 1$ vectors corresponding to stacking Γ_n and Υ_n respectively. The particular optimization formulated in Eq. (31) has a unique optimum [31], given by

$$\hat{\Upsilon} = K\left(\frac{\sigma_x^T}{\sigma_T^2} I_{2N\times 2N} + K\right)^{-1} \hat{\Gamma}, \quad (32)$$

where K is a $2N \times 2N$ matrix consisting of $N \times N$ blocks of 2×2 matrices: the (i,j) block corresponds to $k(x_i,x_j)T_{x_i,x_j}$. The 2×2 linear transformation $T_{x_i,x_j}(\cdot)$ parallel transports a tangent vector along the great circle from $T_{x_i}S^2$ to $T_{x_j}S^2$. $k(x_i,x_j)$ is a non-negative scalar function uniquely determined by the choice of the energetic norm. Typically, $k(x_i,x_j)$

monotonically decreases as a function of the distance between x_i and x_j. The proof of the uniqueness of the global optimum and the form of solution in Eq. (32) follow from the fact that the Hilbert space V is a reproducing kernel hilbert space (RKHS), allowing the exploitation of the Riesz representation theorem [31]. This offers a wide range of choices of regularization depending on the choice of the energetic norm and the corresponding RKHS.

In [31], the spherical vector spline interpolation problem was applied to landmark matching on S^2, resulting in a reasonable sized linear system of equations. Solving the matrix inversion shown in Eq. (32) is unfortunately prohibitive for cortical surfaces with more than 100,000 vertices. If one chooses a relatively wide kernel $k(x_i,x_j)$, the system is not even sparse.

Inspired by the convolution method of optimizing Step 2 in the Demons algorithm [14], [57], [66] and the convolution-based fast fluid registration in the Euclidean space [12], we propose an iterative smoothing approximation to the solution of the spherical vector spline interpolation problem.

In each smoothing iteration, for each vertex x_i, tangent vectors of neighboring vertices x_j are parallel transported to x_i and linearly combined with the tangent vector at x_i. The weights for the linear combination are set to $\lambda(x_i, x_i)=\frac{1}{1+|N_i|\exp(-\frac{1}{2\gamma})}$ and $\lambda\left(x_i, x_j\right)=\frac{\exp(-\frac{1}{2\gamma})}{1+|N_i|\exp(-\frac{1}{2\gamma})}$ for $i \neq j$, where $|N_i|$ is the number of neighboring vertices of x_i. Therefore, larger number of iterations m and values of γ results in greater amount of smoothing.

We note that the iterative smoothing approximation to spline interpolation is not exact because parallel transport is not transitive on S^2 due to the non-flat curvature of S^2 (unlike in Euclidean space), i.e., parallel transporting a tangent vector from point a to b to c results in a vector different from the result of parallel transporting a tangent vector from a to c. Furthermore, the approximation accuracy degrades as the distribution of points becomes less uniform. In Appendix B, we provide a theoretical bound on the approximation error and demonstrate empirically that iterative smoothing provides a good approximation of spherical vector spline interpolation for a relatively uniform distribution of points corresponding to those of the subdivided icosahedron meshes used in this work.

E. Remarks

The Spherical Demons algorithm is summarized in Algorithm 2.

We run the Spherical Demons algorithm in a multi-scale fashion on a subdivided icosahedral mesh. We begin from a subdivided icosahedral mesh (ic4) that contains 2,562 vertices and work up to a subdivided icosahedral mesh (ic7) that contains 163,842 vertices, which is roughly equal to the number of vertices in the cortical meshes we work with. We perform 15 iterations of Step 1 and Step 2 at each level. Because of the fast convergence rate of the Gauss-Newton method, we find that 15 iterations are more than sufficient for our purposes. We also perform a rotational registration at the beginning of each multi-scale level via a sectioned search of the three Euler angles.

Empirically, we find the computation time of the Spherical Demons algorithm is roughly divided equally among the four components: registration by rotation, computing the Gauss-Newton update, performing “scaling and squaring” and smoothing the vector field.

Algorithm 2

Spherical Demons Algorithm

Data: A fixed spherical image F and moving spherical image M.

Result: Diffeomorphism Γ so that $M \circ \Gamma$ is "close" to F.

Set Υ^0 = identity transformation (or some a-priori transformation, e.g., from a previous registration)

repeat

Step 1. Given $\Upsilon^{(t)}$,

foreach *vertex n* **do**

Compute $\overrightarrow{v}_n^{(t)}$ using Eq. (28).

end

Compute $\Gamma^{(t)} = \exp(\bar{v})$ using "scaling and squaring".

Step 2. Given $\Gamma^{(t)}$,

foreach *vertex n* **do**

Compute $\overrightarrow{\Upsilon}_n^{(t)}$ using Eq. (48) implemented via iterative smoothing.

end

until *convergence*;

In practice, we work with spheres that are normalized to be of radius 100, because we find that at ic7, the average edge length of 1mm corresponds to that of the original cortical surface meshes. This allows for meaningful interpretation of distances on the sphere. This requires slight modification of the equations presented previously to keep track of the radius of the sphere.

The Spherical Demons algorithm presented here registers pairs of spherical images. To incorporate a probabilistic atlas defined by a mean image and a standard deviation image, we modify the Demons objective function in Eq. (3), as explained in Appendix C. This requires a choice of warping the subject or warping the atlas. We find that interpolating the atlas gives slightly better results, compared with interpolating the subject. However, interpolating the subject results in a runtime of under 3 minutes, while the runtime for interpolating the atlas is less than 5 minutes. In the next section, we report results for interpolating the atlas.

IV. Experiments

We use two sets of experiments to evaluate the performance of the Spherical Demons algorithm by comparing it to the widely used and freely available FreeSurfer [27] software. The FreeSurfer registration algorithm uses the same similarity measure as Demons, but explicitly penalizes for metric and areal distortion. As we will show, even though the Spherical Demons algorithm does not specifically take into account the original metric properties of the cortical surface, we still achieve comparable if not better registration accuracy than FreeSurfer. Furthermore, FreeSurfer runtime is more than an hour while Spherical Demons runtime is less than 5 minutes.

There are four parameters in the algorithm. $1/\sigma_x^2$ and ε appear in Eq. (28). Larger values of $1/\sigma_x^2$ and ε decrease the size of the update taken in Step 1 of the Spherical Demons algorithm. In the experiments that follow, we set $1/\sigma_x^2 = \varepsilon$ and set their values such that the

largest vector of the update velocity field is roughly two times the edge lengths of the mesh.

The number of iterations m and the weight $\exp(-\frac{1}{2\gamma})$ determine the degree of smoothing. We set $\gamma = 1$ and explore a range of smoothing iterations m in the following experiments.

A. Parcellation of In-Vivo Cortical Surfaces

We validate Spherical Demons in the context of automatic cortical parcellation. Automatic labeling of cortical brain surfaces is important for identifying regions of interests for clinical, functional and structural studies [20], [52]. Recent efforts have ranged from the identification of sulcal/gyral ridge lines [56], [62] to the segmentation of sulcal/gyral basins [20], [28], [38], [41], [42], [51], [52], [67]. Similar to these prior studies, we are interested in parcellation of the entire cortical surface meshes, where each vertex is assigned a label.

We consider a set of 39 left and right cortical surface models extracted from in-vivo MRI [19]. Each surface is spherically parameterized and represented as a spherical image with geometric features at each vertex: mean curvature of the cortical surfaces, mean curvature of the inflated cortical surfaces and average convexity of the cortical surfaces, which roughly corresponds to sulcal depth [26]. These features are intrinsic and thus independent of the parameterization of the surface. The tools used for segmentation [19] and spherical parameterization [26] are freely available [29]. Both hemispheres of each subject were manually parcellated by a neuroanatomist into 35 labels, corresponding to the main sulci and gyri, enumerated in Table II.

We co-register all 39 spherical images of cortical geometry with Spherical Demons by iteratively building an atlas and registering the surfaces to the atlas. The atlas consists of the mean and variance of cortical geometry represented by the surface features described above. We then perform 4-fold cross-validation of the parcellation of the co-registered cortical surfaces. In each iteration of cross-validation, we leave out ten subjects and use the remainder of the subjects to train a classifier [20], [28] that predicts the labels based on location and geometric features. We then apply the classifier to the hold-out set of ten subjects. We perform each iteration with a different hold-out set, i.e., subjects 1-10, 11-20, 21-30 and 31-39.

As mentioned previously, increasing the number of iterations of smoothing results in smoother warps. As discussed in [67], the choice of the tradeoff between the similarity measure and regularization is important for segmentation accuracy. Estimating the optimal registration regularization tradeoff is an active area of research [1], [18], [48], [65], [67], [68] that we do not deal with in this paper. Here, we simply repeat the above experiments using {6, 8, 10, 12, 14} iterations of smoothing. For brevity, we will focus the discussion on using 10 iterations of smoothing and comment on results obtained with the other levels of smoothing.

We repeat the above procedure of performing co-registration and cross-validation with the FreeSurfer registration algorithm [27] using the default FreeSurfer settings. Once again, we use the same features and parcellation algorithm [20], [28]. As before, the atlas consists of the mean and variance of cortical geometry.

To compare the cortical parcellation results, we compute the average Dice measure, defined as the ratio of cortical surface area with correct labels to the total surface area averaged over the test set. Because the average Dice can be misleading by suppressing small structures, we also compute the Dice measure for each structure.

On the left hemisphere, FreeSurfer achieves an average Dice of 88.9, while Spherical Demons achieves an average Dice of 89.6 with 10 iterations of smoothing. While the improvement is not big, the difference is statistically significant for a onesided t-test with the Dice measure of each subject treated as an independent sample ($p = 2 \times 10^{-6}$). Furthermore, the overall Dice is statistically significantly better than FreeSurfer for all levels of smoothing we considered, with the best overal dice achieved with 12 iterations of smoothing.

On the right hemisphere, FreeSurfer obtains a Dice of 88.8 and Spherical Demons achieves 89.1 with 10 iterations of smoothing. Here, the improvement is smaller, but still statistically significant ($p = 0.01$). Furthermore, the overall dice is statistically significantly better than FreeSurfer for all levels of smoothing we considered, except when 6 iterations of smoothing is used ($p = 0.06$). All results we report in the remainder of this section use 10 iterations of smoothing.

We analyze the segmentation accuracy separately for each structure. To compare Spherical Demons with FreeSurfer, we perform a one-sided paired-sampled t-test treating each subject as an independent sample and correct for multiple comparisons using a False Discovery Rate (FDR) of 0.05 [10]. On the left (right) hemisphere, the segmentations of 16 (8) structures are statistically significantly improved by Spherical Demons with respect to FreeSurfer, while no structure is significantly worse.

Fig. 3 shows the percentage improvement of individual structures over FreeSurfer. Fig. 4 displays the average Dice per structure for FreeSurfer and Spherical Demons (10 iterations of smoothing) for the left and right hemispheres. Standard errors of the mean are displayed as red bars. The numbering of the structures correspond to Table II. Parcellation improvements suggest that our registration is at least as accurate as FreeSurfer.

The structures with the worst Dice are the frontal pole and entorhinal cortex. These structures are small and relatively poorly defined by the underlying cortical geometry. For example, the entorhinal cortex is partially defined by the rhinal sulcus, a tiny sulcus that is only visible on the pial surface. The frontal pole is defined by the surrounding structures, rather than by the underlying cortical geometry.

B. Brodmann Area Localization on Ex-vivo Cortical Surfaces

Brodmann areas are cyto-architectonically defined parcellations of the cerebral cortex [13]. They can be observed through histology and more recently, through ex-vivo high resolution MRI [6]. Unfortunately, much of the cytoarchitectonics cannot be observed with current in-vivo imaging. Nevertheless, most studies today report their functional findings with respect to Brodmann areas, usually estimated by visual comparison of cortical folds with Brodmann's original drawings without quantitative analysis of local accuracy. By combining histology and MRI, recent methods for creating probabilistic Brodmann area maps in the Talairach and Colin27 normalized space promise a more principled approach [2], [24], [54], [55], [71].

In this experiment, we consider a data set that contains Brodmann labels mapped to the corresponding MRI volume. Specifically, we work with postmortem histological images of ten brains created using the techniques described in [54], [71]. The histological sections were aligned to postmortem MR with nonlinear warps to build a 3D histological volume. These volumes were segmented to separate white matter from other tissue classes, and the segmentation was used to generate topologically correct and geometrically accurate surface representations of the cerebral cortex using FreeSurfer [19]. The eight manually labeled Brodmann area maps (areas 2, 4a, 4p, 6, 44, 45, 17 and 18) were sampled onto the surface representations of each hemisphere, and errors in this sampling were manually corrected

(e.g., when a label was erroneously assigned to both banks of a sulcus). A morphological close was then performed on each label to remove small holes. We note that Brodmann areas 4a, 4p and 6 were mapped in only eight of the ten subjects. Fig. 5 shows these eight Brodmann areas on the resulting cortical representations for two subjects. Finally, we map the folding patterns and the Brodmann area labels onto a spherical coordinate system [27].

It has been shown that nonlinear surface registration of cortical folds can significantly improve Brodmann area overlap across different subjects [25], [68] compared with volumetric registration. Registering the ex-vivo surfaces is more difficult than in-vivo surfaces because the reconstructed volumes are extremely noisy due to the distortions introduced by the histology, resulting in noisy geometric features, as shown in Fig. 6.

We consider two strategies for aligning Brodmann areas. For both strategies, we will use 10 iterations of smoothing for Spherical Demons as it proved reasonable in the previous set of experiments. The first strategy involves co-registering the 10 ex-vivo surfaces using cortical geometry by repeatedly building an atlas and registering the surfaces to the atlas, similar to the previous experiment on cortical parcellation. We use either Spherical Demons or FreeSurfer for registration. We refer to the co-registration using Spherical Demons and FreeSurfer as SD10 and FS10 respectively (10 refers to the number of subjects in the study, not the number of smoothing iterations).

The second strategy involves registering the 10 ex-vivo surfaces to the in-vivo “Buckner40” atlas, constructed from 40 in-vivo subjects, that is distributed with the FreeSurfer software. Once again, we use either Spherical Demons or FreeSurfer for the registration. We refer to the co-registration using Spherical Demons and FreeSurfer as SD40 and FS40 respectively.

To measure the quality of alignment of the Brodmann areas after cortical registration, we use an adaptation of the modified Hausdorff distance [21]. For each pair of registered subjects, we project each Brodmann area from the first subject onto the second subject and compute the mean distance between the boundaries, measured on the original cortical surface of the second subject. We obtain a second measurement by projecting each Brodmann area from the second subject onto the first subject. Since we have 10 surfaces, we get 90 ordered pairs and 90 alignment errors for each Brodmann area.

Table III reports the mean alignment errors for each Brodmann area and for each method. The lowest errors for each Brodmann area are shown in **bold**. We see that for almost all Brodmann areas, the best alignment come from SD10 or SD40. Similarly, Fig. 7 shows the median alignment error for each Brodmann area. The error bars indicate the lower and upper quartile alignment errors.

We use permutation testing to evaluate statistical significance of the results. We cannot use the t-test because the 90 alignment errors are correlated - since the subjects are co-registered together, good alignment between subjects 1 and 2 and between subjects 2 and 3 necessarily implies a higher likelihood of good alignment between subjects 1 and 3.

The tests show that SD10 is better than FS10 and SD40 is slightly better than FS40. SD10 and SD40 are comparable. Compared with FS10, SD10 improves the median alignment errors of 5 (4) Brodmann areas on the right (left) hemisphere (FDR = 0.05) and no structure gets worse. Compared with FS40, SD40 statistically improves the alignment of 2 (1) Brodmann areas on the right (left) hemisphere (FDR = 0.05) with no structure getting worse. Permutation tests on the mean alignment errors show similar results, except that FS40 performs better than SD40 for BA4p on the left hemisphere when using the mean statistic. These results suggest that the Spherical Demons algorithm is at least as accurate as FreeSurfer in aligning cortical folds and Brodmann areas.

V. Discussion

The Demons algorithms [57], [66] discussed in Section II and the Spherical Demons algorithm proposed in this paper use a regularization term that modulates the final deformation. Motivated by [12], [14], the Diffeomorphic Demons algorithm [66] admits a fluid prior on the velocity fields. This involves smoothing the velocity field updates *before* computing the exponential map to obtain the displacement field updates to be composed with the current transformation. The resulting algorithm is very similar to the fast implementation [12] of Christensen's well-known fluid registration algorithm [16], except that Christensen's algorithm does not employ a higher-order update method like Gauss-Newon. The Spherical Demons algorithm can similarly incorporate a fluid prior by smoothing the velocity field $\vec{v}^{(t)}$ in Eq. (28) before computing the exponential map to obtain the displacement updates $\exp(\vec{v}^{(t)})$.

An alternative interpretation of the smoothing implementation of Christensen's algorithm comes from choosing a different metric for computing the gradient from the derivatives [9]. The choice of the metric also arises in our problem when computing the gradient as discussed in Appendix A-C. This suggests that the Spherical Demons algorithm can incorporate a fluid prior by modifying the Gauss-Newton update step Eq. (28). Unfortunately, this process introduces coupling among the vertices resulting in the loss of the speed-up advantage of Spherical Demons (see for example the derivations of [34]). The exploration of the performance of the different fluid prior implementations is outside the scope of this paper.

Because the tools of differential geometry are general, the Spherical Demons algorithm can be in principle extended to arbitrary manifolds, besides the sphere. One challenge is that the definition of coordinate charts for an arbitrary manifold is more difficult than that for the sphere. Approaches of utilizing the embedding space [15] or the intrinsic properties of manifolds [40] are promising avenues of future work.

VI. Conclusion

In this paper, we presented the fast Spherical Demons algorithm for registering spherical images. We showed that the two-step optimization of the Demons algorithm can also be applied on the sphere. By utilizing the one parameter subgroups of diffeomorphims, the resulting deformation is invertible. We tested the algorithm extensively in two different applications and showed that the accuracy of the algorithm compares favorably with the widely used FreeSurfer registration algorithm [27] while offering more than one order of magnitude speedup. Both matlab and ITK versions of the Spherical Demons algorithm are publicly available[2].

A clear future challenge is to take into account the original metric properties of the cortical surface in the registration process, as demonstrated in previously proposed registration methods [27], [59].

We note that while fast algorithms are useful for deploying the developed tool on large datasets, they can further allow for complex applications that were previously computationally intractable. For example, we have incorporated the ideas behind Spherical Demons into a meta-registration framework that learns registration cost functions which are optimal for specific applications [68].

[2]The matlab code was used for this paper. The ITK code is still preliminary. Please check website http://sites.google.com/site/yeoyeo02/software/sphericaldemonsrelease for updates.

Acknowledgments

We thank Hartmut Mohlberg, Katrin Amunts and Karl Zilles for providing the histological dataset. We also like to thank Xavier Pennec, Wanmei Ou and Serdar Balci for helpful discussions and Tamar Riklin Raviv for reading the conference draft of this paper. We also thank Luis Ibanez and Michel Audette for reading the journal draft of this paper and for their help in implementing the ITK version of the Spherical Demons algorithm. Finally, we thank the reviewers for their many helpful suggestions on improving and clarifying the paper.

Support for this research is provided in part by the NAMIC (NIH NIBIB NAMIC U54-EB005149), the NAC (NIH NCRR NAC P41-RR13218), the mBIRN (NIH NCRR mBIRN U24-RR021382), the NIH NINDS R01-NS051826 grant, the NSF CAREER 0642971 grant, the National Institute on Aging (AG02238), NCRR (P41-RR14075, R01 RR16594-01A1), the NIBIB (R01 EB001550, R01EB006758), the NINDS (R01 NS052585-01) and the MIND Institute. Additional support was provided by The Autism & Dyslexia Project funded by the Ellison Medical Foundation. B.T. Thomas Yeo is funded by the A*STAR, Singapore. Short visits of B.T. Thomas Yeo and Nicholas Ayache between MIT and INRIA were partially funded by the CompuTumor associated teams funding.

Appendix A: Step 1 Gradient Derivation

In this appendix, we provide details on the computation of the spatial gradient of the warped moving image $M \circ \Upsilon^{(t)}$ and the Jacobian of the deformation $\Upsilon^{(t)}$. We also compute the gradients of the demons cost function using the derivatives computed in Eq. (20) and Eq. (21), assuming the l_2 inner product space for vector fields and the canonical metric.

A. Computing Spatial Gradient of $M \circ \Upsilon^{(t)}$

In this appendix, we discuss the computation of m_n^T, the spatial gradient of the warped moving image $M \circ \Upsilon^{(t)}$ at the point x_n. We can think of $M \circ \Upsilon(\cdot)$ as an image $M_s \triangleq M \circ \Upsilon$ defined on the mesh vertices $\{x_n\}$. This image is made continuous by the choice of an interpolation method. In this work, we assume that we are working with a triangular mesh. To evaluate M_s at a point $x \in S^2$, we first find the triangle that contains the intersection between the vector representing the point x (i.e., the vector between the center and the point x of the sphere) and the mesh. The image value at x is then given by the barycentric interpolation of the image values at the intersection point. Mathematically, we can write

$$M_s(x) = I(p(x)), \quad (33)$$

where $p(x)$ is the intersection point and $I(\cdot)$ is the barycentric interpolation. Let p_1, p_2, p_3 denote the vertices of the triangle containing $p(x)$ and $\vec{n}$ denote the 3×1 normal vector to the triangle. Since $p(x) = \alpha x$ for some α and $\langle p(x) - p_1, \vec{n} \rangle = 0$, we can write

$$p(x) = \frac{\langle p_1, \vec{n} \rangle}{\langle x, \vec{n} \rangle} x \quad (34)$$

and

$$I(p) = \frac{A_1(p) M_s(p_1) + A_2(p) M_s(p_2) + A_3(p) M_s(p_3)}{A}, \quad (35)$$

where $A_1(p)$, $A_2(p)$ and $A_3(p)$ are the areas of the triangles $\triangle p p_2 p_3$, $\triangle p p_1 p_3$ and $\triangle p p_1 p_2$ respectively. Note that $A = A_1(p) + A_2(p) + A_3(p)$. $M_s(p_1)$, $M_s(p_2)$ and $M_s(p_3)$ are the image values at the mesh vertices p_1, p_2 and p_3 respectively.

Computing the derivative of the image value at x follows easily from the chain rule:

$$\frac{\partial p(x)}{\partial x} = \frac{\langle p_1, \vec{n} \rangle}{\langle x, \vec{n} \rangle} I_{3\times 3} - \frac{\langle p_1, \vec{n} \rangle}{\langle x, \vec{n} \rangle^2} x\vec{n}^T \quad (36)$$

$$\frac{\partial I(p)}{\partial p} = \frac{\nabla_p A_1(p) M_s(p_1) + \nabla_p A_2(p) M_s(p_2) + \nabla_p A_3(p) M_s(p_3)}{A}, \quad (37)$$

where $\nabla_p A_i(p)$ is the derivative of the triangle area A_i. For example, $\nabla_p A_1(p)$ is a 1×3 vector in the plane of the triangle pp_2p_3, perpendicular and pointing to the edge p_2p_3, with magnitude half the length of p_2p_3. Combining Eq. (36) and Eq. (37) gives the spatial gradient of the warped moving image.

A complication arises when x corresponds to one of the mesh vertices, since the spatial gradient is not defined in this case. The same problem arises in Euclidean space with linear interpolation and the spatial gradient is typically defined via finite central difference. It is unclear what the equivalent definition on a mesh is. Here, for a mesh vertex x, we compute the spatial gradient for each of the surrounding triangles and linearly combine the spatial gradients using weights corresponding to the areas of the triangles.

B. Computing the Jacobian of Deformation $\Upsilon^{(t)}$

In this appendix, we discuss the computation of S_n^T, the Jacobian of the deformation $\Upsilon^{(t)}$ at x_n. We can think of $\Upsilon^{(t)}$ as a vector function on S^2 that maps each mesh vertex $\{x_n\}$ to a new point on the sphere. This vector image is made continuous by the choice of an interpolation method. We use the same interpolation as in Appendix A-A, except we need to normalize the barycentric interpolation so that the interpolated point is constrained to be on the sphere:

$$\Upsilon^{(t)}(x) = I(p(x)) \quad (38)$$

where $p(x)$ is the same as in the previous section and

$$I(p) = \frac{A_1(p)\Upsilon^{(t)}(p_1) + A_2(p)\Upsilon^{(t)}(p_2) + A_3(p)\Upsilon^{(t)}(p_3)}{\|A_1(p)\Upsilon^{(t)}(p_1) + A_2(p)\Upsilon^{(t)}(p_2) + A_3(p)\Upsilon^{(t)}(p_3)\|} \quad (39)$$

The Jacobian is computed via chain rule, just like in the previous section.

C. Computing the Gradients from the Derivatives

In this appendix, we seek to compute the gradients of $f_n(\vec{z}) \triangleq F(x_n) - M \circ \{\Upsilon^{(t)} \circ \exp(\{E_n \vec{z}_n\})\}(x_n)$ and $g_n(\vec{z}) \triangleq \vec{\Upsilon}_n^{(t)} + G_n^2\{\Upsilon^{(t)} \circ \exp(\{E_n \vec{z}_n\})\}(x_n)$, assuming a l_2 inner product for vector fields and the canonical metric R for S^2. These assumptions imply that the inner product of two vector fields $\vec{z}^1 = \left\{\vec{z}_k^1\right\}$ and $\vec{z}^2 = \left\{\vec{z}_k^2\right\}$ are given by

$$\langle \vec{z}^1, \vec{z}^2 \rangle_{l_2} = \langle \{E_k \vec{z}_k^1\}, \{E_k \vec{z}_k^2\} \rangle_{l_2} \quad (40)$$

$$= \sum_{k=1}^{N} \langle E_k \vec{z}_k^1, E_k \vec{z}_k^2 \rangle_R \quad (41)$$

$$= \sum_{k=1}^{N} \langle \vec{z}_k^1, \vec{z}_k^2 \rangle_R, \quad (42)$$

where

- Eq. (40) follows from the equivalence of the tangent bundles $T\mathbb{R}^2$ and TS^2 induced by the coordinate charts $\{\Psi_n\}$.
- Eq. (41) is the result of the l_2 assumption that the inner product of vector fields is given by the sum of the inner products of individual vectors.
- Because we assume the canonical metric, each term in the inner product in Eq. (41) is simply the usual inner product between 3 × 1 vectors $E_k \vec{z}_k^1$ and $E_k \vec{z}_k^2$. Since the columns of E_k are orthonormal with respect to the usual inner product and using linearity of the inner product, Eq. (41) implies Eq. (42), i.e., the inner product $\langle \vec{z}^1, \vec{z}^2 \rangle_{l_2}$ can be computed by the sum of the usual inner product between 2 × 1 tangent vectors $\vec{z}_k^1$ and $\vec{z}_k^2$.

Let $df_n(\vec{z})$ be the directional derivative of f_n for any $\vec{z} = \{\vec{z}_k\}$. The directional derivative is *independent* of the choice of metric. Since the derivative of $f_n(\vec{z})$ with respect to $\vec{z}_k$ is a 1 × 2 vector $-\vec{m}_n^T E_n \delta(k, n)$ (Eq. (20)), we get

$$df_n(\vec{z}) = -\vec{m}_n^T E_n \vec{z}_n. \quad (43)$$

Recall that the gradient $\nabla_{l_2} f_n$ of $f_n(\vec{z})$ is defined to be a tangent vector field such that $df_n(\vec{z}) = \langle \nabla_{l_2} f_n, \vec{z} \rangle_{l_2}$ for any $\vec{z} = \{\vec{z}_k\}$. The gradient is therefore *dependent* on the choice of the inner product. From Eq. (42) and Eq. (43), we can write

$$-\vec{m}_n^T E_n \vec{z}_n = df_n(\vec{z}) \quad (44)$$

$$= \langle \nabla_{l_2} f_n, \vec{z} \rangle_{l_2} \quad (45)$$

$$= \sum_{k=1}^{N} \langle \nabla_{l_2} f_n(x_k), \vec{z}_k \rangle_R. \quad (46)$$

Therefore, the gradient $\nabla_{l_2} f_n$ can be written as a $2N \times 1$ vector consisting of N blocks of 2 × 1 vectors, where all the blocks are zeros, except the n-th block is equal to $-\mathrm{E}_n^T \vec{m}_n$.

Similarly, we denote the gradient of $g_n(\vec{z})$ as $\nabla_{l_2} g_n(j)$ for $j = 1, 2, 3$ corresponding to the 3 output components of $g_n(\vec{z})$. The derivative of ∇g_n with respect to $\vec{z}_k$ is a 3 × 2 matrix $G_n^2 \mathrm{S}_n^T E_n \delta(k, n) \triangleq [\vec{a}_{1n} \vec{a}_{2n} \vec{a}_{3n}]^T \delta(k, n)$ (Eq. (21)), where $\vec{a}_{jn}^T \delta(k, n)$ is a 1 × 2 vector corresponding to the derivative of the j-th component of g_n with respect to $\vec{z}_k$. Using the same derivation as before, we can show that $\nabla_{l_2} g_n(j)$ can be written as a $2N \times 1$ vector

consisting of N blocks of 2×1 vectors, where all the blocks are zeros, except the n-th block is equal to $\vec{a}_{jn}$.

Appendix B: Approximating Spline Interpolation with Iterative Smoothing

In this appendix, we demonstrate empirically that iterative smoothing provides a good approximation of spherical vector spline interpolation for a relatively uniform distribution of points corresponding to those of the subdivided icosahedron meshes used in this work. Once again, we work with spheres that are normalized to be of radius 100.

Recall that we seek $\{\vec{\Upsilon}_n\} = \{E_n \Upsilon_n\}$, which is a smooth approximation of the input vector field $\{\vec{\Gamma}_n\} = \{E_n \Gamma_n\}$. The solution of the spherical vector spline interpolation problem is given in Eq. (32) as

$$\widehat{\Upsilon} = K\left(\frac{\sigma_x^T}{\sigma_T^2} I_{2N\times 2N} + K\right)^{-1} \widehat{\Gamma}, \quad (47)$$

where K is a $2N \times 2N$ matrix consisting of $N \times N$ blocks of 2×2 matrices: the (i, j) block corresponds to $k(x_i, x_j) T_{x_i,x_j}$. T_{x_i,x_j} is the parallel transport operator from x_i to x_j. $k(x_i, x_j)$ is a non-negative scalar function uniquely determined by the choice of the energetic norm that monotonically decreases as a function of the distance between x_i and x_j.

In constrast, the iterative smoothing approximation we propose can be formalized as follows:

$$\widehat{\Upsilon} = (K')^m \widehat{\Gamma} \quad (48)$$

where m is a positive integer and K' is a $2N \times 2N$ matrix consisting of $N \times N$ blocks of 2×2 matrices: the (i, j) block corresponds to $\lambda(x_i, x_j) T_{x_i,x_j}$ if x_i and x_j are neighboring vertices and is a zero matrix otherwise. $\lambda(x_i, x_i) = \frac{1}{1 + |N_i| \exp(-\frac{1}{2\gamma})}$ and $\lambda(x_i, x_j) = \frac{\exp(-\frac{1}{2\gamma})}{1 + |N_i| \exp(-\frac{1}{2\gamma})}$ for $i \neq j$, where $|N_i|$ is the number of neighboring vertices of x_i.

A. Reverse Engineering the Kernel

We now demonstrate empirically that for a range of values of γ, iterations m and the relatively uniform distribution of points corresponding to those of the subdivided icosahedron mesh, there exist kernels $k(x_i, x_j)$ that are well approximated by iterative smoothing. Technically, the resulting $k(x_i, x_j)$ might not correspond to a true choice of the energetic norm. However, in practice, this does not appear to be a problem.

More specifically, given a configuration of mesh points, iterations m and value of γ, we seek $\tilde{k}(x_i, x_j)$, such that $K\left(\frac{\sigma_x^T}{\sigma_T^2} I_{2N\times 2N} + K\right)^{-1}$ is "close" to $(K')^m$. We propose a two-stage estimation of $\tilde{k}(x_i, x_j)$:

1. In the first stage, we seek $k^*(x_i, x_j)$ that is *not* constrained to be a function of the distance between x_i and x_j, such that

$$K\left(\frac{\sigma_x^T}{\sigma_T^2}I_{2N\times 2N}+K\right)^{-1}-(K')^m\approx 0 \quad (49)$$

Rearranging the terms, we get

$$\left(I_{2N\times 2N}-(K')^m\right)^{-1}(K')^m\frac{\sigma_x^2}{\sigma_T^2}\approx K \quad (50)$$

To make the "≈" concrete, we optimize for

$$k^*=\underset{k}{\operatorname{argmin}}\left\|K-\left(I_{2N\times 2N}-(K')^m\right)^{-1}(K')^m\frac{\sigma_x^2}{\sigma_T^2}\right\|_F^2 \quad (51)$$

where $\|\cdot\|_F$ is the Frobenius norm.

The cost function Eq. (51) can be optimized componentwise, i.e., we can solve for $k^*(x_i, x_j)$ for each pair x_i, x_j. For $\gamma = 1$, $m = 10$ and a subdivided icosahedron mesh with 642 vertices, we plot the resulting $k^*(x_i, x_j)$ as a function of the geodesic distance between x_i and x_j in Fig. 8.

2. In the second stage, we perform a least-squares fit of a b-spline function to the estimated $k^*(x_i, x_j)$ to obtain the final estimate of $\tilde{k}(x_i, x_j)$. Fig. 8 illustrates an example kernel $\tilde{k}(x_i, x_j)$ we obtain (c.f., the kernel illustrated in [31]). We note that an alternative to b-spline interpolation is to fit the coefficients of the general kernel function suggested in Appendix A of [31]. This will guarantee that the estimated kernel corresponds to an energetic norm. We leave exploring this direction to future work.

B. Evaluating Approximation

We now investigate the quality of the estimate $\tilde{k}(x_i, x_j)$ by computing:

$$\left\|K\left(\frac{\sigma_x^T}{\sigma_T^2}I_{2N\times 2N}+K\right)^{-1}-(K')^m\right\|_2^2 \quad (52)$$

where $\|\cdot\|_2$ is the l_2 matrix operator norm. The difference metric Eq. (52) measures the *maximum* l_2 difference between smoothed vector fields obtained from iterative smoothing and spherical vector spline interpolation for *any* possible input vector field $\{\vec{\Gamma}_n\}$ of unit l_2 norm, i.e., $\Sigma_n \|\vec{\Gamma}_n\|^2 = 1$. We note that $\tilde{k}(x_i, x_j)$ can be in principle estimated by minimizing Eq. (52) instead of the proposed 2-stage process. However, the optimization is difficult since evaluating the cost function itself requires finding the largest singular value of a large, non-sparse matrix.

Fig. 9 displays the difference metric we obtained with different values of γ and iterations m for meshes ic2, ic3, ic4 and ic5. Each of the meshes is obtained from recursively subdividing a lower resolution mesh: ic2 indicates that the mesh was obtained from subdividing an icosahedron mesh twice. The number of vertices quadruples with each subdivision, so that ic5 corresponds to 10,242 vertices.

We conclude from the figure that the differences between the two smoothing methods are relatively small and increase with mesh resolution. As discussed in the next section, we run Spherical Demons on different mesh resolutions, including ic7. Unfortunately, because of the large non-sparse matrices we are dealing with, we were only able to compute the differences up to ic5. Computing the difference metric for ic5 took an entire week on a machine with 128GB of RAM. However, the plots in Fig. 9 indicate that the differences appear to have converged by ic5.

To better understand the incurred differences, Fig. 10 illustrates the outputs and differences of the two smoothing methods for different inputs on ic4. The first row illustrates an input vector field which is zero everywhere except for a single tangent vector of unit norm. The results of spline interpolation and iterative smoothing correspond to our intuition that smoothing a single tangent vector propagates tangent vectors of smaller magnitudes to the surronding areas. The two methods also produce almost identical results as shown by the clean difference image in the fourth column.

The second row of Fig. 10 demonstrates the worst unit norm input vector field as measured by the difference metric Eq. (52). This worst unit norm input vector field corresponds to the largest eigenvector in Eq. (52). The pattern of large differences correspond to the original 12 vertices of the uniform icosahedron mesh. These original 12 vertices are the only vertices in the subdivided icosahedron meshes with five, instead of six neighbors, as shown by the pentagon pattern. The fact that these 12 vertices are local maxima of differences suggest that these vertices are treated differently by the two smoothing techniques.

The last row of Fig. 10 demonstrates an input vector field that represents the deformation of an actual registration performed in Section IV. The norm of the input vector field is 700 times that in the first two rows, but the discrepancies between spline interpolation and iterative smoothing are less than expected. The differences of 90% of the vectors are less than 0.2mm, with larger differences in the neighborhoods of the 12 vertices identified previously. Since we conclude previously that the difference metric appears to have converged after ic4, the discrepancies are likely to be acceptable at ic7, whose mesh resultion is 1mm.

We should emphasize that the discrepancies between spline interpolation and iterative smoothing do not necessarily imply registration errors. The differences only indicate the deviations of the deformations from true local optima of the Demons registration cost function Eq. (3) assuming the estimated kernel. Approximating smoothing kernels by iterative smoothing is an active area of research in medical imaging [17], [32]. Future work would involve understanding the interaction between the number of smoothing iterations m and the choice of the weights $\exp(-\frac{1}{2\gamma})$ on the quality of the spherical registration.

Appendix C: Atlas-Based Spherical Demons

In this section, we demonstrate how an atlas consisting of a mean image and a standard deviation image can be incorporated into the Spherical Demons algorithm. The standard deviation image replaces Σ in Eq. (3). We first discuss a probabilistic interpretation of the Demons objective function and its relationship to atlases. We then discuss the optimization of the resulting probabilistic objective function.

A. Probabilistic Demons Objective Function

The Demons objective function reviewed in Section II is defined for the pairwise registration of images. To incorporate a probabilistic atlas, we now reformulate the objective function. Consider the following Maximum-A-Posteriori objective function:

$$(\Upsilon^*, \Gamma^*) = \underset{\Upsilon,\Gamma}{\operatorname{argmax}} \log p(\Gamma, \Upsilon | F, M) \quad (53)$$

$$= \underset{\Upsilon,\Gamma}{\operatorname{argmax}} \log p(F, M | \Gamma, \Upsilon)\ p(\Gamma|\Upsilon)\ p(\Upsilon) \quad (54)$$

$$= \underset{\Upsilon,\Gamma}{\operatorname{argmax}} \log p(F, M \circ \Gamma | \Gamma) + \log p(\Gamma|\Upsilon) + \log p(\Upsilon). \quad (55)$$

Assuming a Gaussian noise model, we define

$$p(F, M \circ \Gamma) = \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi}(\sigma(\Gamma, x_n))} \exp\left(-\frac{(F(x_n) - M \circ \Gamma(x_n))^2}{\left(\sqrt{2}\sigma(\Gamma, x_n)\right)^2} \right), \quad (56)$$

$$\log p(\Gamma|\Upsilon) = \frac{1}{Z(\sigma_x^2)} \exp\left(-\frac{1}{\sigma_x^2} \sum_{n=1}^{N} \| \vec{\Upsilon}_n - \vec{\Gamma}_n \|^2 \right), \quad (57)$$

$$p(\Upsilon) = \frac{1}{Z\left(\sigma_T^2\right)} \exp\left(-\frac{1}{\sigma_T^2} \mathrm{Reg}(\Upsilon) \right), \quad (58)$$

where Reg(Υ) is defined via the energetic norm as discussed in Section III-C and for reasons that will soon be clear, we are being purposefully agnostic about the form of $\sigma(\Gamma, x_n)$. The objective function in Eq. (55) becomes

$$(\Upsilon^*, \Gamma^*) = \underset{\Upsilon,\Gamma}{\operatorname{argmin}} \sum_{n=1}^{N} \frac{(F(x_n) - M \circ \Gamma(x_n))^2}{\left(\sqrt{2}\sigma(\Gamma, x_n)\right)^2} + \frac{1}{\sigma_x^2} \sum_{n=1}^{N} \| \vec{\Upsilon}_n - \vec{\Gamma}_n \|^2 + \frac{1}{\sigma_T^2} \mathrm{Reg}(\Upsilon) + \sum_{n=1}^{N} \log \sigma(\Gamma, x_n), \quad (59)$$

which is the instantiation of the Demons objective function Eq. (3), except for the extra term $\sum_{n=1}^{N} \log \sigma(\Gamma, x_n)$. Note that we have omitted the partition functions $Z\left(\sigma_x^2\right)$ and $Z\left(\sigma_T^2\right)$ because σ_x and σ_T are constant with respect to the deformations Γ and (Υ). In this probabilistic interpretation, the two regularization terms $p(\Gamma|(\Upsilon))$ and $p(\Upsilon)$ act as a hierarchical prior on Γ, with the hidden transformation (Υ) as a hyperparameter.

As before, $\sigma(\Gamma, x_n)$ is the standard deviation of the intensity at vertex n. Given a set of co-registered images, we can create an atlas by computing the mean intensity and standard deviation at each vertex. To incorporate the atlas, we need to make the choice of treating the atlas as the fixed or moving image. If we treat the atlas as the fixed image, then we set F to

be the mean image and σ to be the standard deviation. In this case, we do not need to interpolate the mean or standard deviation image. Consequently, $\sigma(\Gamma, x_n) = \sigma(x_n)$ and $\log \sigma(\Gamma, x_n)$ can be omitted from the optimization. The registration becomes identical to the Spherical Demons algorithm for two images.

However, recent work [1], [53] suggests that treating the atlas as a moving image might be more correct theoretically. This involves setting the moving image M to be the mean image. In this case, $\sigma(\Gamma, x_n) = \sigma(\Gamma(x_n))$ is a function of Γ and we must include $\log(\sigma(\Gamma(x_n)))$ in the optimization. We performed experiments for both choices and found the results from interpolating the atlas, i.e., treating it as a moving image, to be only slightly better than interpolating the subject. However, interpolating the subject results in a faster algorithm, whose computational time is less than 3 minutes. We report the results of interpolating the atlas in the experimental section.

B. Optimization of Atlas-Based Spherical Demons

We now discuss the optimization in Eq. (59). Note that the introduction of the new term $\sum_{n=1}^{N} \log \sigma(\Gamma, x_n)$ only affects Step 1 of the Spherical Demons algorithm. By parameterizing $\Gamma^{(t)} = (\Upsilon)^{(t)} \circ \exp(\{E_n \vec{z}_n\})$, we get

$$\begin{aligned}
&\left\{\vec{z}_n^{(t)}\right\} \\
&= \underset{\vec{z}_n}{\operatorname{argmin}} \sum_{n=1}^{N} \frac{\left(F(x_n) - M \circ \left\{\Upsilon^{(t)} \circ \exp\left(\left\{E_n \vec{z}_n\right\}\right)\right\}(x_n)\right)^2}{\left(\sqrt{2}\sigma \circ \left\{\Upsilon^{(t)} \circ \exp\left(\left\{E_n \vec{z}_n\right\}\right)\right\}(x_n)\right)^2} \\
&\quad + \frac{1}{\sigma_x^2} \sum_{n=1}^{N} \left\| \vec{\Upsilon}_n^{(t)} + G_n^2 \left\{\Upsilon^{(t)} \circ \exp\left(\left\{E_n \vec{z}_n\right\}\right)\right\}(x_n) \right\|^2 \\
&\quad + \sum_{n=1}^{N} \log \sigma \circ \left\{\Upsilon^{(t)} \circ \exp\left(\left\{E_n \vec{z}_n\right\}\right)\right\}(x_n)
\end{aligned} \tag{60}$$

$$\triangleq \underset{\vec{z}_n}{\operatorname{argmin}} \sum_{n=1}^{N} f_{n1}^2(\vec{z}) + f_{n2}^2(\vec{z}) + \log f_{n3}(\vec{z}). \tag{61}$$

The second term is the same as before, while the first term has become more complicated. Using the product rule and the techniques described in Appendix A, we can find the first derivatives of the first and second terms and estimate their second derivatives using the Gauss-Newton method. The difficulty lies in the third term, which is not quadratic and is even strictly concave, so we have to make further approximations.

Consider the problem of optimizing a one-dimensional function $f(x)$. Let the current estimate of x be x_0. Newton's optimization [49] involves the following update:

$$\Delta x = -(f'')^{-1}(x_0)\, f'(x_0), \tag{62}$$

where $f'(x_0)$ and $f''(x_0)$ are the gradient and the Hessian of f evaluated at x_0 respectively. When f'' is negative (positive), the update Δx increases (decreases) the objective function, regardless of whether one is attempting to increase or decrease the objective function! The Gauss-Newton approximation of the Hessian for minimizing non-linear quadratic functions actually stabilizes the Newton's method by ensuring the estimated Hessian is positive.

To optimize Eq. (61) with Newton's method, we need to compute the gradient and the Hessian. Because we are using the l_2 inner product and the canonical metric (see Appendix A-C), the gradient and the Hessian correspond to the first and second derivatives. The first derivative or gradient corresponds to

$$\frac{\partial f}{\partial \vec{z}_k} = 2 f_{n1}(\vec{z}) \frac{\partial f_{n1}}{\partial \vec{z}_k} + 2 f_{n2}(\vec{z}) \frac{\partial f_{n2}}{\partial \vec{z}_k} + \frac{1}{f_{n3}} \frac{\partial f_{n3}}{\partial \vec{z}_k} \quad (63)$$

and the second derivative corresponds to

$$\begin{aligned} \frac{\partial^2 f}{\partial \vec{v}'^2_k} = {} & 2\left(\frac{\partial f_{n1}}{\partial \vec{z}_k}\right)^2 + 2 f_{n1}(\vec{z}) \frac{\partial^2 f_{n1}}{\partial \vec{v}'^2_k} + 2\left(\frac{\partial f_{n2}}{\partial \vec{z}_k}\right)^2 + \\ & + 2 f_{n2}(\vec{z}) \frac{\partial^2 f_{n2}}{\partial \vec{v}'^2_k} - \left(\frac{\partial f_{n3}}{\partial \vec{z}_k}\right)^2 + \frac{1}{f_{n3}} \frac{\partial^2 f_{n3}}{\partial \vec{v}'^2_k} \end{aligned} \quad (64)$$

$$\approx 2\left(\frac{\partial f_{n1}}{\partial \vec{z}_k}\right)^2 + 2\left(\frac{\partial f_{n2}}{\partial \vec{z}_k}\right)^2 - \left(\frac{\partial f_{n3}}{\partial \vec{z}_k}\right)^2 . \quad (65)$$

where the last approximation was made using the Gauss-Newton method. Not surprisingly, the third term corresponding to log is negative, which can introduce instability in the Gauss-Newton update. Consequently, we drop the last term, resulting in:

$$\frac{\partial^2 f}{\partial \vec{v}'^2_k} \approx 2\left(\frac{\partial f_{n1}}{\partial \vec{z}_k}\right)^2 + 2\left(\frac{\partial f_{n2}}{\partial \vec{z}_k}\right)^2 . \quad (66)$$

Note that the resulting update Eq. (62) is always in the direction of descent since the estimated second derivative is always positive. Theoretically, it is necessary to do a line search along the Gauss-Newton update direction to ensure convergence. In practice, we find that the objective function decreases reliably for each full Newton's step.

Appendix D: Numerics of Diffeomorphism

While v and $\Phi_v(x) = \exp(v)(x)$ are technically defined on the entire continuous image domain, in practice, v and u are represented by vector fields defined on a discrete set of points in the image, such as at each pixel [57], [66] or control points [4], [9] or in our case, vertices of a spherical mesh. From the theories of ODEs [11], we know that the integral curves or trajectories $u(t) = \Phi_{tv}(\cdot)$ of a velocity field $v(x, t)$ exist and are unique if $v(x, t)$ is Lipschitz continuous in x and continuous in t. This is true in both Euclidean spaces and on manifolds. Uniqueness means that the trajectories do not cross, implying that the deformation is invertible. Furthermore, we know from the theories of ODEs that a C^r continuous velocity field v produces a C^r continuous deformation field. Therefore, a sufficiently smooth velocity field results in a diffeomorphic transformation.

Since the velocity field v is stationary in the case of the one parameter subgroup of diffeomorphisms, v is clearly continuous (and in fact C^∞) in t. A smooth interpolation of v is continuous in the spatial domain and is Lipschitz continuous if we consider a compact domain, which holds since we only consider images that are closed and bounded.

To compute the final deformation of an image, we have to estimate $\exp(\upsilon)$ at least at the set of image grid points. We can compute $\exp(\upsilon)$ by numerically integrating the smoothly interpolated velocity field υ with Euler integration. In this case, the estimate becomes arbitrarily close to the true $\exp(\upsilon)$ as the number of integration time steps increases. With a sufficiently large number of integration steps, we expect the estimate to be invertible and the resulting transformation to be diffeomorphic.

The parameterization of diffeomorphisms by a stationary velocity field is popular due to the "scaling and squaring" approach [3] for computing $\exp(\upsilon)$. Instead of Euler integration, the "scaling and squaring" approach iteratively composes displacement fields. Because we are working on the sphere S^2, the "scaling and squaring" procedure discussed in [3] has to be slightly modified:

$$\Phi_{\frac{1}{2^K}\upsilon}(x)=\left\{\Psi_n\left(E_n\frac{1}{2^K}\upsilon(x_n)\right)\right\}_{n=1,\cdots,N} \tag{67}$$

$$\begin{aligned}\Phi_{\frac{1}{2^{K-1}}\upsilon}(x)&=\Phi_{\frac{1}{2^K}\upsilon}\left(\Phi_{\frac{1}{2^K}\upsilon}(x)\right)\\&\vdots\\\Phi_{\upsilon}(x)&=\Phi_{\frac{1}{2}\upsilon}\left(\Phi_{\frac{1}{2}\upsilon}(x)\right),\end{aligned} \tag{68}$$

where Ψ_n is the local coordinate chart defined in Eq. (10), such that $\Psi_n(0) = x_n$. Eq. (67) differs from "scaling and squaring" in Euclidean space. $E_n\frac{1}{2^K}\upsilon(x_n)$ is the velocity vector at the origin of $\mathbb{R}^2$ corresponding to the velocity vector $\frac{1}{2^K}\upsilon(x_n)$ at x_n. For large enough K, we can approximate a particle at the origin to move to position $E_n\frac{1}{2^K}\upsilon(x_n)$ via the flow of $E_n\frac{1}{2^K}\upsilon(x_n)$. Finally, the coordinate chart Ψ_n maps the point $E_n\frac{1}{2^K}\upsilon(x_n)$ back to the sphere. The correctness of this process follows from the fact that the solution trajectories of the ODEs of a vector field can be consistently transformed via the coordinate charts.

While "scaling and squaring" converges to the true answer as K approaches ∞ in the continuous case, in the discrete case, composition of the displacement fields requires interpolation of displacement fields, introducing errors in the process. In particular, suppose $\Phi_{t_0\upsilon}(x)$ and $\Phi_{2t_0\upsilon}(x)$ are the true trajectories found by performing an accurate Euler integration up to time t_0 and $2t_0$ respectively. Then, there does not exist a trivial interpolation scheme, so that $\Phi_{2t_0\upsilon}(x) = \Phi_{t_0\upsilon}(\Phi_{t_0\upsilon}(x))$. In practice however, it is widely reported that in $\mathbb{R}^2$ and $\mathbb{R}^3$, "scaling and squaring" tends to preserve invertibility even with rather large deformations [4], [66].

As discussed in Appendix A-B, we employ barycentric interpolation, followed by normalization to ensure the warp stays on the unit sphere. In practice, we find that the resulting transformation is indeed diffeomorphic. Technically speaking, since we use linear interpolation for the displacement field, the transformation is only homeomorphic rather than diffeomorphic. This is because the transformation is continuous, but not differentiable across mesh edges. However, we follow the convention of [3], [4], [66] who call their transformation diffeomorphic even though they are homeomorphic.

References

1. Allassonniere S, Amit Y, Trouvé A. Toward a Coherent Statistical Framework for Dense Deformable Template Estimation. Journal of the Royal Statistical Society, Series B. 2007; 69(1):3–29.
2. Amunts K, Malikovic A, Mohlberg H, Schormann T, Zilles K. Brodmann's Areas 17 and 18 Brought into Stereotaxic Space - Where and How Variable? NeuroImage. 2000; 11:66–84. [PubMed: 10686118]
3. Arsigny, V.; Commowick, O.; Pennec, X.; Ayache, N. Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI). Vol. 4190. LNCS; 2006. A Log-Euclidean Framework for Statistics on Diffeomorphisms; p. 924-931.
4. Ashburner J. A Fast Diffeomorphic Image Registration Algorithm. NeuroImage. 2007; 38:95–113. [PubMed: 17761438]
5. Ashburner J, Andersson J, Friston K. High-Dimensional Image Registration using Symmetric Priors. NeuroImage. 1999; 9:619–628. [PubMed: 10334905]
6. Augustinack J, Van der Kouwe A, Blackwell M, Salat D, Wiggins C, Frosch M, Wiggins G, Potthast A, Wald L, Fischl B. Detection of Entorhinal Layer ii Using 7 Tesla Magnetic Resonance Imaging. Annals of Neurology. 2005; 57(4):489–494. [PubMed: 15786476]
7. Avants B, Gee J. Geodesic Estimation for Large Deformation Anatomical Shape Averaging and Interpolation. NeuroImage. 2004; 23:139–150.
8. Bakircioglu M, Joshi S, Miller M. Landmark Matching on Brain Surfaces via Large Deformation Diffeomorphisms on the Sphere. Proc SPIE Medical Imaging. 1999; 3661:710–715.
9. Beg M, Miller M, Trouvé A, Younes L. Computing Large Deformation Metric Mappings via Geodesic Flows of Diffeomorphisms. International Journal on Computer Vision. 2005; 61(2):139–157.
10. Benjamini Y, Hochberg Y. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. Journal of the Royal Statistical Society B. 1995; 57(1):256–300.
11. Birkhoff, G.; Rota, G. Ordinary Differential Equations. John Wiley and Sons Inc; 1978.
12. Bro-Nielsen M, Gramkow C. Fast Fluid Registration of Medical Images. Proceedigs Visualization in Biomedical Computing. 1996; 1131:267–276.
13. Brodmann, K. Vergleichende Lokalisationslehre der Grohirnrinde in Ihren Prinzipien Dargestellt auf Grund des Zellenbaues. 1909.
14. Cachier P, Bardinet E, Dormont D, Pennec X, Ayache N. Iconic Feature Based Non-Rigid Registration: The PASHA algorithm. Compute Vision and Image Understanding. 2003; 89(2-3): 272–298.
15. Chefd'hotel, C. Scale Space and Variational Methods in Computer Vision. Vol. 4485. LNCS; 2007. A Method for the Transport and Registration of Images on Implicit Surfaces; p. 860-870.
16. Christensen G, Rabbit R, Miller M. Deformable Templates Using Large Deformation Kinematics. IEEE Transactions on Image Processing. 1996; 5(10):1435–1447. [PubMed: 18290061]
17. Chung M, Robbins S, Dalton K, Davidson R, Alexander A, Evans A. Cortical Thickness Analysis in Autism with Heat Kernel Smoothing. NeuroImage. 2005; 25(4):1256–1265. [PubMed: 15850743]
18. Commowick, O.; Stefanescu, R.; Fillard, P.; Arsigny, V.; Ayache, N.; Pennec, X.; Malandain, G. Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI). Vol. 3750. LNCS; 2005. Incorporating Statistical Measures of Anatomical Variability in Atlas-To-Subject Registration for Conformal Brain Radiotherapy; p. 927-934.
19. Dale A, Fischl B, Sereno M. Cortical Surface-Based Analysis I: Segmentation and Surface Reconstruction. NeuroImage. 1999; 9(2):179–194. [PubMed: 9931268]
20. Desikan R, Segonne F, Fischl B, Quinn B, Dickerson B, Blacker D, Buckner R, Dale A, Maguire R, Hyman B, Albert M, Killiany R. An Automated Labeling System for Subdividing the Human Cerebral Cortex on MRI Scans into Gyral Based Regions of Interest. NeuroImage. 2006; 31(3): 968–980. [PubMed: 16530430]

21. Dubuisson M, Jain A. A Modified Hausdorff Distance for Object Matching. Proceedings of the 12th IAPR International Conference on Pattern Recognition. 1994; 1:566–568.
22. Durrleman S, Pennec X, Trouvé A, P, Ayache N. Inferring brain variability from diffeomorphic deformations of currents: an integrative approach. Medical Image Analysis. 2008; 12(5):626–637. PMID: 18658005. [PubMed: 18658005]
23. Eckstein, I.; Joshi, A.; Jay Kuo, C.; Leahy, R.; Desbrun, M. Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI). Vol. 4791. LNCS; 2007. Generalized Surface Flows for Deformable Registration and Cortical Matching; p. 692-700.
24. Eickhoff S, et al. A new SPM Toolbox for Combining Probabilistic Cytoarchitectonic Maps and Functional Imaging Data. NeuroImage. 2005; 25(4):1325–1335. [PubMed: 15850749]
25. Fischl B, Rajendran N, Busa E, Augustinack J, Hinds O, Yeo BTT, Mohlberg H, Amunts K, Zilles K. Cortical Folding Patterns and Predicting Cytoarchictecture. Cerebral Cortex. 2008; 18(8):1973–1980. [PubMed: 18079129]
26. Fischl B, Sereno M, Dale A. Surface-Based Analysis II: Inflation, Flattening, and a Surface-Based Coordinate System. NeuroImage. 1999; 9(2):195–207. [PubMed: 9931269]
27. Fischl B, Sereno M, Tootell R, Dale A. High-resolution Inter-subject Averaging and a Coordinate System for the Cortical Surface. Human Brain Mapping. 1999; 8(4):272–284. [PubMed: 10619420]
28. Fischl B, van der Kouwe A, Destrieux C, Halgren E, Segonne F, Salat D, Busa E, Seidman L, Goldstein J, Kennedy D, Caviness V, Makris N, Rosen B, Dale A. Automatically Parcellating the Human Cerebral Cortex. Cerebral Cortex. 2004; 14:11–22. [PubMed: 14654453]
29. http://surfer.nmr.mgh.harvard.edu/fswiki.
30. Geng, X.; Kumar, D.; Christensen, G. Proceedings of the International Conference on Information Processing in Medical Imaging. Vol. 3564. LNCS; 2005. Transitive Inverse-Consistent Manifold Registration; p. 468-479.
31. Glaunès J, Vaillant M, Miller M. Landmark Matching via Large Deformation Diffeomorphisms on the Sphere. Journal of Mathematical Imaging and Vision. 2004; 20:179–200.
32. Hagler D, Saygin A, Sereno M. Smoothing and Cluster Thresholding for Cortical Surface-Based Group Analysis for fMRI Data. NeuroImage. 2006; 33(4):1093–1103. [PubMed: 17011792]
33. Hernandez M, Bossa M, Olmos S. Registration of Anatomical Images Using Geodesic Paths of Diffeomorphisms Parameterized with Stationary Velocity Fields. Proceedings of the Workshop on Mathematical Methods in Biomedical Image Analysis, International Conference on Computer Vision. 2007:1–9.
34. Hernandez M, Olmos S. Gauss-Newton Optimization in Diffeomorphic Registration. Proceedings of the International Symposium on Biomedical Imaging: From Nano to Macro. 2008:1083–1086.
35. Ibanez L, Audette M, Yeo BTT, Golland P. Rotational Registration of Spherical Surfaces Represented as QuadEdge Meshes. Insight Journal. 2009 Jan-Jun;http://hdl.handle.net/10380/3063
36. Ibanez L, Audette M, Yeo BTT, Golland P. Spherical Demons Registration of Spherical Surfaces. Insight Journal. 2009 Jul-Dec;http://hdl.handle.net/10380/3117
37. Ibanez L, Yeo BTT, Golland P. Iterative Smoothing of Field Data in Spherical Meshes. Insight Journal. 2009 Jul-Dec;http://hdl.handle.net/10380/3091
38. Klein A, Hirsch J. Mindboggle: A Scatterbrained Approach to Automate Brain Labeling. NeuroImage. 2005; 24:261–280. [PubMed: 15627570]
39. Kühnel, W. Differential Geometry: Curves-Surfaces-Manifolds. second. American Mathematical Society; 2006.
40. Lefèvre J, Baillet S. Optical Flow and Advection on 2-Riemannian Manifolds: A Common Framework. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2008; 30(6):1081–1092. [PubMed: 18421112]
41. Lohmann G, Von Cramon D. Automatic Labelling of the Human Cortical Surface using Sulcal Basins. Medical Image Analysis. 2000; 4:179–188. [PubMed: 11145307]
42. Mangin JF, Frouin V, Bloch I, Régis J, Lopez-Krahe J. Automatic Construction of an Attributed Relational Graph Representing the Cortex Topography using Homotopic Transformations. Proceedings of SPIE. 1994; 2299:110–121.

43. Marsland, S.; McLachlan, R. Proceedings of the International Conference on Information Processing in Medical Imaging. Vol. 4548. LNCS; 2007. A Hamiltonian Particle Method for Diffeomorphic Image Registration; p. 396-407.
44. Munkres, J. Analysis on Manifold. Westview Press; 1991.
45. Nielsen M, Florack L, Deriche R. Regularization, Scale-Space, and Edge Detection Filters. Journal of Mathematical Imaging and Vision. 1997; 7:291–307.
46. Nielsen, M.; Johansen, P.; Jackson, AD.; Lautrup, B. Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI). Vol. 2489. LNCS; 2002. Brownian Warps: A Least Committed Prior for Non-rigid Registration; p. 557-264.
47. Olver, P. Applications of Lie Groups to Differential Equations. 2nd. Springer-Verlag; 1993.
48. Pennec X, Stefanescu R, Arsigny V, Fillard P, Ayache N. Riemannian Elasticity: A Statistical Regularization Framework for Nonlinear Registration.
49. Press, W.; Teukolsky, S.; Vetterling, W.; Flannery, B. Numerical Recipes in C: The Art of Scientific Computing. second. Cambridge University Press; 1992.
50. Qiu, A.; Miller, M. Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI). Vol. 4791. LNCS; 2007. Cortical Hemisphere Registration via Large Deformation Diffeomorphic Metric Curve Mapping; p. 186-193.
51. Rettmann M, Han X, Xu C, Prince J. Automated Sulcal Segmentation Using Watersheds on the Cortical Surface. NeuroImage. 2002; 15:329–344. [PubMed: 11798269]
52. Riviere D, Mangin JF, Papadopoulos-Orfanos D, Martinez J, Frouin V, Regis J. Automatic Recognition of Cortical Sulci of the Human Brain Using a Congregation of Neural Networks. Medical Image Analysis. 2002; 6:77–92. [PubMed: 12044997]
53. Sabuncu M, Yeo BTT, Vercauteren T, Van Leemput K, Golland P. Assymetric Image-Template Registration. Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI). 2009; 5761:565–573.
54. Schleicher A, Amunts K, Geyer S, Morosan P, Zilles K. Observer-Independent Method for Microstructural Parcellation of Cerebral Cortex: A Quantitative Approach to Cytoarchitectonics. NeuroImage. 1999; 9:165–177. [PubMed: 9918738]
55. Schormann T, Zilles K. Three-Dimensional Linear and Non-linear Transformations: An Integration of Light Microscopical and MRI Data. Human Brain Mapping. 1998; 6:339–347. [PubMed: 9788070]
56. Tao X, Prince J, Davatzikos C. Using a Statistical Shape Model to Extract Sulcal Curves on the Outer Cortex of the Human Brain. IEEE Transactions on Medical Imaging. 2002; 21:513–524. [PubMed: 12071622]
57. Thirion J. Image Matching as a Diffusion Process: an Analogy with Maxwell's Demons. Medical Image Analysis. 1998; 2(3):243–260. [PubMed: 9873902]
58. Thompson P, Toga A. A Surface-Based Technique for Warping 3-Dimensional Images of the Brain. IEEE Transactions on Medical Imaging. 1996; 15(4):1–16.
59. Thompson P, Woods R, Mega M, Toga A. Mathematical/Computational Challenges in Creating Deformable and Probabilistic Atlases of the Human Brain. Human Brain Mapping. 2000; 9(2):81–92. [PubMed: 10680765]
60. Tosun, D.; Prince, J. Proceedings of the International Conference on Information Processing in Medical Imaging. Vol. 3565. LNCS; 2005. Cortical Surface Alignment Using Geometry Driven Multispectral Optical Flow; p. 480-492.
61. Trouvé A. Diffeomorphisms, Groups and Pattern Matching in Image Analysis. International Journal on Computer Vision. 1998; 28(3):213–221.
62. Tu Z, Zheng S, Yuille A, Reiss A, Dutton R, Lee A, Galaburda A, Dinov I, Thompson P, Toga A. Automated Extraction of the Cortical Sulci Based on a Supervised Learning Approach. IEEE Transactions on Medical Imaging. 2007; 26(4):541–552. [PubMed: 17427741]
63. Twining, C.; Davies, R.; Taylor, C. Proceedings of the International Conference on Information Processing in Medical Imaging. Vol. 4584. LNCS; 2007. Non-Parametric Surface-Based Regularisation for Building Statistical Shape Models; p. 738-750.

64. Van Essen D, Drury H, Joshi S, Miller M. Functional and Structural Mapping of Human Cerebral Cortex: Solutions are in the Surfaces. Proceedings of the National Academy of Sciences. 1996; 95(3):788–795.

65. Van Leemput, K. Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI). Vol. 4190. LNCS; 2006. Probabilistic Brain Atlas Encoding Using Bayesian Inference; p. 704-711.

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

67. Yeo BTT, Sabuncu M, Desikan R, Fischl B, Golland P. Effects of registration regularization and atlas sharpness on segmentation accuracy. Medical Image Analysis. 2008; 12(5):603–615. [PubMed: 18667352]

68. Yeo, BTT.; Sabuncu, M.; Golland, P.; Fischl, B. Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI). Vol. 5761. LNCS; 2009. Task-Optimal Registration Cost Functions; p. 598-606.

69. Yeo, BTT.; Sabuncu, M.; Vercauteren, T.; Ayache, N.; Fischl, B.; Golland, P. Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI). Vol. 5241. LNCS; 2008. Spherical Demons: Fast Surface Registration; p. 745-753.

70. Yeo BTT, Vercauteren T, Fillard P, Pennec X, Golland P, Ayache N, Clatz O. DT-REFinD: Diffusion Tensor Registration with Exact Finite-Strain Differential. IEEE Transactions on Medical Imaging. 2009 In Press.

71. Zilles, K.; Schleicher, A.; Palomero-Gallagher, N.; Amunts, K. Quantitative Analysis of Cyto- and Receptor Architecture of the Human Brain. Elsevier; 2002.

72. Zou, G.; Hua, J.; Muzik, O. Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI). Vol. 4791. LNCS; 2007. Non-rigid Surface Registration using Spherical Thin-Plate Splines; p. 367-374.

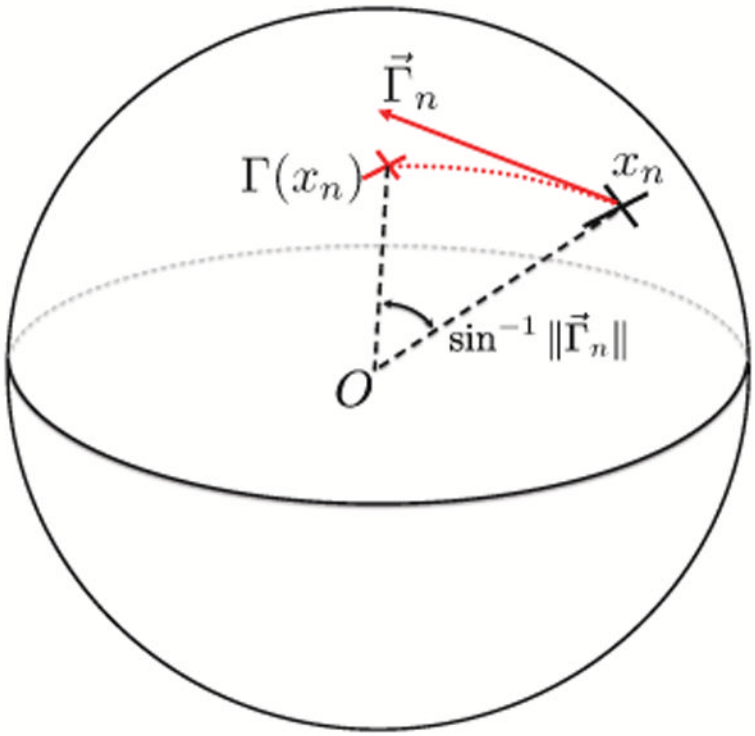

Fig. 1.
Tangent vector representation of transformation Γ. See text for more details.

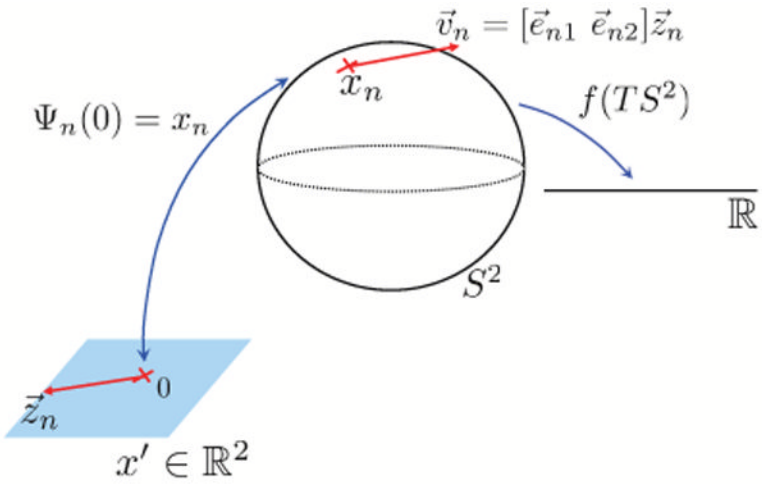

Fig. 2.

Coordinate chart of the sphere S^2. The chart allows a reparameterization of the constrained optimization problem f in step 1 of the Spherical Demons algorithm into an unconstrained one.

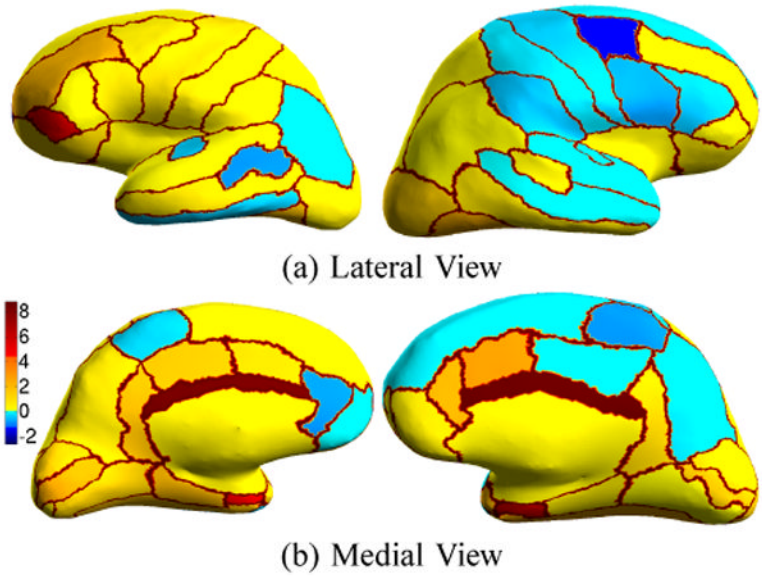

Fig. 3.

Percentage Improvement over FreeSurfer. Yellow regions indicate structures scoring better than FreeSurfer. Blue regions correspond to decrease in accuracy. Note that none of these blue regions are statistically significant. The boundaries between parcellation regions are set to reddish-brown to improve visualization of the regions.

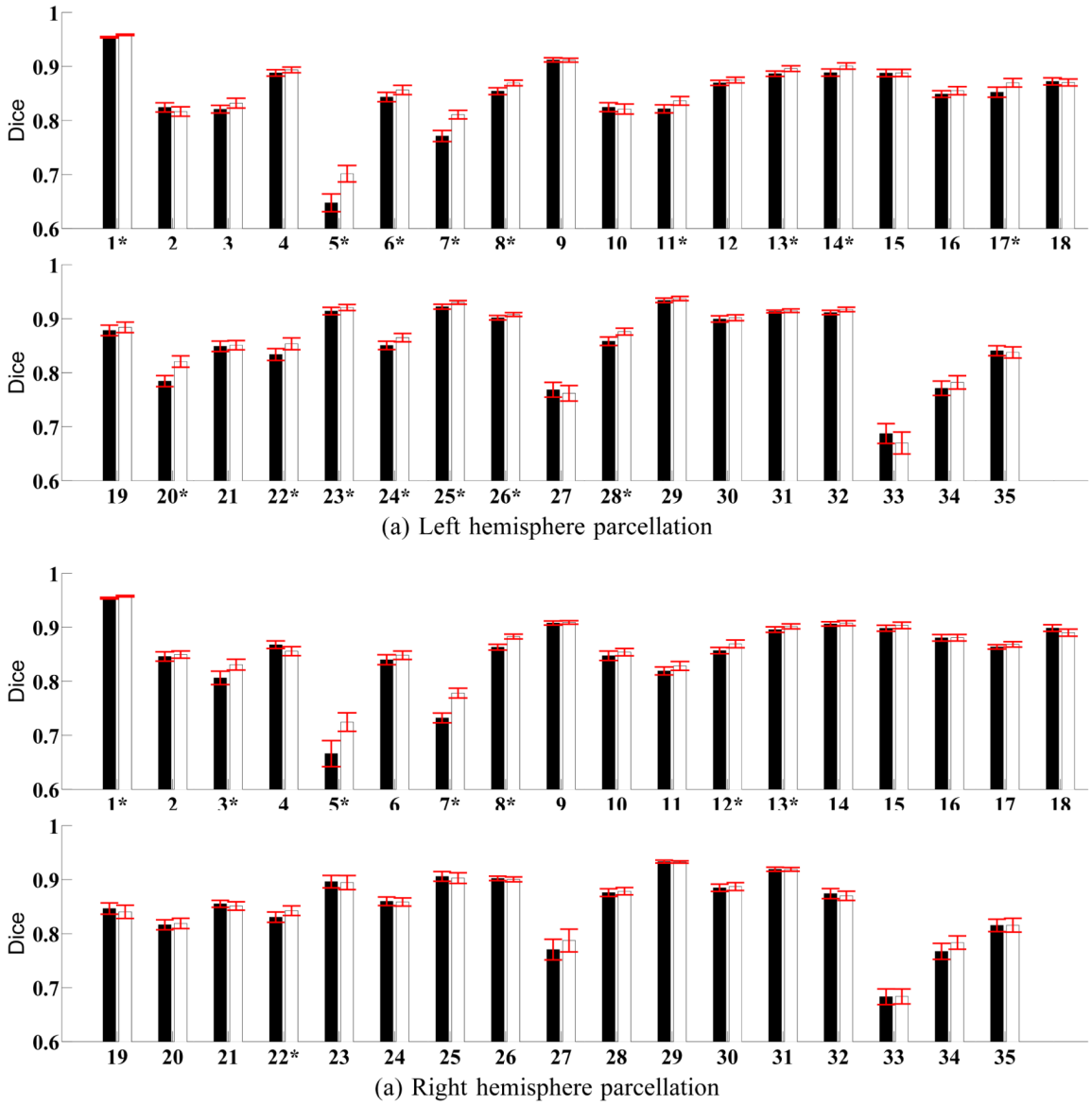

Fig. 4.

(a) Dice measure for each structure in the left hemisphere. (b) Dice measure for each structure in the right hemisphere. Black columns correspond to FreeSurfer. White columns correspond to Spherical Demons. * indicates structures where Spherical Demons shows statistically significant improvement over FreeSurfer (FDR = 0.05). No structure exhibit significant decrease in accuracy.

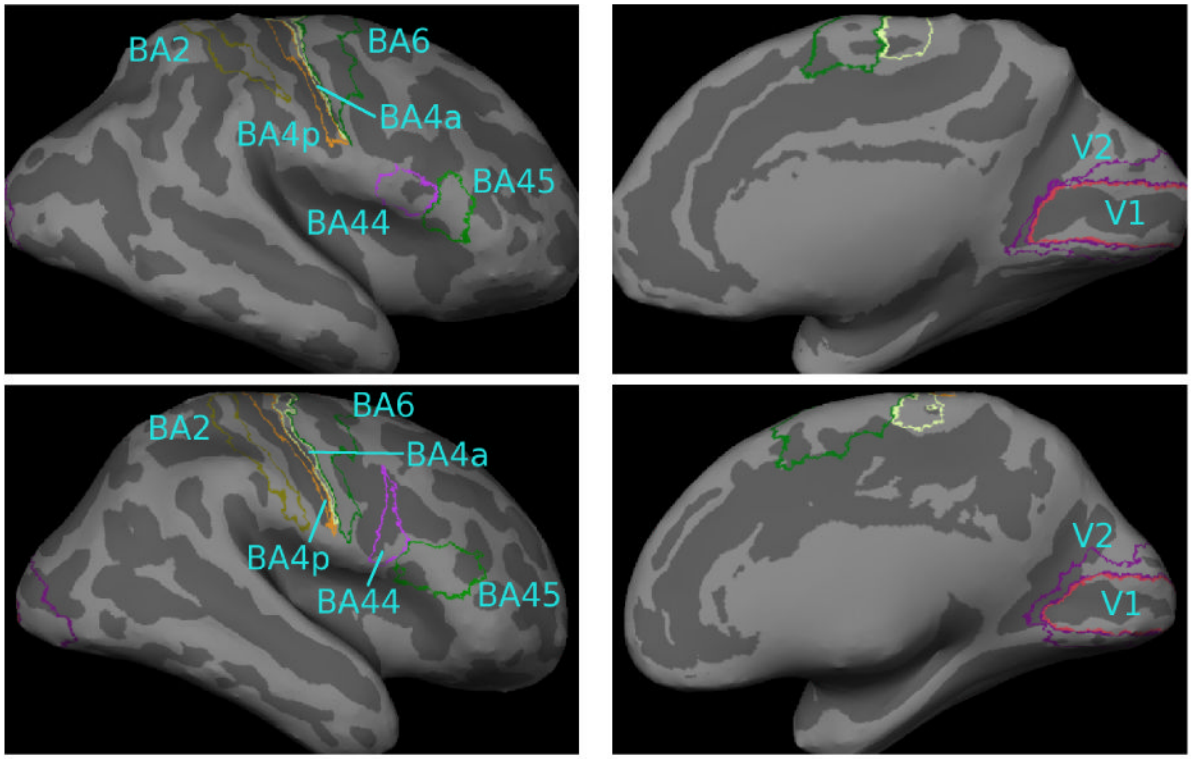

Fig. 5.
Brodmann areas 17 (V1), 18 (V2), 2, 4a, 4p, 6, 44 and 45 shown on inflated cortical surfaces of two subjects. Notice the variability of BA44 and BA45 with respect to the underlying folding pattern.

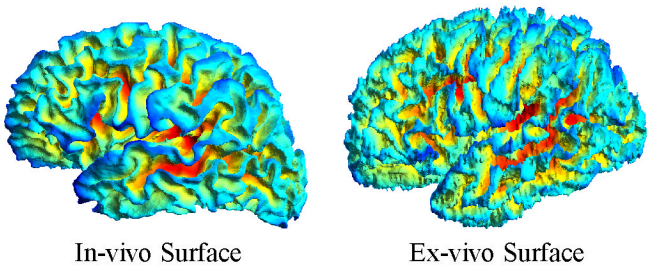

Fig. 6.
Left: example in-vivo surface used in the parcellation study. Right: example ex-vivo surface used in the Brodmann area study.

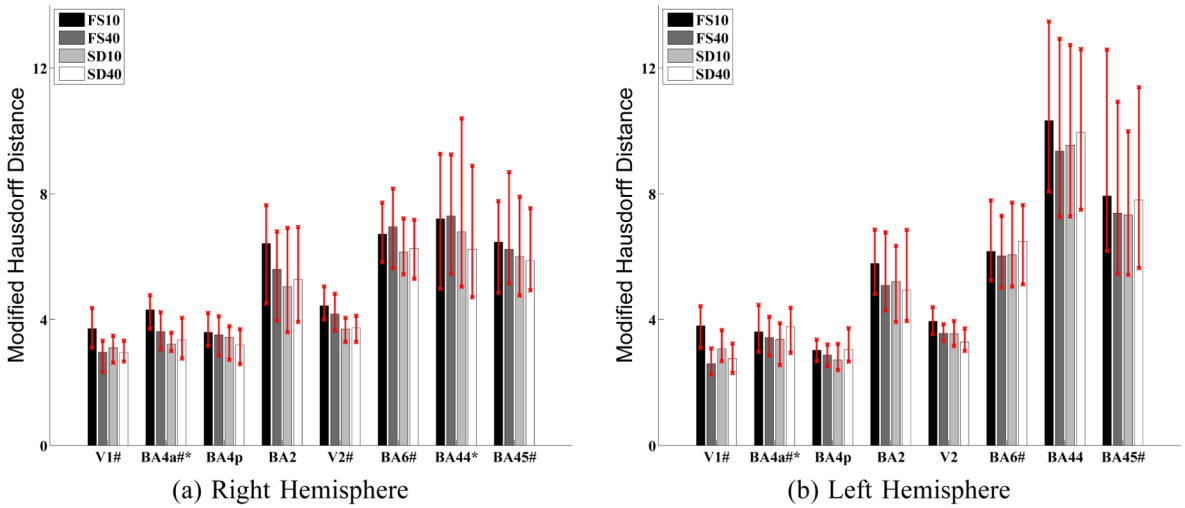

Fig. 7.

Median alignment errors of Brodmann areas in *mm* for the four registration methods. The error bars indicate the upper and lower quartile alignment errors. "#" indicates that the median errors of SD10 are statistically lower than those of FS10 (FDR = 0.05). "*" indicates SD40 outperforms FS40. For no Brodmann area does FreeSurfer outperform Spherical Demons.

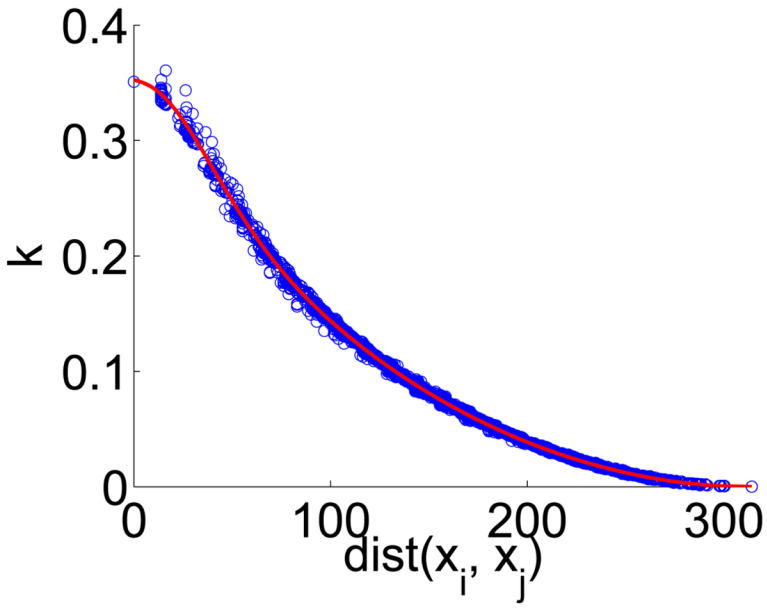

Fig. 8.

Approximating the kernel function $k(x_i, x_j)$. The scattered points corresponds to the estimation of $k^*(x_i, x_j)$ via Eq. (51). The red curve corresponds to fitting the scattered points so that $\tilde{k}(x_i, x_j)$ is strictly a function of the geodesic distance between x_i and x_j.

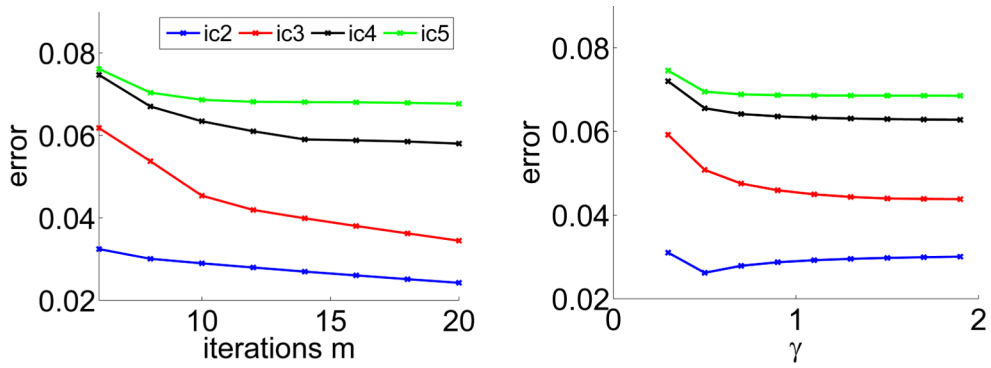

Fig. 9.
Difference metric as a function of the number of iterations m and value of γ.

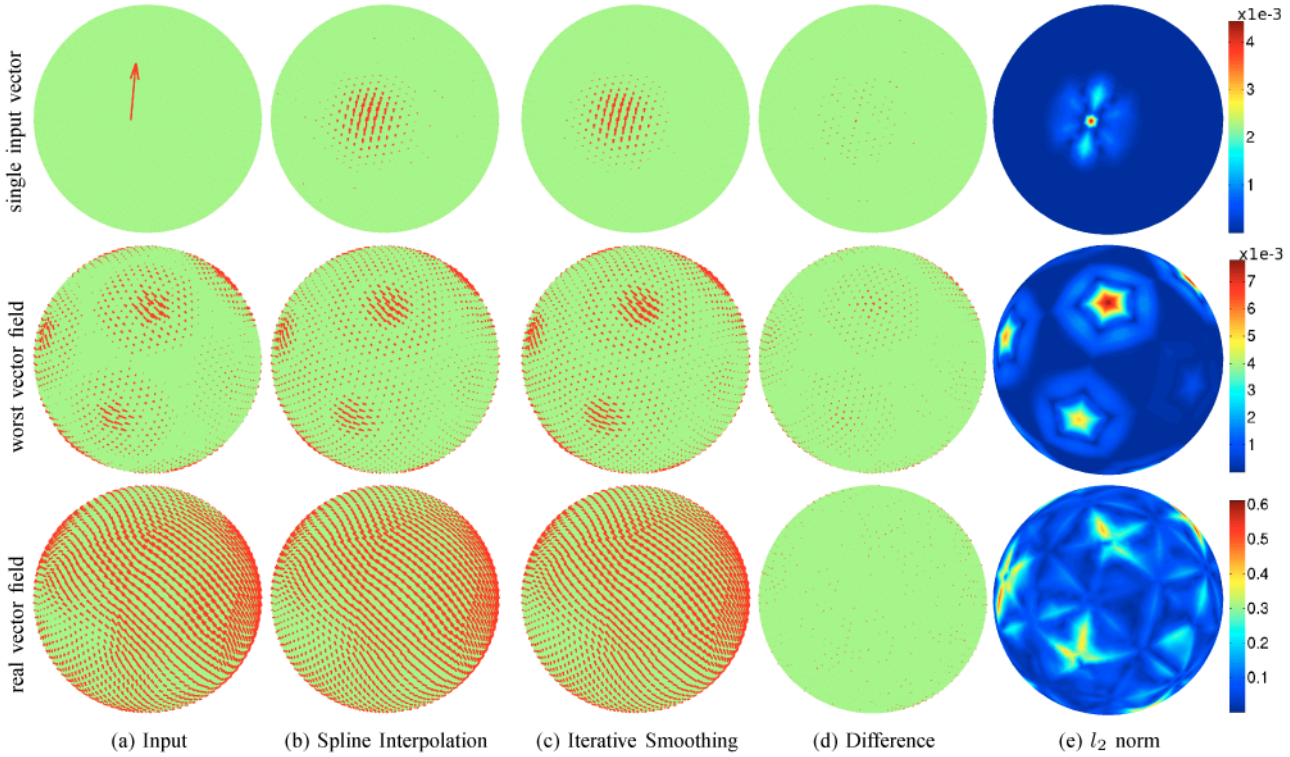

Fig. 10.

Comparison of spline interpolation and iterative smoothing ($m = 10$, $\gamma = 1$). (a) Input vector field (b) Spline Interpolation Output (c) Iterative Smoothing Output (d) Difference between the second and third columns (e) l_2 norm of the difference. The first row uses an input vector field which is zero everywhere except for a single tangent vector of unit norm. Second row illustrates the worst unit norm input as measured by the difference metric Eq. (52). This worst unit norm input vector field corresponds to the largest eigenvector in Eq. (52). The third row uses a vector field from an actual warped image from the experimental section. Note that the input vector field in the first two rows are scaled down for the purpose of display. The vector fields in the entire third row are of the same scale, but are scaled down relative to the first two rows, since the vector field from the warped image is substantially larger in magnitude than the unit norm inputs of the first two rows. This explains the substantially larger difference metric on the third row.

TABLE I

Glossary of terms used throughout the paper.

F, M	Fixed image F, moving image M.
Σ	Typically a diagonal matrix that models variability of feature values at a particular vertex.
σ_x, σ_T	Parameters of Demons cost function in Eq. (3).
Γ, Υ	Transformations from S^2 to S^2. Γ is the transformation we are seeking. Υ is the smooth hidden transformation close to Γ.
$\vec{\Gamma} \triangleq \{\vec{\Gamma}_n\}$, $\vec{\Upsilon} \triangleq \{\vec{\Upsilon}_n\}$	Discrete tangent vector representation of the deformations (see Fig. 1 and Eq. (5)). For example, given the tangent vector $\vec{\Gamma}_n$ at $x_n \in S^2$, one can compute $\Gamma(x_n)$.
$\vec{v} \triangleq \{\vec{v}_n\}$	We parameterize diffeomorphic transformations from S^2 to S^2 by a composition of diffeomorphisms, each parameterized by a stationary velocity field $\vec{v}$. $\vec{v}_n$ is the velocity vector at x_n.
$u(\cdot) \triangleq \exp(\vec{v})(\cdot)$	The diffeomorphism parameterized by the stationary velocity field $\vec{v}$ is the solution of a stationary ODE at time 1.
$E_n \triangleq [\vec{e}^{n1}\ \vec{e}^{n2}]$	$\vec{e}^{n1}$ and $\vec{e}^{n2}$ are orthonormal vectors tangent to the sphere at x_n
Ψ_n	Coordinate chart defined in Eq.(10): $\Psi_n(x') = \frac{x_n + E_n x'}{\|x_n + E_n x'\|}$. Ψ_n is a diffeomorphism between $\mathbb{R}^2$ and a hemisphere centered at $x_n \in S^2$.
$\vec{z}_n$	$\vec{z}_n$ is an arbitrary tangent vector at the origin of $\mathbb{R}^2$. At x_n, the velocity vector $\vec{v}_n = E_n \vec{z}_n$ via the coordinate chart Ψ_n (see Eq. (14)).

TABLE II

List of Parcellation Structures

1. Sylvian Fissure / Unknown	2. Bank of the Superior Temporal Sulcus	3. Caudal Anterior Cingulate
4. Caudal Middle Frontal Gyrus	5. Corpus Callosum	6. Cuneus
7. Entorhinal	8. Fusiform Gyrus	9. Inferior Parietal Complex
10. Inferior Temporal Gyrus	11. Isthmus Cingulate	12. Lateral Occipital
13. Lateral Orbito Frontal	14. Lingual	15. Medial Orbito Frontal
16. Middle Temporal Gyrus	17. Parahippocampal	18. Paracentral
19. Parsopercularis	20. Parsorbitalis	21. Parstriangularis
22. Peri-calcarine	23. Post-central Gyrus	24. Posterior Cingulate
25. Pre-central Gyrus	26. Pre-cuneus	27. Rostral Anterior Cingulate
28. Rostral Middle Frontal	29. Superior Frontal Gyrus	30. Superior Parietal Complex
31. Superior Temporal Gyrus	32. Supramarginal	33. Frontal Pole
34. Temporal Pole	35. Transverse Temporal	

TABLE III

Mean alignment errors of Brodmann areas in *mm* for the four registration methods. Lowest errors are shown in **bold**. SD refers to Spherical Demons; FS refers to FreeSurfer.

	Right Hemisphere								Left Hemisphere							
	V1	**BA4a**	**BA4p**	**BA2**	**V2**	**BA6**	**BA44**	**BA45**	**V1**	**BA4a**	**BA4p**	**BA2**	**V2**	**BA6**	**BA44**	**BA45**
FS10	3.8	4.4	3.8	6.3	4.6	7.0	7.4	6.8	3.8	3.8	3.1	5.9	4.0	6.5	11.5	9.9
FS40	**2.9**	3.8	3.6	5.6	4.2	7.1	7.6	6.9	**2.7**	3.6	2.9	5.7	3.6	**6.3**	10.5	9.2
SD10	3.1	**3.3**	3.3	**5.4**	**3.7**	**6.4**	7.7	6.4	3.2	**3.4**	**2.8**	**5.5**	3.5	6.4	**10.4**	**8.6**
SD40	3.0	3.4	**3.2**	5.5	3.8	**6.4**	**6.8**	**6.3**	2.8	3.8	3.7	5.6	**3.4**	6.6	10.7	9.0