Dipy's Registration module

1 Notation and image processing background

We regard an *image* as a function I that maps voxels of a grid $\mathcal{L}$ to a set G of possible values called the "dynamic range" of I. The images we are interested in represent objects in physical space, $\mathbf{R}^3$. This means that each point $[i, j, k]$ in the 3-dimensional grid $\mathcal{L}$ is associated to a point $(x, y, z) \in \mathbf{R}^3$. We use square brackets to emphasize the fact that i, j, k are coordinates of a grid, which may not be integers. When i, j, k are not integers, the image is evaluated by interpolation.

The function that maps voxel coordinates of a grid $\mathcal{L}$ to their corresponding coordinates in physical space is an invertible affine transformation. Whenever we talk about **a grid** $\mathcal{L}$ we implicitly assume that this $\mathcal{L}$ is associated to a specific grid-to-space transformation. Since our images represent objects in physical space, and these objects are not tied to any specific grid, the same object may be **sampled** over any grid $\mathcal{L}$. Note, however, that the grid $\mathcal{L}$ is finite and it may be the case that none of its voxels map to any point on the object of interest (in that case the object was badly sampled by $\mathcal{L}$).

2 Diffeomorphic Map

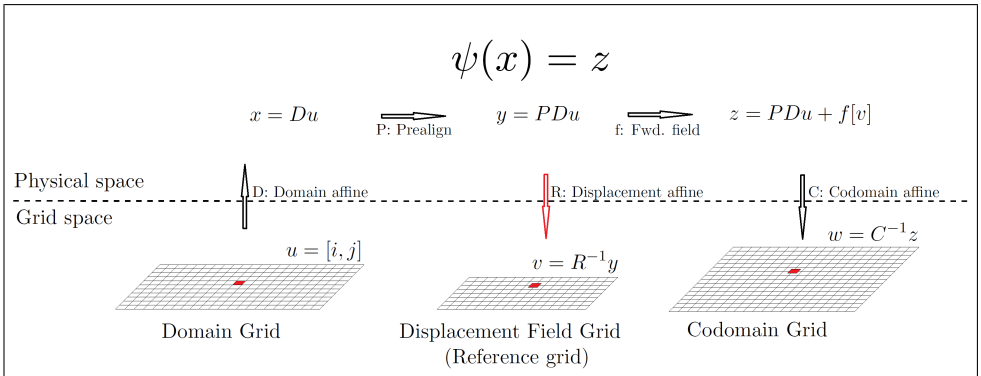
A diffeomorphism is an invertible and differentiable function whose inverse is also differentiable. We implement a diffeomorphism Ψ by means of a deformation field ϕ that assigns to each point x a displacement vector $\phi(x)$ such that $\Psi(x) = x + \phi(x)$ (therefore, the zero deformation field $\phi \equiv 0$ represents the identity diffeomorphism). In non-linear image registration, we usually perform a linear registration first so that the images are roughly aligned. To avoid doing more than one interpolation of the input images (which may

introduce registration inaccuracies), our diffeomorphic map includes a pre-alignment matrix P so that $\Psi(x) = Px + \phi(Px)$ (Fig. 1). The deformation field itself is discretized on its own grid (which of course is associated to a grid-to-space transform), and is evaluated at non-integer coordinates by interpolation. Internally, the diffeomorphic map holds the "default" domain's and codomain's grids, so that images can be "warped" without the need of specifying their discretization.

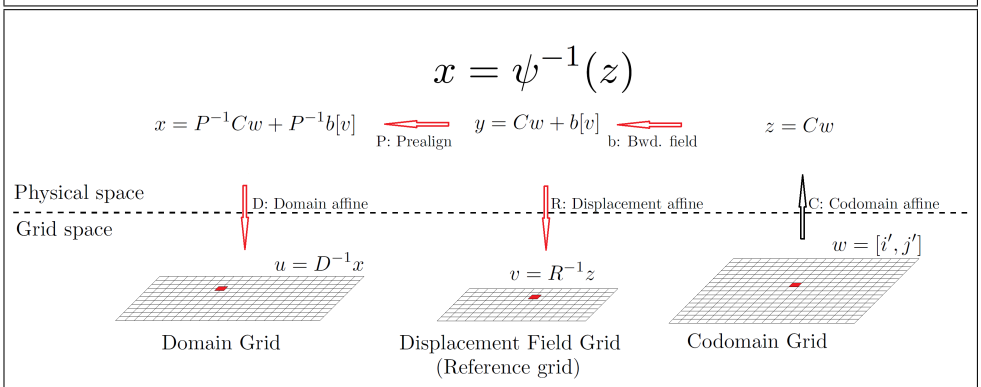
To "warp" an image I, defined on $\mathcal{L}_I$, towards an image J, defined on $\mathcal{L}_J$, we need to take each voxel $j \in \mathcal{L}_J$ and "pull" the intensity value of I at its corresponding point in $\mathcal{L}_I$. This means that by applying a diffeomorphism Ψ **in the forward direction** we "pull" values from the codomain of Ψ towards its domain. In other words, "warping" an image by Ψ **in the forward direction**, means transforming an image from Ψ's codomain towards Ψ's domain. Similarly, "'warping' an image by Ψ **in the backward direction**, means transforming an image from Ψ's domain towards Ψ's codomain (Fig. 1).

The DiffeomorphicMap class in Dipy's registration module contains the following fields (Fig. 1):

```
domain_shape        : Domain grid shape
domain_affine       : Domain grid to physical space transform
domain_affine_inv   : Physical space to domain grid transform
prealign            : Affine transform from points in the domain to points in the displacement field domain
prealign_inv        : Affine transform from points in the displacement field domain to points in the domain
disp_shape          : Displacement field grid shape
disp_affine         : Displacement field grid to physical space transform
disp_affine_inv     : Physical space to displacement field grid transform
codomain_shape      : Codomain grid shape
codomain_affine     : Codomain grid to physical space transform
codomain_affine_inv: Physical space to codomain grid transform
forward             : Forward displacement field
backward            : Backward displacement field
```



(a) **Forward** transform: maps points in the domain to points in the codomain.



(b) **Backward** transform: maps points in the codomain to points in the domain.

Figure 1: Diffeomorphic map. We can map points back and forth between the domain and codomain. The forward transform maps points from the domain to the codomain, which defines the "pull-back" that can be used to warp an image **from the codomain grid towards the domain grid**. Similarly, the backward transform is the pull-back warping images **from the domain grid towards the codomain grid.**

3 Symmetric Diffeomorphic Registration

The greedy algorithm for Symmetric Diffeomorphic Registration ("Greedy SyN") finds a diffeomorphism mapping back and forth between two given images by looking for two diffeomorphisms mapping the given images to an "intermediate" shape and then composing the intermediate mappings to find the diffeomorphism between the two original images (Fig. 2).

Both displacement fields (forward and backward) have the same discretization (same grid shape and grid-to-space transform), which means that the deformation fields actually define endomorphisms. The domain of these endomorphisms (equal to their codomain) is called "reference domain", and similarly, their discretization grid is called "reference grid".

For convenience, the reference discretization (grid shape and grid-to-space transform) is arbitrarily chosen to be the same as the static image. As a consequence, the prealigning matrix corresponding to the static-to-reference diffeomorphism (Ψ_1 in fig. 2) is the identity (only the moving image is pre-aligned to the reference).

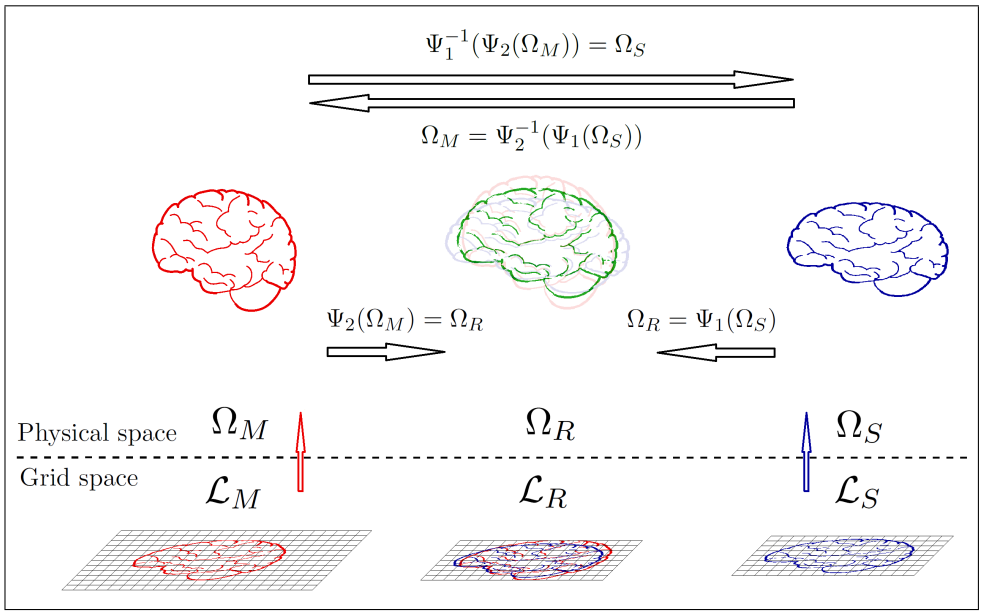


Figure 2: Symmetric Diffeomorphic Registration. The process may be regarded as looking for an "intermediate" shape defined on a reference domain Ω_R such that it can be warped towards the static and moving images by diffeomorphisms Ψ_1, Ψ_2, respectively. The diffeomorphism warping the moving image towards the static, and vice versa, can then be found by composition.

The fields in SymmetricDiffeomorphicRegistration class, in Dipy's registration module, representing Ψ_1, Ψ_2 are

```
static_to_ref : the DiffeomorphicMap between the static image and the reference grid
moving_to_ref : the DiffeomorphicMap between the moving image and the reference grid
```

The following is an overview of the Greedy SyN algorithm:

Algorithm 1 Overview of the Greedy SyN algorithm

Require: Static image S
Require: Moving image M
Initialize diffeomorphism $\Psi_1 = \text{identity}$
Initialize diffeomorphism $\Psi_2 = \text{identity}$
repeat
    Warp the Static image to the reference grid: $S_w = \Psi_1^{-1}(S)$
    Warp the Moving image to the reference grid: $M_w = \Psi_2^{-1}(M)$
    Compute the fwd. step f, (pull M_w towards S_w)
    Update $\Psi_1(\cdot) = f(\Psi_1(\cdot))$
    Compute the bwd. step b, (pull S_w towards M_w)
    Update $\Psi_2(\cdot) = b(\Psi_2(\cdot))$
    Invert: $\Psi_1^{-1} = \text{invert}(\Psi_1)$
    Invert: $\Psi_2^{-1} = \text{invert}(\Psi_2)$
    Invert: $\Psi_1 = \text{invert}(\Psi_1^{-1})$
    Invert: $\Psi_2 = \text{invert}(\Psi_2^{-1})$
until Convergence
return $\Psi_2^{-1}(\Psi_1(\cdot))$