An introduction to ITK version 4 registration

Anonymous

No Institute Given

Abstract. The ITKv4 registration framework is a unified system for performing multi-threaded affine and deformable image registration. The revised framework supports composite transformations, unbiased registration, the simultaneous use of multiple similarity metrics, multi-channel/tensor image registration and geometrically correct transformation of covariant vectors and tensors via the composite transform framework and transformations based on finite element models. ITKv4 also contains new metrics that can be used for registering point sets, curves and surfaces as well as a set of efficiently implemented neighborhood correlation metrics. Despite these significant additions, the user interface to the framework is, at the basic level, unchanged from prior versions of ITK. Furthermore, we provide new optimization strategies that simplify the user experience by reducing the number of parameters that need to be set by the user.

1 Introduction

What is the current state of image registration? As image registration methods mature—and their capabilities become more widely recognized—the number of applications increase. Consequently, image registration has largely transitioned, over the last 10 years, from being a field of active research, and few applied results, to one where the main focus is translational. Image registration is now used to derive quantitative biomarkers from images, plays a major role in some clinical products (especially in radiation oncology), has led to numerous new findings in studies of brain and behavior and is a critical component in applications in pathology, microscopy, surgical planning and more.

What role has ITK filled in the registration world? The Insight ToolKit is a standard-bearer for medical image processing algorithms and, in particular, for image registration methods. Image registration was cited as the number X most important contribution of ITK (cite Jesus's work). Numerous papers use ITK algorithms as standard references for implementations of Demons registration and mutual information-based rigid registration. Multiple toolkits extend ITK registration methods in exciting ways: Elastix, Slicer, ANTs (name more). Thus, the impact of having only a few well-implemented methods, in the 3.x version of ITK, is significant.

ITKv4 Registration: What it is. The original goals of the version 4 registration framework refactoring were to both simplify and extend the techniques available in version 3.x. We aimed to unify the dense (PDE, FEM) registration framework with the low-dimensional (B-Spline, Affine, rigid) framework by introducing composite transforms, deformation field transforms and specializations that allowed these to be optimized efficiently. We intended to add robust metrics to the toolkit. We aimed to simplify

parameter setting by adding helper methods that use well-known principles of image registration to automatically scale transform components and set optimization parameters. We also sought to apply ITK transforms to objects such as variable length vectors, covariant vectors and tensors while taking into account orientation if necessary. Finally, our goal was to reconfigure the whole framework to use multi-threading in as many locations as possible and ultimately compare the multi-core approach to GPU-specific implementations. These goals have largely been achieved.

ITKv4 Registration: What it's not. Although the current framework exploits multi-core architectures, we did not focus on optimal speed implementations (except on the GPU). We did not implement parameterized surface-based registration methods, although the point-set metric framework can be used to achieve something of the sort. Integration with the FEM framework is currently minimal. Additional validation is necessary. The ability to stream large datasets to the registration framework is not tested and has not entered, at a deep level, into design considerations. These issues can all be overcome by external toolkits without fundamental changes to the existing code base.

The remainder of the document will provide an overview of the new framework. We first establish nomenclature. We then summarize the new framework's capabilities. The next section will highlight functionality through a series of examples. We then establish performance benchmarks for ITKv4 release XXX. Finally, we discuss the future of the framework.

2 Nomenclature

We will use the nomenclature below to designate an image registration algorithm pictorially. This nomenclature is intended to be a descriptive, but also technically correct, system for visually representing algorithms and applications of registration. Ideally, any standard algorithm can be written in the nomenclature below.

A position: $\mathbf{x} \in \Omega$ where Ω is the domain.
An image: $I\colon \Omega^d \rightarrow \mathbb{R}^n$ where n is the number of components per pixel and d is dimensionality.
Domain map: $\phi\colon \Omega \rightarrow \Omega$ where $\rightarrow$ may be replaced with any mapping symbol below.
Affine mapping: $\rightarrow$ a low-dimensional transformation: affine, rigid, translation, etc.
Deformation field: $\rightsquigarrow$ deformation field mapping J to I.
Spline-based mapping: $\overset{\rightsquigarrow}{b}$ e.g. B-Spline field mapping J to I.
Diffeomorphic mapping: $\leftrightsquigarrow$ these maps should have an accurate inverse that is computed in the algorithm or can be computed from the results.
Composite mapping: $\phi = \phi_1(\phi_2(\mathbf{x}))$ is defined by $\leftrightsquigarrow\rightarrow$ where ϕ_2 is of type $\leftrightsquigarrow$ which precedes the application of $\rightarrow$.
Not invertible: $\nleftrightarrow$ indicates a mapping that is not guaranteed invertible.

A standard Demons registration application that maps a labeling from one image, I, into another, J, would then be written:

$$J \rightsquigarrow\rightarrow I.$$

The notation means that the algorithm first computes an affine mapping from I to J and then computes a deformation from $I(\rightarrow)$ to J. Note, also, that the tail of the mapping indicates the transform's domain. The head of the arrow indicates its range. This is an important distinction for deformable maps.

3 Overview of the unified framework

4 From Mathematics to Code

The metric is defined as $M(I, J, T(p, x))$. For instance, $M = \|I(x) - J(T(p, x))\|^2$. Its gradient with respect to parameter p is

$$\frac{dM}{dp} = \frac{dM}{dJ}\frac{dJ(T(p,x))}{dT}\frac{dT}{dp}|_x \ .$$

This is the generic form of the metric gradient specified for sum of squared differences but similar forms arise for the correlation and mutual information.

In terms of code $\frac{dT}{dp}|_x$ corresponds to *ComputeJacobianWithRespectToParameters(mappedFixedPoint, jacobian);* . Note that it is evaluated at point x not at point $T(p, x)$. We then use *ComputeMovingImageGradientAtPoint(mappedMovingPoint, mappedMovingImageGradient);* to compute the moving image gradient when there is no pre-warping. *ComputeMovingImageGradientAtPoint* uses central differences (or a gradient filter) in the moving image space to compute the image gradient, $\frac{dJ(T(p,x))}{dT}$.

If one is doing pre-warping, then we have an index access to the warped moving image. We compute the warped J as $J_w(x) = J(T(p, x))$. Then,

$$\begin{aligned} \frac{dJ_w}{dx} &= \frac{dJ(T(p,x))}{dT}\frac{d(T(p,x))}{dx} \\ \frac{dJ(T(p,x))}{dT} &= \frac{dJ_w}{dx}\frac{d(T(p,x))}{dx}^{-1} \end{aligned} \tag{1}$$

In code, we use *ComputeMovingImageGradientAtIndex(index, mappedMovingImageGradient);* to get $\frac{dJ_w}{dx}$ and transform this image gradient via the inverse jacobian by calling *mappedMovingImageGradient= TransformCovariantVector(mappedMovingImageGradient, mappedMovingPoint);*

4.1 Simplified registration example

4.2 Transform changes

Deformation field transforms

Composite transforms Composite transform I/O

4.3 Optimizer changes

Composite transform optimization

4.4 Metric changes

The model used for multi-threading.

Correlation metrics

Pointset metrics I/O and data representation

Revised mutual information metric

Multivariate metric

Multi-channel metric

4.5 The virtual domain

Application in multi-resolution optimization

Unbiased registration

5 ITKv4 registration by example

5.1 Composite registration

5.2 Demons registration

5.3 BSpline registration

5.4 Diffeomorphic registration

5.5 Multi-channel registration

5.6 Tensor registration

5.7 Diffeomorphic registration

5.8 Diffeomorphic registration with two metrics: Image and PointSet

5.9 Optimized SyN registration

5.10 GPU Demons

6 ITKv4 registration benchmarks

Comparison with ANTs and Flirt.
Speed comparison for multi-threaded implementation.
etc.

7 Discussion and future work