Questions and Answers

Chapter 6

Questions

(Q1) *What is the main benefit of using inverse consistent image registration compared uni-directional registration?*

(Q2) *How much computation time is added by converting a uni-directional registration algorithm into a inverse consistent image registration algorithm?*

(Q3) *How do you compute in the inverse of a transformation?*

Answers

(A1) The main benefit of using inverse consistent image registration is that the estimated correspondence between two images improves compared to using a similar uni-directional registration algorithm. The usual task in image registration is to find the pointwise correspondence between two images. However, there is rarely if ever a "gold standard" to evaluate the performance of a non-rigid image registration algorithm. Therefore in the absence of a gold standard, we take the approach of defining necessary conditions or properties that the transformations need to satisfy to give the desired correspondence. The properties we normally impose on

the transformations include continuity, differentiability, and one-to-one. In addition, it is logical to assume that the correspondence defined by the transformation should be unique assuming that a unique correspondence mapping exists. However, unidirectional registration algorithms do not in general define a unique correspondence between two images. This can be seen by comparing the forward transformation computed from image A to B to the reverse transformation computed from image B to A. Since the forward and reverse transformations are estimated separately, they rarely if ever agree with each other and therefore do not define a unique correspondence between the images. Inverse consistent registration provides a unique correspondence when the inverse consistency error is zero over the whole domain of the images. This rarely happens in practice, so the best that can be said about inverse consistent registration is that it minimizes one source of registration error and therefore provides better correspondence between the images.

(A2) The added computation cost for converting a uni-directional registration algorithm to an inverse consistent algorithm is more than double the unidirectional algorithmic time. The main computational components of the algorithm double because both the forward and reverse transformations are estimated at each iteration. In addition, the inverse of the forward and reverse transformations are computed at each iteration. The computation of the inverse is a substantial fraction of the computation time per iteration. The exact time depends on the allowable error in estimating the inverse. For some applications 1% of a pixel dimension is an allowable error and in others 0.01% is required. The smaller the allowable error, the more computations are required. The typical additional computational cost required for computing the inverses is between 20–60% depending on the desired accuraccy.

(A3) The following algorithm computes the discrete inverse h^{-1} of the transformation h where Ω_d is the discrete domain of h^{-1}. This procedure only works for computing the inverse of a small deformation transformation, i.e., a transformation with a positive Jacobian at all points in the domain.

For each $n \in \Omega_d$ do {
  Set $\delta = [1, 1, 1]^T$, $x = [\frac{n_1}{N_1}, \frac{n_2}{N_2}, \frac{n_3}{N_3}]^T$, iteration $= 0$.
  While ($||\delta|| >$ threshold) do {
    $\delta = [\frac{n_1}{N_1}, \frac{n_2}{N_2}, \frac{n_3}{N_3}]^T - h_d[\frac{x_1}{N_1}, \frac{x_2}{N_2}, \frac{x_3}{N_3}]$
    $x = x + \frac{\delta}{2}$
    iteration = iteration + 1
    if (iteration > max_iteration) then
      Report algorithm failed to converge and exit.
  }
  $h_d^{-1}[n] = x$
}

This algorithm searches for the inverse for each voxel in the domain of h^{-1} independently. The result from the previous voxel can be used to initialize the search for the inverse at the current voxel to speed up the algorithm. This algorithm is computationally fast but gets stuck in oscillations. To avoid oscillations, the step size can be reduced when an oscillation is detected. When reducing the step size does not work, a gradient descent algorithm can be used to find the inverse value.