Lecture 8
Registration with ITK

Methods in Medical Image Analysis - Spring 2012
BioE 2630 (Pitt) : 16-725 (CMU II)
18-791 (CMU ECE) : 42-735 (CMU BME)
By Dr. John Galeo2 & Dr. Damion Shelton

For more info/gory detail...

- Please see the following for exhaustive detail:
  - Chapter 8 in the ITK Software Guide
    - Currently, this does not cover the newer registration methods
  - Insight into Images
  - ITK Source Tree
    - E.g. Examples/Registration/ImageRegistration1.cxx
  - ITK Doxygen
    - http://www.itk.org/Doxygen40/html/group__RegistrationFilters.html

What is registration?

- The process of aligning a target image to a source image
- More generally, determining the transform that maps points in the target image to points in the source image
Transform types

- Rigid (rotate, translate)
- Affine (rigid + scale & shear)
- Deformable (affine + vector field)
- Many others

Registration in ITK

- ITK uses an extensible registration framework
- Various interchangeable classes exist
- Relatively easy to “twiddle” the part you’re interested in while recycling prior work

ITK terminology

- Fixed image $f(x)$ - stationary in space
- Moving image $m(x)$ - the fixed image with an unknown transform applied
- Goal: recover the transform $T(x)$ which maps points in $f(x)$ to $m(x)$
Registration framework pieces

- 2 input images, fixed and moving
- Metric - determines the “fitness” of the current registration iteration
- Optimizer - adjusts the transform in an attempt to improve the metric
- Interpolator - applies transform to image and computes sub-pixel values

ITK registration flowchart

[Diagram showing the flowchart of ITK registration process]

ITK’s “Hello world” example

- 2D floating point image inputs
- Please see the software guide (section 8.2) for code specifics
- I am going to cover what each piece does, not look at code per se
ITK’s “Hello World” Example: Flow Chart for Everything

Figure 8.5 from the ITK Software Guide v2.4, by Luis Ibáñez, et al.

Input images

- 2D floating point
- Floating point avoids loss of precision problems with integer pixel types

Transform

- TranslationTransform
- Permits translation only in 2D
- Documentation notes similarity to AffineTransform, but improved speed (why...?)
Transform Coordinate System

Metric

- MeanSquaresImageToImageMetric
- Sum of squared differences between 2 images on a pixel-by-pixel basis
- A bit naive
- Works for 2 images that were acquired with the same imaging modality

Optimizer

- RegularStepGradientDescent
- Follows the derivative of the metric
- Step size depends on rapid changes in the gradient’s direction
- Step size eventually reaches a user-defined value that determines convergence
Interpolator

- LinearInterpolateImageFunction
- Fast and conceptually simple

Wrapper

- ImageRegistrationMethod
- Combines all of the previous classes into a master class

registration->SetMetric( metric );
registration->SetOptimizer( optimizer );
registration->SetTransform( transform );
registration->SetInterpolator( interpolator );

Other steps

- Set the region of the fixed image the registration will operate on (e.g. to ignore bad data)
- Initialize the transform
- Twiddle the optimizer for best performance

*may involve pain and suffering
Hello world input

Figure 8.3 from the ITK Software Guide v.2.4, by Luis Ibáñez, et al.

X & Y translation vs. time

Figure 8.6 (top) from the ITK Software Guide v.2.4, by Luis Ibáñez, et al.

Metric vs. time

Figure 8.6 (bottom) from the ITK Software Guide v.2.4, by Luis Ibáñez, et al.
Registration results

- After registration converges/terminates you call GetLastTransformParameters to recover the final transform
- For the Hello World example there are 2 parameters, X & Y translation

Double checking results

- Use ResampleImageFilter to apply the transform for the fixed image
- Take the output, compute a difference image with the moving image, examine the results
- Good registration results in nothing much to see

Image comparison

Figure 8.4 from the ITK Software Guide v 2.4, by Luis Ibáñez, et al.
Keeping tabs on registration

- Registration is often time consuming
- It's nice to know that your algorithm isn't just spinning its wheels
- Use the observer mechanism in ITK to monitor progress
  * See the software guide, 3.2.6 and 8.4
- We’ll see this again later, when we discuss how to write your own ITK filters
  * itk::ProgressEvent is one example

Observer steps

- Write an observer class that will process "iteration" events
  * (Just copy some code from an example)
- Add the observer to the optimizer
  * As a generic note, observers can observe any class derived from itk::Object
  * Start registration as usual

Things observers can do

- Print debugging info
- Update GUI
- Other small management functions
  * Should not do anything too processor intensive
Multi-modality registration

- Remember how I said sum-of-squares difference is relatively naïve?
- Mutual information helps overcome this problem
- Section 8.5 shows how to implement a simple MI registration

Notes about the MI example

- Significantly, largely the same piece of code as Hello World
- Mutual Information is a metric, so we can keep the optimizer, the interpolator, and so on
- Majority of differences are in tweaking the metric, not in rewriting code

MI Inputs

T1 MRI  Proton density MRI

Figure 8.9 from the ITK Software Guide v2.4, by Luis Ibáñez, et al.
MI Output: Image Comparison

This is an example of a checkerboard visualization

Centered transforms

• More natural (arguably) reference frame than having the origin at the corner of the image
• In SimpleITK, transforms are automatically centered by default!
• Details are not appreciably different from other rigid registrations, see 8.6

SimpleITK Registration (Python)

```python
import SimpleITK as sitk

imgT1 = sitk.ReadImage("MRI_T1.tif")
imgPD_shifted = sitk.ReadImage("MRI_PD_shifted.tif")

transform = sitk.AffineTransform()
interp = sitk.LinearInterpolate()
metric = sitk.MattesMutualInformationMetric()
optimizer = sitk.RegularStepGradientDescentOptimizer()

params = sitk.Register( imgPD_shifted, imgT1, transform,\ interp, metric, optimizer )

print(params)
```
SimpleITK Registration (C++)

```cpp
#include <SimpleITK.h>

int main(void) {
    itk::simple::ImageFileReader reader;
    itk::simple::Image fixed = reader.SetFileName("FixedImage.nii").Execute();
    itk::simple::Image moving = reader.SetFileName("MovingImage.nii").Execute();
    itk::simple::AffineTransform transform;
    itk::simple::MattesMutualInformationMetric metric;
    itk::simple::LinearInterpolate interpolate;
    itk::simple::RegularStepGradientDescentOptimizer optimizer;

    // Longer form (Python can likewise also set these one at a time):
    itk::simple::Registration registration;
    registration.SetTransform(&transform);
    registration.SetMetric(&metric);
    registration.SetInterpolate(&interpolate);
    registration.SetOptimizer(&optimizer);

    std::vector<double> params;
    params = registration.Execute(fixed, moving);
}
```

An aside: “Twiddling”

- A common criticism of many/most registration techniques is their number of parameters
- A successful registration often depends on a very specific fine-tuning of the algorithm
- “Generalized” registration is an open problem

**WARNINGS for SimpleITK beta 0.3:**
- Beta v. 0.3 currently does NOT allow twiddling of any of the registration parameters
- The beta’s choices of registration parameters are not necessarily very good yet.

Multi-Resolution registration

- Useful to think of this as algorithmic “squinting” by using image pyramids
- Start with something simple and low-res
- Use low-res registration to seed the next higher step
- Eventually run registration at high-res
- Also called “coarse to fine"
Multi-resolution schematic

Image pyramids

Optimization

- Parameter dependency rears its ugly head
- You often/usually need to adjust optimizer parameters as you move through the pyramid
- You can do this using the Observer mechanism
Multi-resolution example

- Again, mostly the same code as Hello World
- Use MultiResolutionPyramidImage filter to build fixed and moving pyramids
- MultiResolutionImageRegistrationMethod is now the overall framework

Benefits of multi-resolution

- Often faster
- More tolerant of noise (from “squinting”)
- Minimizes initialization problems to a certain extent, though not perfect

See the software guide for...

- Detailed list of:
  - Transforms
  - Optimizers
  - Interpolation methods
  - You’re encouraged to mix and match!

- Note: ITKv4’s new registration methods are still being developed, and the documentation is not yet complete for them.
- Check Doxygen and ITK source code for ImageRegistrationMethodv4
Deformable registration

- ITK has 2 primary techniques:
  - Finite element: treat small image regions as having physical properties that control deformation
  - Demons: images are assumed to have iso-intensity contours (isophotes); image deformations occur by pushing on these contours

Model based registration

- Build a simplified geometric model from a training set
- Identify parameters that control the characteristics of the model
- Register the model to a target image to adapt to a particular patient

Model based, cont.

- Uses the Spatial Objects framework for representing geometry
- Useful because it derives analytical data from the registration process, not just a pixel-to-pixel mapping
Model-based example

Note: This is what we want, NOT the output of an actual registration

Figure 8.60 from the ITK Software Guide v2.4, by Luis Ibáñez, et al.

Model-based reg. schematic

Figure 8.59 from the ITK Software Guide v2.4, by Luis Ibáñez, et al.

Model-based registration: Warning!

- ITK does not yet directly support generic model-based registration “out of the box”
- ITKv4 does support point-set to image registration
- Otherwise, model-based reg. requires writing your own custom ITK transform, with new parameters
  - Transform’s new parameters → Spatial Object parameters
  - You must individually map your custom transform’s new parameters to the specific spatial object parameters you want to allow registration to adjust
- This isn’t too complicated if you know what you’re doing
- Search Insight Journal for examples
Speed issues

- Execution time can vary wildly
  - Optimizer (more naïve = faster)
  - Image dimensionality (fewer = faster)
  - Transform (fewer DOF = faster)
  - Interpolator (less precise = faster)

New in ITKv4 (ImageRegistrationMethodv4, etc.)

- New unified and fully multi-threaded optimization and registration framework
- Unified framework supports sparse and dense metric computation
- Unified framework supports low and high-dimensional mapping
- Improved multi-threaded metrics for rigid, affine and deformable registration
- New metrics support sparse or dense sampling
- Metrics for landmark or label guided registration

- Automatic parameter scale estimation for registration
- Automatic step size selection for gradient-based registration optimizers
- Composite transforms and composite transform optimization
- Displacement field and diffeomorphic velocity field-based transforms
- Better support for multi-threading in optimizers and metrics
- Additional evolutionary optimizers
- Improved B-spline registration approach available and bug fixes to old framework
- Accurately transform and reorient covariant tensors and vectors

List taken from http://www.itk.org/Wiki/ITK_Release_4/Why_Switch_to_ITKv4

Take home messages

- Exactly what parameters do what is not always obvious, even if you are familiar with the code
- Successful registrations can be something of an art form
- Multi-resolution techniques can help
- Work within the framework!