Notes

For nonlinear registration, there are three options: Geodesic SyN, BSpline SyN, and Greedy SyN. For the purpose of time, I chose Greedy SyN for this pseudocode. We have been having trouble with time in nearly every step of our pipeline and registration is done on 3 times the amount of data we've been dealing with. Thus, I'd like to try steering towards algorithms that are not very time-intensive, like Greedy SyN. If we find that Greedy SyN is not as accurate as we'd like it to be, I think we should explore Geodesic and BSpline SyN. But, I believe we should explore time-efficient algorithms first, such as Greedy SyN.

Additionally, you might note that the first section of this algorithm is very similar to Richard's linear ANTs algorithm. This is because the first step of the nonlinear ANTs algorithm is linear alignment.

Pseudocode Outline

1. Roughly align datasets using linear registration
    (a) Align centers
    (b) Align orientations
    (c) Accunt for scaling factors
2. Fine alignment using nonlinear registration

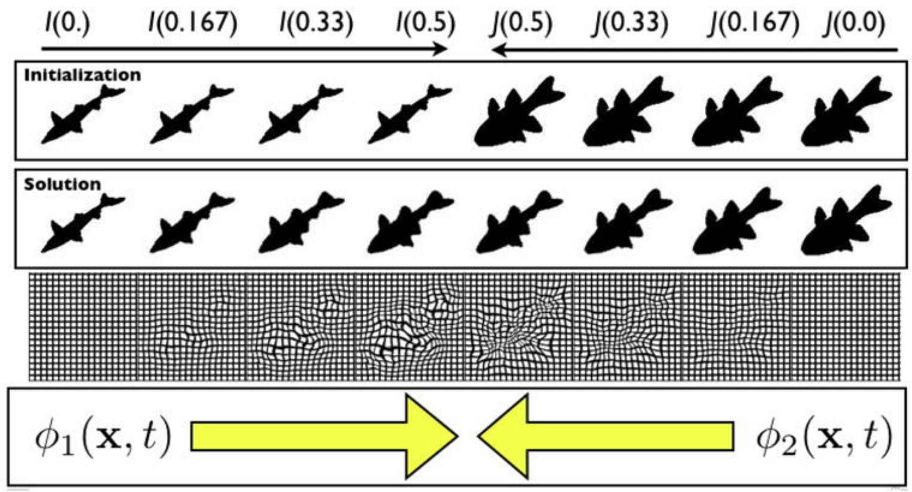
Actual Pseudocode

```
Algorithm: ANTS Non-Linear Registration Pseudocode
Input: Fixed image, moving image, similarity metric (SSD, MSQ, or MI)
Output: Moving image after registration to fixed image

1.  similarity = calculateSimilarityMetric(similarity metric, fixed image, moving image)      Compare images to see how similar they are.
2.  image = moving image
3.  while similarity not MAX:                                                                  Optimize similarity between images
4.      image = alignCenters(fixed image, image)                                               Translation movement to align object centers.
5.      similarity = calculateSimilarityMetric(similarity metric, fixed image, image)
6.  endWhile
7.  while similarity not MAX:
8.      image = alignOrientation(fixed image, image)                                           Rigid body transform to match orientation of objects.
9.      similarity = calculateSimilarityMetric(similarity metric, fixed image, image)
10. endWhile
11. while similarity not MAX:
12.     image = scale(fixed image, image)                                                      scale fixed image to match moving image.
13.     similarity = calculateSimilarityMetric(similarity metric, fixed image, image)
14. endWhile
15. while similarity not MAX:
16.     image = affine(fixed image, image)                                                     Linearly match objects as close as possible using affine transformations
17.     similarity = calculateSimilarityMetric(similarity metric, fixed image, image)
18. \phi_i = identitiy matrix
19. \phi_i^{-1} = identitiy matrix
20. n = 1
21. while \phi_i not converged:                                                                \phi_i for i ∈{1,2} are diffeomorphic "half-maps" (explained in notes section below)
22.     v_1^n = calculateSimilarityMetric(similarity metric, \phi_1^{n-1}(image, 0.5), \phi_2^{n-1}(moving image, 0.5)
23.     v_2^n = calculateSimilarityMetric(similarity metric, \phi_2^{n-1}(moving image, 0.5), \phi_1^{n-1}(image, 0.5))
24.     v_i^n = S_v(v_i^n) for i ∈{1,2}                                                        S_v is a smoothing operation on the transform field v_i^n that filters noise
25.     \phi_i^n = S_\phi(v_i^n ∘ \phi_i^{n-1}) for i ∈{1,2}                                   S_\phi is a smoothing operation on the total transform field that filters noise
26.     n = n + 1
27. endWhile
28. \phi = \phi_1 ∘ \phi_2^{-1}
29. return \phi
```

Notes On Pseudocode

For line 25, it's important to note that $\frac{d\phi_i(x,t)}{dt} = v_i(\phi_i(x,t),t), \phi_i(x,0) =$ Identity Matrix, for $i \in \{1,2\}$



The above image is an illustration of the difference between ϕ, ϕ_1, and ϕ_2 – mainly, that ϕ_1 and ϕ_2 are "half-maps" of the total diffeomorphic map, ϕ. That is, if ϕ is a map from I to J, then $\phi_1(\mathrm{I}) = \phi_2(J)$. In other words, $\phi = \phi_1 \circ \ \phi_2^{-1}$ (as stated in line 28 of the pseudocode).

The images in top row are the original images, with I = image and J = moving image. After the SyN solution converges (meaning $\phi_1(\mathrm{I}) \approx \ \phi_2(J)$) (as they are in the middle of second row) these images deform in time along the series of diffeomorphisms that connect them. The deforming grids associated with these diffeomorphisms are shown in the bottom row. The maps, ϕ_1 and ϕ_2, map image and moving image to the mean shape between the images as shown in the middle of the second row. The full paths, ϕ and ϕ^{-1}, are found by joining the paths ϕ_1 and ϕ_2. ϕ and ϕ^{-1} are diffeomorphic maps from image (I) to moving image (J) and moving image (J) to image (I), respectively.