ITK v4 refactoring notes

Anonymous

March 9, 2011

Abstract

Current status and goals for the refactoring.

1 Introduction

graceful failure of an algorithm. Should not be catastrophic. A nice feedback loop for software.

procedure: 1. clinicians are conservative. 2. first, prove robustness. 3. provide useful and encouraging feedback. 4. fail gracefully when do fail.

good average performance versus good performance on a given dataset.

2 Nomenclature

Transport versus diffeomorphism intrasubject versus intersubject.

Resolution effects ...

accuracy vs precision obsessed with accuracy because we dont know the precision we need

Evaluation data is there any?

Serial vs longitudinal short time scale versus long time scale

Pathology appearance changes ...

whole body atlas map any image to whole body

radiation oncology techs understand translation versus rotation versus deformation.

3 Deliverables

3.1 Usability

Automate parameter scaling base this on empirical statistics "learning" on a per registration problem basis.

GetParameterSuggestion metric and trasformation classes should recommend parameters from a developer-defined set.

Multi-core implementations multi threading of metric, regularization, parameter update, etc. stephen indicates that the setparameters function may cause problems.

Unify the dense and sparse frameworks metrics and transformations should be reusable across frameworks.

3.2 Data Types

Transform vectors, curves and tensors with reorientation.

3.3 Metrics

metrics derivatives should be bi-directional.

MI and NMI Shreyas — MI and NMI multicore.

ATG Neighborhood Cross Correlation our approximation to the NCC derivative.

PSE Metric with arbitrary data type. ObjectMetric ...

Tractography/vector flow metric vector based. distance transform?

Multivariate metric plug in metrics and weights and a "combination" strategy. e.g. match 1-norm, 2-norm, etc before weighting.

3.4 Transformations

BSpline Nick DMFFD and refactoring , bug fixing. Usability and speed.

Composite transformation need to get the composite derivatives right.

Deformation and rotation transform have jeff implement. smooth rotation component internally (after update parameters).

Velocity field with fourth order integration.

Reorientation transformation and derivative tensor and vector.

Rigid and deformable takes a mask that defines which parts are deformable and which are rigid. the rigid parts are fixed by pre-computation. the deformable parts are modified according to a registration strategy.

3.5 Regularization

For instance, we should be able to compute a "demons" registration without passing deformations to the filter. We should pass transformations (e.g. affine) and regularization (e.g. distance from identity) in addition to the demons metric.

3.6 Longitudinal and serial registration

Facilitate through transformations and regularization and unbiased design.

longitudinal multiple images of the same subject over a longer time-scale.

serial or time series multiple images of the same subject or scene with high frequency sampling (samples that are dense in time).

Longitudinal Image Given a set of 3D images sampled longitudinally (see above), we package these n images—along with n rigid or affine transforms that map them to a common domain—into a standard itk image interface.

Serial image this concept may already be supported.

3.7 Algorithms

SyN fully unbiased and lives as an algorithm (not multi-resolution) within ITK. Multi-resolution SyN is a 2nd algorithm.

DMFFD nick's style—greg's?

Longitudinal Diffeomorphic Mapping gang's version.

vector versions of above ... with reorientation in transform. also in optimization?

4 The new framework

Changes that we need to implement ASAP.

4.1 optimizers with regularization

We use the following idea: any gradient based optimizer can be altered to perform the following update scheme $T_{i+1} = R(T_i + \lambda F(g))$ where g is the gradient and λ is the update step. The function F is the regularization on the gradient which will be dealt with by regularized metrics. So, this object should take a transform, an update to a transform and return the regularized transform T_{i+1}. This requires a class of optimizers that takes a regularization function as input `->SetRegularizationFunction.` We will start with the gradient descent optimizer. Build a derived class that has a regularization function and that overrides the update step to use that function. It can be an identity function for now.

4.2 transform changes

The transforms should add a *thread-safe* function called IncrementalUpdateToTransformParameters that takes a delta to the parameters and updates the transform. There will be an implementation in the base class that just adds the parameters. But special transforms may want to override this function. We also want a function LocalJacobian

which will default to the Jacobian computation except in the cases where we override the function for the local, dense transform types.

4.3 deformation field transform

We need to resolve the issues with the dense transform parameters. I.e. map them to an image efficiently or define a new local parameter type that is based on the itkVector image.

4.4 multivariate metric

Takes two ObjectToObjectMetrics as input along with a weight function. Defaults to uniform weighting unless otherwise specified. Weights are applied to the gradients of the metric, not the metric itself.

4.5 multi-threading

We need to be careful in making decisions about all the above. For instance, what is needed to make the transforms thread-safe? The regularizers?

4.6 lightweight resampling

Can we implement a lightweight resample function that is local, thread-safe and samples an image in such a way that code becomes more readable? I.e. does the bounds checking, reorientation, etc. Relatedly, can the resample image filter be rewritten with code reuse as an objective?

4.7 multi-threaded metrics

Gang wrote an initial implementation. We need to continue to work on this — the demons version needs to take a generic iterator type or allow different iterators to be selected by the user.

4.8 the first new registration object

SetImages

SetRegistrationMachineryObjects — what type of object is the registration machinery? really just a holder with reasonable defaults for the stuff that needs to be set up. Relies on get recommended parameters and baohua's scale parameter setter to set up the defaults for optimizer, metric etc.

SetOptimizer — that will be in the machine but the optimizer might have a regularization function which will have parameters. Alternatively, we define regularized metrics that takes an unregularized metric as input along with a regularizer and just returns the regularized gradient—maybe a better idea! A regularized metric will project the gradient to the space in which it should live.

SetResolutions — virtual domains along with scales? kind of unclear

SetVirtualDomain — related to above

RunRegistration — given above, runs the registration. Need a convergence criterion.

5 Frobenius norm registration

A three-dimensional problem with a derivative on $\mathbf{x}$, the spatial domain. Let us assume that $\boldsymbol{I}\colon \Omega \in [0,1]^d \rightarrow \mathbb{R}^d$ where d defines the dimensionality. Homogeneous coordinates make this easier.

$$T(\mathbf{x}) = \mathbf{y} = [A(\theta)]\phi(\mathbf{x}) = [A(\theta)](\mathbf{x} + U(\mathbf{x})) \tag{1}$$

$$\boldsymbol{J}(T) = \boldsymbol{R}^T \boldsymbol{J}(\mathbf{y}) \boldsymbol{R}$$

$\boldsymbol{R}$ = R part of polar decomposition of

$$[A(\theta)](\mathbf{Id} + U_{\mathbf{x}}(\mathbf{x}))$$

must compose affine with deformation gradient

in order that correctly reorients the data.

$$\|\boldsymbol{I} - \boldsymbol{J}(T)\|^2$$

$$\frac{\partial}{\partial \mathbf{x}} \|\boldsymbol{I} - \boldsymbol{J}(T)\|^2$$

$$< \boldsymbol{I} - \boldsymbol{J}(T), \frac{\partial}{\partial \mathbf{x}} (\boldsymbol{I} - \boldsymbol{J}(T)) >$$

$$\frac{\partial}{\partial \mathbf{x}} (\boldsymbol{I} - \boldsymbol{J}(T)) =$$

$$\frac{\partial}{\partial \mathbf{x}} \boldsymbol{J}(T) =$$

chain rule gets ugly ...