Parallelising Image Registration and the HPC Porting Journey

P. Tooley[1,2], I. Benemerito[2], A. Melis[2],
A. Narracott[2], A. Marzo[2], M. Viceconti[1,3]

[1]Numerical Algorithms Group
[2]University of Sheffield
[3]University of Bologna

27 September 2019

Optical Flow Image Registration

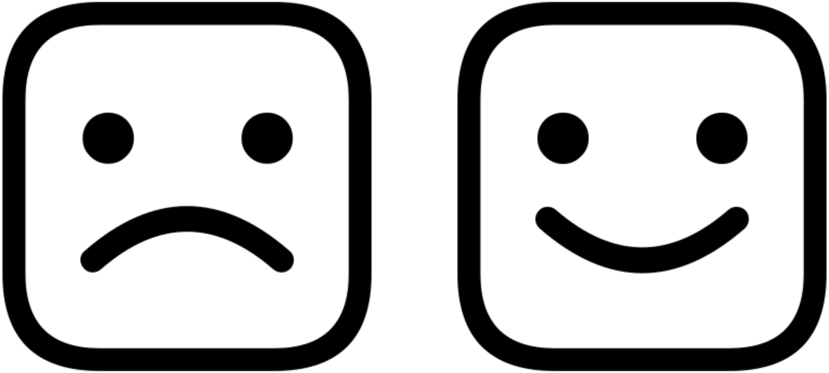
Image Registration

Image registration is the process by which one image is transformed by displacing pixels to match another image as closely as possible.

Optical Flow Image Registration

Image Registration

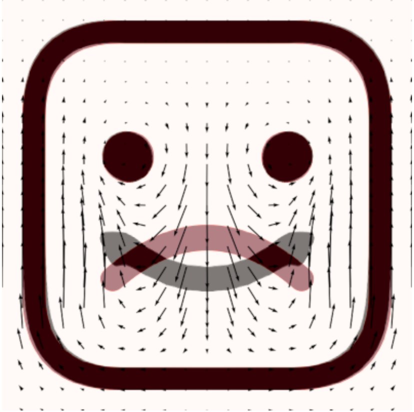
Image registration is the process by which one image is transformed by displacing pixels to match another image as closely as possible.



Optical Flow Image Registration

Image Registration

Image registration is the process by which one image is transformed by displacing pixels to match another image as closely as possible.



Optical Flow Image Registration

Optical Flow Method

"Automatic segmentation of medical images using image registration"

DC Barber & DR Hose (2005) , Journal of Medical Engineering & Technology, 29:2

- Vector "flow" field $\vec{A}$ displaces pixels between images F and M

$$F(\vec{x}) = M(\vec{x} + \vec{A}(\vec{x})) \tag{1}$$

- Taylor expand and linearise to get registration equation

$$F(\vec{x}) - M(\vec{x}) \simeq \frac{1}{2}\vec{A}(\vec{x}) \cdot \nabla(F(\vec{x}) + M(\vec{x})) \tag{2}$$

- F, M, A discretised as pixels or nodes
- Solve non-linear problem in many linear "steps"

Optical Flow Image Registration

Optical Flow Method

- Discrete problem is expressed in matrix form

$$\vec{F} - \vec{M} = \mathbf{T}\vec{A} = \mathbf{G}\boldsymbol{\Phi}\vec{A} \tag{3}$$

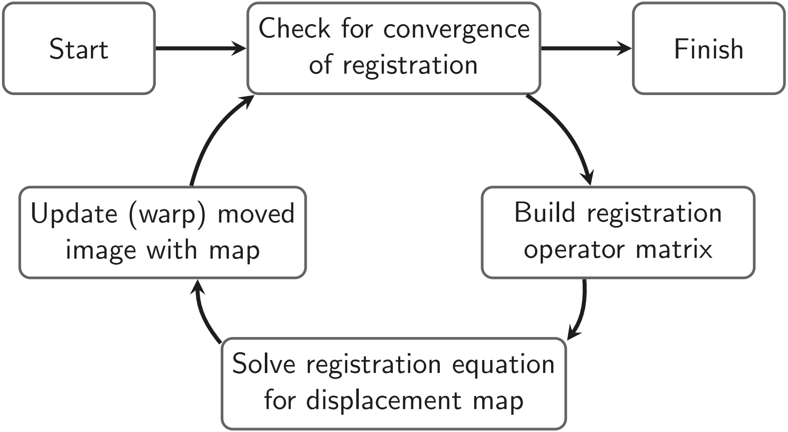
- Fewer nodes in $\vec{A}$ than pixels in $\vec{F}$ or $\vec{M}$, gradient matrix $\mathbf{G}$ is square but $\boldsymbol{\Phi}$ is non-square interpolation matrix
- Multiply both sides by $\mathbf{T}^t$ to gain final registration equation

$$\mathbf{T}^t(\vec{F} - \vec{M}) = \mathbf{T}^t\mathbf{T}\vec{A} \tag{4}$$

- System is underdetermined → least squares problem
- Use Tikhonov regularisation with Laplacian matrix → constrains to the smoothest available solution

Optical Flow Image Registration

The pFIRE Algorithm



Start

Check for convergence of registration

Finish

Build registration operator matrix

Solve registration equation for displacement map

Update (warp) moved image with map

Why are we writing a new code?

Sheffield Image Registration Toolkit (ShIRT)

- Currently heavily used at Sheffield
- Efficient registration of smaller images (2D and 3D)
- Written in early 2000's
- Serial execution - no parallelism

Why are we writing a new code?

Sheffield Image Registration Toolkit (ShIRT)

- Currently heavily used at Sheffield
- Efficient registration of smaller images (2D and 3D)
- Written in early 2000's
- Serial execution - no parallelism

The need for a new code

- Want to register 3D synchrotron images ($> 100\ \mathrm{GB}$)
- Too big for a single machine → too big for ShIRT
- Need distributed memory parallel code

pFIRE

pFIRE — Parallel Framework for Image Registration

- Parallel implementation of the ShIRT algorithm
- Open source license
- Modern C++ with MPI Parallelism
- PETSc for parallel linear algebra

pFIRE

pFIRE — Parallel Framework for Image Registration

- Parallel implementation of the ShIRT algorithm
- Open source license
- Modern C++ with MPI Parallelism
- PETSc for parallel linear algebra

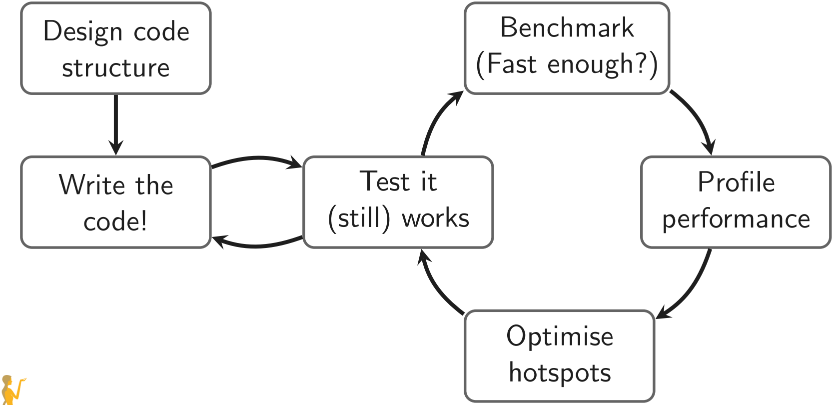
Design Goals

- Drop-in replacement for ShIRT
- Scalable from laptop to HPC
- Modular, extensible application design
- Compatible with wide range of image formats

The Path to Parallelisation

(Ideal) development journey

Writing the code is only part of the picture



Design code structure

Write the code!

Test it (still) works

Benchmark (Fast enough?)

Profile performance

Optimise hotspots

Choosing the Appropriate Parallel Paradigm

Shared Memory - Multiple cores on one node

- Code fits in a single node's memory, but needs to be faster
- Parallel algorithms for "bottlenecks" in the code
- Can often be added to existing serial code

Distributed Memory - Multiple cores on multiple nodes

- Need multiple nodes to fit problem in memory
- Pass data between nodes as needed
- Parallelise data structures and algorithms
- Serial code usually needs a complete rewrite

Parallel Frameworks — Why PETSc?

Parallel codes need lots of housekeeping

- Domain decomposition
- Halo cell communication
- Environment setup and teardown
- Parallel algorithm implementation

Parallel Frameworks — Why PETSc?

Parallel codes need lots of housekeeping

- Domain decomposition
- Halo cell communication
- Environment setup and teardown
- Parallel algorithm implementation

PETSc does all this already

- MPI environment management
- Domain decomposition of vectors/matrices
- Parallel linear algebra routines
- Well tested and widely used

Testing the Code

Are we getting the right answer?

Check everything is correct as often as possible

- Check as many scenarios as we can
- End-to-end as well as individual components
- Automate tests so we actually run them

Testing the Code

Are we getting the right answer?

Check everything is correct as often as possible

- Check as many scenarios as we can
- End-to-end as well as individual components
- Automate tests so we actually run them

What is the right answer anyway?

Non-trivial problem when testing numerical codes

- Compare with reference code (old version, serial version)
- Test with analytic solutions
- Compare with competitor codes
- Visualise the results(!)

Are we being efficient?

Potential Parallel Performance Issues

- Load balance
- Communication Efficiency
- Memory Efficiency
- Computational Efficiency

Are we being efficient?

pFIRE Parallel Performance Issues

- Load balance
- Communication Efficiency
- Memory Efficiency
- Computational Efficiency

Are we being efficient?

pFIRE Parallel Performance Issues

- Load balance
- Communication Efficiency
- Memory Efficiency
- Computational Efficiency

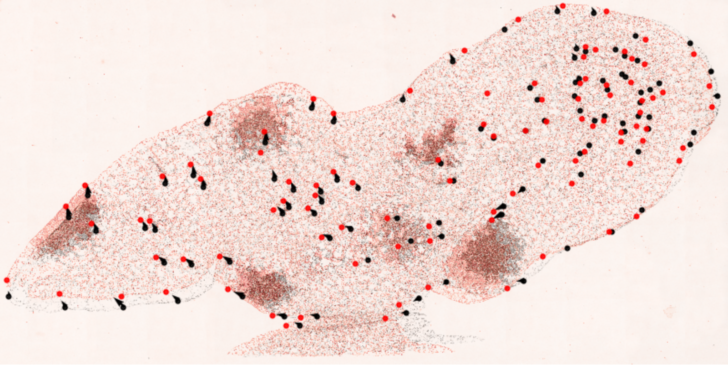
Image warping intrinsically unbalanced

- Elastic registration has arbitrary displacement maps
- Any pixel might be sourced from any location
- Communication pattern unknown ahead of time
- Load balance of warper different to solver

Distributed Memory Image Registration

Working 2D image registration

- Workshop at CompBioMed Winter School 2019
- Online tutorial available: https://insigneo.github.io/pFIRE
- Laptop and HPC registration of small and large images



Memory Usage

Matrix methods are memory inefficient

- Primary memory costs are matrix $\mathbf{T}$ and warping operator
- Sparse matrices of size $n \times m$ (for n pixels and m map nodes)
- 8 entries per pixel in 3D
- Requires $16\times$ the memory that the image does

Matrix Free Methods

- Trade algorithmic complexity for memory efficiency
- Directly constuct the (smaller) matrix $\mathbf{T}^t\mathbf{T}$
- Calculate entries in $\mathbf{T}$ as needed
- Need careful algorithm design for efficient communication
- Use similar approach for image warping

Development Roadmap

Ongoing Development

- Matrix free operator assembly
- Enhanced image format support (stacked .tiffs, dicom)
- Ansys/abaqus mesh output support

Future Plans

- Result visualisation
- Alternative interpolators
- Rigid pre-registration support

Performance Optimisation and Productivity

A Centre of Excellence in HPC

Contact Us:

- 🌎 https://pop-coe.eu
- ✉ pop@bsc.es
- @POP_HPC
- ▶ POP_HPC

The POP project has received funding from the EU Horizon 2020 research and innovation programme under grant agreement Nos. 676553 and 824080.