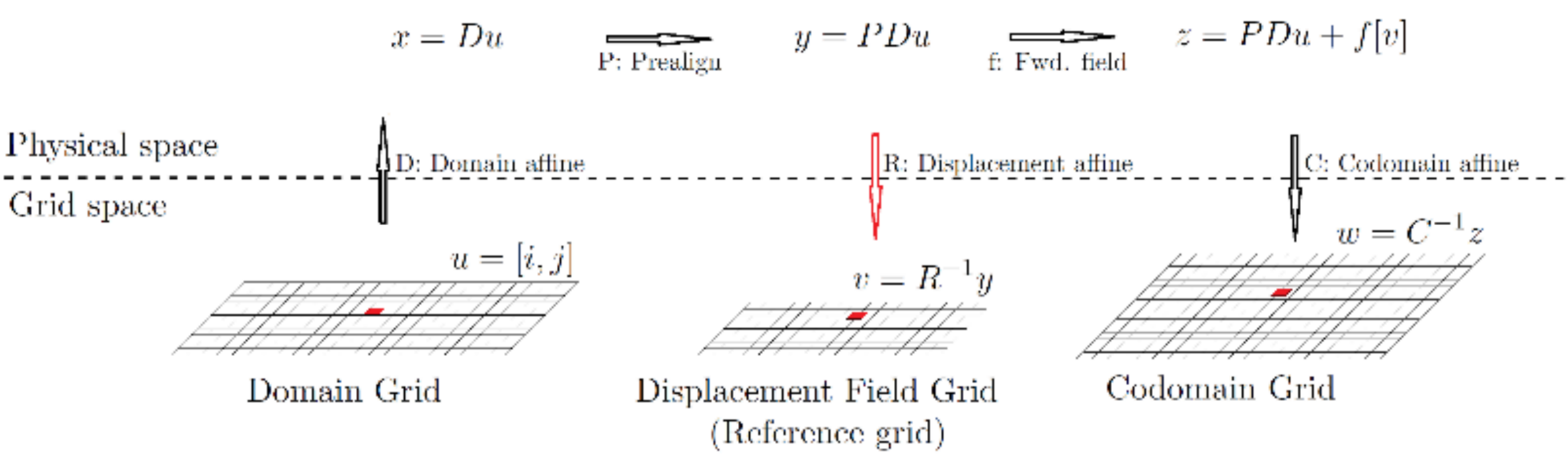
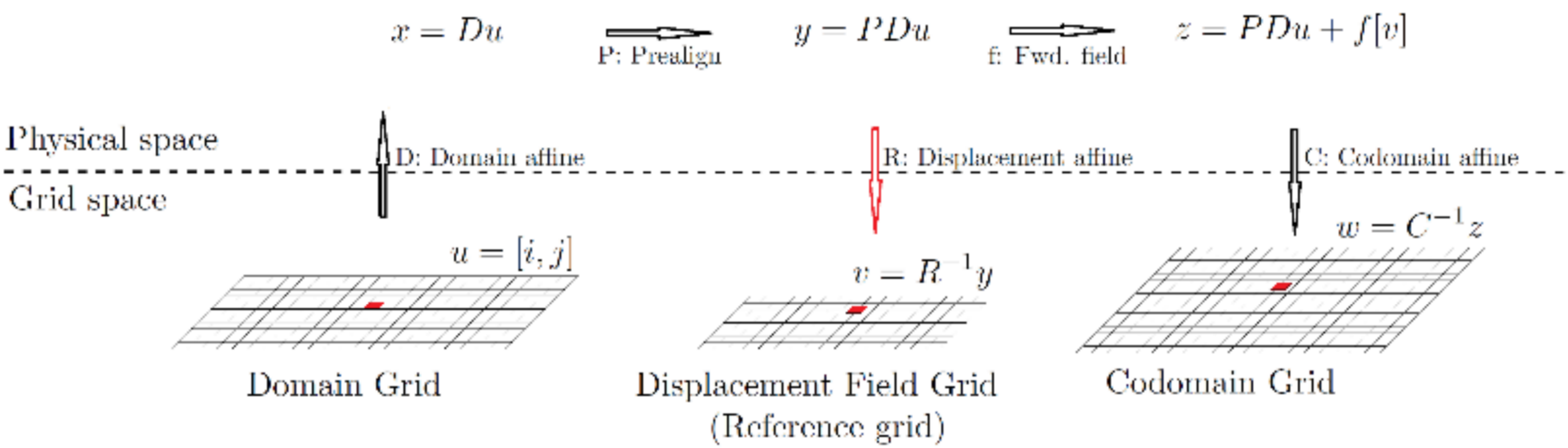
We regard an *image* as a function I that maps voxels of a grid $\mathcal{L}$ to a set G of possible values called the "dynamic range" of I. The images we are interested in represent objects in physical space, $\mathbf{R}^3$. This means that each point $[i, j, k]$ in the 3-dimensional grid $\mathcal{L}$ is associated to a point $(x, y, z) \in \mathbf{R}^3$.

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.

$$\psi(x) = z$$

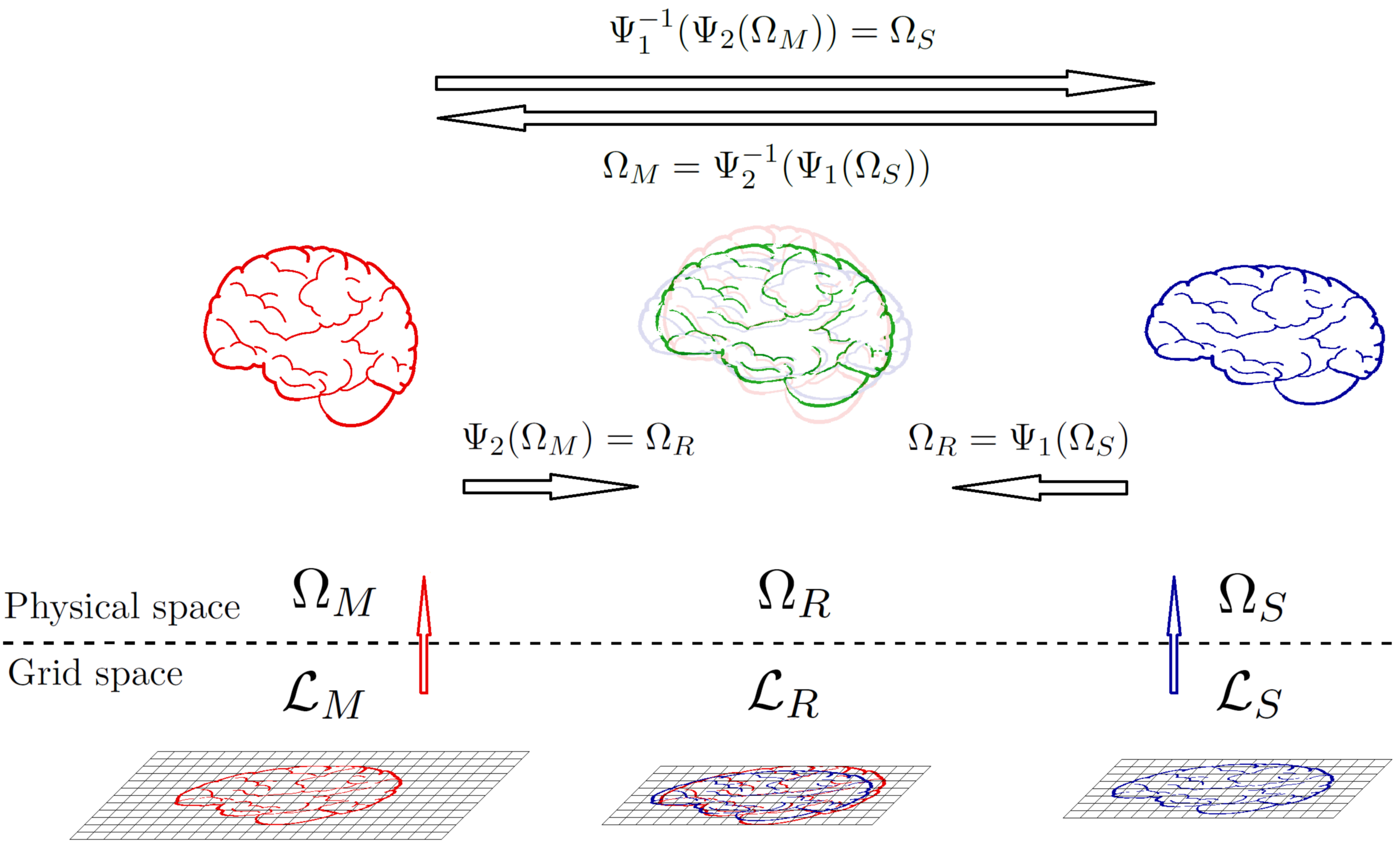


$$\psi(x) = z$$



The DiffeomorphicMap class in Dipy's registration module contains the following fields

```
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
```



$\Psi_1^{-1}(\Psi_2(\Omega_M)) = \Omega_S$
$\Omega_M = \Psi_2^{-1}(\Psi_1(\Omega_S))$
$\Psi_2(\Omega_M) = \Omega_R$
$\Omega_R = \Psi_1(\Omega_S)$
Physical space
Ω_M
Ω_R
Ω_S
Grid space
$\mathcal{L}_M$
$\mathcal{L}_R$
$\mathcal{L}_S$

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 $\Psi_2^{-1}(\Psi_1(\cdot))$

$$\iint (f-g)^2$$

$$NCC(u,v) = \frac{\iint_A f(x,y) \cdot g(x+u, y+u)\,dx\,dy}{\left[\iint_A f^2(x,y)\,dx\,dy \cdot \iint_A g^2(x+u, y+v)\,dx\,dy\right]^{\frac{1}{2}}}$$

- $H = -\sum p_{i,j} \log p_{i,j}$ is the Shannon entropy for a joint distribution; p_{ij} is probability of co-occurence of i and j.

  - Def. 1: I(A,B) = H(B) – H(B|A)
  - Def. 2: I(A,B) = H(A) + H(B) – H(A,B)
  - Def. 3: Kullback-Leibler distance

$$I(A,B) = \sum_{a,b} p(a,b) \log \frac{p(a,b)}{p(a)p(b)}.$$

joint gray values

joint in case of independent images