Diffeomorphic Image Registration by Geodesic Shooting

John Ashburner

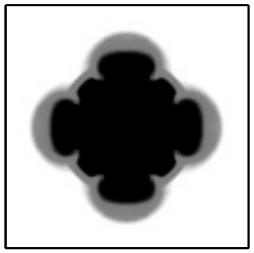
Wellcome Trust Centre for Neuroimaging
UCL Institute of Neurlogy
London, UK.

Outline

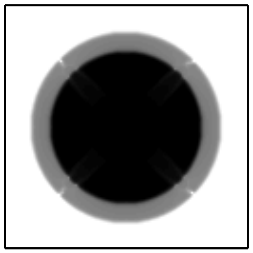
Small Deformation Approximation

$\mu \circ (x - v(x))$

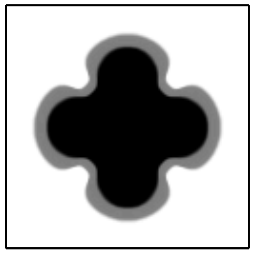
$f \circ (x + v(x))$



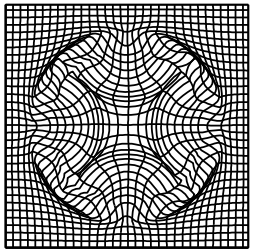
$f \circ (x + v(x))$



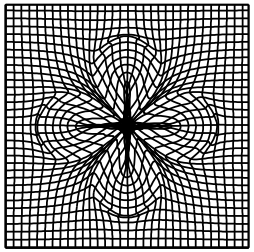
$\mu \circ (x - v(x))$



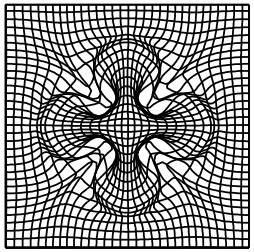
$x - v(x)$



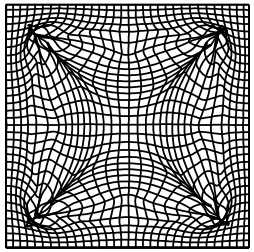
$x + v(x)$



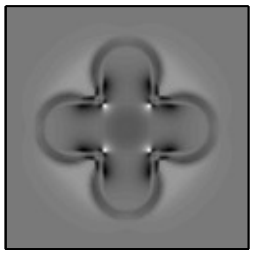
$x + v(x)$



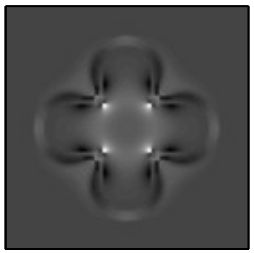
$x - v(x)$



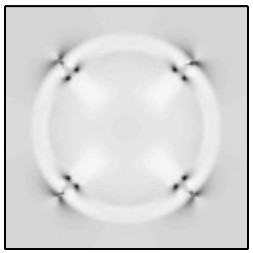
$|J^{x - v(x)}|$



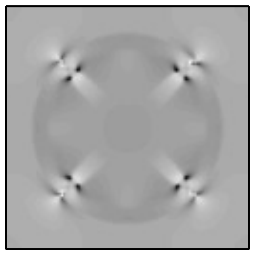
$|J^{x + v(x)}|$



$|J^{x + v(x)}|$



$|J^{x - v(x)}|$



Small Deformation Model

- Should fit a model to the data (as opposed to fitting the data to the model).
- The objective function for least-squares fitting is often expressed as:

$$\mathcal{E} = \frac{1}{2}||\mathbf{Lv}||^2 + \frac{1}{2\sigma^2}||f - \mu\left(\mathrm{Id} - \mathbf{v}\right)||^2$$

  - f is our image (data).
  - μ is the template (model).
  - $\mathbf{v}$ is a displacemet field.
  - Id is an identity transform.
  - $\mathbf{L}$ is a differential operator.

Linear Elasticity Operator

The differential operator used is essentially the same linear elasticity operator used by Christensen (where **D** indicates computing the Jacobian:

$$||\mathbf{Lv}||^2 = \int_{\mathbf{x}\in\Omega} \left(\frac{\lambda_1}{4} ||\mathbf{Dv} + (\mathbf{Dv})^T||^2 + \lambda_2 \mathrm{tr}(\mathbf{Dv})^2 + \lambda_3 ||\mathbf{v}||^2 \right) d\mathbf{x}$$

This has three terms that penalise different things.

1. Length changes ($\mathbf{Dv} + (\mathbf{Dv})^T$).
2. Volume changes ($\mathrm{tr}(\mathbf{Dv})$).
3. Absolute displacement ($\mathbf{v}$ – needed to make the operator invertable).

Discretisation

- Images and deformations are really stored discretely, and made continuous by interpolation.
- Generalised interpolation involves representing fields etc by linear combinations of basis functions.
- Linear interpolation, which involves very local triangular basis functions ($b_i(\mathbf{x})$), so:

$$\mathbf{v}(\mathbf{x}) = \sum_i w_i \mathbf{b}_i(\mathbf{x})$$

Differential Operator in 1D

- In 1D, a simple operator could compute the gradients by $\mathbf{Gw}$, where:

$$\mathbf{G} = \begin{pmatrix} -1 & 1 & 0 & 0 & \dots \\ 0 & -1 & 1 & 0 & \dots \\ 0 & 0 & -1 & 1 & \dots \\ 0 & 0 & 0 & -1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

- Penalty ($||\mathbf{Lv}||^2$) could be the sum of squares of the gradients.
- Computed by $(\mathbf{Gw})^T(\mathbf{Gw}) \equiv \mathbf{w}^T\mathbf{G}^T\mathbf{Gw} \equiv \mathbf{w}^T\mathbf{Aw}$.
- Note that the norm $||\mathbf{Lv}||$ (ie $\sqrt{\mathbf{w}^T\mathbf{Aw}}$) can be thought of as a distance.

Differential Operator in 2D

- For more dimensions, we can use Kroneker tensor products.

$$\mathbf{G} \otimes \mathbf{B} = \begin{pmatrix} g_{11}\mathbf{B} & \cdots & g_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ g_{m1}\mathbf{B} & \cdots & g_{mn}\mathbf{B} \end{pmatrix}$$

- So, computing horizontal and vertical gradients of a square image (reshaped to a column vector) may be achieved using the following matrix:

$$\begin{pmatrix} \mathbf{I} \otimes \mathbf{G} \\ \mathbf{G} \otimes \mathbf{I} \end{pmatrix}$$

Differential Operator for Computing Jacobians (2D)

- If we stack the x component of the displacement field on top of the y component, then we can compute elements of the Jacobians by:

$$\begin{pmatrix} \mathbf{I} \otimes \mathbf{G} & \mathbf{0} \\ \mathbf{G} \otimes \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \otimes \mathbf{G} \\ \mathbf{0} & \mathbf{G} \otimes \mathbf{I} \end{pmatrix} \mathbf{w}$$

- The divergence of the displacement is simply the trace of the Jacobian, so can be computed using:

$$\begin{pmatrix} \mathbf{I} \otimes \mathbf{G} & \mathbf{G} \otimes \mathbf{I} \end{pmatrix} \mathbf{w}$$

Symmetric and Anti-symmetric Matrices

- A square matrix ($\mathbf{J}$) can be decomposed into the sum of a symmetric and anti-symmetric (skew symmetric) matrix.
  - The symmetric part is $(\mathbf{J} + \mathbf{J}^T)/2$.
  - The anti-symmetric part is $(\mathbf{J} - \mathbf{J}^T)/2$.
- The symmetric elements of the Jacobian of the displacement can be obtained from:

$$\begin{pmatrix} \mathbf{I} \otimes \mathbf{G} & \mathbf{0} \\ \mathbf{0} & \mathbf{G} \otimes \mathbf{I} \\ \frac{1}{2}\mathbf{G} \otimes \mathbf{I} & \frac{1}{2}\mathbf{I} \otimes \mathbf{G} \end{pmatrix} \mathbf{w}$$

- Similar large sparse matrices can be generated for 3D data.

Matrix Exponentials

- A matrix exponential ("expm" in MATLAB) is given by:

$$e^{\mathbf{J}} = \sum_{k=0}^{\infty} \frac{1}{k!} \mathbf{J}^k$$

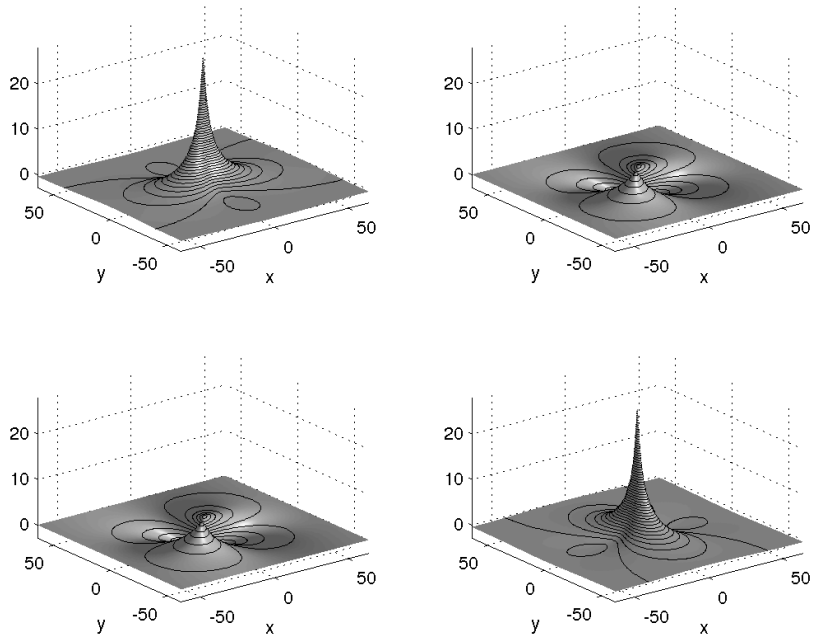
- For matrices very close to zero, $e^{\mathbf{J}} \simeq \mathbf{I} + \mathbf{J}$.
- The determinant $|e^{\mathbf{J}}|$ may be computed from $e^{\mathrm{tr}(\mathbf{J})}$.
- If $\mathbf{J}$ is anti-symmetric, then $e^{\mathbf{J}}$ is orthogonal (ie a rigid rotation).

$$\mathbf{J} = w_1 \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} + w_2 \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} + w_3 \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix}$$

- We do not wish to penalise rotations, so only penalise the symmetric parts of the Jacobians.

Greens Function

- Note that the Green's function ($\mathbf{K}$) of the operator may be computed from the (pseudo)inverse of $\mathbf{A}$, although it is often easier to use Fourier methods.
- If you think of applying a differential operator as a convolution, then applying the Green's function is a deconvolution.



Gauss-Newton Optimisation

An iterative procedure, using both first and second derivatives.

$$\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} - \left[\frac{\partial^2 \mathcal{E}}{\partial \mathbf{w}^2}\bigg|_{\mathbf{w}^{(i)}} \right]^{-1} \left[\frac{\partial \mathcal{E}}{\partial \mathbf{w}}\bigg|_{\mathbf{w}^{(i)}} \right]$$

$$\frac{\partial \mathcal{E}}{\partial \mathbf{w}} = \mathbf{A}\mathbf{w} + \mathbf{g}$$

$$\frac{\partial^2 \mathcal{E}}{\partial \mathbf{w}^2} = \mathbf{A} + \mathbf{H}$$

Gradients

$$\mathbf{g} = \begin{bmatrix} g_{11} \\ g_{21} \\ g_{31} \\ \vdots \\ g_{I1} \\ g_{11} \\ g_{21} \\ g_{31} \\ \vdots \\ g_{I1} \end{bmatrix}$$

Using trilinear interpolation to represent the displacement fields, where the density of the control points is equal to the pixel spacing. Therefore $\boldsymbol{\vartheta}(\mathbf{x}_i) = [x_{i1} - w_{i1}, x_{i2} - w_{i2}]$.

$$g_{il} = \frac{1}{\sigma^2} \left(f(\mathbf{x}_i) - \mu(\boldsymbol{\vartheta}(\mathbf{x}_i))\right) \left((\nabla_l \mu) \circ (\boldsymbol{\vartheta}(\mathbf{x}_i))\right)$$

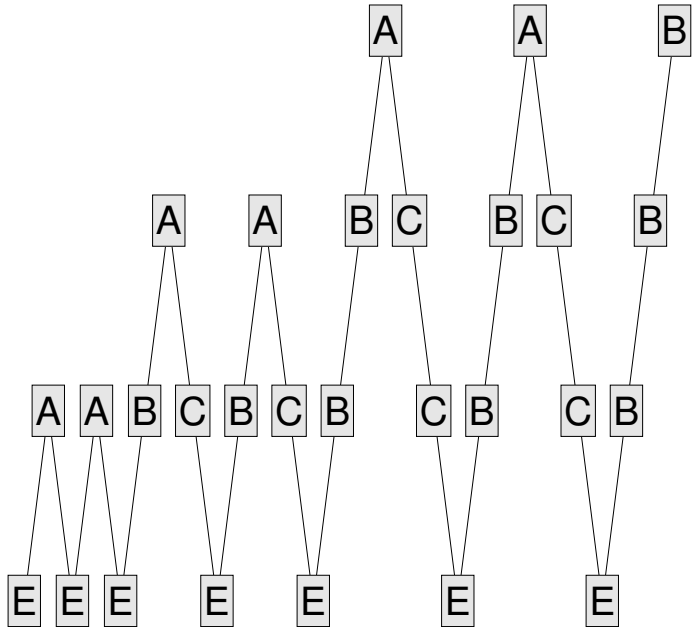
Hessian

$$\mathbf{H} = \begin{bmatrix}
h_{111} & 0 & 0 & \dots & 0 & h_{121} & 0 & 0 & \dots & 0 \\
0 & h_{211} & 0 & \dots & 0 & 0 & h_{221} & 0 & \dots & 0 \\
0 & 0 & h_{311} & \dots & 0 & 0 & 0 & h_{321} & \dots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \dots & h_{I11} & 0 & 0 & 0 & \dots & h_{I21} \\
h_{112} & 0 & 0 & \dots & 0 & h_{122} & 0 & 0 & \dots & 0 \\
0 & h_{212} & 0 & \dots & 0 & 0 & h_{222} & 0 & \dots & 0 \\
0 & 0 & h_{312} & \dots & 0 & 0 & 0 & h_{322} & \dots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \dots & h_{I12} & 0 & 0 & 0 & \dots & h_{I22}
\end{bmatrix}$$

$$h_{ilm} = \frac{1}{\sigma^2}((\nabla_l \mu) \circ \boldsymbol{\vartheta}(\mathbf{x}_i))((\nabla_m \mu) \circ \boldsymbol{\vartheta}(\mathbf{x}_i))$$

Matrix Solution

A multigrid approach used to solve $(\mathbf{A}+\mathbf{H})^{-1}(\mathbf{A}\mathbf{w}+\mathbf{g})$. Based on a relaxation scheme at several resolutions. See Numerical Recipes (2nd edition onwards) for more details.



At The Solution

At the solution, the derivatives of the objective function are zero.

$$\frac{\partial \mathcal{E}}{\partial \mathbf{w}} = \mathbf{A}\mathbf{w} + \mathbf{g} = 0$$

Therefore

$$\mathbf{w} = \mathbf{A}^{-1}\mathbf{g}$$

We see that the displacements are a smoothed version of a difference image multiplied by the warped gradients of the template.

$$\mathbf{v} = \mathbf{K}\left((\mathbf{f} - \boldsymbol{\mu}(\boldsymbol{\vartheta}))((\nabla \boldsymbol{\mu}) \circ \boldsymbol{\vartheta})\right)$$

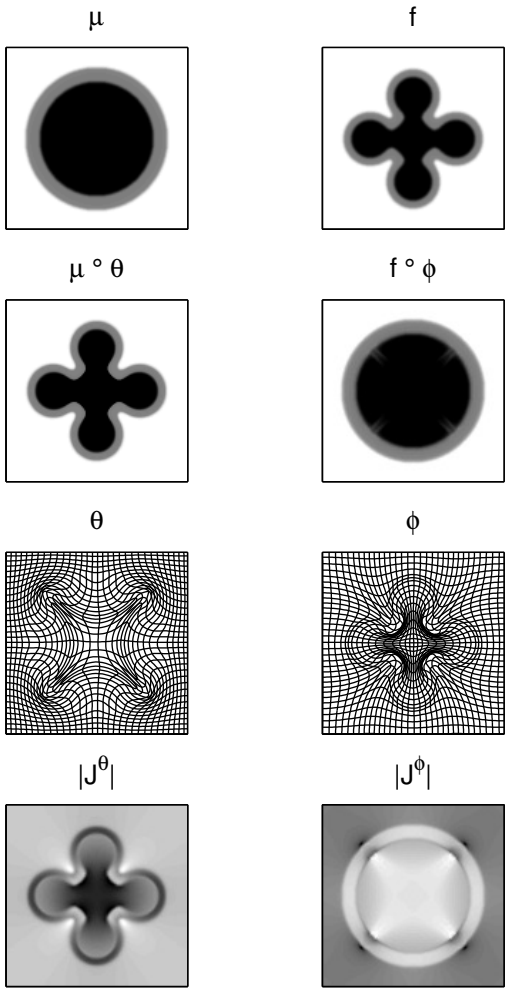
Outline

Nonlinearity of Shapes

According to David Mumford (Fields Medal, 1974):

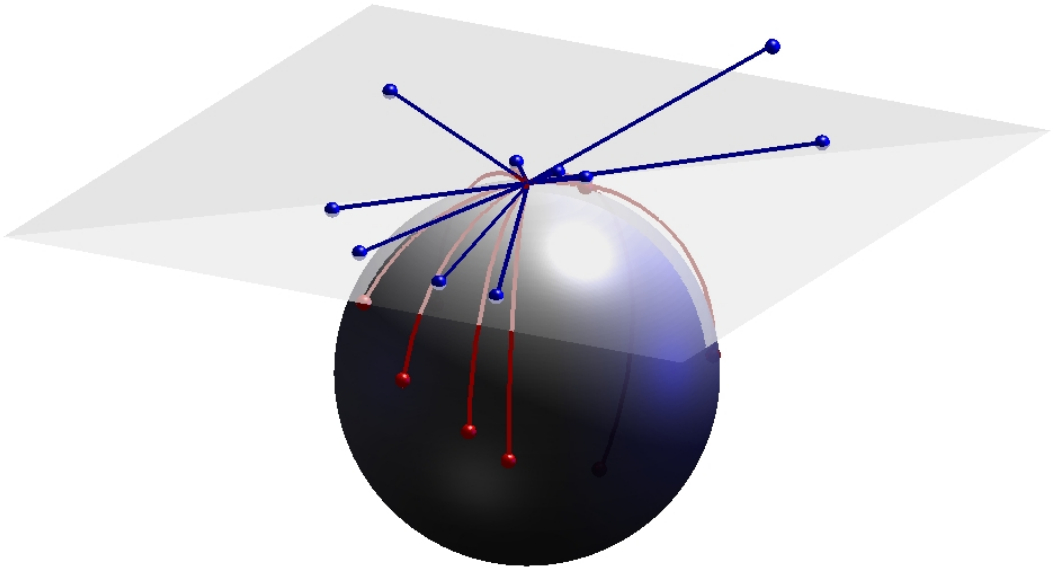
> *"Shapes are the ultimate non-linear sort of thing"*

Relative shapes can not be added and subtracted (ie, they are nonlinear). Instead, deformations should combined by composing them together.
Deformations that are smooth and invertable are known as *diffeomorphisms*, and form a mathematical *group*.



Linear Methods for Data on Manifolds

We are dealing with non-Euclidean geometry, so conceptualise the curved spaces as manifolds embedded in higher dimensions.



Diffeomorphism Groups

A group is a set, G, together with an operation, $\circ$, that combines any two elements a and b to form another element, denoted $a \circ b$. To qualify as a group, the set and operation must satisfy four requirements known as the group axioms:

- **Closure**. For all a, b in G, the result of the operation, $a \circ b$, is also in G.
- **Associativity**. For all a, b and c in G, $(a \circ b) \circ c = a \circ (b \circ c)$.
- **Identity element**. There exists an element Id in G, such that for every element a in G, the equation $Id \circ a = a \circ Id = a$ holds.
- **Inverse element**. For each a in G, there exists an element b in G such that $a \circ b = b \circ a = Id$.

The order in which the group operation is carried out matters for diffeomorphisms ($a \circ b \neq b \circ a$). They are not *abelian*.

Tiny Small Deformation Approximation

Positive real numbers form a group under the multiplication operation. We can not multiply numbers together by addition, but if they are very close to 1 (the identity element of the group), then we almost can. For example
$1.3 \times 0.8 = 1.04$, $1 + (1.3 - 1) + (0.8 - 1) = 1.1$
$1.03 \times 0.98 = 1.0094$, $1 + (1.03 - 1) + (0.98 - 1) = 1.01$
$1.003 \times 0.998 = 1.000994$, $1 + (1.003 - 1) + (0.998 - 1) = 1.001$

Similarly, we can treat extremely small displacement fields as almost linear.

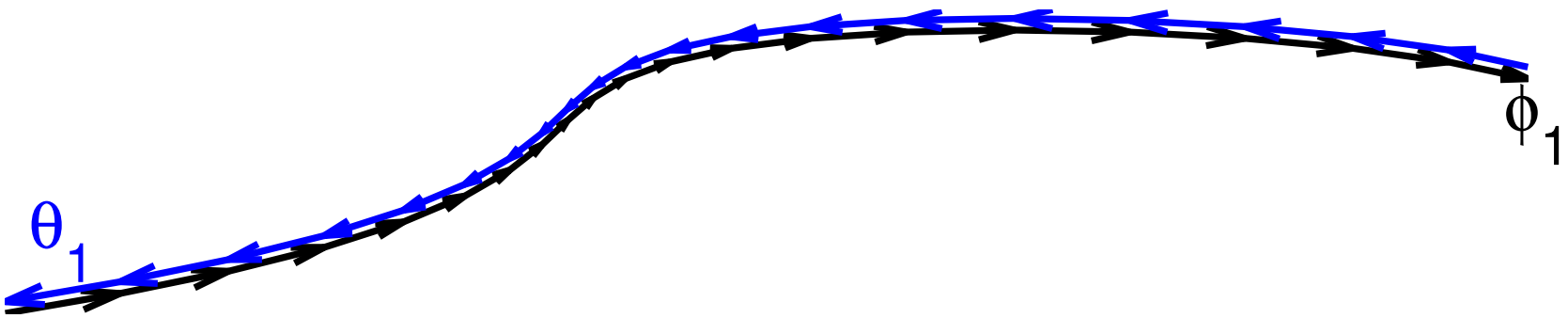
Large Deformations

We can consider a large deformation as the composition of a series of small deformations:

$$\varphi_1 = \left(\mathrm{Id} + \frac{\mathbf{v}_{t_{N-1}}}{N}\right) \circ \left(\mathrm{Id} + \frac{\mathbf{v}_{t_{N-2}}}{N}\right) \circ \ldots \circ \left(\mathrm{Id} + \frac{\mathbf{v}_{t_1}}{N}\right) \circ \left(\mathrm{Id} + \frac{\mathbf{v}_0}{N}\right)$$

The inverse of the deformation can be computed from:

$$\vartheta_1 = \left(\mathrm{Id} - \frac{\mathbf{v}_0}{N}\right) \circ \left(\mathrm{Id} - \frac{\mathbf{v}_{t_1}}{N}\right) \circ \ldots \circ \left(\mathrm{Id} - \frac{\mathbf{v}_{t_{N-2}}}{N}\right) \circ \left(\mathrm{Id} - \frac{\mathbf{v}_{t_{N-1}}}{N}\right)$$



Geodesic Distances

By modelling the trajectories as piecewise linear, geodesic distances (locally shortest distances along a curved manifold) can be computed using:

$$d = \frac{1}{N}\sum_{n=0}^{N-1}\sqrt{\mathbf{v}_{t_n}^T\mathbf{A}\mathbf{v}_{t_n}} = \frac{1}{N}\sum_{n=0}^{N-1}||\mathbf{L}\mathbf{v}_{t_n}||$$

If N approaches infinity, the evolution of a deformation may be conceptualised as integrating the following equation:

$$\frac{d\varphi}{dt} = \mathbf{v}_t(\varphi)$$

Geodesic distances (from zero) are then measured by:

$$d = \int_{t=0}^{1}||\mathbf{L}\mathbf{v}_t||dt$$

LDDMM

Large Deformation Diffeomorphic Metric Mapping is an image registration algorithm that minimises the following:

$$\mathcal{E} = \frac{1}{2}\int_{t=0}^{1} ||\mathbf{L}\mathbf{v}_t||^2 dt + \frac{1}{2\sigma^2}||f - \mu\left(\varphi_1^{-1}\right)||^2$$

$$\text{where } \varphi_0 = \mathrm{Id},\ \frac{d\varphi}{dt} = \mathbf{v}_t\left(\varphi_t\right)$$

The first term minimises the squared distance measure of the deformations, whereas the second term simply minimises the difference between the warped template and the individual scan. The objective is to estimate a series of velocity fields $(\mathbf{v}_t)$.

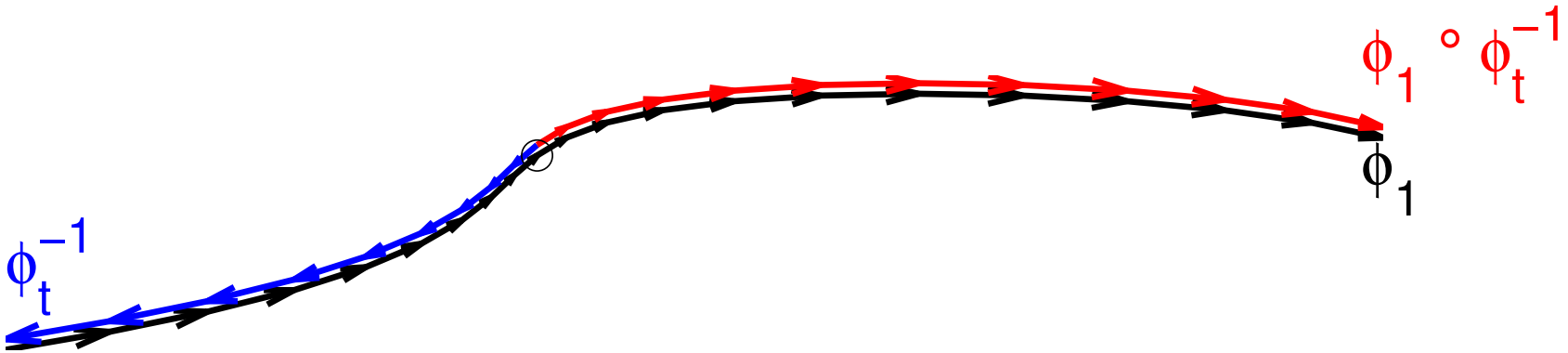
Change of Variables

When we warp images, we should usually account for expansion/contraction via a change of variables.

$$\int_{\mathbf{x}\in\varphi(\Omega)} f(\mathbf{x})d\mathbf{x} = \int_{\mathbf{x}\in\Omega} f(\varphi(\mathbf{x}))|(\mathbf{D}\varphi)(\mathbf{x})|d\mathbf{x}$$

where $(\mathbf{D}\varphi)(\mathbf{x})$ means the Jacobian of φ at $\mathbf{x}$.

Matching Term



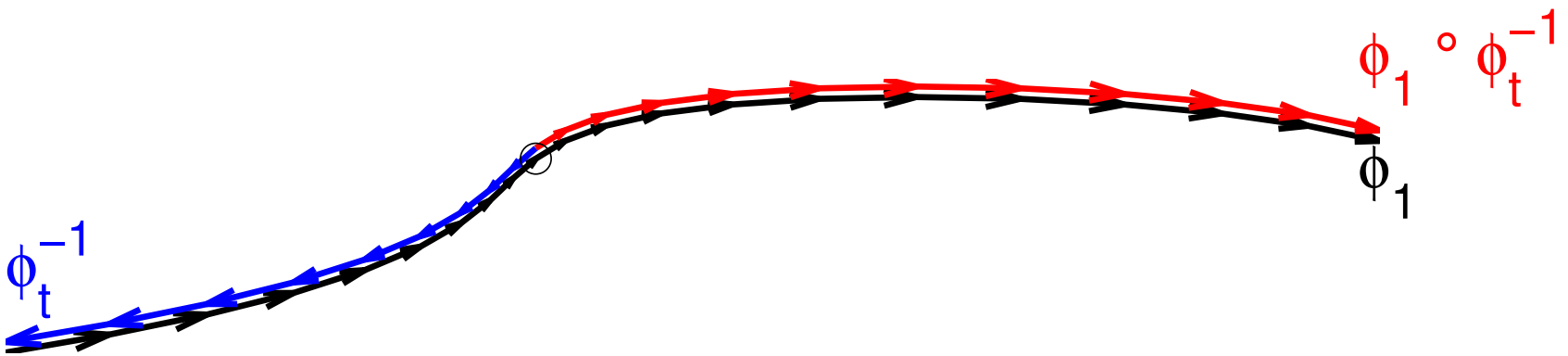
The matching term of the objective function is:

$$\frac{1}{\sigma^2}\int_{x\in\Omega}(f\circ\mathbf{x}-\mu\circ\varphi_1^{-1}\circ\mathbf{x})^2 d\mathbf{x}$$

This may be re-written (including a change of variables via $\varphi_1\circ\varphi_t^{-1}$) as:

$$\frac{1}{\sigma^2}\int_{x\in\Omega}|\mathbf{D}(\varphi_1\circ\varphi_t^{-1})\circ\mathbf{x}|(f\circ\varphi_1\circ\varphi_t^{-1}\circ\mathbf{x}-\mu\circ\varphi_t^{-1}\circ\mathbf{x})^2 d\mathbf{x}$$

Matching Term



After registration.

- $f(\mathbf{x})$ matches $\mu(\varphi_1^{-1}(\mathbf{x}))$.
- $\mu(\mathbf{x})$ matches $f(\varphi_1(\mathbf{x}))$.
- $f(\varphi_1(\varphi_t^{-1}(\mathbf{x})))$ matches $\mu(\varphi_t^{-1}(\mathbf{x}))$.

Derivatives for LDDMM (need to check these equations)

For $n = 0..N$, the objective function is:

$$\frac{1}{\sigma^2}\int_{x\in\Omega} |\mathbf{D}(\varphi_1 \circ \varphi_{\frac{n}{N}}^{-1}) \circ \mathbf{x}|(f \circ \varphi_1 \circ \varphi_{\frac{n}{N}}^{-1} \circ \mathbf{x} - \mu \circ \varphi_{\frac{n}{N}}^{-1} \circ \mathbf{x})^2 d\mathbf{x}$$

We can create two images $f' = f \circ \varphi_1 \circ \varphi_{\frac{n}{N}}^{-1}$ and $\mu' = \mu \circ \varphi_{\frac{n-1}{N}}^{-1}$, as well as some weights $p = |\mathbf{D}(\varphi_1 \circ \varphi_{\frac{n}{N}}^{-1})|$. This (almost) means that the updates at each time step can be treated as small deformations.

$$\mathcal{E}_t = \frac{1}{2}||\mathbf{L}\mathbf{v}_t||^2 + \frac{1}{2\sigma^2}\int_{\mathbf{x}} p(\mathbf{x})\left(f'(\mathbf{x}) - \mu'(\mathbf{x} - \frac{\mathbf{v}_t(\mathbf{x})}{N})\right)^2 d\mathbf{x}$$

Derivatives for LDDMM

The derivatives for the matching term turn out to be:

$$\frac{1}{\sigma^2}|\mathbf{D}(\boldsymbol{\varphi}_1 \circ \boldsymbol{\varphi}_t^{-1})|(\nabla(\mu \circ \boldsymbol{\varphi}_t^{-1}))(f \circ \boldsymbol{\varphi}_1 \circ \boldsymbol{\varphi}_t^{-1} - \mu \circ \boldsymbol{\varphi}_t^{-1})$$

Note that:

- $\nabla(\mu \circ \boldsymbol{\varphi}_t^{-1}) = (\mathbf{D}\boldsymbol{\varphi}_t^{-1})^T((\nabla\mu) \circ \boldsymbol{\varphi}_t^{-1})$
- $|\mathbf{D}(\boldsymbol{\varphi}_1 \circ \boldsymbol{\varphi}_t^{-1})| = |\mathbf{D}\boldsymbol{\varphi}_t^{-1}|(|\mathbf{D}\boldsymbol{\varphi}_1| \circ \boldsymbol{\varphi}_t^{-1})$.
- $(f \circ \boldsymbol{\varphi}_1 \circ \boldsymbol{\varphi}_t^{-1} - \mu \circ \boldsymbol{\varphi}_t^{-1}) = (f \circ \boldsymbol{\varphi}_1 - \mu) \circ \boldsymbol{\varphi}_t^{-1}$
- $\boldsymbol{\varphi}_t^{-1} = \boldsymbol{\vartheta}_t$

So the derivatives can be re-written:

$$\frac{1}{\sigma^2}|\mathbf{D}\boldsymbol{\vartheta}_t|(|\mathbf{D}\boldsymbol{\varphi}_1| \circ \boldsymbol{\vartheta}_t)(\mathbf{D}\boldsymbol{\vartheta}_t)^T((\nabla\mu) \circ \boldsymbol{\vartheta}_t)((f \circ \boldsymbol{\varphi}_1 - \mu) \circ \boldsymbol{\vartheta}_t)$$

$$|\mathbf{D}\boldsymbol{\vartheta}_t|(\mathbf{D}\boldsymbol{\vartheta}_t)^T\left(\left(\frac{1}{\sigma^2}|\mathbf{D}\boldsymbol{\varphi}_1|(\nabla\mu)(f \circ \boldsymbol{\varphi}_1 - \mu)\right) \circ \boldsymbol{\vartheta}_t\right)$$

Gradient Descent for LDDMM

LDDMM is optimised via gradient descent on the so called *Hilbert gradient*:

$$\mathbf{v}_t^{(i+1)} = \mathbf{v}_t^{(i)} - \epsilon \mathbf{v}_t^{(i)} - \epsilon \mathbf{K} \left(\left| \mathbf{D}\boldsymbol{\vartheta}_t^{(i)} \right| \left(\mathbf{D}\boldsymbol{\vartheta}_t^{(i)} \right)^T \left(\left(\frac{|\mathbf{D}\boldsymbol{\varphi}_1^{(i)}|}{\sigma^2} (\nabla \mu)(f \circ \boldsymbol{\varphi}_1^{(i)} - \mu) \right) \circ \boldsymbol{\vartheta}_t^{(i)} \right) \right)$$

Note that Gauss-Newton can not easily be used because of covariance between velocities at different times.
Also note that lots of deformations and velocity fields need to be computed, which either involves using loads of memory, or writing loads of temporary files to disk.

Outline

Euler-Lagrange Equations

At the solution, the derivatives of the objective function are zero, which means that the velocity at any time point is given by:

$$\mathbf{v}_t + \mathbf{K}\left(|\mathbf{D}\boldsymbol{\vartheta}_t|\,(\mathbf{D}\boldsymbol{\vartheta}_t)^T\left(\left(\frac{1}{\sigma^2}|\mathbf{D}\boldsymbol{\varphi}_1|(\nabla\mu)(f\circ\boldsymbol{\varphi}_1-\mu)\right)\circ\boldsymbol{\vartheta}_t\right)\right)=0$$

If we introduce something that we'll call initial momentum:

$$\mathbf{u}_0=\mathbf{A}\mathbf{v}_0=\frac{1}{\sigma^2}|\mathbf{D}\boldsymbol{\varphi}_1|(\nabla\mu)(\mu-f\circ\boldsymbol{\varphi}_1)$$

we see that the velocity at any time point is determined by the initial momentum (or velocity), according to:

$$\mathbf{v}_t=\mathbf{K}\left(|\mathbf{D}\boldsymbol{\vartheta}_t|(\mathbf{D}\boldsymbol{\vartheta}_t)^T(\mathbf{u}_0\circ\boldsymbol{\vartheta}_t)\right)$$

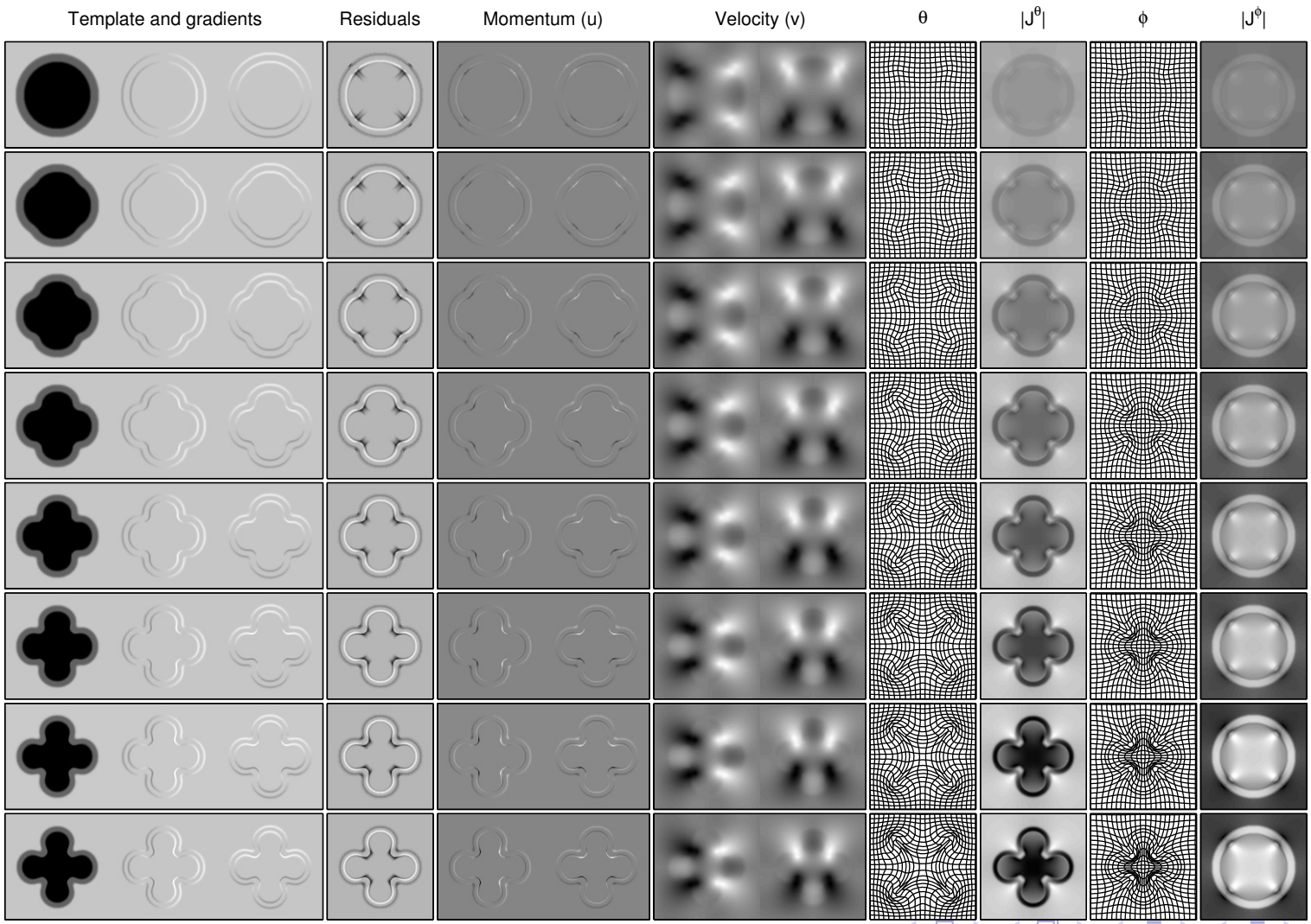
Geodesic Shooting

This all means that we do not need to estimate a series of velocity fields. We just need to estimate an initial velocity ($\mathbf{v}_0$), from which we compute the initial momentum by $\mathbf{u}_0 = \mathbf{A}\mathbf{v}_0$.
We set the deformation at time 0 to an identity transform ($\varphi_0 = Id$), and then evolve the following dynamical system for unit time:

$$\frac{d\varphi}{dt} = \mathbf{v}_t(\varphi_t)$$

$$\mathbf{v}_t = \mathbf{K}\left(|\mathbf{D}\varphi_t^{-1}|(\mathbf{D}\varphi_t^{-1})^T(\mathbf{u}_0 \circ \varphi_t^{-1})\right)$$

Evolution


Template and gradients
Residuals
Momentum (u)
Velocity (v)
θ
|J^θ|
φ
|J^φ|

Objective Function

LDDMM objective function:

$$\mathcal{E} = \frac{1}{2}\int_{t=0}^{1} ||\mathbf{L}\mathbf{v}_t||^2 dt + \frac{1}{2\sigma^2}||f - \mu\left(\boldsymbol{\varphi}_1^{-1}\right)||^2$$

A possible Geodesic Shooting objective function:

$$\mathcal{E} = \frac{1}{2}||\mathbf{L}\mathbf{v}_0||^2 dt + \frac{1}{2\sigma^2}||f - \mu\left(\boldsymbol{\varphi}_1^{-1}\right)||^2$$

The objective function I use:

$$\mathcal{E} = \frac{1}{2}||\mathbf{L}\mathbf{v}_0||^2 dt + \frac{1}{2\sigma^2}\int_{\mathbf{x}\in\Omega} |(\mathbf{D}\boldsymbol{\varphi}_1)\circ\mathbf{x}|\left(f\left(\boldsymbol{\varphi}_1(\mathbf{x})\right) - \mu(\mathbf{x})\right)^2 d\mathbf{x}$$

Gradient Descent for Geodesic Shooting

Could simply optimise $\mathbf{v}_0$ via gradient descent:

$$\mathbf{v}_t^{(i+1)} = \mathbf{v}_t^{(i)} - \epsilon \mathbf{v}_t^{(i)} - \epsilon \mathbf{K} \left(\frac{|\mathbf{D}\boldsymbol{\varphi}_1^{(i)}|}{\sigma^2} (\nabla \mu)(f \circ \boldsymbol{\varphi}_1^{(i)} - \mu) \right)$$

We no longer need save a series of deformations and velocity fields. Note also that Gauss-Newton can now be used because all the parameters are in $\mathbf{v}_0$ - so no covariance to worry about.

Gauss-Newton Algorithm

- Set the initial velocity $\mathbf{v}_0$ (parameterised by $\mathbf{w}$) to zero.
- Repeat the following until convergence or for a fixed number of iterations
  - Shoot from the initial velocity $\mathbf{v}_0$ to obtain φ_1.
  - Compute the objective function, and approximate gradient and Hessian ($\mathcal{E}$, $\mathbf{g}$ and $\mathbf{H}$), using the current φ_1.
  - Check $\mathcal{E}$ improves.
  - Gauss-Newton update of the coefficients, which parameterise $\mathbf{v}_0$.

Gradients

$$\mathbf{g} = \begin{bmatrix} g_{11} \\ g_{21} \\ g_{31} \\ \vdots \\ g_{I1} \\ g_{11} \\ g_{21} \\ g_{31} \\ \vdots \\ g_{I1} \end{bmatrix}$$

$$g_{il} = \frac{1}{\sigma^2} |\mathbf{J}_1^{\varphi}(\mathbf{x}_i)| \left(f(\varphi_1(\mathbf{x}_i)) - \mu(\mathbf{x}_i)\right) \left((\nabla_I \mu) \circ \mathbf{x}_i\right)$$

Hessian

$$\mathbf{H} = \begin{bmatrix}
h_{111} & 0 & 0 & \dots & 0 & h_{121} & 0 & 0 & \dots & 0 \\
0 & h_{211} & 0 & \dots & 0 & 0 & h_{221} & 0 & \dots & 0 \\
0 & 0 & h_{311} & \dots & 0 & 0 & 0 & h_{321} & \dots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \dots & h_{I11} & 0 & 0 & 0 & \dots & h_{I21} \\
h_{112} & 0 & 0 & \dots & 0 & h_{122} & 0 & 0 & \dots & 0 \\
0 & h_{212} & 0 & \dots & 0 & 0 & h_{222} & 0 & \dots & 0 \\
0 & 0 & h_{312} & \dots & 0 & 0 & 0 & h_{322} & \dots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \dots & h_{I12} & 0 & 0 & 0 & \dots & h_{I22}
\end{bmatrix}$$

$$h_{ilm} = \frac{1}{\sigma^2} |\mathbf{J}_1^{\varphi}(\mathbf{x}_i)| ((\nabla_l \mu) \circ \mathbf{x}_i)((\nabla_m \mu) \circ \mathbf{x}_i)$$

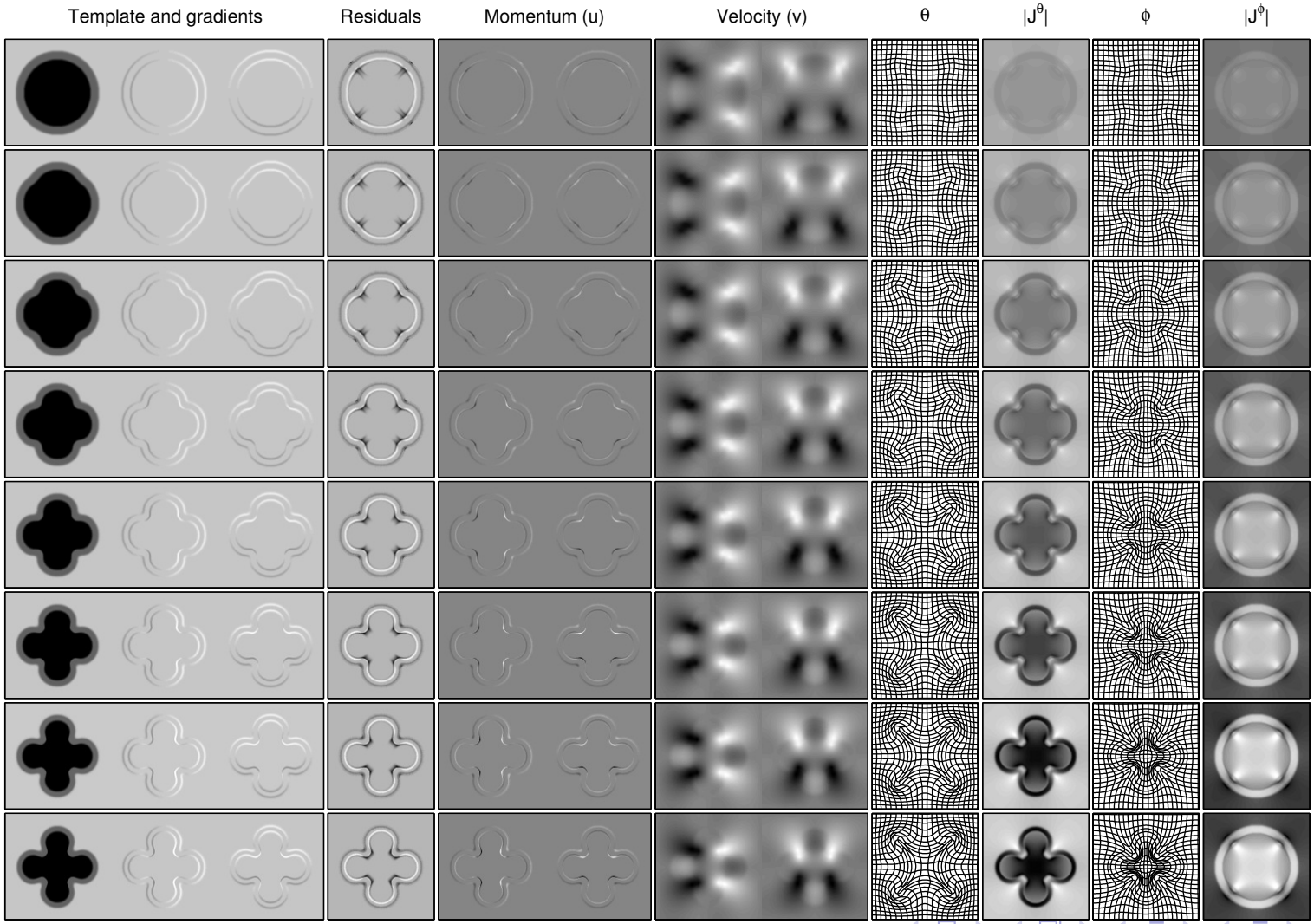
Initial Momentum

Remember that

$$\mathbf{u}_0 = \frac{1}{\sigma^2}|\mathbf{D}\boldsymbol{\varphi}_1|(\nabla\mu)(\mu - f \circ \boldsymbol{\varphi}_1)$$

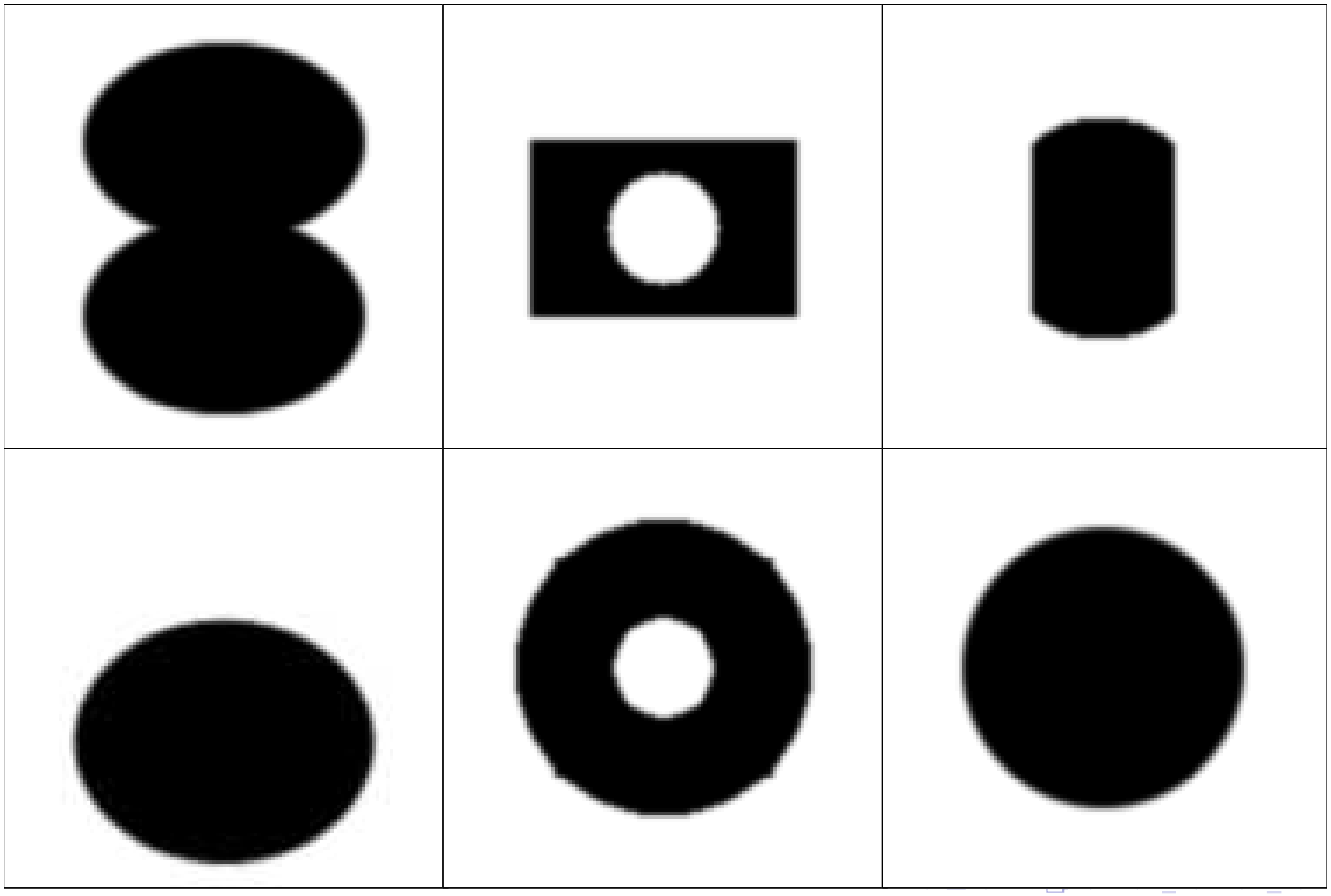
If a population of subjects are all aligned with the same template image, $\frac{1}{\sigma^2}(\nabla\mu)$ will be the same for all subjects. Deviations from the template are encoded by the residuals, $|\mathbf{D}\boldsymbol{\varphi}_1|(\mu - f \circ \boldsymbol{\varphi}_1)$. This is a scalar field, and in principle is all that is needed (along with the template) to reconstruct the original images.

Evolution


Template and gradients
Residuals
Momentum (u)
Velocity (v)
θ
|J^θ|
φ
|J^φ|

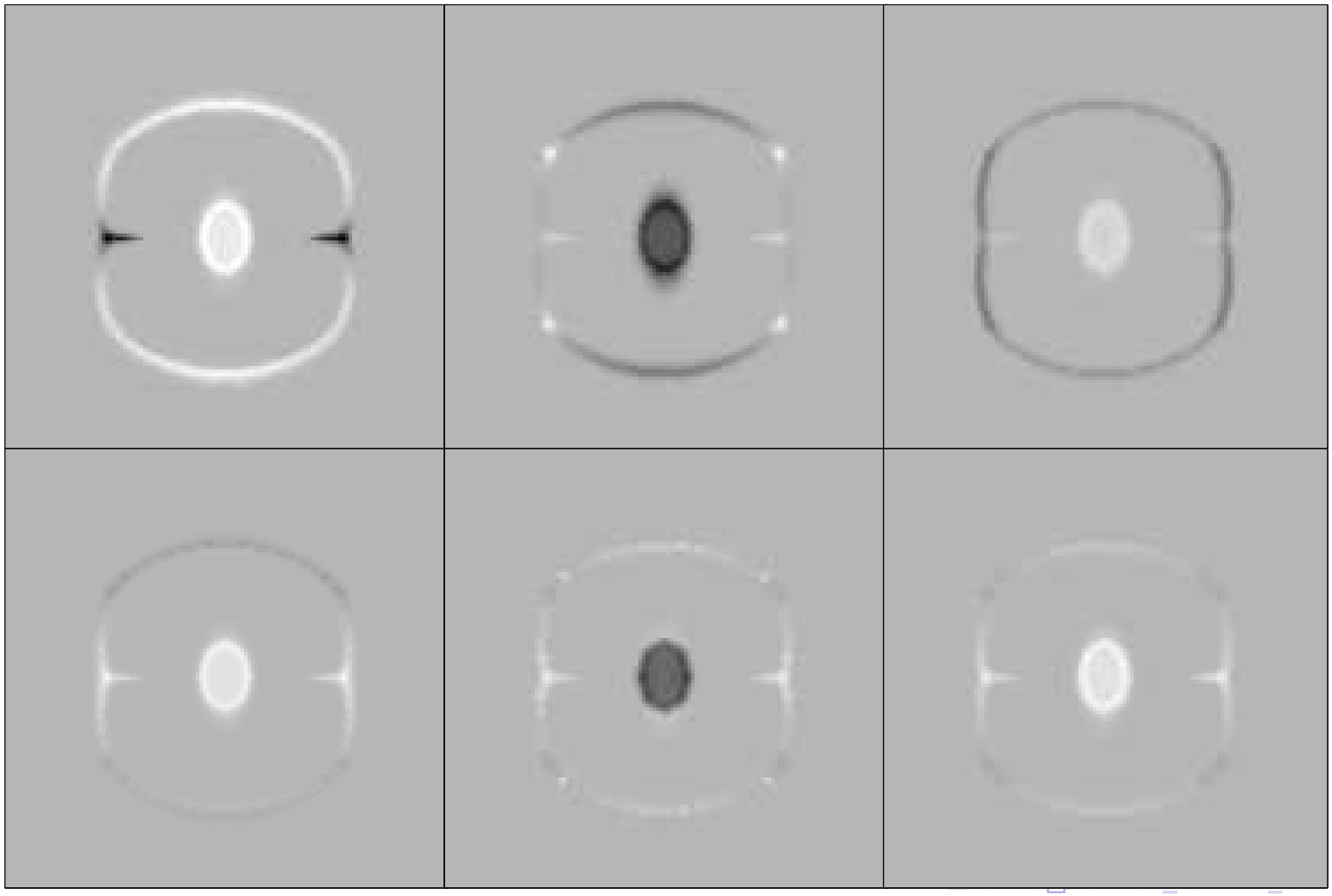
Example Images

Some example images.



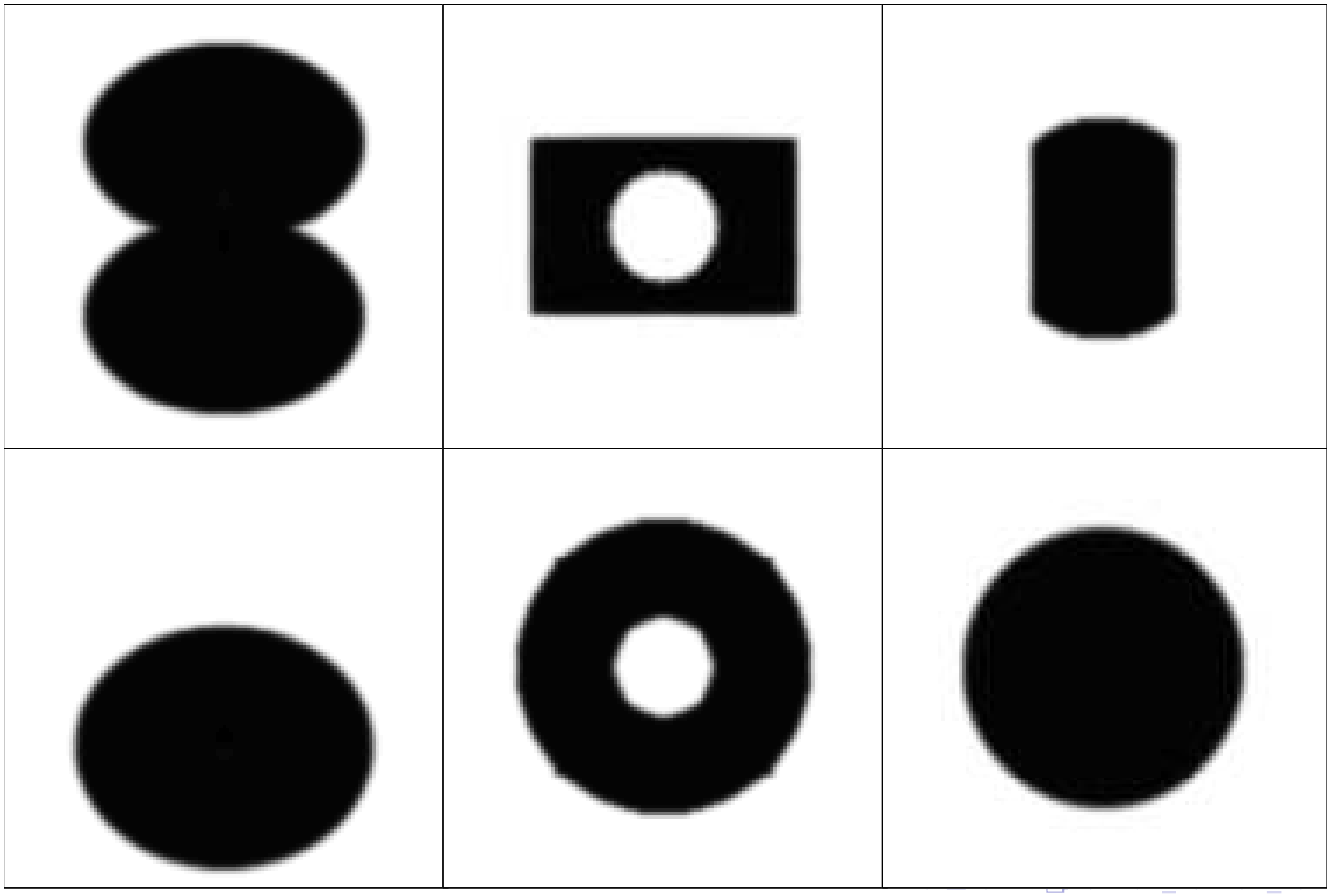
Residuals

Residuals after aligning the example images to a common template.

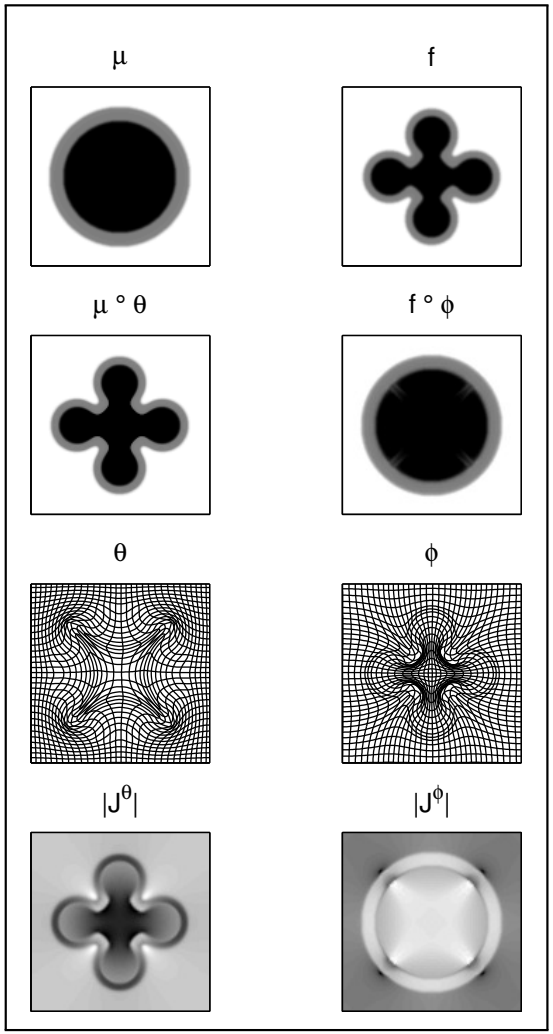


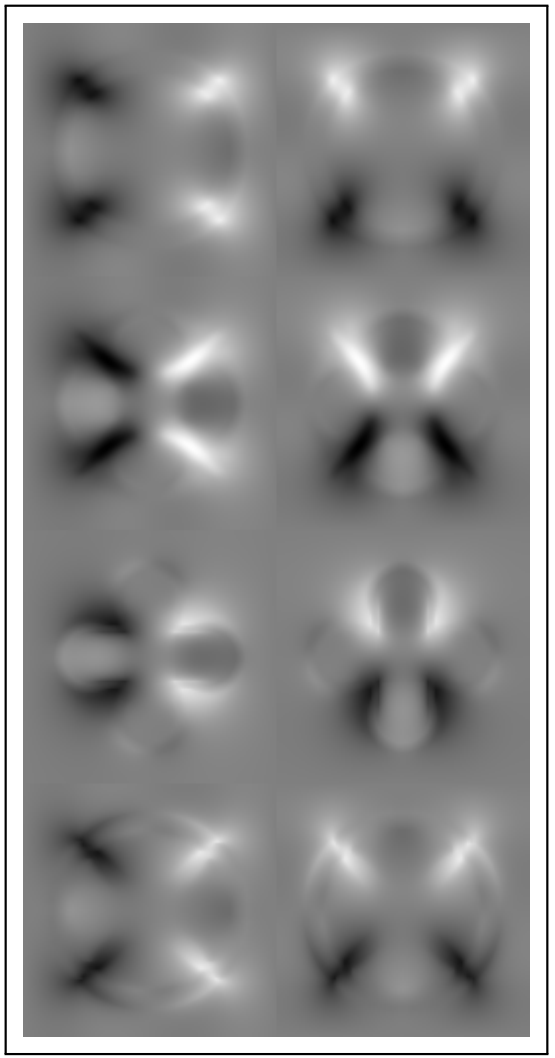
Reconstructed Images

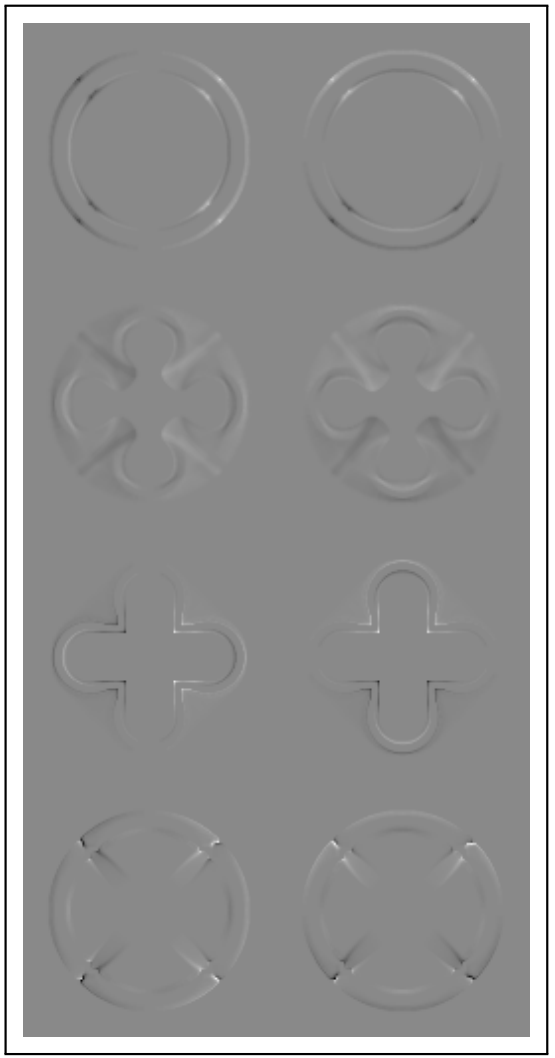
Images reconstructed using just the template and residuals.



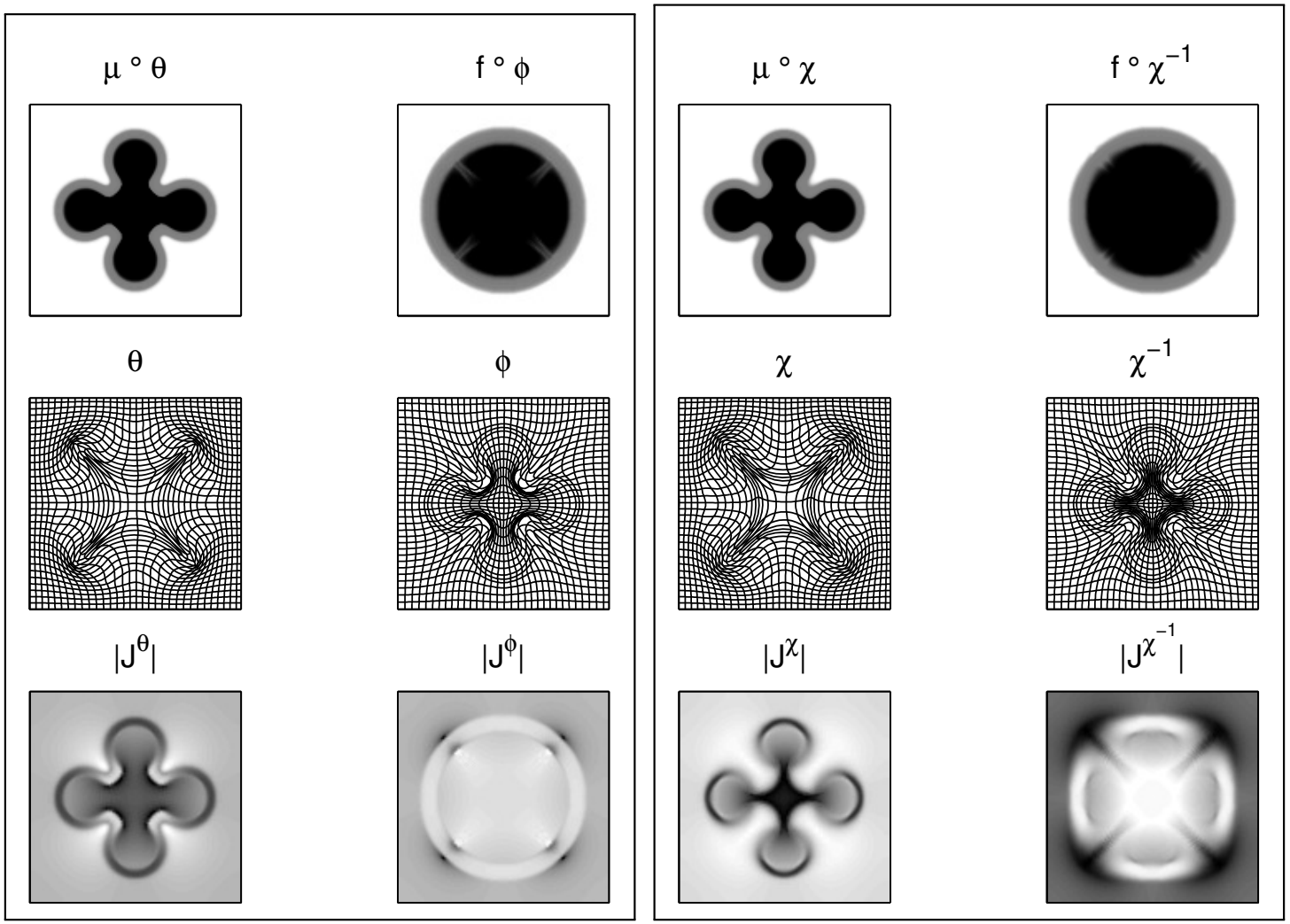
Parameterisations



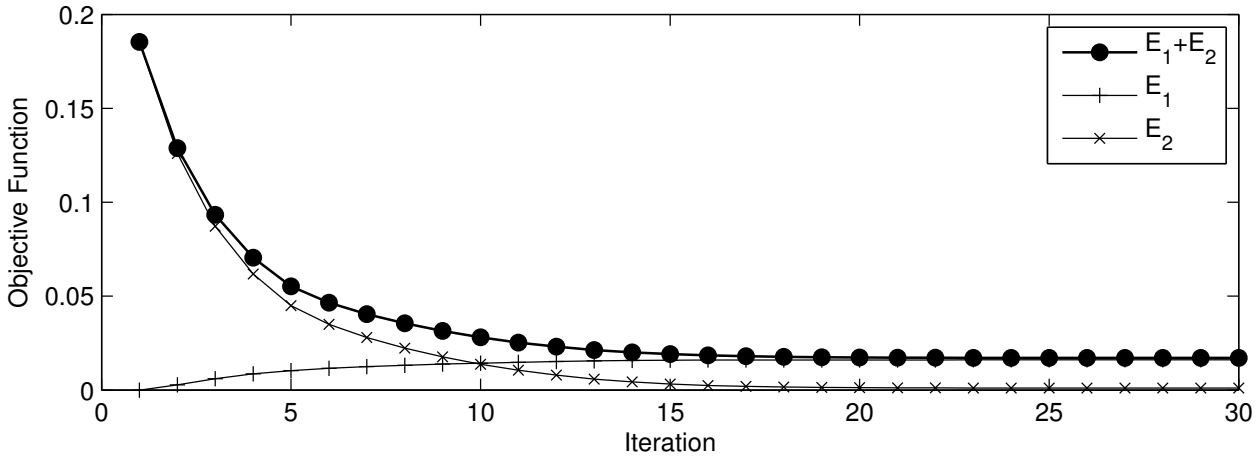


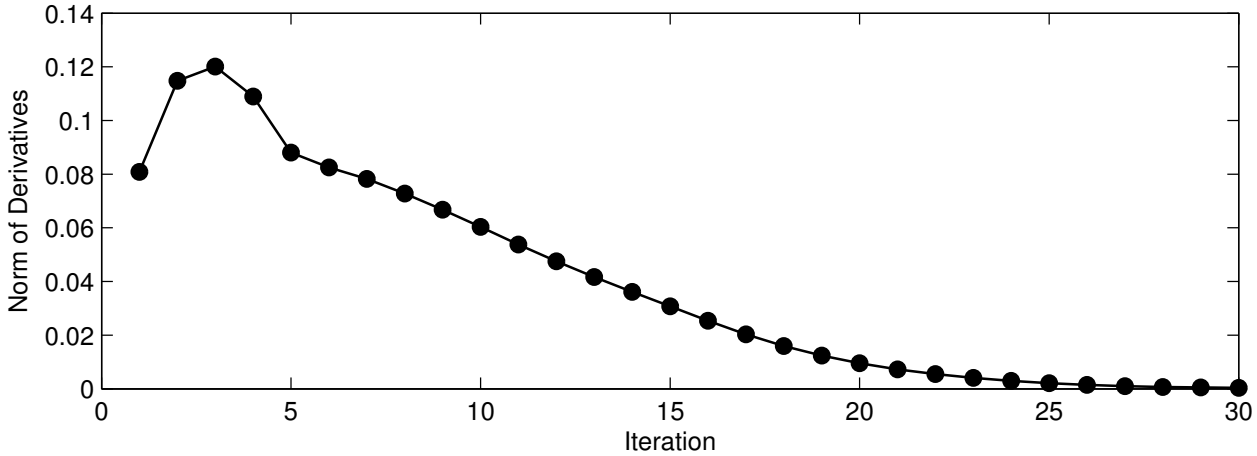


Geodesic Shooting versus Dartel

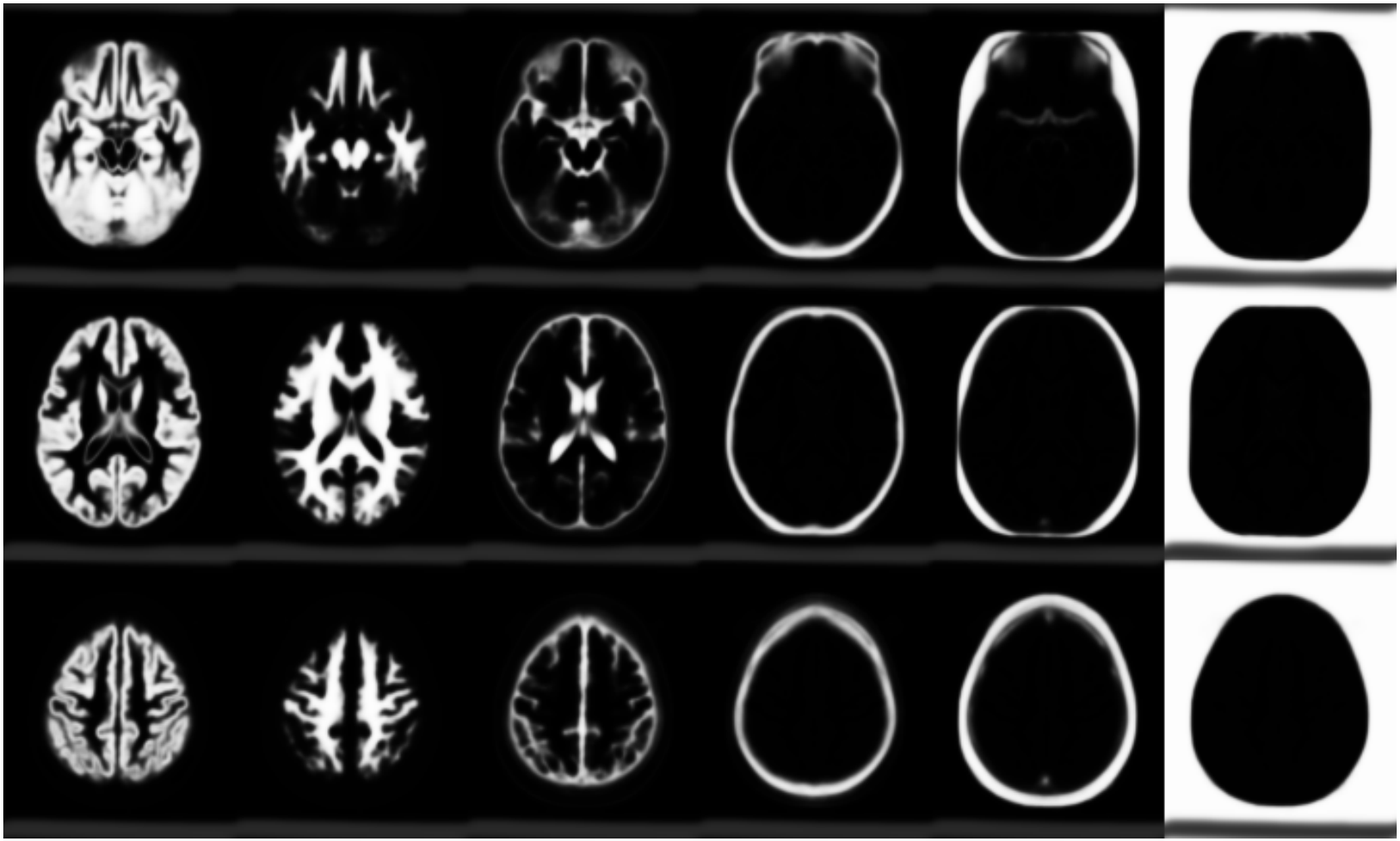


Convergence of Gauss-Newton

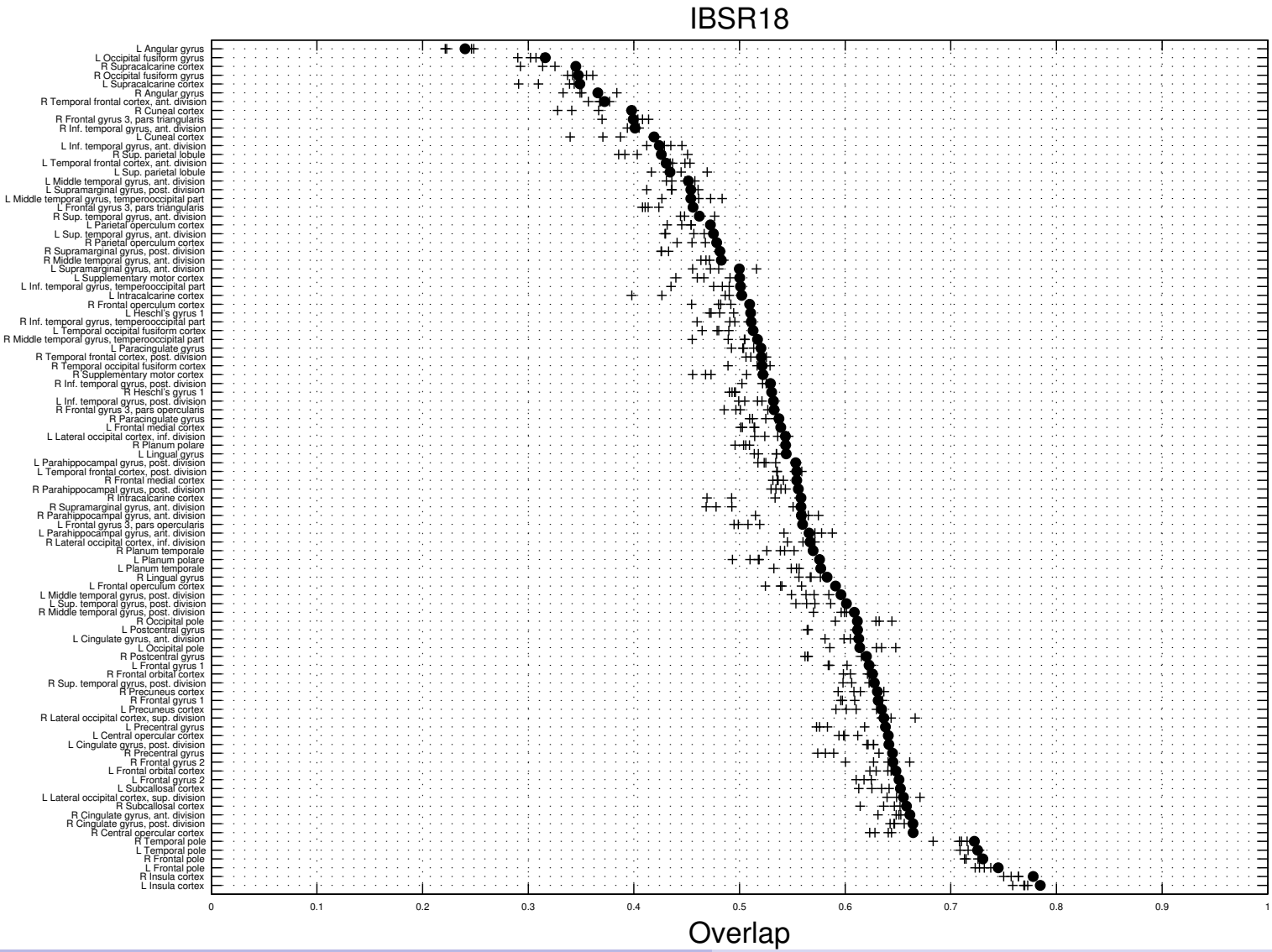




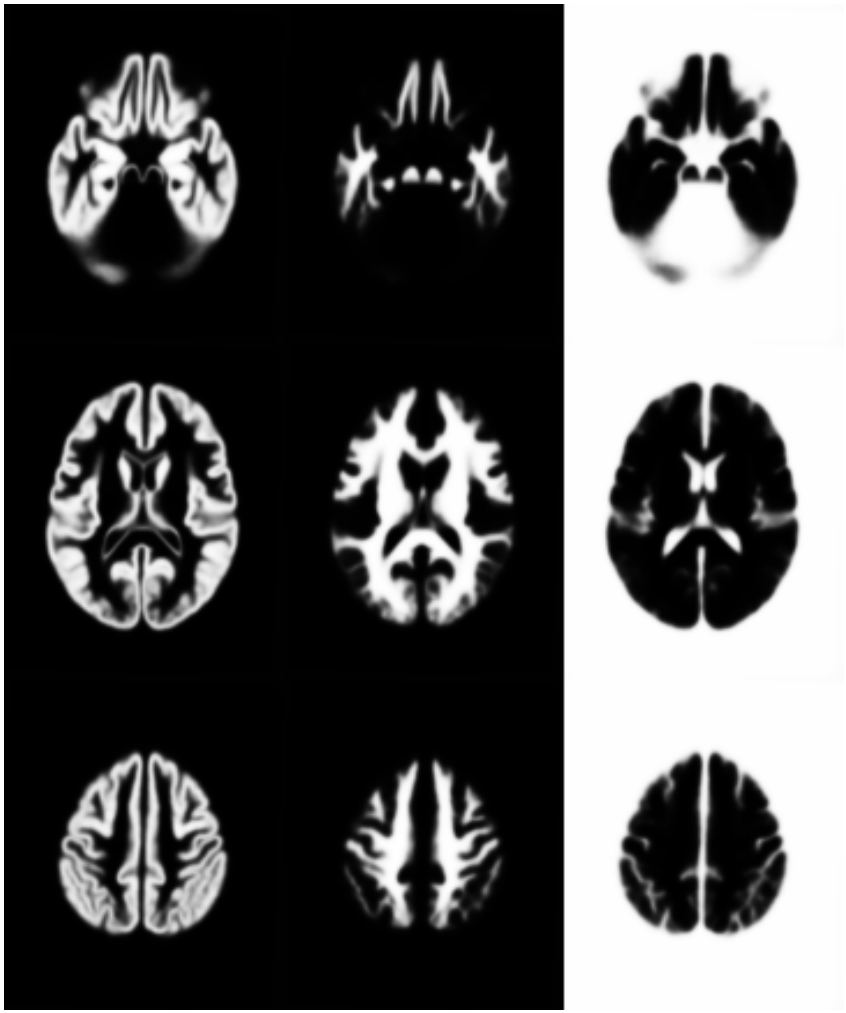
Template (IBSR18)



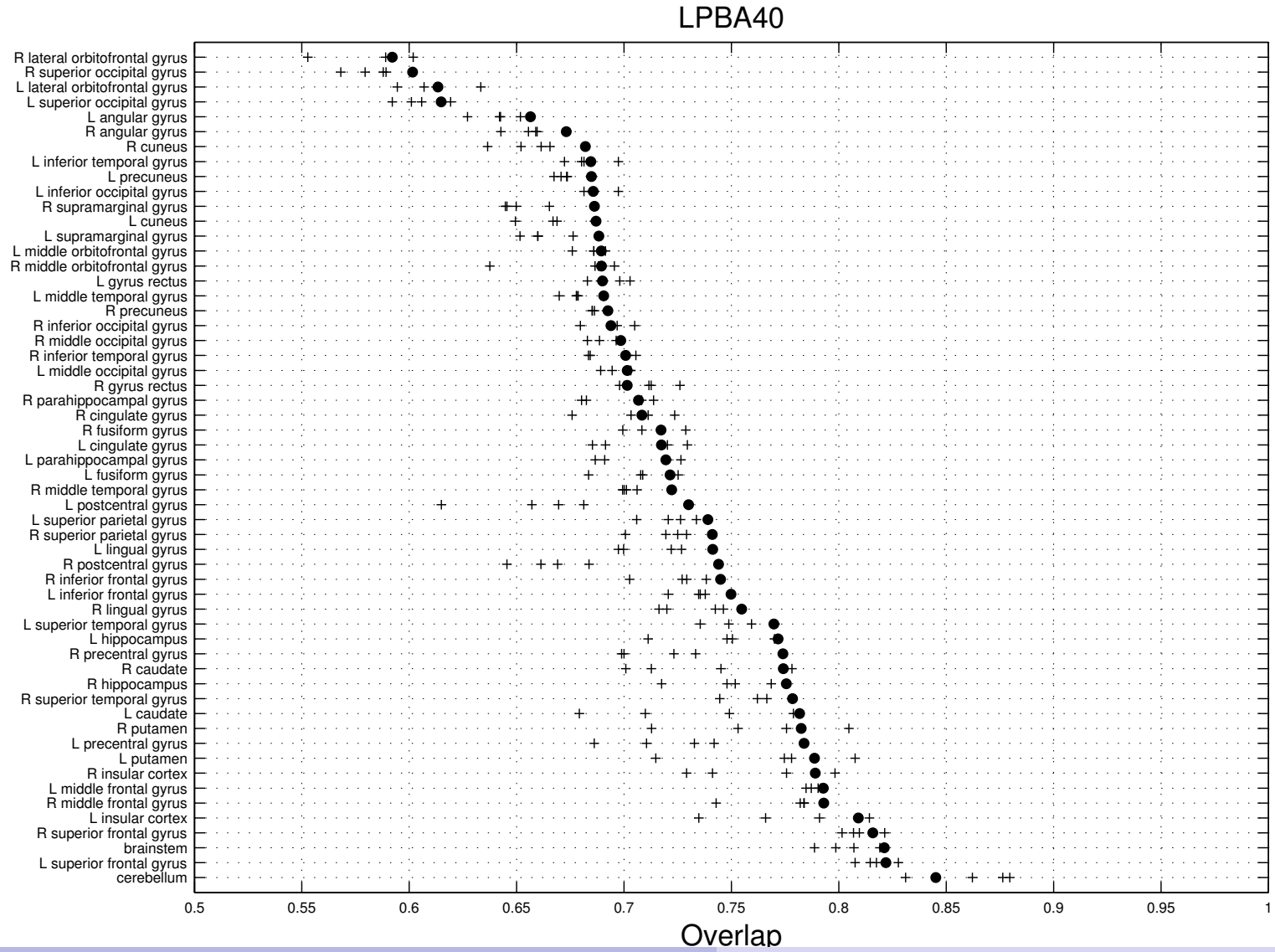
Label Overlaps (IBSR18)



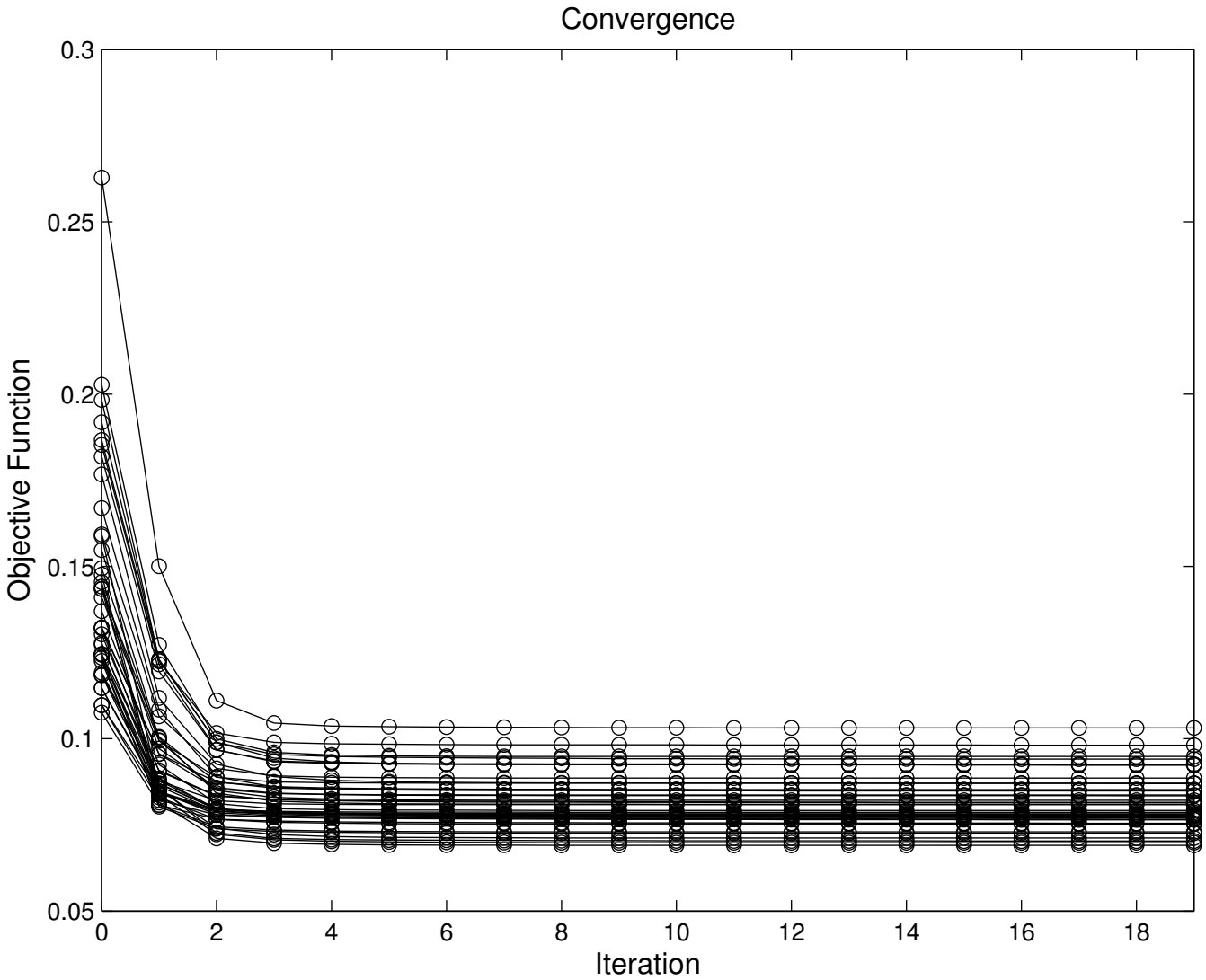
Template (LPBA40)



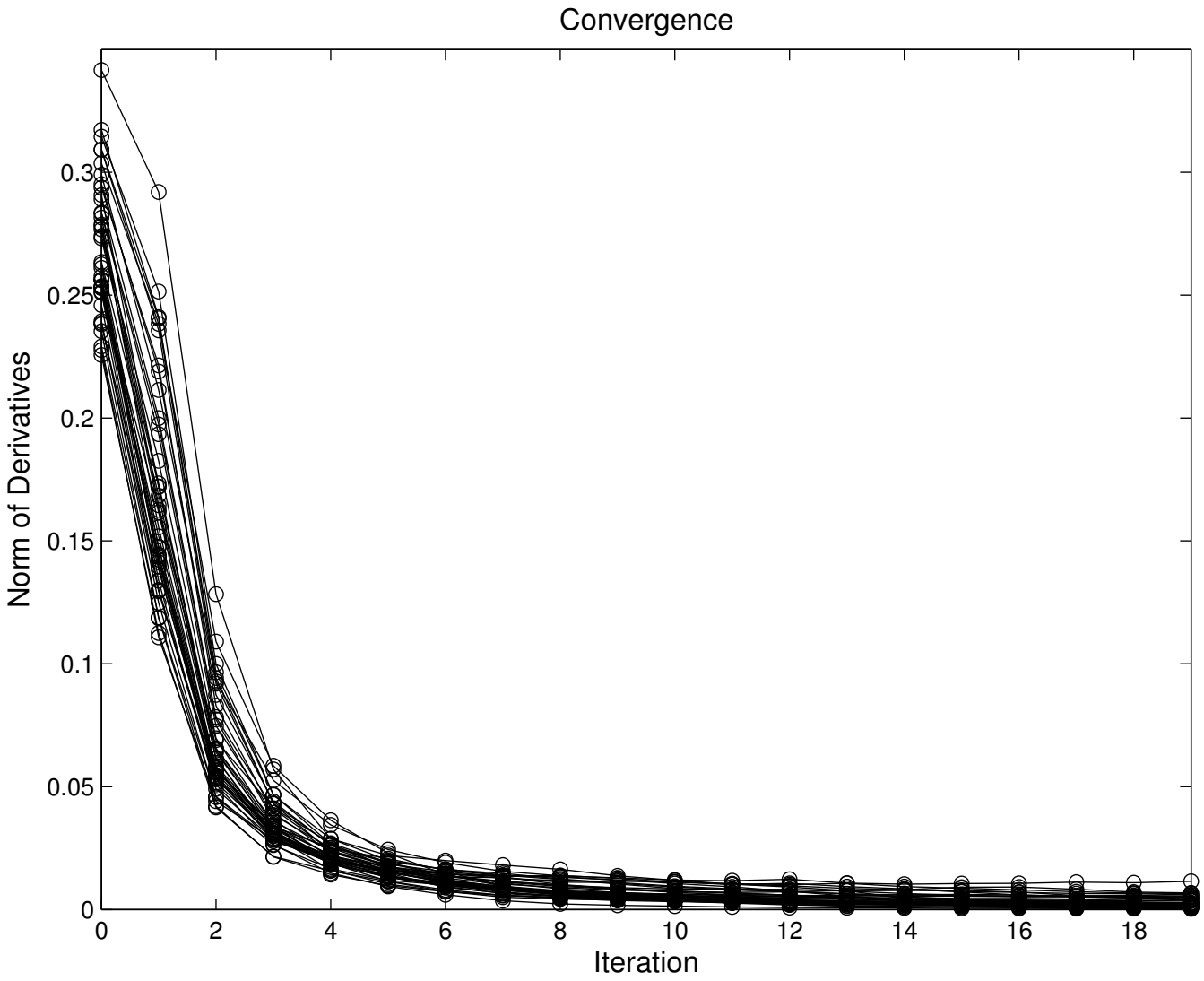
Label Overlaps (LPBA40)



Convergence (LPBA40)



Convergence (LPBA40)



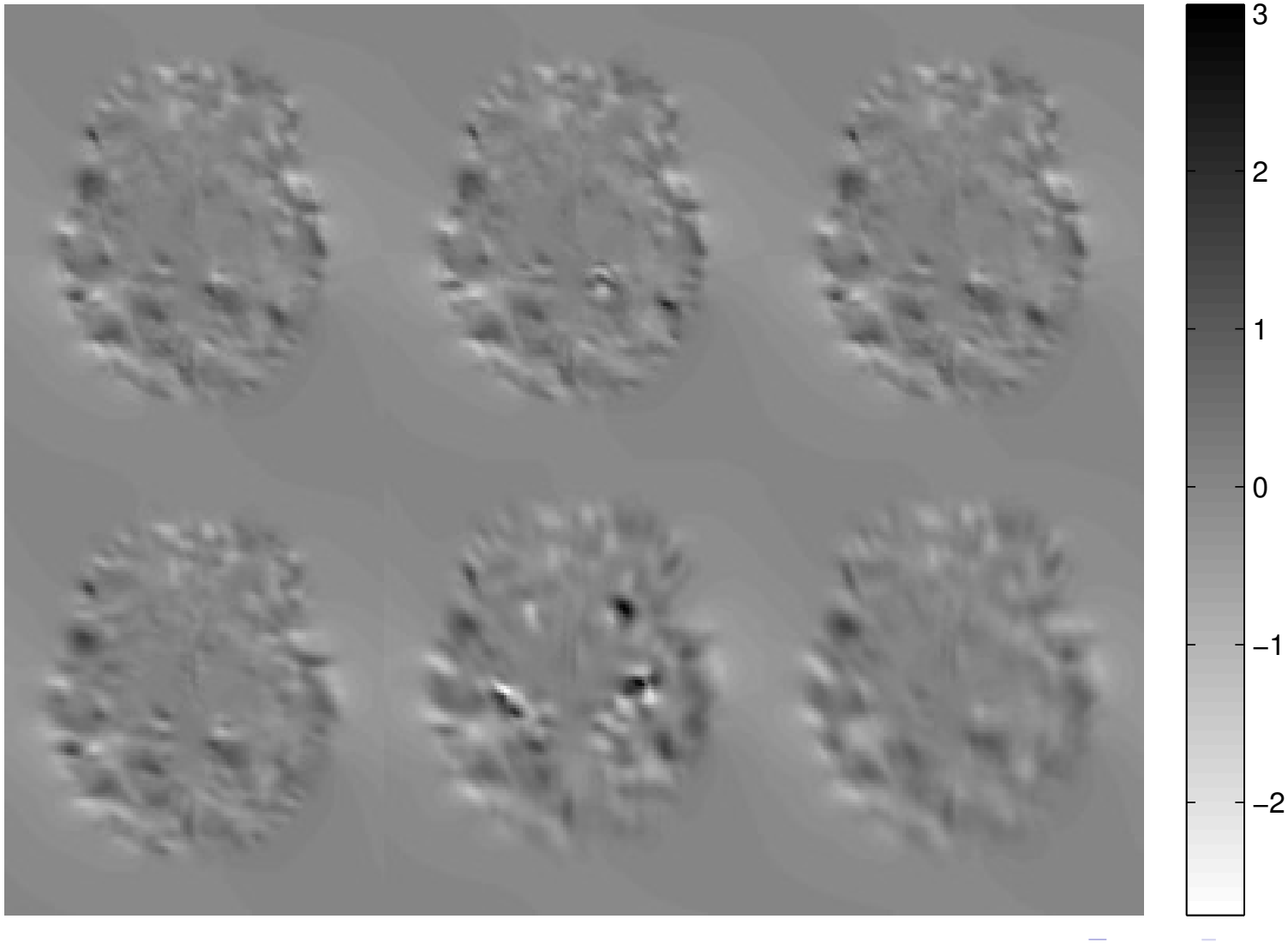
More GS versus Dartel

To assess the effects of larger displacements, the IBSR40 images were all translated along the anterior-posterior direction by 12mm (8 voxels), and re-registered with the template previously generated from un-translated versions of the data.
Reproducibility was assessed from correlation coefficients with original GS velocities:

$$r_{ab} = \frac{\mathbf{w}_a^T \mathbf{A} \mathbf{w}_b}{\sqrt{\mathbf{w}_a^T \mathbf{A} \mathbf{w}_a} \sqrt{\mathbf{w}_b^T \mathbf{A} \mathbf{w}_b}}$$

	No Translations	Poor Estimates	Good Estimates
GS	0.98	0.52	0.98
Dartel	0.84	0.19	0.47

More GS versus Dartel



Inverse Consistency

Could have achieved inverse consistency by aligning scans to a template at the "half-way point".
The objective here was to try to achieve inverse consistent registration, but with the template aligned to one of the original scans:

$$\mathcal{E} = \frac{1}{2}\int_{t=0}^{1} ||\mathbf{L}\mathbf{v}_t||^2 dt + \frac{1}{2\sigma^2}||f_1 - \mu\left(\varphi_1^{-1}\right)||^2 + \frac{1}{2\sigma^2}||f_0 - \mu||^2$$

where $\varphi_0 = \mathrm{Id}$, $\dfrac{d\varphi}{dt} = \mathbf{v}_t\left(\varphi_t\right)$, $\mu = (|\mathbf{J}_1^{\varphi}| f_1(\varphi_1) + f_0)/(|\mathbf{J}_1^{\varphi}| + 1)$

Inverse Consistency