The Dissertation Committee for Naveen Himthani

certifies that this is the approved version of the following dissertation:

High performance algorithms for medical image registration with applications in neuroradiology

Committee:

George Biros, Supervisor

Omar Ghattas

Keshav Pingali

Thomas Yankeelov

Andreas Mang

High performance algorithms for medical image registration with applications in neuroradiology

by

Naveen Himthani

Dissertation

Presented to the Faculty of the Graduate School
of the University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

The University of Texas at Austin
May 2022

Acknowledgements

During the course of my PhD, I have been extremely fortunate in having exceptional mentors, colleagues, family and friends by my side. They have helped me grow as an individual both professionally and personally. I would love to express my sincere gratitude for their invaluable support and guidance.

Foremost, I would like to express my deepest and profound gratitude to my family. They have always been suportive of me and all my endeavours. The amout of sacrifices they have made to make who I am today cannot be expressed in written words. I am forever indebted to my parents for their love, encouragement, and unwavering support.

I am extremely grateful to my doctoral advisor Professor George Biros. This dissertation would not have been possible without his expert guidance. George has spent countless hours in discussions with me. His expertise, experience and insight have been invaluable towards the completion of my doctoral thesis. He has always been very patient with me during the times I got stuck on any problem. I have learned a tremendous amount and grown as a researcher under his mentorship.

I am deeply grateful to all my collaborators: Shashank Subramanian, Malte Brunn, Klaudius Scheufele, Amir Gholami, Professor Andreas Mang, Professor Miriam Mehl. The ideas in this dissertation build upon the doctoral work of Amir Gholami, a former student of Professor Biros. Although I interacted with Amir only for a short while, the discussions I had with him have been very helpful. I thank him for his guidance and friendship. Malte has been a close collaborator on all the software elements and methods presented in this dissertation.

I am thankful for our discussions and collective efforts to make this a productive endeavour. Shashank has been a very dear friend and colleague during the entire duration of my PhD. I have spent a lot of time discussing new ideas and coming up with new projects with him. Many thanks to him for those intellectual and invaluable conversations.

I would like to thank my thesis committee members—Professor Keshav Pingali, Professor Thomas Yankeelov, Professor Omar Ghattas, and Professor Andreas Mang. I am grateful for their time and effort spent in reviewing this thesis. It is a great honor to have their endorsement on my PhD. I am extremely grateful to Professor Andreas Mang who has always been very keen to discuss any issues I have faced during my thesis work. I have collaborated with him for most of the work in this thesis. He has always provided constructive feedbacks and valuable comments.

Special thanks to the faculty, staff, and fellow students of the Oden Institute, and especially the PADAS lab (Shashank Subramnian, Siddhant Wahal, Gokberk Kabacaoglu, Dhwanit Agarwal, James Levitt and Sameer Tharakan), for creating an enjoyable and rewarding working climate. I would like to acknowledge Stephanie Rodriguez, Jeff Early, Lauren Constant, Melissa Manifold and Donna Taylor for their administrative support. A very special thanks to my dear friend Sneha Goenka, who has always been there to talk me through any difficult time I have faced during the PhD.

High performance algorithms for medical image registration with applications in neuroradiology

by

Naveen Himthani, Ph.D.

The University of Texas at Austin, 2022

Supervisor: George Biros

This dissertation concerns the design, analysis and High Performance Computing (HPC) implementation of fast algorithms for large deformation diffeomorphic registration and its application in quantifying abnormal anatomical deformations in Magnetic Resonance Image (MRI) scans of brain tumor patients. Image registration finds point correspondences between two images by solving an optimization problem. It is a fundamental and computationally expensive operation that finds applications in computer vision and medical image analysis. Diffeomorphic registration is a non-convex and nonlinear inverse problem and, as a result, presents significant numerical and computational challenges. Designing and implementing efficient and accurate numerical schemes on modern computer architectures is the key to accelerating and sometimes even enabling the development of image analysis workflows. In this dissertation, we contribute on several aspects of diffeomorphic registration: *(i)* a novel preconditioner that improves performance and scalability, *(ii)* algorithms and their scalable implementation on heterogeneous compute architectures, and *(ii)* applications in neuroradiology. Our work on diffeomorphic image registration is based on `CLAIRE` – a formulation, algorithmic framework and software developed at the University of Texas at Austin. As a first highlight of our contributions, we introduced a novel two-level Hessian

preconditioner that results in an improvement of 2.5× in CLAIRE's performance. As a second highlight, our optimized HPC implementation yields orders of magnitude speedup as CLAIRE now supports GPU architectures and distributed memory parallelism via GPU-aware message passing interface (MPI). CLAIRE can register clinical grade brain MRI scans of size 256^3 in under 5 seconds on a single NVIDIA V100 GPU. For research grade high-resolution volumetric images, e.g., mouse brain CLARITY images of size $2816 \times 3016 \times 1162$, CLAIRE takes under 30 minutes using 256 NVIDIA V100 GPUs on the Texas Advanced Computing Center's (TACC) Longhorn supercomputer. To the best of our knowledge, CLAIRE is the most scalable image registration algorithm and software. CLAIRE has been open-sourced under the GNU v3 license and available on Github at https://github.com/andreasmang/claire.

Our target clinical application concerns the utilization of image registration to characterize the mass effect in MRI scans of patients with glioblastoma, a fatal brain cancer. Mass effect is the mechanical deformation in surrounding healthy tissue caused by the growing tumor. The location and degree of mass effect could aid in the differential diagnosis and treatment planning. Towards this end, we introduce an algorithm which integrates CLAIRE, statistical analysis for abnormality detection and machine learning to quantify and localize mass effect. Given a patient's brain tumor scan, we generate a clinical summary with *(i)* an estimate of the degree of mass effect along with a severity label – mild, moderate or severe with up to 62% accuracy, *(ii)* a heatmap of mass effect for the brain scan and, *(iii)* a list of specific anatomical regions, e.g. frontal lobe, which are statistically likely to possess significant mass effect.

Table of Contents

Acknowledgements **4**

List of Tables **11**

List of Figures **13**

Chapter 1. Introduction **15**

1.1 Motivation 15

1.1.1 Contributions 20

1.1.2 Outline 23

1.2 CSEM area contributions 23

Chapter 2. Diffeomorphic Image Registration **26**

2.1 Methods 28

2.1.1 Formulation 28

2.1.2 Discretization and Numerical Algorithms 30

2.2 Chapter conclusions 33

Chapter 3. Single GPU image registration **34**

3.1 Computational Kernels 39

3.2 Kernel Performance Analysis 45

3.2.1 Cubic Interpolation Kernel 45

3.2.2 Finite Difference Kernel 50

3.3 Image Registration Results 52

3.3.1 Data and Setup 53

3.3.2 Results 55

3.4 Chapter conclusions 60

Chapter 4. Multi-node multi-GPU image registration and novel preconditioning . . . 62
4.1 Discretization and Numerical Algorithms . . . 65
4.2 Computational Kernels . . . 68
4.2.1 Interpolation . . . 71
4.2.2 Finite Differences . . . 73
4.2.3 FFT . . . 74
4.3 Results . . . 77
4.3.1 Preconditioning . . . 77
4.3.2 Registration Performance . . . 78
4.3.3 Strong and Weak Scaling Results . . . 84
4.4 Chapter conclusions . . . 85

Chapter 5. Scalable image registration applications . . . 89
5.1 Methods . . . 91
5.1.1 Key Solver Parameters . . . 91
5.1.2 Parameter Search Scheme . . . 94
5.2 Results . . . 95
5.2.1 Measures of Performance . . . 95
5.2.2 Parameter Search Scheme . . . 97
5.2.3 Experiment 1A: High Resolution Synthetic Data Registration . . . 101
5.2.4 Experiment 1B: High Resolution Real Data Registrations . . . 111
5.2.5 Experiment 2: Registration of Mouse Brain CLARITY Images . . . 117
5.3 Chapter conclusions . . . 120

Chapter 6. Mass effect characterization using image registration . . . 123
6.1 Method . . . 128
6.1.1 Problem formulation . . . 129
6.1.2 Statistical mass effect model . . . 131
6.2 Datasets . . . 136
6.3 Detection problem . . . 144
6.3.1 Classification problem . . . 152
6.3.2 Regression problem . . . 157
6.4 Localization problem . . . 158

6.5 Results . . . 161
6.5.1 Classification problem . . . 162
6.5.2 Regression problem . . . 163
6.5.3 Localization problem . . . 165
6.5.4 Clinical summary . . . 168
6.6 Limitations . . . 169
6.7 Chapter conclusions . . . 172

Chapter 7. Conclusions and future work . . . 174
7.1 Conclusions . . . 174
7.2 Future work . . . 177

List of Tables

2.1 Notation and main symbols 28

3.1 Computational complexity of the main building blocks of single GPU CLAIRE 41
3.2 Comparison of analytic and experimental arithmetic intensity for the IP kernels. 46
3.3 Performance of the overall semi-Lagrangian transport using different IP kernels 47
3.4 Runtime (in seconds) and error of different interpolation kernels on the NVIDIA Tesla V100 48
3.5 Runtime (in seconds) of first order differential operators using FFT and 8^{th} finite differences. 52
3.6 Variants of combinations of computational kernels 53
3.7 Registration performance for different GPU image registration solvers 57

4.1 Weak scaling study for the IP kernel 73
4.2 Scalability for our finite difference (FD) scheme for first order derivatives . . 74
4.3 MPI performance analysis for the proposed FFT kernel 76
4.4 Weak and strong scaling for the proposed 3D FFT kernel in slab decomposition 76
4.5 Results for the registration of different NIREP and CLARITY datasets . . . 81
4.6 Strong and weak scaling results for CLAIRE using synthetic data 86

5.1 Performance of the parameter search scheme implemented in CLAIRE 98
5.2 Registration performance of ANTs 99
5.3 Registration performance for CLAIRE for multi-resolution registration of spherical harmonic images - case 1 104
5.4 Registration performance for CLAIRE for multi-resolution registration of spherical harmonic images - case 2 105
5.5 Registration performance for CLAIRE for multi-resolution registration of human brain images: case 1 114
5.6 Registration performance for CLAIRE for multi-resolution registration of human brain images: case 2 115
5.7 Registration performance for CLAIRE for multi-resolution registration of human brain images: case 3 116

5.8 Registration performance for CLAIRE for the CLARITY imaging data 120

6.1 Notation and main symbols 128
6.2 Image datasets for mass effect analysis experiments 137
6.3 List of top feature sets for comparing statistical model and tumor model . . 160
6.4 Best mass effect classification models 161
6.5 Regression performance of top five models for tumor displacement $\|u^*\|_1$ prediction problem 163
6.6 Mass effect localization using top displacement features sets for synthetic dataset 167
6.7 Mass effect localization using top divergence of displacement features sets for synthetic dataset 168
6.8 Mass effect localization using top displacement features sets for clinical dataset 169
6.9 Mass effect localization using top divergence of displacement features sets for clinical dataset 172

List of Figures

1.1 3D diffeomorphic image registration problem for human neuroimaging data . 16
1.2 Mass effect visualization 17
1.3 3D image registration problem for murine CLARITY imaging data 19

3.1 Illustration of the computation of the characteristic in the semi-Lagrangian scheme 39
3.2 Accuracy of first order differential operators using FFT and 8^{th} finite difference 51
3.3 Runtime breakdown for the main kernels of single GPU `CLAIRE` 56
3.4 Results comparison for different GPU image registration solvers 58
3.5 Registration results for single GPU `CLAIRE` 58

4.1 Plot of PCG residual versus PCG iterations for the benchmark and proposed preconditioner 79
4.2 Visualization of the allocated runtime for different parts of the optimization solver for different preconditioners 82
4.3 Strong and weak scaling results for `CLAIRE` 83

5.1 Comparison of Dice scores for `CLAIRE` and `ANTs` 99
5.2 Exemplary regisration results using parameter search scheme implemented in `CLAIRE` 100
5.3 Visualization of registration results for multi-resolution registration of spherical harmonic images. 106
5.4 Dice score versus label volume fraction for the multi-resolution registration results corresponding to spherical harmonics 107
5.5 Dice versus resolution plots for the multi-resolution registration results corresponding to spherical harmonics 108
5.6 Illustration of registration results for the multi-resolution registration experiment on real brain images 113
5.7 Illustration of the registration performance for `CLAIRE` for the CLARITY mouse brain imaging data 119

6.1 Exemplar brain parcellation 131

6.2 Template image MUSE parcellation volume normalization 138
6.3 Automated Anatomical Labelling (AAL)) template image parcellation 138
6.4 Reconstructed tumor atlas 140
6.5 Exemplar visualization of synthetic tumors 142
6.6 Joint Gaussian kernel density of tumor volume fraction and tumor mass effect for real and synthetic tumor datasets 143
6.7 Best abnormality score metric for estimation of statistical mass effect 147
6.8 Scatter plots of aggregate abnormality score with tumor model displacements 148
6.9 Visualization of three brain tumor cases with mass effect ratings from experts 150
6.10 Comparison of expert mass effect ratings with tumor imaging features 151
6.11 Comparison of expert mass effect ratings with tumor model displacement features 151
6.12 Comparison of expert mass effect ratings with abnormality score features from statistical mass effect model 152
6.13 Nested cross-validation workflow diagram 156
6.14 Best model results for regression for predicting tumor displacements $\|u^*\|_1$. 164
6.15 Spearman rank correlation between abnormality scores and tumor displacement features for synthetic and clinical dataset 165
6.16 Spearman rank correlation between abnormality scores and tumor divergence of displacement features for synthetic and clinical dataset 166
6.17 Clinical report of mass effect - part 1 170
6.18 Clinical summary of mass effect - part 2 171

Chapter 1

Introduction

1.1 Motivation

Clinical significance

3D image registration or image alignment or image matching is the process of establishing spatial correspondences between different image acquisitions. It is a critical task in biomedical imaging applications [57, 113, 142]. Mathematically, the image registration problem is typically formulated as an inverse problem where the inputs are two (or more) images $m_0(\boldsymbol{x})$ (the *template or moving image*) and $m_1(\boldsymbol{x})$ (the *reference or fixed image*) of the same type of object, compactly supported on a domain $\Omega \subset \mathbb{R}^3$ (see Figure 1.1). The task of image registration is to compute a spatial transformation or mapping $\boldsymbol{y}(\boldsymbol{x})$ such that $m_0(\boldsymbol{y}(\boldsymbol{x})) \approx m_1(\boldsymbol{x})$ for all points $\boldsymbol{x} \in \Omega$ [113]. There are several applications of image registration in medical imaging:

- In imaging studies of patient populations across different cohorts and different image acquisition techniques.
- Analysis and study of morphological changes associated with the progression of diseases in time series of medical images or in a single time-snapshot medical image.

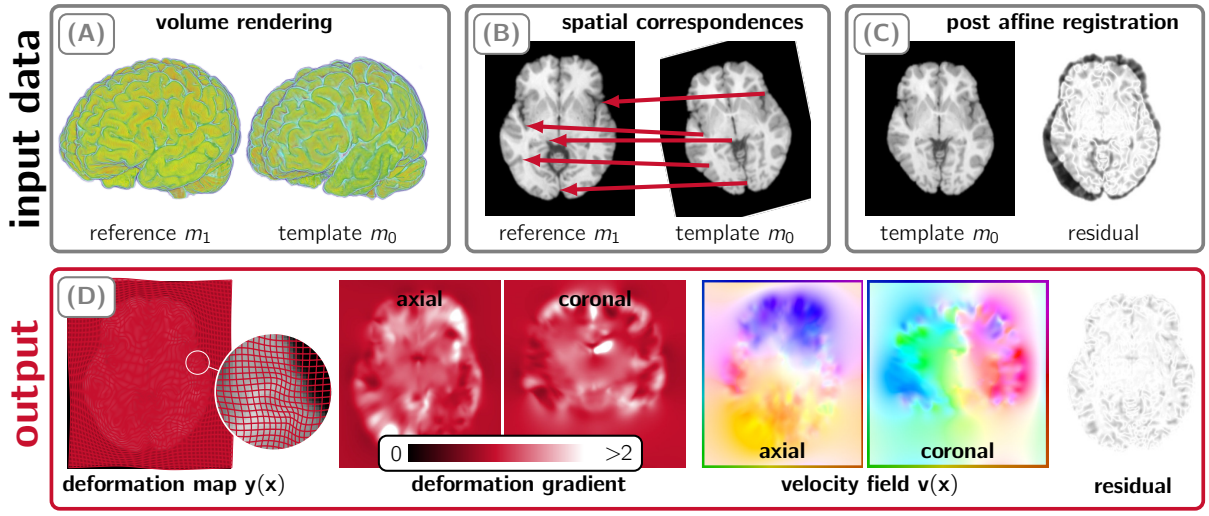


Figure 1.1: *3D diffeomorphic image registration problem for human neuroimaging data. We illustrate the input data and the registration problem in panel (A) and the results in panel (B) for a multi-subject registration problem (NIREP dataset [39]; reference image:* `na01`*; template image:* `na10`*). Panel (A) [from left to right]: Volume rendering of the input images, axial view of the reference image, axial view of the template image, and the residual of these views before registration (white: small residual; black: large residual). The image registration problem is to identify spatial correspondences that map points from one image (the template image) to points in another image (the reference image); see red arrows. Panel (B) [from left to right]: residual after registration, computed velocity field* $\boldsymbol{v}(\boldsymbol{x})$ *that parameterizes the deformation map* $\boldsymbol{y}(\boldsymbol{x})$ *(color denotes orientation), and an illustration of* $\boldsymbol{y}(\boldsymbol{x})$*. Qualitatively, the computed map is a smooth diffeomorphism (confirmed numerically).*

- Mapping structures in diseased tissue images to a reference atlas image for identification of precise diseased tissue boundaries for the purpose of surgery or radiation therapy.

- For semantic segmentation of medical images where the task is to assign a specific label to each voxel in the image, for example, gray matter, white matter, cerebrospinal fluid and lateral ventricles.

In this thesis, we explore a clinical application of image registration in characterizing the deformation abnormalities in MRIs, especially those of patients with Glioblastoma (GBM). Glioblastoma is the most common primary brain tumor in adults and is characterized with a highly proliferative and aggressive invasiveness in the brain [48]. GBMs have a median survival rate of 15 months [144]. Patients diagnosed with GBM are often presented with significant mechanical deformation of healthy brain tissue surrounding the tumor. This

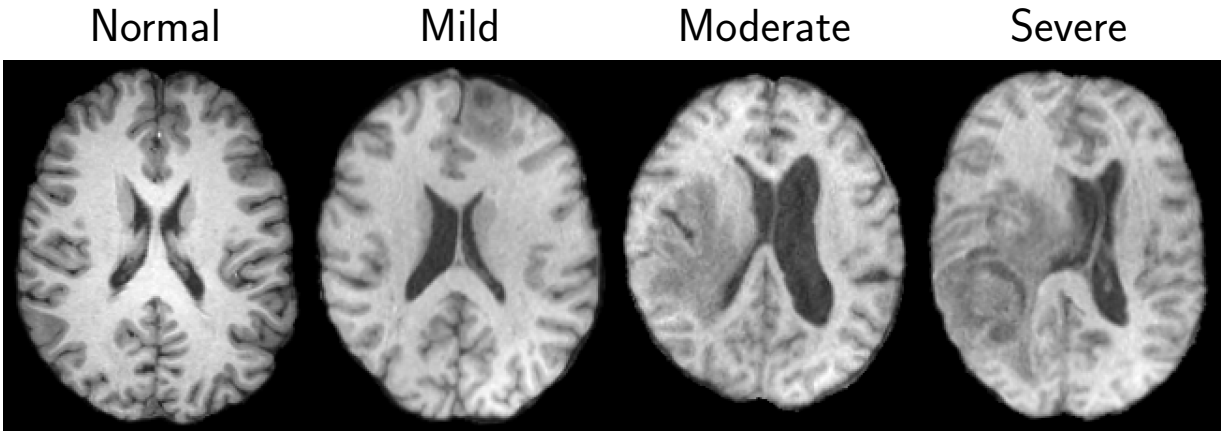


Figure 1.2: *T1-weighted Magnetic Resonance Imaging (MRI) scans (2D slices) of individuals with different levels of mass effect. The leftmost scan is of a healthy individual without any disease and second to fourth cases have a brain tumor called Glioblastoma. The severity levels of mass effect were assigned by expert radiologists. Significant compression of the lateral ventricles (dark structure in the center of the brain) and brain midline shift can be seen for moderate and severe cases respecitvely.*

mechanical deformation is clinically termed as *mass effect* (see Figure 1.2). For example, midline shift is typically found in patients with mass effect and this can be easily observed and treated immediately. However, subtle *mass effect* in the brain is often not visually evident and may cause neurological deficits impacting survival. Mass effect often results in alterations of consciousness, attention, and even awareness in a GBM patient. Radiomic features provide sub-pixel quantitative measurements to uncover such minute manifestation of the disease. These features are image driven and registration is typically required for further analysis. Image registration based methods are often used to construct models of the anatomy of a healthy brain and then characterize abnormalities such as mass effect based on these models.

Challenges

Methods for the registration of images can be classified according to the parameterization for $\boldsymbol{y}$ [113]. We consider maps $\boldsymbol{y}$ that are ***diffeomorphisms***, i.e., maps that are a differentiable bijection, and have a differentiable inverse. In this thesis, we consider formulations that

belong or are related to a class of methods referred to as *large-deformation diffeomorphic metric mapping* (**LDDMM**) [24,152,164]. These methods parameterize diffeomorphisms in terms of a smooth (time-dependent) velocity field. The associated mappings provide maximal flexibility [142] but are expensive to compute – the problem is infinite-dimensional, and upon discretization it becomes a nonlinear system with millions or even billions of unknowns. For example, registering two volumes of grid size 256^3 (a typical data size for clinical images) necessitates solving for approximately 50 M unknowns (three vector components per image grid point). This is further complicated by the fact that image registration is a highly nonlinear, ill-posed inverse problem [57], resulting in ill-conditioned inversion operators. As a result, image registration can take several minutes on multi-core CPUs with existing software packages [64,65,108]. As clinical workflows for multi-center population-studies that require thousands of registrations become increasingly more common, execution time of a single registration becomes more and more critical; reducing the runtime to seconds corresponds to a reduction of clinical study time from weeks to a few days. GPUs with their compute parallelism and high memory bandwith are an attractive choice to achieve this goal. However, despite the need for high computational throughput and the existence of several software packages for LDDMM [42,69,153,154,167], there is little work on high-performance GPU implementations, and even less work on multi-node multi-GPU implementations for large-scale applications. One such application is the registration of CLARITY images [41,88,93,94,151,157] of resolution in the order of $20\,\mathrm{K} \times 20\,\mathrm{K} \times 1\,\mathrm{K}$, which corresponds to a problem with about 1.2 trillion unknowns (see Figure 1.3).

Objectives

The objective of this thesis is to develop algorithms, implement corresponding HPC systems and study applications of diffeomorphic image registration. Specifically, we want to:

- *Develop* novel algorithms for diffeomorphic image registration in order to *(i)* improve

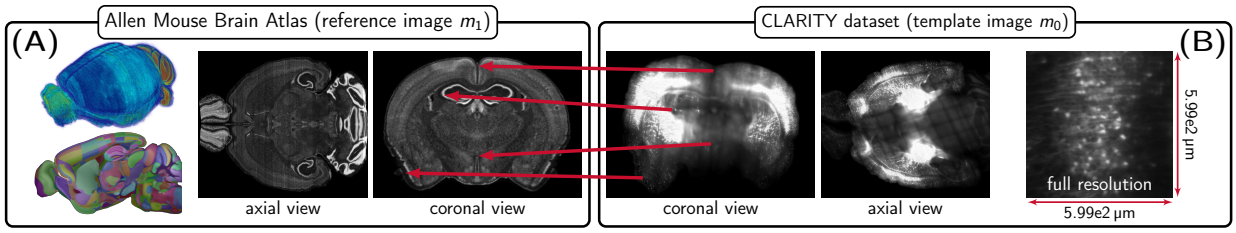


Figure 1.3: *3D image registration problem for murine CLARITY imaging data. We illustrate a multi-subject registration problem. In panel (A), we show the Allen Mouse Brain Atlas [85] with a grid size of* $800 \times 1140 \times 1320$*. We show a volume rendering (top left), annotations of anatomical regions (bottom left), an axial view (middle) and a coronal view (right). In panel (B), we show a coronal and an axial view (after affine registration to the atlas image), as well as a closeup of a sub-region in full resolution. This CLARITY volume (Control 189) has a resolution of* $585\,\text{nm} \times 585\,\text{nm} \times 5\,\mu\text{m}$ *with a grid size of* $20\,084 \times 24\,618 \times 1333$*. Once we have found the diffeomorphism, we can transfer the annotations of the anatomical regions identified in the atlas (see panel (A)) to the CLARITY dataset, and study anatomical sub-regions.*

solver convergence performance and scalability and, *(ii)* automate solver parameter selection to reduce user inputs and enable precise control over the mathematical properties of the deformation.

- *Implement* a high performance scalable image registration solver which can *(i)* efficiently utilize heterogeneous computing platforms and, *(ii)* allow processing of ultra–high resolution images in a reasonably short amount of time and, *(iii)* provide highly optimized kernels for important mathematical operations which can be used in other application domains.

- *Deploy* the high performance registration code to *(i)* demonstrate scalability for high resolution clinical images and show improvement in registration accuracy over low resolutions and, *(ii)* develop a statistical estimation framework for characterization of deformation abnormalities to perform important clinical tasks such as normal vs abnormal classification and identification of brain areas with aggressive deformations.

1.1.1 Contributions

The contributions and summary of the research in this thesis are as follows:

1. ***Single GPU image registration:*** In Chapter 3, we extend the open source diffeomorphic image registration framework, `CLAIRE`, to single-node single-GPU architectures. Previously `CLAIRE` [64,65,108], it has been shown that most of the computation time in `CLAIRE` is spent in scattered interpolation and Fast Fourier Transformations (FFT) operations. We developed novel algorithms for mapping these kernels to a single GPU and study several alternatives and optimizations of the scattered interpolation kernel. We use a mixed-precision approach to calculate derivatives and introduce finite difference kernels to replace FFT operations for first order derivatives. We demonstrate 20× improvement in computational performance compared to single node multi-core CPU implementation for similar accuracy. While over state-of-the art GPU implementations, we show 23× improvement in runtime with better accuracy. The results from this chapter are published in the following paper:

   - M. Brunn(∗), N. Himthani(∗), G. Biros, M. Mehl and A. Mang: *Fast GPU 3D diffeomorphic image registration*, Journal of Parallel and Distributed Computing, Vol 149, 149-162, 2021 (∗ – equal contribution)

2. ***Multi-node multi-GPU image registration:*** We extend the single GPU implementation of `CLAIRE` to a highly optimized multi-node multi-GPU setup. We minimize CPU-GPU communication and increase the computational throughput in the bottleneck kernels – scattered interpolation and FFT. We introduce a new conditioner for the Hessian system arising from the Gauss-Newton second order optimization method which significantly reduces the convergence time of `CLAIRE`. We present scaling results for upto 256 GPUs for all important kernels independently and the solver as a whole. We demonstrate the performance of our solver on synthetic as well as real datasets.

The results from this chapter are published in the following paper:

- M. Brunn(*), N. Himthani(*), G. Biros, M. Mehl and A. Mang: *Multi-node multi-GPU diffeomorphic image registration for large-scale imaging problems*, Proceedings of ACM/IEEE Supercomputing Conference (SC20), 2020 (* – equal contribution)

3. ***Large scale biomedical imaging applications:*** We demonstrate the scalability of `CLAIRE` for ultra-high resolution synthetic and clinical datasets. We demonstrate the need for performing image registration in high resolution to be able to morph fine structures in the image to improve performance metrics such as Dice coefficient which measures the voxel-wise overlap of two segmentations. `CLAIRE` has many solver parameters which can sometimes make it difficult for new users to tune and to get optimum registration performance. To this end, we introduce a new version of an automated regularization parameter search scheme which helps in making `CLAIRE` a closer to a black-box solver. We study the performance of `CLAIRE` on two pairs of ultra-high resolution murine CLARITY brain images with resolution $2816 \times 3016 \times 1162$ and demonstrate the effect of high resolution on registration accuracy. The results from this chapter are in the process of submission to the NeuroImage journal.

4. ***Abnormality characterization in medical imaging:*** We develop a statistical framework for characterization of mass effect observed in Glioblastoma MR images. We use a multi-template registration approach to build a distribution of normal anatomical structures using the registration deformation map between several healthy brain images. We solve a couple of clinically relevant problems – *(i)* mass effect detection and *(ii)* localization. In the detection problem, given a new patient image, we quantify the mass effect and assign it mass effect severity. For the localization problem, we identify brain regions that are statistically most likely to have high mass effect. We evaluate the framework on a large set of synthetic and real brain tumor images

and, demonstrate the effectiveness of using a statistical outlier method in quantifying abnormalities in a highly complex brain geometry.

5. ***Clinical applications and evaluation:*** We released `CLAIRE` as an open-source software (https://github.com/andreasmang/claire). We have used `CLAIRE` or its component kernels for several clinical applications[1]. These tasks are: *(i) semantic segmentation of MRI scans:* we used `CLAIRE` to create a segmentation of healthy brain tissues using a multi-atlas scheme. We used this segmentation as an auxiliary augmentation in a novel domain adaptation-based convolutional neural network to segment different tumorous tissue types in human brain, *(ii) Multi-atlas calibration for tumor growth models and mass effect quantification:* Our optimized interpolation GPU kernels were used to accelerate a 3D tumor forward growth model for calibration of tumor growth model parameters using a multi-atlas scheme. The results from these contributions have appeared in the following papers:

    - M. Brunn, N. Himthani, G. Biros, M. Mehl, A. Mang: CLAIRE: Constrained Large Deformation Diffeomorphic Image Registration on Parallel Computing Architectures. Journal of Open Source Software, 6(61), 3038 (2021)
    - A. Gholami, S. Subramanian, V. Shenoy, N. Himthani, X. Yue, S. Zhao, P. Jin, G. Biros, K. Keutzer: A novel domain adaptation framework for medical image segmentation, International Conference on Medical Image Computing and Computer-Assisted Intervention, 289-298 (2018)
    - S. Subramanian, K. Scheufele, N. Himthani, C. Davatzikos, G. Biros: Ensemble inversion for brain tumor growth models with mass effect, submitted to IEEE Transactions in Medical Imaging (2021)

[1] these contributions were not done as a primary author and are not included in detail in this thesis

1.1.2 Outline

In Chapter 3, we discuss the mathematical formulation, optimization algorithm and discretization methods and numerical schemes for `CLAIRE`. We discuss several kernel optimizations and parameters for executing scattered data interpolation on GPUs, quantify errors and runtime performance for each of those. We compare `CLAIRE` with previous CPU versions and other state-of-the-art GPU implementations for clinical datasets and discuss registration performance. In Chapter 4, we extend the single GPU implementation to distributed memory multi-GPU setup and perform weak and strong scaling analysis for the important kernels in `CLAIRE` using synthetic and real datasets. We discuss the performance of the novel preconditioner for the Gauss-Newton Hessian system and perform the convergence analysis. In Chapter 5, we analyze the accuracy of `CLAIRE` for ultra-high resolution and low resolution images and demonstrate improved registration accuracy for a variety of test datasets, both synthetic and real datasets. We demonstrate the scalability of `CLAIRE` for a couple of real high resolution murine brain images acquired using the CLARITY technique. In Chapter 6, we first discuss the abstract formulation for brain tumor mass effect characterization followed by a discussion on a concrete framework to construct healthy brain anatomy distribution using image registration. Subsequently, we discuss the pipeline for evaluating a patient GBM image which corresponds to detecting and localizing mass effect in the image. We conduct several experiments to study the accuracy of detection and localization for real and synthetic datasets. Finally in Chapter 7, we conclude the thesis, list limitations and propose future work.

1.2 CSEM area contributions

Next, we outline our contributions to the three different areas of the CSEM program:

Area A: Applicable Mathematics

1. We proposed a preconditioner based on the zero-velocity approximation of the reduced-space Hessian system (parameterized by the velocity) to accelerate the Krylov subspace solver for the solution of image registration problem using a PDE-constrained formulation.

Area B: Numerical Analysis and Scientific Computation

1. We developed single and multi-GPU implementations of `CLAIRE` [28, 29] image registration code using PETSc-CUDA toolkit and cuFFT for 3D fast Fourier transforms.

2. We developed highly optimized interpolation kernels for single and multi-GPU implementations of the semi-Lagrangian scheme for solving the linear advection equation. We extended the B-Spline polynomial based texture interpolation kernel to work with Lagrange polynomials, hence avoiding the need to compute B-Spline filter coefficients.

Area C: Mathematical Modeling and Applications

1. We studied the image registration performance of `CLAIRE` on ultra-high resolution images and demonstrated the need for conducting image registration at native high resolution in order to be able to better align fine image structures.

2. We introduced a multi-atlas image registration based statistical method to quantify and localize mass effect due to tumor growth from a single MRI scan. The method calculates an overall brain mass effect score along with scores for different functional regions in the brain. We validated the mass effect quantification method on a data set of synthetic tumor bearing brain MRIs with known mass effect displacement field. We tested several registration features to get the best correlation with ground truth. We qualitatively validated the mass effect quantification method on the BraTS challenge

2018 data set.

3. We implemented a multi-atlas healthy tissue brain semantic segmentation using `CLAIRE` for improving the performance of whole tumor segmentation in BraTS challenge 2018.

Chapter 2

Diffeomorphic Image Registration

This chapter introduces and describes the problem formulation, numerical methods and algorithms for diffeomorphic 3D image registration.

Image registration (also known as image alignment, warping, or matching) is an important task in medical image analysis [142]. It is used in computer aided diagnosis and clinical population studies. A comprehensive overview can be found in [43, 57, 79, 100, 113, 114, 142]. The process of image registration involves finding transformations that relate spatial information conveyed in one image to those in another [75]. We illustrate this in Figure 1.1. In mathematical terms, we are given two images $m_0(\boldsymbol{x})$ (the template image) and $m_1(\boldsymbol{x})$ (the reference image; here, $\boldsymbol{x} \in \Omega \subset \mathbb{R}^3$), we seek a spatial transformation $\boldsymbol{y} : \mathbb{R}^3 \to \mathbb{R}^3$, $\boldsymbol{x} \mapsto \boldsymbol{y}(\boldsymbol{x})$, such that the deformed template image $m_0(\boldsymbol{y}(\boldsymbol{x}))$ is similar to $m_1(\boldsymbol{x})$ [113]. Registration methods can be classified according to the parameterization for $\boldsymbol{y}$. In this dissertation, we consider methods that belong or are related to large-deformation diffeomorphic metric mapping (**LDDMM**) [24, 164]. In the original LDDMM formulation, the map $\boldsymbol{y}$ is parameterized by a time-dependent velocity field $\boldsymbol{v}$, modeled through the ordinary differential equation $\mathrm{d}_t\boldsymbol{y} = \boldsymbol{v}(\boldsymbol{y})$. In our formulation, we model the deformation through a hyperbolic transport equation; the map $\boldsymbol{y}$ does not appear explicitly (see §**??**). Such mappings parameterized through a velocity field $\boldsymbol{v}$ provide maximal flexibility [142]. LDDMM maps are expensive to compute since they are infinite-dimensional. Upon discretization, the number

of unknowns of our problem is still in the millions even if we restrict ourselves to stationary velocity fields $\boldsymbol{v}(\boldsymbol{x}) \in \mathbb{R}^3$: For each grid point of the discretized domain Ω, we seek a vector in $\mathbb{R}^3$. Consequently, solving a problem for two images of resolution 256^3 requires us to find the vector entries of $\boldsymbol{v}$ for 256^3 grid points, which results in $\approx$50 M unknowns (three unknowns per grid point). Furthermore, LDDMM registration is a highly non-linear and ill-conditioned inverse problem [57].

Related Work

We refer to [57, 113, 114, 142] for recent developments in image registration. Surveys of GPU accelerated solvers can be found in [55, 60, 139]. As mentioned above, this work extends `CLAIRE` [65, 103, 105, 108]. Popular (in clinical studies) software packages for deformable registration are `IRTK` [132], `elastix` [89], `NiftyReg` [112], and `FAIR` [114]. GPU implementations of (low-dimensional) parametric approaches are described in [56, 112, 137, 138]. Fast GPU implementations of (high-dimensional) nonparametric formulations available in `FAIR` are presented in [31, 90]. Unlike `CLAIRE`, these methods do not guarantee that the computed map $\boldsymbol{y}$ is a diffeomorphism. One possibility to safeguard against non-diffeomorphic maps $\boldsymbol{y}$ is by augmenting the formulation by hard and/or soft constraints on $\boldsymbol{y}$ [33], which introduces significant algorithmic complications. Another approach to enable diffeomorphic registration is to parametrize $\boldsymbol{y}$ via a smooth time-dependent velocity field $\boldsymbol{v}$ [50, 152]. This approach has been termed LDDMM [24]. The formulation in `CLAIRE` is closely related to LDDMM. A key difference is that LDDMM is based on non-stationary (time-dependent) $\boldsymbol{v}$ but `CLAIRE` uses stationary $\boldsymbol{v}$. Moreover, in the original formulation, the map $\boldsymbol{y}$ appears explicitly. In our work, we eliminate $\boldsymbol{y}$ and model the deformation using a transport equation (see §**??**; the work in [77, 102] and establishes a connection to LDDMM [24]).

Other approaches that use stationary $\boldsymbol{v}$ are described in [6,7,78,97,98,155]. There exists a large body of literature on LDDMM-type approaches that, in many cases, mostly focuses on

Table 2.1: *Notation and main symbols.*

Symbol	Description
Ω	spatial domain; $\Omega := [0, 2\pi)^3 \subset \mathbb{R}^3$ with boundary $\partial\Omega$
$\boldsymbol{x}$	spatial coordinate; $\boldsymbol{x} := (x_1, x_2, x_3)^\mathsf{T} \in \mathbb{R}^3$
t	(pseudo-)time variable; $t \in [0, 1]$
$m_1(\boldsymbol{x})$	reference image (fixed image)
$m_0(\boldsymbol{x})$	template image (moving image)
$\boldsymbol{v}(\boldsymbol{x})$	stationary velocity field
$\boldsymbol{y}(\boldsymbol{x})$	(diffeomorphic) deformation map
$m(\boldsymbol{x}, t)$	state variable (transported intensities of m_0)
$\lambda(\boldsymbol{x}, t)$	adjoint variable
$\mathcal{A}$	regularization operator
$\beta_v > 0$	regularization parameter for $\boldsymbol{v}$
$\beta_w > 0$	regularization parameter for $\nabla \cdot \boldsymbol{v}$
$\mathbf{F}$	deformation gradient
J	determinant of deformation gradient (Jacobian determinant)
n_t	number of time steps in PDE solver
CFL	Courant-Friedrichs-Lewy (number/condition)
FD	finite differences
FFT	Fast Fourier Transform
IP	scattered data interpolation
LDDMM	Large Deformation Diffeomorphic Metric Mapping
MPI	Message Passing Interface
PCG	Preconditioned Conjugate Gradient (method)

theoretical considerations [50,111,163–165]. There is much less work on the design of efficient solvers; examples are [7,9,13,15,24,124,155,167,168]. Popular software packages for LDDMM are diffeomorphic `Demons` [155], `ANTs` [12,13], `DARTEL` [7], `deformetrica` [25,26,51,58], and `PyCA` [127].

Outline

We summarize the overall formulation §2.1.1 and algorithms §2.1.2 in `CLAIRE`. All material in §2.1.1 and §2.1.2 is discussed in detail in the works [65,103,104,106,108].

2.1 Methods

2.1.1 Formulation

We summarize our notation in Table 2.1. `CLAIRE` uses an optimal control formulation. We parameterize the deformation map $\boldsymbol{y}(\boldsymbol{x})$ through a smooth, stationary velocity field $\boldsymbol{v}(\boldsymbol{x})$. We then define the optimization problem as follows: Given two images $m_0(\boldsymbol{x})$ (template

image; image to be deformed) and $m_1(\boldsymbol{x})$ (reference image), we seek a *stationary* velocity field $\boldsymbol{v}(\boldsymbol{x})$ by solving

$$\underset{\boldsymbol{v},m}{\text{minimize}} \quad \frac{1}{2}\int_\Omega (m(\boldsymbol{x},1) - m_1(\boldsymbol{x}))^2 \mathrm{d}\boldsymbol{x} + \frac{\beta_v}{2}\,\mathrm{reg}_v(\boldsymbol{v}) + \frac{\beta_w}{2}\,\mathrm{reg}_w(w) \tag{2.1a}$$

subject to

$$\partial_t m(\boldsymbol{x},t) + \boldsymbol{v}(\boldsymbol{x}) \cdot \nabla m(\boldsymbol{x},t) = 0 \qquad \text{in } \Omega \times (0,1], \tag{2.1b}$$

$$m(\boldsymbol{x},t) = m_0(\boldsymbol{x}) \qquad \text{in } \Omega \times \{0\}, \tag{2.1c}$$

$$\nabla \cdot \boldsymbol{v} - w = 0 \qquad \text{in } \Omega. \tag{2.1d}$$

on a three-dimensional rectangular domain $\Omega \subset \mathbb{R}^3$ with periodic boundary conditions on $\partial\Omega$. The first term in (2.1a) is a similarity measure for the proximity between the deformed template image $m(\boldsymbol{x}, t = 1)$ and the reference image $m_1(\boldsymbol{x})$. Without loss of generality, we consider a squared L^2-distance. The objective functional in (2.1a) additionally consists of two regularization models that act on the controls $\boldsymbol{v}$ and w with regularization parameters $\beta_v > 0$ and $\beta_w > 0$, respectively. The regularization operators are introduced to prescribe sufficient regularity requirements on $\boldsymbol{v}$ and its divergence $\nabla \cdot \boldsymbol{v}$ [103]. By adjusting their values, we can ensure that the deformation map $\boldsymbol{y}$ is a diffeomorphism [21, 24, 27, 38, 156, 164]. The default configuration of `CLAIRE` is an H^1-Sobolev-seminorm for $\boldsymbol{v}$ and H^1-Sobolev-norm for w. The transport equation (2.1c) describes the geometric transformation of the template image $m_0(\boldsymbol{x})$ by advecting the intensities forward in time. We use a reduced-space Gauss–Newton–Krylov method to solve (2.1). Details can be found in §2.1.2.

To solve (2.1), we apply the method of Lagrange multipliers to obtain the Lagrangian

functional

$$\mathcal{L}(\boldsymbol{\phi}) := \frac{1}{2}\int_\Omega (m(\boldsymbol{x},1) - m_1(\boldsymbol{x}))^2 \,\mathrm{d}\boldsymbol{x} + \frac{\beta_v}{2}\,\mathrm{reg}_v(\boldsymbol{v}) + \frac{\beta_w}{2}\,\mathrm{reg}_w(\nabla\cdot\boldsymbol{v}) + \int_0^1\int_\Omega \lambda(\boldsymbol{x},t)(\partial_t m + \boldsymbol{v}\cdot\nabla m)\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}t$$
$$+ \int_\Omega \lambda(\boldsymbol{x},0)(m(\boldsymbol{x},0) - m_0(\boldsymbol{x}))\,\mathrm{d}\boldsymbol{x} + \int_\Omega p(\boldsymbol{x})(\nabla\cdot v - w)\,\mathrm{d}\boldsymbol{x} \tag{2.2}$$

with state, adjoint, and control variables $(m, \lambda, p, \boldsymbol{v}) := \boldsymbol{\phi}$, respectively.

2.1.2 Discretization and Numerical Algorithms

Optimality Conditions & Reduced Space Approach

We derive first-order optimality conditions by taking variations of $\mathcal{L}$ with respect to the state variable m, the adjoint variables λ and p, and the control variable $\boldsymbol{v}$. This results in a set of coupled, hyperbolic-elliptic PDEs in *4D (space-time)*. At optimality, we require that the gradient of our problem vanishes. `CLAIRE` uses a ***reduced-space approach***, in which one iterates only on the reduced-space of $\boldsymbol{v}$. We require $\boldsymbol{g}(\boldsymbol{v}^\star) = \boldsymbol{0}$ for an admissible solution $\boldsymbol{v}^\star$, where

$$\boldsymbol{g}(\boldsymbol{v}) := \beta_v \mathcal{A}\boldsymbol{v}(\boldsymbol{x}) + \mathcal{K}\int_0^1 \lambda(\boldsymbol{x},t)\nabla m(\boldsymbol{x},t)\,\mathrm{d}t \tag{2.3}$$

is the so-called ***reduced gradient*** system (variation of $\mathcal{L}$ with respect to $\boldsymbol{v}$). The operator $\mathcal{A}$ corresponds to the first variation of the regularization model for $\boldsymbol{v}$ (i.e., reg_v in (2.1a)) and the operator $\mathcal{K}$ projects $\boldsymbol{v}$ onto the space of near-incompressible velocity fields (see [103] for details). To evaluate (2.3), we first solve the forward problem (2.1c) (variation of $\mathcal{L}$ with respect to λ) and then the *adjoint problem* (variation of $\mathcal{L}$ with respect to m) given by

$$-\partial_t\lambda(\boldsymbol{x},t) - \nabla\cdot\lambda(\boldsymbol{x},t)\boldsymbol{v}(\boldsymbol{x}) = 0 \quad \text{in } \Omega\times[0,1) \tag{2.4}$$

with final condition $\lambda(\boldsymbol{x},t) = m_1(\boldsymbol{x}) - m(\boldsymbol{x},t)$ in $\Omega\times\{1\}$ and periodic boundary conditions

on $\partial\Omega$.

Discretization

The forward and adjoint PDEs in the space-time interval $\Omega \times [0,1]$, $\Omega := [0,2\pi)^3 \subset \mathbb{R}^3$, with periodic boundary conditions on $\partial\Omega$, are discretized on a regular grid with $N = N_1 N_2 N_3$ grid points $\mathbf{x}_{ijk} \in \mathbb{R}^3$ in space and $n_t + 1$ grid points in time. A semi-Lagrangian scheme is used to solve the transport equations that appear in the optimality system [104, 106]. That is, the advection term is discretized in space and time based on backward trajectories of grid points. The total time derivative is evaluated by means of the difference of the current value of the transported variable at a grid point and the previous time step's value at the end point of a backward trajectory in time. An interpolation in space is needed at the end points of the backward trajectories that are, in general, off-grid points. The backward trajectories themselves are calculated by solving an ODE of the form $\partial_t \boldsymbol{y}(t) = \boldsymbol{v}(\boldsymbol{y}(t))$ in $[t, t+\delta t)$ with final condition $\boldsymbol{y}(t+\delta t) = \boldsymbol{x}$ using a second-order Runge–Kutta scheme.

Aside from integrating the PDEs in time, we need to apply gradient and divergence operators to evaluate $\boldsymbol{g}$ in (2.3) and to solve (2.4) for λ. We use finite difference (**FD**) operators for these differential operators [28, 29]. The reduced gradient (2.3) also involves the vector-Laplacian $\mathcal{A}$ and a Leray-like operator (see [103]). These operators are implemented in the spectral domain since (*i*) as we will see below, the solver requires the application of the inverse of $\mathcal{A}$ and (*ii*) the Leray projection $\mathcal{K}$ also involves the inverse of a Laplacian operator. In spectral methods, inverting higher order differential operators can be done at the cost of two FFTs and one a Hadamard product.

Gauss–Newton–Krylov Solver

`CLAIRE` uses a Gauss–Newton–Krylov method globalized with an Armijo line search to solve the non-linear problem $\boldsymbol{g}(\boldsymbol{v}) = \boldsymbol{0}$. The iterative scheme is given by

$$\mathbf{v}_{k+1} = \mathbf{v}_k + \alpha_k \tilde{\mathbf{v}}_k, \quad \mathbf{H}\tilde{\mathbf{v}}_k = -\mathbf{g}_k, \quad k = 0, 1, 2, \ldots, \tag{2.5}$$

where $\mathbf{H} \in \mathbb{R}^{3N,3N}$ is the discretized reduced-space Hessian operator, $\tilde{\mathbf{v}}_k \in \mathbb{R}^{3N}$ the search direction, $\mathbf{g}_k \in \mathbb{R}^{3N}$ a discrete version of the gradient in (2.3), $\alpha_k > 0$ a line search parameter, and $k \in \mathbb{N}$ the Gauss–Newton iteration index. We have to solve the linear system in (2.5) at each Gauss–Newton step. We do not form or assemble $\mathbf{H}$; we use a matrix-free preconditioned conjugate gradient (**PCG**) method. This only requires an expression for applying the Hessian matrix to a vector that we term ***Hessian matvec***. In continuous form, the Gauss–Newton approximation of this matvec is given by

$$\mathcal{H}\tilde{\boldsymbol{v}} = \beta_v \mathcal{A}\tilde{\boldsymbol{v}}(\boldsymbol{x}) + \mathcal{K} \int_0^1 \tilde{\lambda}(\boldsymbol{x}, t) \nabla m(\boldsymbol{x}, t) \, \mathrm{d}t. \tag{2.6}$$

The evaluation of the Hessian matvec in (2.6) requires several steps. For a candidate $\boldsymbol{v}$ we find the state variable m during the evaluation of (2.3); we keep m in memory. To find $\tilde{m}$ for a candidate $\tilde{\boldsymbol{v}}$, we solve

$$\partial_t \tilde{m}(\boldsymbol{x}, t) + \boldsymbol{v}(\boldsymbol{x}) \cdot \nabla \tilde{m}(\boldsymbol{x}, t) + \tilde{\boldsymbol{v}}(\boldsymbol{x}) \cdot \nabla m(\boldsymbol{x}, t) = 0 \quad \text{in } \Omega \times (0, 1] \tag{2.7}$$

with initial condition $\tilde{m}(\boldsymbol{x}, t) = 0$ in $\Omega \times \{0\}$ forward in time. To find $\tilde{\lambda}$ we solve

$$-\partial_t \tilde{\lambda}(\boldsymbol{x}, t) - \nabla \cdot \tilde{\lambda}(\boldsymbol{x}, t) \boldsymbol{v}(\boldsymbol{x}) = 0 \quad \text{in } \Omega \times [0, 1) \tag{2.8}$$

with final condition $\tilde{\lambda}(\boldsymbol{x}, t) = -\tilde{m}(\boldsymbol{x}, t)$ in $\Omega \times \{1\}$ backward in time.

Solving the linear system with $\mathbf{H}$ in (2.5) is the most expensive part of `CLAIRE`. A common choice in PDE-constrained optimization to alleviate the computational costs of solving for the search direction $\tilde{\boldsymbol{v}}$ is to precondition the reduced-space Hessian system using a spectral preconditioner $\mathrm{Inv}\mathcal{A} = (\beta_v \mathbf{A})^{-1}$ [5, 32, 107]. Application of $\mathrm{Inv}\mathcal{A}$ requires two

volumetric FFTs and a Hadamard product in the Fourier domain. However, Inv$\mathcal{A}$ does not take into account any information from the input template image m_0. In Chapter 4, we will introduce two novel preconditioners which are based on a zero-velocity approximation of $\mathbf{H}$ and result in faster convergence of the linear system in (2.5).

2.2 Chapter conclusions

In this chapter, we described the image registration problem using `CLAIRE` and formulated it is as a PDE-constrained optimization problem. We then obtained the associated optimality conditions using the reduced space approach and then described the second-order Newton-Krylov solver to obtain the solution to the optimization problem. In the next chapter, we discuss the implementation aspect of `CLAIRE` including the important computational kernels and their optimized high performance implementation on heterogenous architectures.

Chapter 3

Single GPU image registration

In the previous chapter, we formulated the image registration problem and described the methods and algorithms for solving it. In this chapter[2], we focus on the implementation aspects of image registration in `CLAIRE`. We discuss the important computational kernels, their high performance implementations and comparison of our optimized LDDMM registration with other registration software packages.

As discussed in Chapter 2, image registration can take a few minutes on multi-core high-end CPUs. As large clinical, cross-center, population-study workflows require thousands of registrations, reducing the compute time of a single registration to seconds translates to a reduction of clinical study time from weeks to a few days. GPUs with their inherent parallelism and low energy consumption are an attractive choice to achieve this goal. However, despite the need for high-throughput computational performance for registration, and the existence of several software libraries for LDDMM registration, there is little work on highly optimized GPU implementations.

[2]This chapter is based on the author's contribution in the following publication *M. Brunn(*), N. Himthani(*), G. Biros, M. Mehl and A. Mang: Fast GPU 3D diffeomorphic image registration, Journal of Parallel and Distributed Computing, Vol 149, 149-162, 2021 (* - equal contribution)*. The author of this dissertation contributed to model and software development for the interpolation and finite difference schemes, design and execution of numerical accuracy and GPU performance experiments for interpolation and finite difference kernels. The author also performed the overall image registration runs which included writing scripts and the analysis and interpretation of the results. The author also contributed towards manuscript writing and submission process.

Contributions

Based on the open source diffeomorphic image registration framework `CLAIRE` [102, 103, 105, 106, 108], we introduce a new, optimized, GPU implementation of LDDMM registration. In previous works [106] it has been demonstrated that the main computational kernels of `CLAIRE`–interpolation and differentiation—take up 90% of the runtime of the CPU version of `CLAIRE` [65, 106, 108]. To address this, we propose several modifications of the differentiation and interpolation kernels. More specifically, our contributions are:

- **Interpolation:** The first important computational kernel is scattered-data interpolation used for semi-Lagrangian advection. `CLAIRE` originally employed a Lagrange-basis cubic interpolation. We study several alternative methods on GPUs using a combination of pre-filtering, texture, and polynomial interpolation. We study their accuracy and performance using simple performance models and vendor performance profiling tools in §3.2.

- **Differentiation:** The second important computational kernel is computing derivatives (gradient and divergence) of 3D images (scalar fields). `CLAIRE` in its original version used FFTs to calculate all derivatives and their inverse in the reduced gradient and in the forward and adjoint PDE systems. We introduce a mixed-precision implementation using 8^{th} order finite-difference (**FD8**) kernels to replace FFT-based spectral derivatives. In particular, we replace all first order derivatives that appear in the partial differential equations (**PDE**) of our optimality systems. Note that FFTs are still retained for higher-order derivatives and their inverse. We discuss this in detail in §3.2.

- **Evaluation:** We evaluate the new algorithm on four exemplary magnetic resonance imaging **(MRI)** scans and for three different image resolutions. We compare the proposed method with the original `CLAIRE` in §3.3 as well as with the GPU packages

PyCA [127] and deformetrica [26,51]. The focus of this study is not to provide a complete benchmark for registration accuracy of CLAIRE. It is to establish a baseline to compare our improved solver to our prior work and the work of other groups. Overall, we will see that our method is over 20× faster than the original CPU-based CLAIRE and produces registration maps of similar quality. This speedup does not only reflect hardware differences but mostly algorithmic changes, some of which could also be implemented in a CPU version. Furthermore, reducing the accuracy of certain calculations to exploit hardware acceleration has no negative effects on the quality of the registration.

Limitations

The original implementation of CLAIRE was built to support the Message Passing Interface (**MPI**) for parallelism [65,106,108]. Our proposed adaption for GPUs has not been integrated with MPI yet. This will be subject to future work, in particular the integration of the high-speed GPU interface *NVLink* in a multi-node multi-GPU context. Thus, our solver does not scale to the image sizes that can be handled by CLAIRE. However, this is not an issue for clinical images since typical image sizes fit in a single GPU[3]. CLAIRE currently only supports periodic boundary conditions. Most medical imaging data is embedded into a zero background and therefore naturally periodic. If the data is not periodic, it can be mollified. If effects of boundary conditions on the computed solution are of concern, the data can be embedded into a larger domain.

[3]The GPU implementation is for a single GPU only and, therefore, limited by the memory available on the considered card (NVIDIA Tesla V100 in our case). The typical size for clinical images (magnetic resonance imaging) is approximately 256^3 and fits into memory of a single GPU for the current implementation.

Related Work

Before we discuss available GPU implementations in more detail, we note that the runtimes reported below are included to put our work into perspective. In general, numerous implementation aspects and settings such as complexity of the computational kernels, programming languages, employed optimization algorithms, used tolerances, computational accuracy, considered hardware, and even varying input data (complexity of the problem) will have an effect on the runtime. This is why we include our own experiments for software of other groups (see §3.3.2).

A GPU implementation of the diffeomorphic `Demons` algorithm is described in [42,69]. The runtime reported in [42] is in the order of 60 s on a Quadro FX 1400 for a dataset of size 128^3 [42] (2 s per iteration)[4]. A multi-GPU implementation of `DARTEL` is described in [153,154]. The work in [167] introduces `FLASH`, a fast CPU implementation for LDDMM. It is based on a band-limited spectral discretization targeting low resolution images to speed up the computations. By truncating the problem to 16 frequencies along each spatial dimension, the runtime is reduced from 45 s to under 2 s per iteration, resulting in an overall execution time of ≈200 s for 100 gradient descent steps; we use Newton–Krylov method, and our formulation of the problem is different. In [71,72], a (multi-)GPU implementation of the LDDMM approach described in [86] is presented; This work follows closely the fluid dynamics inspired framework for diffeomorphic registration proposed in [40]. This is different to our formulation. They use the greedy gradient descent strategy introduced in [40] to solve for the velocity field that parameterizes the deformation map (we use a Newton–Krylov algorithm). The optimization scheme is accelerated by separating the time and space dimensions of the problem. The resulting method is locally-in-time optimal and does not produce a deformation map that represents the shortest path connecting images on the space of diffeomorphisms (our method shares this property), one of the main virtues of the tra-

[4] All timings here are for single-precision calculations, which is typically used in practice. Our results for the proposed method are for single-precision as well.

ditional LDDMM formulation [24]. The computational acceleration is primarily achieved by using GPU texture hardware for trilinear interpolation for ODE integration (to compute the deformation map) and for computing spatial gradients. They employ a Successive-Over-Relaxation method (reported to be 2.5-3× faster than using FFTs) to solve the Navier-Stokes equation associated with their fluid dynamics inspired formulation of the diffeomorphic registration problem. The runtime of this solver is in the order of 12 s on a single NVIDIA Quadro FX5600 for a dataset of size 256^3 [72]. In [68], a GPU accelerated LDDMM implementation called `FastReg` is introduced. They use a gradient descent type optimization algorithm that is based on a generalization of the Nesterov accelerated momentum approach for finite-dimensional spaces [117] to infinite-dimensional manifolds [149]. The work in [68] aims to provide an alternative to computationally expensive convolution operations that appear in Sobolev gradient descent approaches typically used in LDDMM [24]. Sobolev gradient descent strategies are motivated by Riesz representation theorem; they use the inverse of the regularization operator to *precondition* the gradient descent direction [102]. We note that we provide efficient numerics to apply the inverse regularization operator based on spectral differentation, which exploits the convolution theorem and by that avoids invoking convolution operations (see also, e.g., [23, 129, 167, 169]). We also note that we have compared Sobolev gradient descent to our second order methods for optimization in our past work [102] and found that Newton–Krylov methods can be much more effective. The work in [68] report results for neuroimaging data with an average DICE of ≈0.67 (much smaller than our results) and a runtime of ≈35 s on a GeForce RTX 2080Ti. A GPU implementation of an LDDMM formulation for point cloud matching (not images) is described in [141]. The software package `deformetrica` [26] parametrizes $\boldsymbol{y}$ by a finite set of control points [52]. The gradient is computed via automatic differentiation [123]. The timings reported in [26] for the registration of an image of size $181 \times 217 \times 181$, executing 50 iterations, are 102 s and 202 s (Nvidia Quadro M4000) for two variants of the GPU implementation, respectively. The execution time for the CPU version of deformetrica is ≈10 h (Intel Xeon E5-1630). The

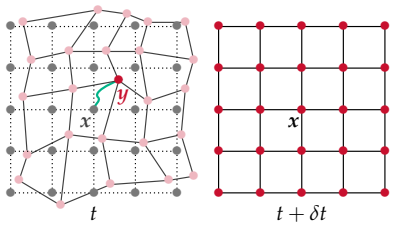


Figure 3.1: *Illustration of the computation of the characteristic in the semi-Lagrangian scheme. We start with a regular grid at time $t + \delta t$ and solve for the characteristic $\boldsymbol{y}$ at a given point $\boldsymbol{x}$ backward in time (green line in the graphic on the left). The deformed grid configuration is overlaid onto the initial regular grid at time t. Evaluating variables at off-grid locations requires interpolation. (Figure modified from [104].)*

runtime for the GPU variant of `PyCA` [127] reported in [162] for a $229 \times 193 \times 193$ neuroimaging dataset is 648 s (Nvidia TitanX (Pascal)). Many of these methods reduce the unknowns by using coarser resolutions, and use algorithms that produce a registration quality that is not as good as `CLAIRE` in terms of Jacobians.

Another approach that can speed up image registration is deep learning [17,91,161,162]. As an example, the training in [161] is performed with `PyCA`; it takes ≈72 h. After training, the reported runtime for the registration of $229 \times 193 \times 193$ images is 18.43 s on a single Nvidia TitanX (Pascal) [162], which is significantly slower than our method. Most importantly, it is unclear how deep learning performs on unseen clinical datasets.

Outline

In §3.1, we present the two main computational kernels in `CLAIRE`, the scattered-data interpolation and the approximation of first-order spatial derivatives. In §3.2, we analyze the performance of computational kernels on a single GPU and in §3.3, we present overall image registration results using single GPU and compare with other GPU software packages.

3.1 Computational Kernels

Let us first summarize the overall algorithm from Chapter 2. We use a Gauss–Newton–Krylov method (2.5) to solve the reduced gradient system $\boldsymbol{g}(\boldsymbol{v}) = \boldsymbol{0}$ for $\boldsymbol{v}$ [102]. The matrix-free

Gauss–Newton approximation of the Hessian involves solving forward and adjoint hyperbolic PDEs for the linearized (2.1c) and (2.4). If we use N_t time steps, each Hessian matvec requires $2N_t$ semi-Lagrangian steps, $2N_t$ gradient operators, and N_t divergence operators. In addition, the Hessian matvec needs $\mathbf{A}$ and its inverse, which are computed as spectral operators using FFTs. All these operators have $\mathcal{O}(N)$ complexity per time step, up to a logarithmic prefactor. The total number of Hessian matvecs is the sum of PCG iterations across Newton steps. Table 3.1 lists the number of FFTs and interpolations in more detail. The overall computational complexity can be estimated by $\mathcal{O}(N_{\text{GN}}(2N_{\text{PCG}}+2)(N_t N \log N))$ with N_{GN} as the number of outer iterations of the Gauss–Newton solver and N_{PCG} as the number of inner PCG iterations per Newton step to invert the Hessian system. The memory pressure is $\mathcal{O}((N_t+7)N_1N_2N_3)$ for the gradient and $\mathcal{O}((N_t+10)N_1N_2N_3)$ for the Hessian matvec, respectively.

The original `CLAIRE` implementation for CPUs used FFTs for gradients, divergences, $\mathbf{A}$ and $\mathbf{A}^{-1}$, and a highly optimized cubic Lagrange interpolation for the semi-Lagrangian method [108]. We transformed all computational kernels to GPU architectures, and most importantly, we introduced several algorithmic innovations to speed-up both derivatives and interpolations. First, we discuss several options for the interpolation. Second, we replace all gradient and divergence operators with high-order finite-difference (**FD**) operators.[5] Notice that we keep the spectral differentiation for high-order differential operators, since we need to evaluate their inverses in our solver (spectral preconditioner and Leray projection). Computing their inverses can be done efficiently in the spectral domain; for FD it would require linear solves. We show that, for the given image resolution and floating point accuracy, replacing the spectral methods with high-order FD discretizations allows us to maintain accuracy but significantly increase efficiency on GPUs. To the best of our knowledge, we are the first group to implement this type of mixed-precision code in a hardware and resolution adaptive way. Again, the spectral differentiation is kept for evaluating $\mathbf{A}$ (and its inverse

[5] We note that low-order (first and second order) FD (and finite volume) operators are a common choice in image registration [113, 114].

Table 3.1: *Computational complexity of the main building blocks of our solver in terms of the number of evaluations of the main computational kernels. We report the number of FFT operators (#FFTs, split into first order derivatives and other, i.e., higher order or inverse operators) and the number of scattered data interpolations (#IPs) that need to be performed for evaluating the objective functional, the gradient, and the Hessian matvec (Gauss–Newton approximation). (Notice, that the evaluation of the gradient requires forward and adjoint PDE solves and the Hessian matvec requires the evaluation of the incremental adjoint and state equations as subfunctions.) The first order operators are either implemented as FFT or finite differences (#FD). We report generic numbers; $d \in \{2, 3\}$ denotes the dimension of the ambient space ($d = 3$ in our case) and N_t is the number of time steps (we set $N_t = 4$). FFTs and IPs are executed for scalar fields of size $N_1 \times N_2 \times N_3$, where N_i denote the number of grid points in each spatial direction. We demonstrated in [65, 106, 108] (CPU implementation of CLAIRE) that about 90% of the runtime is spent on evaluating FFTs and the IP model. In the present work, we propose improvements to these kernels and their deployment in GPU architectures (see §??). To reduce the memory footprint of our solver, we evaluate parts of the gradient and Hessian matvec during the solution of the adjoint operators. The memory pressure is $\mathcal{O}((N_t + 7)N_1N_2N_3)$ for the gradient and $\mathcal{O}((N_t + 10)N_1N_2N_3)$ for the Hessian matvec, respectively.*

function	subfunction	symbol	#FFTs / #FD (1st order)	#FFTs (other)	#IPs
objective functional		—	—	d	$d + N_t$
	state equation (SE)	m	—	—	$d + N_t$
gradient		$\boldsymbol{g}$	$d(N_t + 2)$	d	$d + N_t + 1$
	adjoint equation (AE)	λ	d	—	$d + N_t + 1$
Hessian matvec		$\mathbf{H}\tilde{\boldsymbol{v}}$	$d(2N_t + 3)$	d	$d + (d + 2)N_t + 1$
	incremental SE	$\tilde{m}$	$d(N_t + 1)$	—	$d + (d + 1)N_t$
	incremental AE	$\tilde{\lambda}$	d	—	$N_t + 1$

to avoid an additional need for linear solvers); the GPU implementation of the proposed method employs a hybrid differentiation scheme that uses both FFTs and finite differences.

GPU Interpolation

The semi-Lagrangian scheme requires costly interpolation of velocities and scalar image fields along backward characteristics as shown in Figure 3.1; the data is sampled on a regular voxel grid (see §2.1.2) and the end-points of the backward characteristic in general correspond to off-grid locations. The CPU version of `CLAIRE` uses Lagrange-based cubic interpolation. GPUs provide two technologies that we exploit in our schemes: texture fetches and hardware support for trilinear interpolation (although not fully single-precision). In addition to these modifications, we also consider another change: switching from Lagrange cubic

to B-spline cubic interpolation. The generic formula for interpolating at an off-grid point $\boldsymbol{x} := (x_1, x_2, x_3) \in \mathbb{R}^3$ is given by

$$f(x_1, x_2, x_3) = \sum_{i,j,k=0}^{d} c_{ijk}\phi_i(x_1)\phi_j(x_2)\phi_k(x_3), \tag{3.1}$$

where $c_{ijk} \in \mathbb{R}$ are scalar coefficients associated with each grid point, $d \in \mathbb{N}$ is the polynomial order, and $\phi_i(x_1)$, $\phi_j(x_2)$, $\phi_k(x_3)$ are the basis functions. For Lagrange interpolation, the coefficients equal the grid values ($c_{ijk} = f_{ijk}$), and the ϕ's are the Lagrange polynomials. We use third order cubic ($d = 3$) but we also consider first-order trilinear interpolation ($d = 1$) since GPUs offer hardware acceleration for it. So, we need to evaluate a set of 64 (cubic) or 8 (linear) grid values f_{ijk}. However, there are other options. For example, we can use uniform B-splines for ϕ. In that case, the coefficients c_{ijk} are non-local—they depend on all grid values f_{ijk} unlike the Lagrange case [135]. Below we give the implementation details for the different schemes.

- **GPU-TXTLIN:** Here we use NVIDIA's libraries for trilinear interpolation [1,140]. It is efficiently performed using NVIDIA's hardware-accelerated texture units (using the `tex3D()` function). The texture units store the coefficients of the trilinear interpolation in 9-bit precision and return the result in single precision. We observed some effects in the registration quality in terms of smoothness of the deformation and the overall mismatch—especially in lower-resolutions or when the image has high frequency components. For this kernel, the asymptotic memory complexity to evaluate the scattered intepolation on $N_1N_2N_3$ points is $\mathcal{O}(12N_1N_2N_3)$ and the computational complexity is $\mathcal{O}(30N_1N_2N_3)$.

- **GPU-LAG:** This is our baseline since it represents a direct translation of the existing algorithm in `CLAIRE` to GPUs. The c_{ijk} values required to evaluate f are ordered lexicographically. This ordering results in non-coalesced memory accesses that reduce performance. To partially improve this, we use the texture function `tex3D()` as a

table lookup to access c_{ijk} and evaluate (3.1). We remark that we use the texture memory only for look ups and not for trilinear interpolation. For this kernel, the asymptotic memory complexity is $\mathcal{O}(68N_1N_2N_3)$ and the computational complexity is $\mathcal{O}(221N_1N_2N_3)$.

- **GPU-TXTLAG:** This is also a cubic Lagrange interpolation but now we use texture-based interpolation (as opposed to using textures as a table lookup), and thus the accuracy is reduced compared to GPU-LAG. However, in our experiments we don't observe any significant difference in the accuracy. The algorithm is based on the same principle as presented in [134]. Instead of doing *eight* weighted trilinear interpolations, we do *27 weighted trilinear* interpolations at off-grid points. The different number of trilinear interpolations arises due to differences in the Lagrange and B-spline polynomials. Nevertheless, because of hardware acceleration, GPU-TXTLAG significantly outperforms GPU-LAG. For this kernel, the asymptotic memory complexity is $\mathcal{O}(68N_1N_2N_3)$ and the computational complexity is $\mathcal{O}(482N_1N_2N_3)$.

- **GPU-TXTSPL:** The algorithm we use is exactly the one presented in [134]. The implementation is based on the open source library [133], with a major modification related to pre-filtering. We replaced the pre-filter in [133] with a finite convolution inspired by [35]. The pre-filtering to compute the coefficients c_{ijk} then becomes a 15-point axis aligned stencil operation on f_{ijk} and is implemented using the FD scheme used in the CUDA SDK example [76]. We also modified the code to support periodic boundary conditions. Then, following [134], we use *eight* weighted trilinear $(8 \times 8 f_{ijk})$ interpolations to compose the cubic B-spline interpolation. These interpolations require *eight* texture fetches at off-grid points. The resulting asymptotic memory complexity including the cost for the pre-filter kernel is $\mathcal{O}(68N_1N_2N_3 + 6N_1N_2N_3)$. The computational complexity is $\mathcal{O}(294N_1N_2N_3 + 90N_1N_2N_3)$

GPU Derivatives

The CPU `CLAIRE` uses FFTs to perform spatial differentiation [65]. Since our functions are periodic, all such operators are diagonal in the spectral domain. But in the proposed GPU implementation, we use an FD scheme that is more accurate (only for the given resolutions—not asymptotically) and faster than FFTs (see §3.2).

- **Finite Difference Scheme:** In particular, we use an 8^{th} order central difference scheme to evaluate first-order partial derivatives for the gradient and divergence operators. To evaluate the partial derivative at a regular grid point, we require nine axis-aligned function evaluations f_{ijk}. We load the grid values f_{ijk} from global memory to a shared memory tile and then evaluate the finite difference stencil. The derivative evaluations in the x_1, x_2 and x_3 spatial dimensions are independent of each other. Our implementation is the same as the CUDA SDK finite difference code [76] except that our implementation works for general grid sizes and supports periodic boundary conditions. The memory complexity to evaluate the volumetric gradient is $\mathcal{O}(6N_1N_2N_3)$ and the computational complexity is $\mathcal{O}(54N_1N_2N_3)$.

- **FFT (Spectral Differentiation):** `CLAIRE` uses `AccFFT` [63,66], which supports MPI for both CPU and GPUs. Here, we just use `cuFFT` [122] as we focus on a single GPU implementation. When we use FFTs for gradient and divergence operations we compute 3D FFTs. This avoids an explicit transpose operation on the data and misaligned memory accesses. Additionally, 3D FFTs reduce the number of memory accesses of the spectral data from global device memory. For the gradient all partial derivatives can be computed with only a single read and three write operation per element (instead of $3 + 3$ as for one-dimensional FFTs). Similarly, the divergence operator only needs a single store operation after summing all partial derivatives. Each volumetric FFT has a computational complexity of $\mathcal{O}(N_1N_2N_3 \log(N_1N_2N_3))$.

3.2 Kernel Performance Analysis

In this section, we evaluate the performance of interpolation (**IP**) and finite difference (**FD**) kernels. We calculate their arithmetic intensity (or simply "*intensity*") defined as the ratio of FLOP (total number of floating point operations) to MOP (total number of memory operations). We compare the kernel intensity to the device intensity. If the kernel intensity is less than the device intensity (peak floating point performance divided by peak device memory bandwidth), then the kernel is memory bound, otherwise it is compute bound. This is a simplification of the roofline model [160] since here we do not account for the cache hierarchy and latency effects. We also perform benchmark experiments to identify performance ceilings for our kernels.

As reference system for the CPU code, we used a two-socket Intel Skylake system. It is equipped with two Xeon Gold 5120 with a maximum frequency of 2.20 GHz and a maximum bandwidth of $107.30\,\mathrm{GB\,s^{-1}}$ with a TDP of 105 W per socket. We used a 32GB NVidia Tesla V100 with a memory bandwidth $B_{\max}$ of $900\,\mathrm{GB\,s^{-1}}$ and a TDP of 300 W for GPU experiments. The V100 is part of a two socket IBM Power9 system featuring NVLink as inter-device bus. Our implementation is in C++ and CUDA, and uses the PETSc library [20] for the Gauss–Newton–Krylov solvers.

3.2.1 Cubic Interpolation Kernel

Both cubic and linear interpolation (IP) are memory bound. The IP kernel has two main inputs: the target point coordinates ($3N$ floats), and the grid point scalar values (N floats). The output is the scalar field at the target points (N floats). Thus, the number of MOP is five floats (20 B) per target point. Formula (3.1) applies to both B-spline and Lagrange interpolation: the value at each target point depends on 64 regular grid values for cubic and 8 for trilinear interpolation, and these are not contiguous in memory.

Table 3.2: *Experiment 2: Comparison of analytic and experimental arithmetic "intensity" for the IP kernels. The analytic FLOP value is given in operations per target point assuming that each FPADD (add), FPMUL (multiply), FPSP (other ops like division) is one FLOP, and an FMA (multiply add) is two FLOP. The analytic MOP is given in MByte per target point assuming that each c_{ijk} value is loaded only once from the device memory. For the experimental data, we executed two interpolations with $N = 256^3$ on an NVIDIA Tesla V100 and measured total FLOP and MOP using the NVidia Visual Profiler. The MOP is the total number of bytes read from and written to the GPU device memory from/to the L2 cache. GPU-TXTSPL* corresponds to GPU-TXTSPL w/o prefilter. All FLOP values include the operations done internally by the texture unit. The intensity value is computed as FLOP per MOP. The device intensity is calculated as FLOPS per MOPS.*

	Analytic			Experimental			
Kernel	**FLOP**	**MOP**	**intensity**	**GFLOP**	**GMOP**	**intensity**	**bound by**
PRE-FILTER	22	8	2.75	0.37	0.14	2.64	memory
GPU-TXTLIN	30	20	1.50	0.10	0.34	0.30	memory
GPU-LAG	221	20	11.05	3.66	1.55	2.36	memory
GPU-TXTLAG	482	20	24.10	3.00	0.34	8.94	memory
GPU-TXTSPL*	294	20	14.70	2.97	0.27	10.86	memory
NVIDIA Tesla V100				14 000 GFLOPS	900GB/s	15.56	

Assuming an infinite amount of fast memory and ignoring latency cost, an analytic calculation of The number of FLOP divided by the number of MOP for each kernel gives the arithmetic intensity that shows that the kernels are memory bound. We overestimate the analytic intensity because we assume that all c_{ijk} values in (3.1) are loaded exactly once from device memory, which will typically not be the case, unless the memory accesses are fully coalesced. We evaluate performance experimentally using an effective bandwidth in GB/s defined as $\frac{(b_w+b_r)}{tx10^9}$, where b_r and b_w are the kernel loads/stores in bytes and t is the kernel total run time. We tuned the threadblock configuration to obtain optimal performance for the interpolation kernel. We used a one dimensional threadblock configuration with 256 threads for all our experiments. We perform two experiments for a localized and for a scattered target point distribution.

Experiment 1—Localized Target Points

As we discussed, each target point requires a set of c_{ijk} values. To isolate the memory issues related to streaming the target points, we conducted a run in which *all target points*

Table 3.3: *Performance of the overall semi-Lagrangian transport using different interpolation kernels on the V100. We report runtimes (in seconds) for applying an LDDMM transformation on a real 3D brain MR image using a semi-Lagrangian scheme. We deform the brain image using a velocity field (generated by registering two images from a clinical dataset) forward in time, followed by deforming the resulting image backward in time. We then compare the original image to the resulting image and compute the relative mismatch between the two. CPU-LAG, GPU-LAG and GPU-TXTLAG have a relative error of* 5.3E−2 *and* 2.4E−2 *for* $N = 64^3$ *and* 256^3*, respectively. GPU-TXTSPL is 2× more accurate, and has a relative error* 2.5E−2 *and* 1.7E−2*, respectively. GPU-TXTLIN has a relative error of* 1.2E−1 *and* 5.5E−2*, respectively. We also report wall-clock time for two advection solves, which incurs 14 interpolation kernel calls in total. The corresponding effective global memory bandwidth is also reported. The run time and bandwidth reported for GPU-TXTSPL include the overhead of the pre-filter operation. The CPU Lagrange (CPU-LAG) interpolation kernel is executed on a single intel-skylake node with 24 MPI tasks.*

	CPU-LAG	GPU-LAG		GPU-TXTLAG		GPU-TXTSPL (w/pre-filter)		GPU-TXTLIN	
N	**time**	**time**	**BW**	**time**	**BW**	**time**	**BW**	**time**	**BW**
64^3	16	1.5	50	6.4E−1	115	6.7E−1	240	1.3E−1	552
128^3	124	1.1E1	54	4.0	146	2.9	442	8.3E−1	705
256^3	1000	8.4E1	56	3.5E1	136	2.2E1	461	6.0	790

use the same 64 grid values for interpolation. This ensures full reuse of regular grid values among targets and provides an upper limit for the performance of the kernel. We run this test on the GPU-LAG and GPU-TXTSPL kernels. The performance of GPU-TXTLAG is somewhere in between and we omitted it in these runs. In this model the number of MOP changes. We only read and write $4N$ floats, and read 64 grid values for all points. Since all thread blocks need to read these values, the number of total number of MOP (in bytes) is equal to $4(4N + 64\#\text{threadblocks})$. We use this to estimate an upper performance bound. It is important to note here that the number of threadblocks only matters for a theoretical estimate without accounting for cache effects. In experimental runs, since all threadblocks are accessing the same set of 64 grid values, they will be cached. Hence, different threadblock configurations will not significantly affect the kernel performance, except for extremely small threadblocks where latency effects are dominant.

- **GPU-LAG Kernel (w/shared memory):** All CUDA thread-blocks load the same set of 64 c_{ijk} values from device memory and store them in the on-chip shared memory for reuse. All threads evaluate the result at their corresponding target points using

Table 3.4: *Experiment 2: Runtime (in seconds) and error of different interpolation kernels on the NVIDIA Tesla V100. We report results for synthetic data and real data. We report the relative interpolation error and the average runtime for one kernel call (in seconds). The relative interpolation error is given in the ℓ^2-norm with respect to an analytically known function. The evaluation is done on a grid with randomly perturbed grid points. We use sinusoidal test functions with different frequencies to analyze the interpolation error. The first test function f_1 is given by $(\sin^2(8x_1) + \sin^2(2x_2) + \sin^2(4x_3))/3$. The second and third test functions f_2 and f_3 are given by $\sum_{i=1}^{3} \sin^2(\omega_i * x_i)/3$ where $\omega_i = 1$ and $\omega_i = 16$, respectively, for f_2 and f_3.. For this synthetic setup, the measured runtime is averaged over 100 interpolations. The runtimes are independent of the interpolated function. We therefore only list runtimes of the interpolation of f_1. The faster variants GPU-TXTSPL and GPU-TXTLIN are used in the experiments reported in Table 3.7 in §3.3. For these experiments, we also report the per-call duration averaged over all Gauss–Newton iterations (right most column). The reported runtimes include all pre- and post-processing needed for the interpolation method.*

		synthetic data				real data
N	**method**	**error(f_1)**	**error(f_2)**	**error(f_3)**	**runtime**	**runtime**
64^3	GPU-LAG	9.9E−3	4.2E−6	2.5E−2	1.2E−4	—
	GPU-TXTLAG	9.8E−3	7.8E−5	2.5E−2	7.5E−5	—
	GPU-TXTSPL	2.2E−3	3.8E−5	3.4E−3	1.1E−4	1.1E−4
	GPU-TXTLIN	2.6E−2	5.9E−4	3.6E−2	3.8E−5	2.7E−5
128^3	GPU-LAG	7.2E−4	2.7E−7	1.4E−2	7.4E−4	—
	GPU-TXTLAG	7.3E−4	3.9E−5	1.4E−2	4.1E−4	—
	GPU-TXTSPL	1.1E−4	1.8E−5	3.1E−3	3.6E−4	3.3E−4
	GPU-TXTLIN	6.8E−3	1.5E−4	3.6E−2	1.3E−4	1.4E−4
256^3	GPU-LAG	4.7E−5	7.2E−8	1.0E−3	5.2E−3	—
	GPU-TXTLAG	8.7E−5	1.9E−5	1.1E−3	3.0E−3	—
	GPU-TXTSPL	5.0E−5	8.9E−6	1.9E−4	2.3E−3	2.1E−3
	GPU-TXTLIN	1.7E−3	4.0E−5	9.3E−3	8.4E−4	1.0E−3

the data available in shared memory and then apply (3.1). Using the MOP estimate from above (and the observed timings), we achieve an effective bandwidth of 570 GB/s ($63.3\% B_{\max}$).

- **GPU-TXTSPL Kernel:** To calculate the effective bandwidth, we assume that each of the 64 c_{ijk} are fetched by the texture exactly once from the device memory. Using this assumption (and the observed timings), the effective bandwidth for this method is 350 GB/s ($39\% B_{\max}$). Note that the reported bandwidth here does not account for the prefilter operation. Also note that, in reality, textures cannot take significant advantage of the fact that the target points have exactly the same regular grid dependencies. As a result, there are more memory dependencies (than our MOP estimate) and, thus, the observed performance drops—compared to the GPU-LAG kernel.

Experiment 2 —Scattered Target Points

We consider a real distribution (generated via random perturbation of grid points or actual trajectory backward tracking) of target points (and switch to the original 20 B/point MOP model). Here, unlike "Experiment 1", the implementation of GPU-LAG does not use shared memory to load target point dependencies. The implementation of GPU-LAG which uses shared memory to load target point dependencies for the scattered case is future work. However, the implementation of GPU-TXTSPL remains the same as in "Experiment 1". The analytic observation that the interpolation is memory bound result is confirmed by measurements with the NVIDIA Visual Profiler summarized in Table 3.2.

For a random distribution of target points, GPU-TXTSPL achieves an effective global memory bandwidth of 335 GB/s ($37.6\% B_{\max}$), which is nearly identical to "Experiment 1". Hence, GPU-TXTSPL is insensitive to target point dependencies. In contrast, GPU-LAGs performance drops by a factor of 10 to 56 GB/s because we are no longer making explicit use of shared memory to load and reuse the target point dependencies. Also note that,

once we coupled the GPU-TXTSPL to the overall semi-Lagrangian scheme in Table 3.3, the effective bandwidth increases to 461 GB/s, which is slightly over 50% relative to the peak bandwidth. Finally, in Table 3.4, we compare the accuracy and time of the four different methods. We consider functions of varying smoothness, with the highest frequency determined by the resolution limit on the coarsest grid (see Table 3.4). The differences in accuracy are somewhat significant only in lower resolutions. Note that we get different accuracy results for real brain MR images in Table 3.3. This is expected since the cubic spline interpolation of GPU-TXTSPL gives better interpolation accuracy than third-order Lagrange polynomials used in CPU-LAG or GPU-LAG in cases where the image resolution is not sufficiently high relative to the highest frequency in the image. For the synthetic low frequency image used in Table 3.4, Lagrange polynomials tend to perform better for higher image resolutions. Here, GPU-LAG gives more accurate results than GPU-TXTSPL for a 256^3 resolution. We compare our new GPU implementation to the original MPI based CPU version of CLAIRE [108]; the CPU version of CLAIRE does not support OpenMP.

As a byproduct, this analysis also addresses to some extent the following question: Would it make sense to reorder (say in Morton order) the target and grid points in order to achieve better locality (but possible sacrifice texture memory)? As we show, an ideal ordering would result in 570 GB/s; we observe about 460 GB/s for GPU-TXTSPL and conclude that our implementation is nearly optimal.

3.2.2 Finite Difference Kernel

In our implementation, each CUDA thread block evaluates the derivatives for a 2D tile of data. We refer to the points contained in this tile as *inner points*. To evaluate the derivatives at the edge of a tile, we load a set of neighboring points known as *halo points*. We load the set of inner points and halo points from device memory to a 2D shared memory tile, evaluate the derivatives, and store the result back to shared memory. The inner points of one thread-

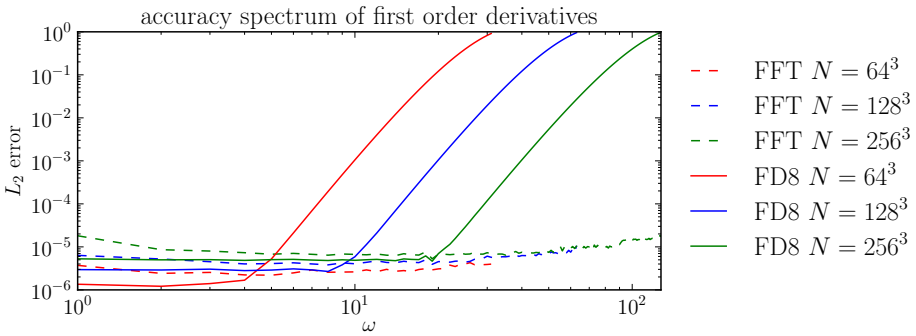


Figure 3.2: *Accuracy of first order differential operators (gradient and divergence) using FFT and 8^{th} order finite differences on a Nvidia Tesla V100 for different problem sizes. We report the L_2 error of our operators. The error is measured using the computed partial derivative in x_3-direction of the function $\sin(\omega x_3) + \cos(\omega x_3)$ compared to the analytical derivative. The error is plotted over the frequency up to the Nyquist frequency. Finite differences are more accurate for low frequency modes and have an increasing error for higher modes. By replacing FFTs with finite differences, we trade faster computation (due to a higher data locality and a reduced algorithmic complexity) against lower accuracy for high frequency modes.*

block are halo-points of the adjacent thread-block and are loaded twice. We quantify this experimentally. We first repeat the FLOP-MOP experiment for the FD kernel and observe that the kernel is memory bound.

We compare the bandwidth performance of our general kernel to the parent SDK example. The SDK code works only for a fixed grid size $N = 64^3$ and a 9-point stencil. CUDA SDK reports an effective bandwidth of 310 GB/s whereas our implementation achieves 212 GB/s. The reported bandwidth includes the cost of loading halo points. Both values are much smaller than B_{max} because the grid size is not large enough to hide latency. Unlike the SDK example, the CUDA threads on the boundary of the domain load halo points from global memory instead of shared memory. The observed performance drops due to the thread divergence caused by reading out-of-bound halo points. For large N, as we show later, this overhead is greatly reduced as a direct consequence of decreased latency caused by higher occupancy.

We perform a baseline memory copy, i.e., copy within the HBM2 device memory to put an absolute upper bound on the performance of our implementation. We load each element

Table 3.5: *Runtime (in seconds) of first order differential operators (gradient and divergence) using FFT and 8^{th} order finite differences (FD8) on a NVIDIA Tesla V100 for different problem sizes. We report the runtime in s per kernel call averaged over the whole registration run from experiments shown in §3.3 including all pre- and post-processing needed.*

N	Operator	FFT	FD8
64^3	grad	1.7E−4	3.6E−5
	div	1.7E−4	3.9E−5
128^3	grad	6.0E−4	1.4E−4
	div	5.7E−4	1.6E−4
256^3	grad	4.1E−3	9.4E−4
	div	3.8E−3	1.2E−3

of an array of size $N = 256^3$ from the global device memory and store it in another array. The peak performance we get for this copy routine is 780 GB/s. To quantify the halo points load overhead, we perform another experiment. Each thread-block loads its inner points and halo points into a 2D shared memory tile and copies only the inner points back to the output array. The effective bandwidth for this benchmark is 766 GB/s. The reported bandwidth includes the cost of loading halo points. We only lose 1.8% of the memory bandwidth in comparison to the baseline memory copy experiment. This indicates that the overhead due to loading of out-of-bound halo points gets smaller as the kernel occupancy increases. We verify our claims by profiling the kernels using the NVIDIA Visual Profiler. For the smaller grid size of 64^3, the kernel is bound by instruction and memory latency, for the larger grids (128^3 and 256^3) by memory bandwidth.

3.3 Image Registration Results

We evaluate the overall algorithm using four 3D MRI images. We study convergence, time-to-solution, and registration accuracy for different variants of the computational kernels of our new GPU implementation of `CLAIRE`. The purpose of this section is to show that (*i*) our new (mixed-precision) GPU implementation yields the same registration accuracy as our CPU implementation of `CLAIRE` [108] and (*ii*) to compare our method against GPU implementations of other groups. In the present work, we are interested in computational throughput; we will see for exemplary datasets that the obtained results are equivalent to those reported in [108]. We refer to [108] for a more detailed study of registration accuracy.

Table 3.6: *Variants of combinations of computational kernels and the respective tag used in this work. IP stands for interpolation and FD8 for finite difference operators of 8^{th} order.*

Tag	Variant
cpu-fft-cubic	FP32, CPU, FFT, cubic IP
gpu-fft-cubic	FP32, GPU, FFT, cubic IP
gpu-fd8-cubic	FP32, GPU, FD8, cubic IP
gpu-fd8-linear	FP32, GPU, FD8, trilinear IP

3.3.1 Data and Setup

Images

We report results for the `NIREP` data, a commonly used data set to evaluate the performance of deformable registration algorithms [39]. `NIREP` consists of 16 rigidly aligned T1-weighted MR scans (`na01`–`na16`) of different individuals. The original resolution is $256 \times 300 \times 256$. Each scan comes with a label map that identifies 32 gray matter regions [39]. We select four scans from this data set, `na01` as reference image and `na02`, `na03`, and `na10` as template images, respectively. The initial DICE coefficient (spatial overlap index) for the union of the gray matter regions of the template images versus the reference image is 0.55, 0.50 and 0.48, respectively. A perfect matching corresponds to a value of 1.00. Currently, we only support image sizes $N_1 N_2 N_3$ dividable by 256. We resample the data to grid sizes of 64^3, 128^3, 256^3, and 384^3, using linear and a nearest-neighbor interpolation models for the image data and the label maps, respectively.

Numerical & Floating Point Accuracy Parameters

Unless specified otherwise, we use the default solver parameters from [105, 108]. For regularization, we use the default of `CLAIRE`, H^1-div—an H^1-seminorm with an additional penalty for the divergence of the velocity. We execute the proposed solver with a parameter continuation scheme for the regularization parameter β_v. This scheme is describe in detail in [102]. In all runs, we use the target parameter $\beta_v = 5\text{E}{-4}$ selected based on experiments reported in [108]. We set the parameter for the penalty for the divergence of $\boldsymbol{v}$ to $\beta_w = 1\text{E}{-4}$.

- **Convergence Criteria:** As a stopping criterion for the optimizer we use a tolerance of 5E−2 for the relative reduction of the gradient (2.3) together with a maximal number of iterations of 50 (never reached). We use an inexact Newton method with a superlinear forcing sequence (see [47, 54] for details) and set the maximum number of iterations for the PCG method to 500 (never reached). We globalize our Gauss–Newton–Krylov method using an Armijo line search [120].

- **Interpolation:** We consider different IP methods to evaluate values of variables at off grid locations. In particular, we select either a linear or a cubic IP scheme. For cubic IP, we use GPU-TXTSPL as proposed in §3.1.

- **First Order Derivatives:** For the calculation of first order derivatives, we compare the FFT-based scheme and the 8^{th} order FD (**FD8**) scheme, as proposed in §3.1.

- **Floating Point Accuracy:** Our new implementation uses single precision (**FP32**). For validation, we compare against results achieved with the `CLAIRE` CPU implementation in single precision [108]. We summarize the settings in Table 3.6.

Performance Metrics

We report two groups of metrics: To assess computational performance, we report runtimes. To assess accuracy, we report (*i*) the relative mismatch $\|m(\bullet, 1) - m_1\|_2 / \|m_1 - m_0\|_2$ of the template image $m_0(\boldsymbol{x})$, the reference image $m_1(\boldsymbol{x})$, and the transformed template image $m(\boldsymbol{x}, 1)$ obtained by solving the forward problem (2.1c) as well as (*ii*) the DICE coefficient (overlap) between the union of the gray matter labels. The latter assesses how well anatomical structures identified by expert observers are aligned after registration. For a perfect matching, the value is 1.00[6]. To assess the quality of the computed deformation map, we

[6]The DICE coefficient is a metric that has been widely adopted by the registration community to assess registration accuracy. We provide a more detailed study in [108]. We note that DICE and mismatch values do not provide a complete picture about registration accuracy. Other metrics include the Haussdorff distance between the contours of label maps or landmark errors (an example for a database that considers landmarks to evaluate registration performance is DIRLAB; see `www.dir-lab.com`). We note that the focus of the

report min, mean and max values of the determinant of the deformation gradient. The mapping is locally non-diffeomorphic if the determinant changes sign or is zero. In general, if the determinant is either very small (but still positive) or very big, the mapping is of poor quality. In our case, the values are between 0.5 and 10, which indicates well-behaved deformation maps.

To assess the convergence of our solver, we report (*i*) the relative gradient norm $\|\boldsymbol{g}\|_{\text{rel}} := \|\boldsymbol{g}^\star\|_2/\|\boldsymbol{g}^0\|_2$, where $\boldsymbol{g}^\star$ is the gradient of the optimization problem after convergence and $\boldsymbol{g}^0$ is the gradient for a zero initial guess, (*ii*) the number of iterations for the Newton–Krylov solver, and (*iii*) the total number of Hessian matvecs.

3.3.2 Results

Next, we report results for our improved implementation of `CLAIRE`. We use the same experimental setup as for the kernel performance analysis in §3.2.

Performance Analysis of the Proposed Method

Purpose: We study performance for different combinations of the computational kernels.

Results: We report exemplary results in Table 3.7 for images of size 256^3 and 384^3. We note that we have conducted additional experiments for grid sizes of 64^3, 128^3, 256^3, and 384^3, respectively, for three different datasets. These experiments can be found in the preprint of this manuscript. The breakdown of the execution time with respect to the individual kernels is shown in Figure 3.3 (for `na02`). The maximum allocated memory on the GPU during the experiments was 0.60 GB, 1.30 GB, 6.10 GB, and 20.00 GB for image sizes of 64^3, 128^3, 256^3, and 384^3, respectively. The maximum allocated memory on the host CPU was below 2 GB for all GPU experiments and only used for management and IO purposes.

manuscript is on computational performance and not registration accuracy. The accuracy results included in this study serve as a baseline to compare our improved solver to our past work [108].

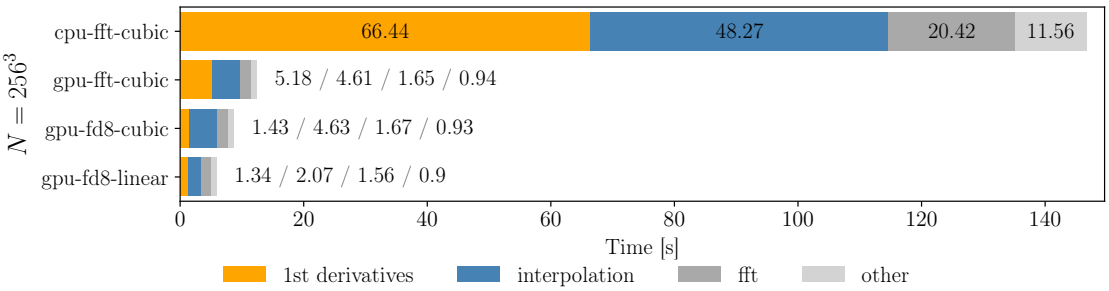


Figure 3.3: *Runtime breakdown for the main kernels of the proposed method for all implementations (cpu and gpu, first order derivatives via FFT or FD8, cubic or linear interpolation). The dark gray parts indicate the contribution of higher order operators in spectral space to the overall execution time of the solver. We consider the registration of* `na02` *to* `na01` *at a resolution* 256^3.

Observations: The critical result is that we can accurately solve 3D image registration problems for clinically relevant sizes (256^3) on a single GPU in less than 10 seconds (see Table 3.7) for the variant gpu-fd8-linear. The gpu-fd8-cubic approximation is almost as fast. Switching to lower accuracy regimes hardly changes the iteration counts, registration quality (mismatch and DICE), and number of Hessian matvecs. For all implementations, we reach the tolerance of 5E−2 for the relative reduction of the gradient.

All implementations produce well-behaved determinants of the deformation gradient. The highest DICE is achieved for `na02` (0.86). In our experiments we observed a speedup between the baseline method cpu-fft-cubic and gpu-fd8-linear of 8–11 for 64^3, 16–18 for 128^3, and 23–25 for 256^3.

For the considered test problems with image sizes 64^3, 128^3, and 256^3, the number of Gauss–Newton iterations increases only slightly with the resolution. The number of Hessian matvecs increases by up to a factor of two as we change resolution levels. There are several reasons. First, we can resolve finer details in the velocity and the images, which results in a more complicated deformation. Second, we use $\beta_v = 1\text{E}{-4}$ for all resolutions, to be consistent. Given the observed change of information content, one should in general adapt β_v according to the resolution level. Our experiments for the image size 384^3 have a higher variation in the number of Newton steps and matvecs. We use relative tolerances in our

Table 3.7: *Registration performance for* `PyCA` *[127],* `deformetrica` *[51], and the improved GPU variant of* `CLAIRE` *executed on a V100 and a P100 for a typical neuroimaging data sets (*`na02` *to* `na01`*; grid size:* 256^3 *and* 384^3 *(*`CLAIRE` *only); initial DICE:* 0.55*).*

	N	variant	det $\boldsymbol{F}$ min	det $\boldsymbol{F}$ mean	det $\boldsymbol{F}$ max	DICE DICE	DICE mism.	DICE $\|\boldsymbol{g}\|_{\text{rel}}$	#iter	#MV	time V100	time P100
CLAIRE	256^3	cpu-fft-cubic	0.41	1.01	3.62	0.86	2.9E−2	3.7E−2	14	81	146.69	—
		gpu-fft-cubic	0.41	1.01	3.57	0.85	3.0E−2	3.8E−2	14	81	12.38	—
		gpu-fd8-cubic	0.41	1.01	3.57	0.85	3.0E−2	3.7E−2	14	81	8.66	—
		gpu-fd8-linear	0.43	1.01	3.83	0.86	2.7E−2	3.1E−2	14	75	5.87	9.01
CLAIRE	384^3	gpu-fft-cubic	0.37	0.59	3.78	0.86	2.6E−2	3.4E−2	16	152	72.82	—
		gpu-fd8-cubic	0.40	0.59	3.55	0.85	3.4E−2	4.3E−2	15	91	31.59	—
		gpu-fd8-linear	0.41	0.59	3.71	0.85	3.1E−2	3.8E−2	15	85	21.69	—

	N	#iter	mism.	time V100	P100
PyCA	256^3	100,50	4.2E−1	1.1E1	1.9E1
		100,100	3.4E−1	1.8E1	3.4E1
		300,300	2.4E−1	5.3E1	1.0E2
		500,500	2.1E−1	8.9E1	1.7E2
		1000,1000	1.9E−1	1.8E2	3.4E2

	N	#iter	mism.	time V100	P100
deformetrica	256^3	10	4.8E−1	1.4E2	–
		25	4.0E−1	2.5E2	–
		50	3.5E−1	4.4E2	–
		100	3.2E−1	8.2E2	–
		300	2.8E−1	2.4E3	–

algorithm. Consequently, we expect that differences in numerical accuracy and changes in the resolution (more frequencies can be resolved) have an effect on the number of iterations required until convergence.

For the CPU baseline in Figure 3.3, the runtime is dominated by the application of first-order derivatives and IP operations. If we add the execution time of high-order spectral derivatives (bars in dark gray in the "*other*" category), we see that almost all runtime goes to differentiation and IP. We observe a similar behavior for the GPU implementation. We expect a significant reduction in the runtime of our GPU accelerated version of `CLAIRE` compared to the CPU version if we can speed up the evaluation of these kernels. This is reflected in Table 3.7.

The breakdown in Figure 3.3 provides additional insight. We can see that the execution time for the first-order derivatives reduces by a factor of ≈ 3.5 when switching from spectral methods to an optimized FD8 implementation (Figure 3.3, bottom block; yellow bars for the 1st derivative). If we switch from cubic to linear IP, we see a reduction in the execution time by a factor of ≈ 2. The runtime of the other operations remains almost constant. So, overall we went from a solver that is bound by the throughput of first order derivatives and

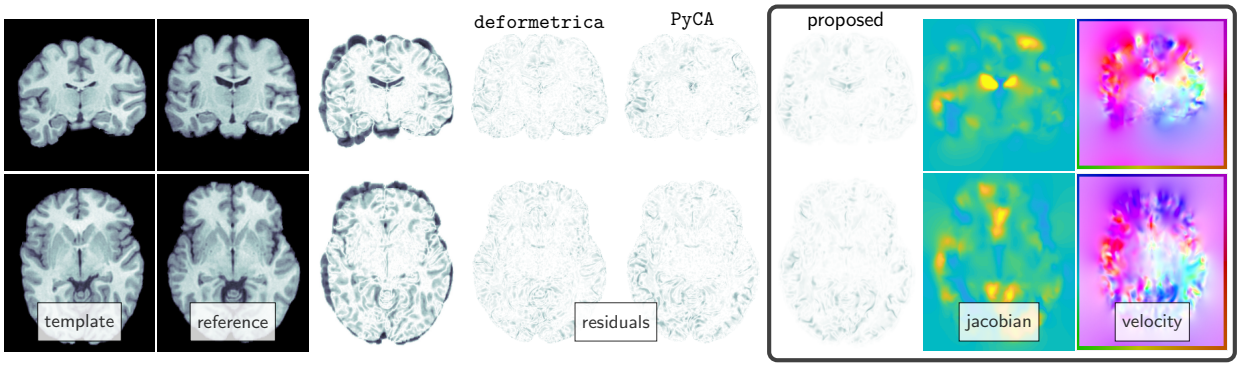


Figure 3.4: *Registration results for image* `na03` *to* `na01` *for* `deformetrica`, `PyCA`, *and our improved implementation of* `CLAIRE`.

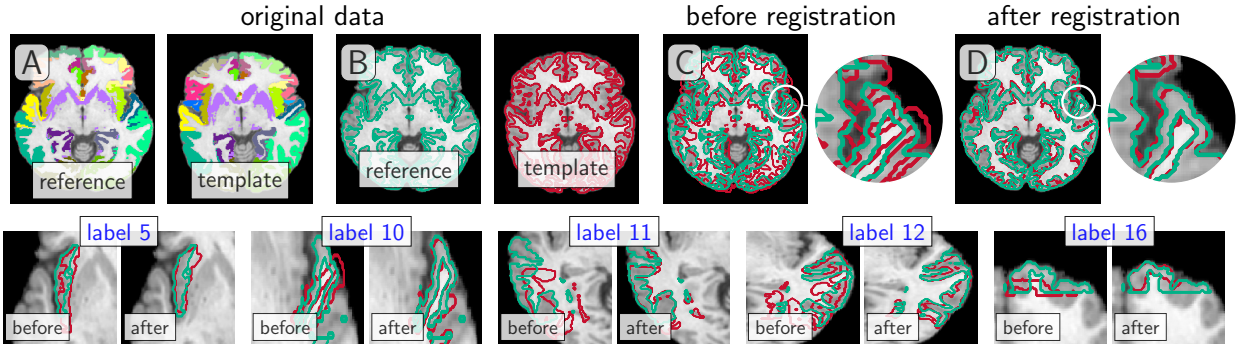


Figure 3.5: *Registration results for* `CLAIRE`. *Top row: In (A) we show the image data overlaid with the 32 gray matter labels (datasets* `na03` *and* `na01`*). In (B) we show the contours of the union of these labels overlaid onto the reference and template image, respectively. In (C) we show the two contours overlaid onto the reference image before registration and in (D) after registration (red contour: template image; green contour: reference image). The circles show a closeup. In the bottom row we show contours before and after registration (left and right, respectively) for five of the 32 gray matter labels visualized in (A) (top row).*

IP operations, to a solver that is bound by the execution time of high-order derivatives.

Comparison to other GPU Implementations

Purpose: We compare the performance of our new GPU version of `CLAIRE` to other GPU implementations of LDDMM-type methods [25, 26, 58, 161, 162].

Setup: The first software package is `PyCA` [127]. `PyCA` uses gradient descent for optimization. Its interface is written in `python`. The libraries and modules used for the com-

pilation of `PyCA` and `deformetrica` are listed in the citations [127] and [51], respectively. The second software package is `deformetrica` [51]; `deformetrica` uses a LBFGS method for optimization. The gradient of the optimization problem is computed based on automatic differentiation [26]. We have executed both registration packages on three different neuroimaging data sets; in particular, we selected `na02`, `na03`, and `na10` as template images and `na01` as the reference image. The runs are performed on full resolution (i.e, $N = 256^3$). We only include results for one exemplary pair of images; additional results can be found in the preprint of this manuscript. We slightly modify scripts available in the repositories of these two software packages to execute these runs (using the default parameters available in the scripts). We vary the number of iterations for `PyCA` and `deformetrica` to make sure we (*i*) do not terminate early, (*ii*) do not perform unnecessary iterations, and (*iii*) (possibly) generate the most accurate results attainable for the default settings (subject to a reasonable iteration count/runtime).

Results: We report exemplary results of our study in Table 3.7. We showcase exemplary registration results in Figure 3.4 and Figure 3.5. In Figure 3.4, we show (from left to right; coronal views: top row; axial views: bottom row) the reference image, the template image, the mismatch before and after registration for `deformetrica`, `PyCA`, and the proposed method, respectively. We also provide point wise maps for the determinant of the deformation gradient and a map of the orientation of the velocity field for `CLAIRE`. Figure 3.5 shows image data overlaid with the 32 gray matter labels, contours of the union of these labels overlaid onto the data. More extensive experiments can be found in [108]. In the present work, we are only interested in demonstrating that switching to our GPU implementation (with mixed-precision accuracy) does not deteriorate the results.

Observations: The most important observation is that the proposed method delivers a mismatch that is about one order of magnitude better than `PyCA` and `deformetrica` for the default settings, with a more than one order of magnitude decrease in runtime. Our approach is up to 30× faster with a 6× better mismatch (peak performance for the fastest

variant of CLAIRE). Note that PyCA uses first order methods for optimization. Therefore, each iteration is much cheaper. In CLAIRE, we use second order information (Newton). Our method makes more progress per iteration but also requires more work; we need to iteratively solve a linear system to compute the search direction. Thus, time per iteration is not a good measure on its own. We need to compare how much work (runtime) it requires to reach a certain accuracy (mismatch between the data). For the proposed method, we use convergence criteria based on the relative reduction of the gradient norm. The two other methods considered here terminate when they reach an upper bound for the iterations. The best result is obtained for PyCA with 1,000 gradient descent steps per level. If we would further increase the runtime (number of iterations) we would probably obtain results that are closer to those obtained for the proposed method (in terms of mismatch). We observe a linear increase in the runtime with respect to the number of iterations for both methods. We note that the differences in accuracy between the methods can be attributed to various factors (e.g., different optimization methods; convergence criteria; different regularization weights and norms; different parameters for the algorithm; or different mathematical formulations). The findings reported here are in accordance with timings reported in the literature [25, 161, 162]. Figure 3.5 shows that not only the DICE coefficients indicate good quality od registration results, but also the label contours match very well after registration.

3.4 Chapter conclusions

In this chapter, we presented algorithms, analysis, and numerical experiments for an improved GPU implementation of the CPU registration solver CLAIRE for LDDMM image registration. This problem is resource constrained because clinical workflows require high-throughput, with one or more registration tasks per node. Typical image sizes fit into the memory of a single GPU in our optimized implementation. MPI parallelism cannot help since multiple registration tasks can take place in an embarrassingly parallel way. Therefore, our focus is on single node and, in particular, on single device optimizations. Our work applies to

other transport dominated forward and inverse problems. For e.g., the semi-Lagrangian GPU algorithm applies to particle-in-cell and weather/climate codes. We demonstrated over 10× speedup over state-of-the-art GPU implementations of LDDMM registration. We showed that the problem is memory-bound but it utilizes over 50% of the peak bandwidth and has sufficient arithmetic intensity to deliver multi TFLOP/s performance.

Chapter 4

Multi-node multi-GPU image registration and novel preconditioning

In the previous chapter, we discussed the important kernels and described an optimized single GPU implementation for `CLAIRE`. We compared our LDDDMM registration implementation with other software packages and demonstrated a speedup of over 23×. However, despite the need for high computational throughput and the existence of several software packages for LDDMM, there is very little work on multi-node multi-GPU implementations for large-scale applications. One such application is the registration of CLARITY images [41, 88, 93, 94, 151, 157] of resolution in the order of $20\,\mathrm{K} \times 20\,\mathrm{K} \times 1\,\mathrm{K}$, which corresponds to a problem with about 1.2 trillion unknowns (see Figure 1.3). Registration problems of such proportions cannot be solved on a single GPU due to memory limitations as these images occupy several Terabytes(TB) of space. Therefore, there is a need for a scalable image registration framework[7] which can utilize multiple GPUs. Towards this end, we extend `CLAIRE` from a single GPU setup to a multi-node multi-GPU setup. To reiterate, `CLAIRE`

[7]This chapter is based on the author's contribution in the following publication *M. Brunn(*), N. Himthani(*), G. Biros, M. Mehl and A. Mang: Multi-node multi-GPU diffeomorphic image registration for large-scale imaging problems, Proceedings of ACM/IEEE Supercomputing Conference (SC20), 2020 (* - equal contribution)*. The author contributed to the developement of the multi-GPU model for the interpolation and finite difference numerical schemes and writing the C++ code for the same. The author also conducted scaling experiments for the following – interpolation kernel and semi-Lagrangian scheme, full 3D image registration strong and weak scaling runs, writing scripts for image pre-processing and the analysis and interpretation of results. In addition to this, the author was also involved in the writing and review of the research manuscript.

has already been developed to scale on standard x86 CPU clusters using the Message Passing Interface (**MPI**) for parallelism [65,105,106,108]. The overall mathematical formulation and solution strategy has not been altered from Chapter 2.

Challenges and Contributions

In this chapter, we propose a new, highly optimized multi-node multi-GPU implementation of `CLAIRE`. The main ***challenges*** are (*i*) eliminating costly host-to-device copies, (*ii*) addressing significant communication costs between devices, (*iii*) reducing memory pressure to enable large-scale runs on limited resources, and (*iv*) identifying an adequate balance between parallelism and local computational throughput. (Per GPU we need to hold enough data to locally perform a sufficient amount of computations, since the computational kernels are extremely fast. Too few data to process per GPU deteriorates scalability. This effect is much more pronounced on GPUs compared to CPUs.) Our main ***contributions*** are:

1) We propose an efficient GPU-only single- and multi-node multi-GPU implementation of `CLAIRE`. The proposed multi-GPU implementation [29,30] is available for download at [105].

2) We minimize communication between host and device through CUDA-aware MPI, and increase the computational throughput in the most important computational kernels of the solver, scattered-data interpolation (**IP**) and differentiation.

3) We propose several improvements to reduce memory pressure and, thus, further increase the computational throughput. With the proposed implementation, it is possible to solve problems with datasets of grid sizes of 512^3 on a single node using four NVIDIA Tesla V100 GPUs in under 30 s.

4) We propose a completely new preconditioner for the reduced-space Hessian based on a zero-velocity approximation, which we term Inv$\mathcal{H}_0$. This allows us to eliminate expensive incremental forward and adjoint PDE solves (hyperbolic transport equations) in the evaluation of the preconditioner. Our method is matrix-free (we do not store or assemble the precon-

ditioner or the Hessian). To further amortize computational costs, we propose a two-level coarse grid approximation.

5) We report results for synthetic and real data, which includes results for CLARITY imaging data for a grid size of 1024×768×768. Overall, we achieve a speedup of up to about 70% on a single GPU compared to the state-of-the-art [28]. This makes the proposed solver 34× faster than the CPU version [65, 106, 108] and 50× faster than other, exemplary GPU-accelerated implementations for LDDMM (c.f., benchmark study in [28]). Moreover, our multi-GPU implementation allows us to solve problems that are approximately 152× larger ($N = 2048^3$, 25 B unknowns) compared to [28].

Limitations

We have optimized memory allocation for the core components of `CLAIRE`. Additional optimizations by sharing memory across external libraries and parallel-in-time integration methods to further reduce the memory pressure remain subject to future work. Moreover, `CLAIRE` uses stationary velocities. This drastically improves efficiency, but results in theoretical limitations.

Related Work

The work in this chapter builds upon the open source framework termed `CLAIRE`. Related LDDMM software packages include `Demons` [155], `ANTs` [12–14], `DARTEL` [7], `deformetrica` [25, 26, 53, 58], `FLASH` [167], `LDDMM` [24, 34], `ARDENT` [118], `ITKNDReg` [83], and `PyCA` [126]. Literature surveys of image registration can be found in [113, 142]. We refer to [108] for a recent overview of existing LDDMM methods. Surveys of GPU-accelerated solvers for image registration are [55, 60, 139]; particular examples for various formulations are [26, 31, 42, 52, 56, 68, 69, 71, 72, 86, 90, 112, 137, 138, 141, 153, 154]. Multi-GPU implementations for LDDMM in the context of atlas construction are described in [71, 72, 153, 154]. None of the hardware-

accelerated LDDMM methods cited above, except for `CLAIRE` [28, 65, 102, 103, 105, 106, 108], use second-order information for numerical optimization. Many of the available methods reduce the number of unknowns by using coarser resolutions either through parameterization or by solving the problem on coarser grids; they use simplified algorithms and deliver subpar registration quality.

The work most pertinent to ours is [28, 71, 72]. In [71, 72], a multi-node multi-GPU implementation of the algorithm in [86] is presented. The considered application is atlas construction from multiple image volumes. While computational throughput on a single GPU is optimized, the focus is on data-parallelism: Multiple input images are loaded and synchronously processed on distinct GPUs. We propose a multi-node multi-GPU framework with high computational throughput for single (large-scale) registration problems. This problem is no longer embarrassingly parallel. The computational bottlenecks in [71, 72, 86] are the repeated solution of a Helmholtz-type PDE and trilinear scattered data interpolation to apply the deformation map. The PDE is solved via an implicit successive over-relaxation method. The trilinear interpolation kernel is hardware accelerated with 3D texture volume support. The runtime for a single dataset of size $160 \times 192 \times 160$ is 20 s on an NVIDIA Quadro FX 5600. The work in [28] presents a single-node single-GPU implementation of `CLAIRE`. The present work ports `CLAIRE` to a heterogeneous multi-node multi-GPU environment by exploiting CUDA-aware MPI. We present several improvements over the computational kernels described in [28] (see contributions above).

4.1 Discretization and Numerical Algorithms

We use the same unaltered formulation from §2.1.1. As discussed in Chapter 2, inverting $\mathbf{H}$ in (2.5) is the most expensive part of `CLAIRE`. To this end, we propose a new preconditioner for the reduced-space system in (2.5) to amortize the computational costs.

Preconditioning

As we can see in (2.6), the Hessian operator consists of two terms. In a discrete setting, we have $\mathbf{H} = \mathbf{A} + \tilde{\mathbf{H}}$. Here, $\mathbf{A} \in \mathbb{R}^{3N,3N}$ corresponds to the regularization operator $\mathcal{A}$ and forming $\tilde{\mathbf{H}} \in \mathbb{R}^{3N,3N}$ involves $3N$ solutions of (2.7) and (2.8). Given that each Hessian matvec involves two PDE solves, we have to keep the number of PCG iterations as small as possible. With this in mind, we propose a new preconditioner.

As a benchmark, we consider a spectral preconditioner Inv$\mathcal{A}$ based on the inverse of $\mathbf{A}$—a common choice in PDE-constrained optimization [5, 32, 107] and the default option in `CLAIRE` [65, 102, 106]. This preconditioner is given by

$$\mathbf{s} = (\beta_v \mathbf{A})^{-1}\mathbf{r}, \tag{4.1}$$

where $\mathbf{r}$ is the residual of the Krylov solver. The cost of applying $(\beta_v \mathbf{A})^{-1}$ to a vector is two FFTs and a Hadamard product in spectral space.

The proposed preconditioner is based on a zero-velocity approximation of $\mathbf{H}$. This allows us to evaluate the Hessian matvec without having to solve (2.7) and/or (2.8). We term this preconditioner Inv$\mathcal{H}_0$. For $\mathbf{v} = \mathbf{0}$, the reduced-space Hessian system in (2.5) becomes $\mathbf{H}_0\tilde{\mathbf{v}} = -\mathbf{g}$, where $\mathbf{H}_0 := (\beta_v \mathbf{A} + \nabla \mathbf{m}_0 \otimes \nabla \mathbf{m}_0)$. Here, $\mathbf{m}_0$ is a discrete representation of the template image and $\otimes$ denotes the outer product. It is important to notice that $\mathbf{m}_0$ does not change during the course of the iterations. We use an (approximate) inverse of $\mathbf{H}_0$ as a preconditioner. To compute the action of $\mathbf{H}_0^{-1}$ we iteratively solve the linear system

$$(\beta_v \mathbf{A} + \nabla \mathbf{m}_0 \otimes \nabla \mathbf{m}_0)\mathbf{s} = \mathbf{r} \tag{4.2}$$

using a matrix-free PCG method with a relative tolerance $\epsilon_{\mathcal{H}_0}\epsilon_K$. Here, $\epsilon_K > 0$ is the tolerance for the outer PCG and $\epsilon_{\mathcal{H}_0} \in (0, 1)$. (We need to use a smaller tolerance in the

inner PCG since the preconditioner would not act as a linear operator otherwise. We set $\epsilon_{\mathcal{H}_0}$ to 1E−3 for the NIREP data and to 1E−2 for the CLARITY data for our runs (see Table 4.5). These values were determined by experimentation in an attempt to obtain optimal runtimes per type of dataset.)

To compute the inverse of $\mathbf{H}_0$ efficiently, we propose several twists. First, we left-precondition $\mathbf{H}_0$ in (4.2) with $(\beta_v\mathbf{A})^{-1}$ (this adds vanishing computational costs; see above). Second, since $\mathbf{H}_0$ represents a zero-velocity approximation to $\mathbf{H}$, we expect the performance of the preconditioner to deteriorate as we iterate. As a remedy, we replace $\mathbf{m}_0$ in (4.2) with the deformed template image obtained for the current iterate $\mathbf{v}_k$ at the beginning of each Gauss-Newton iteration. Third, to further amortize the computational costs, we consider a second variant of Inv$\mathcal{H}_0$ that exploits a coarse grid discretization. We term this variant 2LInv$\mathcal{H}_0$. Here, we invert $\mathbf{H}_0$ on a coarse grid with half the resolution of the fine grid. We restrict the residual $\mathbf{r}$ and $\nabla\mathbf{m}_0$ in (4.2). The restriction and prolongation operators are implemented in the spectral domain. 2LInv$\mathcal{H}_0$ operates only on the low frequency components of $\mathbf{r}$. The solution of the iterative solver, $\mathbf{s}_c$, found on the coarse grid is prolonged to the fine grid and added to the filtered high frequency part of the original residual on the fine grid. In this context, the left-preconditioner $(\beta_v\mathbf{A})^{-1}$ can be viewed as a (poor) approximation of a multi-grid smoother. Algorithm 1 gives an overview of the two proposed preconditioner variants.

```
1  func InvH0PC(r)
2      s_f ← (β_v A)^{-1} r,    tol ← ε_{H_0} ε_K
3      s_f ← run CG(H_0, s_f, (β_v A)^{-1}; tol)          ▷ solve (4.2)
4      return s_f

5  func TwoLvlInvH0PC(r)
6      s_f ← (β_v A)^{-1} r,    tol ← ε_{H_0} ε_K
7      s_c ← Restrict(s_f)
8      s_c ← run CG(H_{0,c}, s_c, (β_v A)^{-1}; tol)      ▷ solve (4.2) on coarse grid
9      s_f ← Prolong(s_c) + HighPass(s_f)
10     return s_f
```

Algorithm 1: Algorithmic overview of the two variants of the Inv$\mathcal{H}_0$ preconditioner.

We observed that the performance of Inv$\mathcal{H}_0$ deteriorates for vanishing β_v. We found by experimentation that, if we use a lower bound of 5E−2 for β_v in (4.2), the preconditioner remains effective even for vanishing β_v for the overall problem. That is, if $\beta_v < 5\text{E}{-}2$, we set β_v in (4.2) to 5E−2.

Finally, the suggested setting for `CLAIRE` is to use a β_v-continuation scheme for the solution of the inverse problem (2.1) [28, 102, 108]. That is, `CLAIRE` solves the registration problem for a vanishing sequence of values for β_v. For each new value, the velocity obtained at the former step is used as an initial guess for the Gauss-Newton-Krylov solver. For large β_v, the problem is dominated by the regularization operator $\mathbf{A}$. As a consequence, the problem is not only easy to solve but the spectral preconditioner is also quite effective. Therefore, if `CLAIRE` is executed using a β_v-continuation scheme we use Inv$\mathcal{A}$ for $\beta_v > 5\text{E}{-}1$ and switch to either variant, Inv$\mathcal{H}_0$ or 2LInv$\mathcal{H}_0$, for $\beta_v \leq 5\text{E}{-}1$ (this bound has been determined by experimentation).

4.2 Computational Kernels

In this section, we describe the multi-node multi-GPU implementation of our computational kernels. In Alg. 2, we summarize the overall algorithm. We identify the three most important kernels and their overall contribution to the computational cost: interpolation (**IP**), finite differences (**FD**), and fast-Fourier transforms (**FFT**s). The costs of solving $\mathbf{g}(\mathbf{v}) = \mathbf{0}$ (first-order optimality conditions, where $\mathbf{g}$ is a discrete version of (2.3)) for $\mathbf{v}(\boldsymbol{x})$ are

$$c_{\text{total}} \approx n_{\text{GN}} \left(n_{\text{CG}} \left(2c_{\text{PDE}} + c_{\mathbf{H}} + c_{\text{PC}} \right) + 2c_{\text{PDE}} \right), \tag{4.3}$$

where n_{GN} is the number of Gauss–Newton iterations, $n_{\text{CG}}(2c_{\text{PDE}} + c_{\mathbf{H}} + c_{\text{PC}})$ summarizes the cost of computing the Gauss-Newton step in (2.5), n_{CG} is the number of PCG iterations per Gauss-Newton step (assuming that it is constant to simplify the analysis). The cost for

evaluating (2.6) is denoted by $c_{\mathbf{H}}$. The cost c_{PC} is for the application of the preconditioner (e.g., iteratively solving (4.2)). c_{PDE} is a prototypical cost for solving the forward or adjoint equations; in particular, (2.7) and (2.8). Let c_{FD} denote the cost for the FD gradient and c_{IP} the cost for evaluating the IP kernel for a scalar field, then c_{PDE} for the RK2 implementation of the semi-Lagrangian scheme is $\mathcal{O}(N_t(c_{\text{FD}} + 4c_{\text{IP}}))$ for (2.7) (if we choose to not store the gradient of the state variable during the solution of (2.1c)) and $\mathcal{O}(N_t c_{\text{IP}})$ for (2.8). The remaining $2c_{\text{PDE}}$ in (4.3) are for evaluating the objective functional (2.1) (which involves the solution of (2.1c)) and the solution of the adjoint problem in (2.4). The cost $c_{\mathbf{H}}$ for evaluating (2.6) is dominated by $2c_{\text{FFT}}$ for applying the regularization operator in the spectral domain (or its inverse) and $N_t c_{\text{FD}}$ (if we choose to not store the gradient of the state variable). The cost for the preconditioner c_{PC} depends on the choice of the preconditioner. That is, c_{PC} is $\mathcal{O}(2c_{\text{FFT}})$ for Inv$\mathcal{A}$, $\mathcal{O}(2c_{\text{FFT}} n_{\text{CG,PC}})$ for Inv$\mathcal{H}_0$, and $\mathcal{O}\left(2c_{\text{FFT}}\frac{1}{8}\left(2c_{\text{FFT}} n_{\text{CG,PC}}\right)\right)$ for 2LInv$\mathcal{H}_0$, where $n_{\text{CG, PC}}$ is the number of PCG iterations to compute the action of the inverse of $\mathbf{H}_0$. (We kept some of the constant factors to explicitly document the computational steps.). The computational and communication components of c_{IP}, c_{FD} and c_{FFT} are reported in §4.2.1, §4.2.2 and §4.2.3, respectively. We refer to [28], where a DRAM based (ignoring cache heirarchy) roofline analysis is performed for the IP and FD kernels (on a single GPU). DRAM memory accesses for each kernel are modelled analytically assuming full reuse. The number of floating point operations are also estimated analytically. The arithmetic intensity, which is defined as the ratio of total number of floating point operations to number of bytes accessed, is assessed based on this model. The analytical value is compared with the experimental value obtained by the NVIDIA profiler. It is found that both kernels are bound by the GPU DRAM bandwidth.

The work in [28] discusses several technical optimizations beyond a pure transition to GPUs, in particular, several options for the IPs as the most important kernel in the semi-Lagrangian solver. In addition, [28] suggests to replace FFTs used in [108] for first order derivatives by FD approximations. In [28], it is shown empirically that this does not

deteriorate the accuracy if FD kernels of high enough order are used. In the following, we describe the implementation of different variants of these kernels, which includes optimizations compared to the work in [28] for efficient execution on a multi-node multi-GPU architecture.

```
1  v ← v_init,    ε_N ← 5E−2
2  run NewtonSolver(v, ε_N)
3      m ← solStateEQ(v, m_0)                          ▷ solve (2.1c)
4      λ ← solAdjointEQ(v, m, m_1)                     ▷ solve (2.4)
5      g ← evalGrad(v, m, λ)                           ▷ evaluate (2.3)
6      ε_K ← min(√(‖g‖_rel), 0.5)
7      run PCG(MatVec, −g, ε_K)                        ▷ solve (2.5)
8          MatVec(ṽ)
9              m̃ ← solIncStateEQ(v, ṽ, m)              ▷ solve (2.7)
10             λ̃ ← solIncAdjointEQ(v, m̃)               ▷ solve (2.8)
11             Hṽ ← evalMatVec(ṽ, m, λ̃)                ▷ eval (2.6)
12             r ← −g − Hṽ
13         applyPrecond(r)
14             see Alg. 1
15     run LineSearch(α)
16         m ← solStateEQ(v + αṽ, m_0)                 ▷ solve (2.1c)
17         evalObjective(v + αṽ, m)                    ▷ eval (2.1a)
18     v ← v + αṽ                                      ▷ Newton step
```

Algorithm 2: Overview of the Gauss–Newton–Krylov solver implemented in `CLAIRE`.

The total memory consumption mostly depends on the domain size $N = N_1 N_2 N_3$. The state variable $m(\boldsymbol{x}, t)$ has to be stored for all time steps to avoid additional PDE solves. The memory footprint for the proposed method is

$$\begin{aligned}\mu_{\text{total}} &\approx \mu_{\text{PDE}} + \mu_{\text{FFT}} + \mu_{\text{FD}} + \mu_{\text{SL}} + \mu_{\text{GN/CG}} + \mu_{\text{IP}} + \mu_{\text{API}} \\ &= ((24 + N_t) + 7 + 2 + 11 + 30)^{N\mu_0}/_p + \mu_{\text{IP}} + \mu_{\text{API}} \\ &= (74 + N_t)^{N\mu_0}/_p + \mu_{\text{IP}} + \mu_{\text{API}},\end{aligned}$$

where μ_0 is word size of the datatype (i.e. 4 byte for single precision floating point values). The memory required for the ghost layer communication in the IP model is $\mu_{\text{IP}} \approx 30 d N_2 N_3 \mu_0$ with polynomial degree d. Note that the runtime API overhead, μ_{API}, depends on N (especially for `cuFFT` [122] and `PETSc` [18, 19]), but is not further estimated.

4.2.1 Interpolation

The semi-Lagrangian scheme requires IP of vector and scalar fields along backward characteristics. We use Lagrange polynomial-based cubic IP but also consider first-order trilinear IP since GPUs offer hardware acceleration through texture units (not fully single-precision). The formula for interpolating at an off-grid query point $\boldsymbol{x} = (x_1, x_2, x_3)$ is given by

$$f(\boldsymbol{x}) = \sum_{i,j,k=0}^{d} f_{ijk}\phi_i(x_1)\phi_j(x_2)\phi_k(x_3),$$

where f_{ijk} is the function value at a grid point, d is the polynomial order and ϕ_l, $l = 0, \ldots, d$, are the Lagrange polynomial basis functions. The numerical accuracy and compute performance of variants of the IP kernel on a single GPU have been discussed in [28]. We focus on optimizations for the multi-GPU implementation. We follow the workflow described in [65, 106] with the following major modifications:

1) We use CUDA-aware MPI to reduce or eliminate expensive on-node host-device transfers.

2) We use the `thrust` library [80] to efficiently determine, which query points need to be processed by which GPU, thereby completely eliminating host-side computation.

3) We use a sparse point-to-point communication to send points on the backward characteristics to other processors, as proposed in [65]. We adaptively allocate memory for the respective MPI send and receive buffers using an estimate of the maximal displacement of grid points along backward trajectories based on the CFL number of the velocity field.

4) Following [28], we perform local IP on a single GPU using GPU-TXTLAG or GPU-TXTLIN (for high-resolution images). Although GPU-TXTSPL in [28] is much faster than GPU-TXTLAG on a single GPU, for the distributed memory implementation it requires ghost layer communication for the pre-filtering step, which makes it slower than GPU-TXTLAG.

The computational cost c_{IP} of applying the IP kernel GPU-TXTLAG is $\mathcal{O}(^{482N}/_{p})$ (see [28]), where p is the number of processors and $N = N_1N_2N_3$. For GPU-TXTLIN, it is $\mathcal{O}(^{30N}/_{p})$. The total cost of communicating ghost points, query points and interpolated values is $\mathcal{O}(u_{\max}N_2N_3)$ where $u_{\max} \in \mathbb{R}$ is an estimate of the maximum displacement of a voxel from a regular grid point along the coordinate directions. For the IP kernel we do not consider overlapping communication and computation because of the data dependencies in the semi-Lagrangian scheme.

We perform a weak scaling experiment for an isolated semi-Lagrangian solve on a real dataset and present the runtime breakdown in Table 4.1. We use a realistic velocity field for this experiment (obtained by registration of two brain images) to ensure a representative scenario for the communication of query points between MPI ranks. The major **observations** are:

1) Since we use slab decomposition in x_1-dimension, the message size for `ghost_comm` is $\mathcal{O}(N_2N_3)$. Hence, it roughly doubles every time N_2 or N_3 is doubled.

2) We see a similar increase for `interp_comm` and `scatter_comm`. Due to the non-uniformity in space of the query points, communication time does not double exactly and we observe an imbalance in the communication for different MPI ranks.

3) The time spent in `interp_kernel` is almost the same across all cases and takes up the majority of the time for up to 16 GPUs. Beyond 16 GPUs, communication dominates the overall runtime.

4) Since we are performing scattered IP, determining which and how many query points need to be processed locally or sent to other MPI ranks in `scatter_mpi_buffer` leads to expensive scattered memory accesses.[8] This explains why `scatter_mpi_buffer` requires almost one third of `interp_kernel` runtime.

[8] We rely on the `thrust::copy_if` algorithm for this purpose.

Table 4.1: *Weak scaling study for the IP kernel. We report runtimes for the semi-Lagrangian scheme. We advect a real brain MRI (`na10` of the NIREP data; see §4.3) with a velocity field obtained from the registration of `na10` to `na01`. We use cubic IP GPU-TXTLAG and $N_t = 4$ time steps. We report the runtime (in seconds) of the major components in the algorithm and their percentages with respect to the total runtime. These components are `ghost_comm` (communication of ghost points), `interp_comm` (communication of interpolated values), `scatter_comm` (communication of query points), `interp_kernel` (IP kernel), `scatter_mpi_buffer` (creation of MPI buffer for sending query points to other ranks). The experiments were performed on TACC's Longhorn system with four Nvidia V100 GPUs per node and a single GPU per MPI rank. We scale from a single GPU to 64 GPUs for grid resolutions ranging from 256^3 to 1024^3.*

size	256×256×256		512×256×256		512×512×256		512×512×512		1024×512×512		1024×1024×512		1024×1024×1024	
#GPUs	1	%	2	%	4	%	8	%	16	%	32	%	64	%
`ghost_comm`	0.00	0.0	2.48E−3	7.6	3.49E−3	9.9	7.51E−3	18.0	8.66E−3	19.1	1.31E−2	24.0	2.23E−2	31.3
`interp_comm`	0.00	0.0	1.71E−3	5.2	1.80E−3	5.1	3.62E−3	8.7	4.17E−3	9.2	5.92E−3	10.9	9.73E−3	13.6
`scatter_comm`	0.00	0.0	2.65E−4	0.8	7.81E−4	2.2	2.02E−3	4.8	2.85E−3	6.3	5.42E−3	10.0	8.72E−3	12.2
`interp_kernel`	1.77E−2	93.3	1.79E−2	54.8	1.76E−2	49.8	1.76E−2	42.0	1.83E−2	40.2	1.84E−2	33.9	1.87E−2	26.2
`scatter_mpi_buffer`	0.00E0	0.0	5.88E−3	18.0	7.16E−3	20.3	6.63E−3	15.9	6.98E−3	15.4	7.00E−3	12.9	7.30E−3	10.2
total	1.90E−2	100.0	3.28E−2	100.0	3.53E−2	100.0	4.18E−2	100.0	4.54E−2	100.0	5.44E−2	100.0	7.13E−2	100.0

4.2.2 Finite Differences

The CPU version of `CLAIRE` uses FFTs for spatial derivatives [65, 106, 108]. Since our functions are periodic, these spectral operators are diagonal. [28] proposes a mixed-accuracy implementation that replaces the spectral discretization of the divergence and gradient operators with a FD scheme. This mixed scheme is more accurate (for the considered grid sizes—not asymptotically) and faster than differentiation via FFTs. In particular, an 8^{th} order central difference scheme is used. We extend the single-GPU FD kernel described in [28] to a multi-node multi-GPU environment. The computational cost c_{FD} of applying the FD kernel is $\mathcal{O}(^{20N}/_{p})$, where p is the number of processors and $N = N_1 N_2 N_3$. To compute derivatives at the boundary of our 2D slab decomposition, we communicate a ghost layer of size $\mathcal{O}(N_2 N_3)$ to neighboring MPI ranks. We perform strong and weak scaling experiments for computing the gradient of a synthetic scalar field; see Table 4.2. For a single GPU, no communication is involved. It is much faster than using multiple GPUs (for small problem sizes). In the weak scaling setup, the runtime increases when we switch from one to eight

Table 4.2: *Scalability for our finite difference (FD) scheme for first order derivatives. We show strong scaling for* 512^3 *from one to eight MPI ranks, and weak scaling for* 256^3 *to* 1024^3 *from one to 64 MPI ranks. We report the breakdown of runtime (in seconds) into* `comm` *(communication of ghost points) and* `kernel` *(FD kernel) and show percentages with respect to total runtime.*

#GPUs	size	comm	%	kernel	%	total
1	256^3	0.0	0.0	6.32E−4	100.0	6.32E−4
1	512^3	0.0	0.0	4.82E−3	100.0	4.82E−3
2	512^3	9.37E−4	21.9	3.33E−3	78.1	4.27E−3
4	512^3	7.01E−4	29.2	1.70E−3	70.8	2.40E−3
8	512^3	9.86E−4	53.2	8.66E−4	46.8	1.85E−3
16	512^3	8.94E−4	66.0	4.60E−4	34.0	1.35E−3
64	1024^3	2.85E−3	76.0	9.03E−4	24.0	3.76E−3

to 64 GPUs because the size of the ghost layer increases (N_2 and N_3 increase), while the kernel execution time itself remains constant. In the strong scaling setting, the kernel scales well for up to 8 GPUs. Beyond 8 GPUs, the kernel execution time becomes much smaller than the communication time (which is constant); this negatively impacts the scalability. Since the FD kernel is not a bottleneck—as seen in Table 4.6—we did not explore the idea of overlapping communication and computation when evaluating the kernel.

4.2.3 FFT

The distributed memory implementation of `CLAIRE` [65, 106, 108] uses `AccFFT` [63, 66], which supports MPI for CPUs and GPUs. In [28], `cuFFT` [122] is used, as they focus on a single-GPU implementation. Higher order derivatives and their inverses require 3D FFTs. `AccFFT` uses a pencil decomposition (see, e.g., [106]), which is efficient for 1D FFTs (needed for divergence and gradient operators). In [28], 1^{st} order derivatives have been replaced by FD kernels. For the proposed multi-GPU implementation, we use a combination of `cuFFT` and a new 2D slab decomposition, which allows us to use the highly optimized 2D `cuFFT` on each GPU. We decompose the spatial domain in the outer-most dimension (i.e., x_1) and in the spectral domain in x_2 dimension. Thus, the inner-most x_3 dimension is always continuous in memory. This reduces misaligned memory accesses for communication and transpose operations. The real-to-complex transformation is divided into three steps. (*i*) We use `cuFFT`'s batched 2D

FFTs in the x_2–x_3 plane. (*ii*) The complex data are transposed to a decomposition in x_2 dimension. (*iii*) We apply `cuFFT`'s batched 1D FFTs to the x_1 dimension, which is non-continuous in memory. For the inverse complex-to-real transformation, these three steps are executed in reverse order, using the respective inverse transformations. The complexity for communication of the 2D slab decomposition is $\mathcal{O}(N/P - N/P^2)$ per process. If the FFT is executed on a single rank, we still use `cuFFT`'s 3D FFT to avoid additional operations, in particular an explicit transpose operation on the data and misaligned memory accesses. Also, it reduces the number of memory accesses of the spectral data from device memory.

For communication between GPUs, we use CUDA-aware MPI. We found that `MPI_Alltoallv` (IBM Spectrum MPI 10.3 [2]) is not optimized for direct GPU communication. For communication volumes larger than ∼500 kB, all-to-all communication using direct GPU-optimized peer-to-peer routines is faster on our test system (see Table 4.3). We implement a threshold of 512 kB to switch between an asynchronous peer-to-peer communication scheme or `MPI_Alltoallv`. For FFTs on a single node (four GPUs), we always use the peer-to-peer scheme to utilize the *NVLink* inter-GPU bus. The communication is only overlapped with the process-local transpose operation due to data dependencies.

In addition to the memory footprint of `cuFFT` our 2D slab decomposition needs twice the local domain size to execute an out-of-place transformation. The temporary memory consumtion of `cuFFT` is between $2N/p$ and $16N/p$ real valued elements [122]. Table 4.4 shows that our 3D FFT with 2D slab decomposition is almost as fast as `cuFFT` 3D-FFT, but can be accelerated and scaled to data sizes beyond the memory capacity of a single GPU. Given the $\mathcal{O}(N \log N)$ computational complexity of the FFT (with data size N) and the huge amount of data communication inherent to FFTs, we observe good scalability up to 128 GPUs, for the large problem sizes—even in strong scaling.

Table 4.3: *MPI performance analysis for the proposed FFT kernel. We use one GPU per MPI rank. We report the sustained bidirectional CUDA MPI bandwidth in GB/s. The results are averaged over ten runs and the smallest value for all ranks is presented. We compare* `MPI_Alltoall` *to our own implementation using asynchronous peer-to-peer routines. The local data size per rank is* $8N_1N_2(\lfloor N_3/2 \rfloor + 1)/p$ *byte. The peer-to-peer communication volume is* $8N_1N_2(\lfloor N_3/2 \rfloor + 1)/p^2$ *byte. Runs in the shaded cells have a communication volume larger than* 512 kB*. The fastest runs are highlighted in bold.*

setup		**MPI tasks**					
size	type	4	8	16	32	64	128
256^3	MPI	5.6	5.0	**3.3**	**2.2**	**2.0**	**1.5**
	P2P	**35.7**	**9.3**	2.2	1.3	1.6	1.4
512×256^2	MPI	5.1	5.2	3.5	**1.5**	**1.9**	**1.9**
	P2P	**36.0**	**9.5**	**5.8**	1.0	1.5	1.4
$512^2 \times 256$	MPI	5.4	4.6	3.5	**2.8**	1.6	**2.7**
	P2P	**36.6**	**9.9**	**6.1**	0.4	**1.7**	1.4
512^3	MPI	5.9	4.9	3.9	2.7	**2.5**	**2.7**
	P2P	**37.1**	**9.5**	**5.9**	**4.7**	0.5	1.5
1024×512^2	MPI	6.4	5.4	3.9	3.4	**3.2**	**2.2**
	P2P	**32.6**	**10.1**	**5.9**	**4.8**	0.4	0.5
$1024^2 \times 512$	MPI	6.7	5.5	4.2	3.6	3.4	**2.7**
	P2P	**36.6**	**10.5**	**5.4**	**4.7**	**4.5**	0.3
1024^3	MPI	6.7	5.6	4.4	3.7	3.4	**3.1**
	P2P	**36.8**	**10.6**	**5.2**	**4.6**	**4.3**	0.4

Table 4.4: *Weak (diagonals) and strong (rows) scaling for the proposed 3D FFT kernel in slab decomposition (forward and inverse). We use one GPU per MPI rank. We report the runtime in ms. The FFT uses CUDA-aware MPI. We switch from point-to-point communication to* `MPI_Alltoall` *for small slabs. Results are averaged over 20 runs. For a single rank, the runtime is also given for* `cuFFT` *3D-FFTs (3D). The highlighted runs use peer-to-peer communication.*

		MPI tasks						
size	3D	1	4	8	16	32	64	128
256^3	1.41	1.86	**2.83**	**3.92**	4.17	3.88	2.93	3.76
512×256^2	3.20	3.87	**5.39**	**7.65**	**7.33**	5.21	4.09	4.30
$512^2 \times 256$	7.30	7.70	**8.48**	**13.8**	**13.3**	8.29	5.67	5.12
512^3	16.9	16.9	**15.6**	**25.7**	**24.5**	**16.7**	9.63	7.23
1024×512^2	31.2	40.1	**31.8**	**51.3**	**43.6**	**31.3**	17.8	11.8
$1024^2 \times 512$	—	—	**65.7**	**100**	**90.5**	**54.2**	**33.4**	21.4
1024^3	—	—	**132**	**198**	**182**	**116**	**62.0**	38.4

4.3 Results

We (i) analyze the numerical and runtime efficiency of our new preconditioner and (ii) assess the overall scalability and efficiency of our multi-GPU multi-node implementation.

We use the following datasets: **1) SYN** is a synthetic test problem, where the template image is $m_0(\boldsymbol{x}) := \sum_{i=1}^{3} \sin^2(x_i)/3$ and the reference image $m_1(\boldsymbol{x})$ is computed by solving (2.1c) with initial condition $m_0(\boldsymbol{x})$ and given velocity

$$\boldsymbol{v}(\boldsymbol{x}) := (\sin(x_i), \cos(x_k), \sin(x_k))_{(i,k)=(3,2),(1,3),(2,1)}$$

2) NIREP [39] is a standardized repository for assessing registration accuracy that contains 16 T1-weighted MR neuroimaging datasets (`na01`–`na16`) of different individuals (see Figure 1.1). The original image size is 256×300×256 voxels. **3) CLARITY** [36, 41, 88, 93, 94, 151, 157] are biomedical imaging datasets with a resolution of 0.60 µm×0.60 µm×6 µm and a grid size at the order of 20 K×20 K×1 K (see Figure 1.3) . We have affinely pre-registered these datasets (at a much lower resolution) using `FAIR` [114] prior to executing `CLAIRE`.

All runs were executed on ***TACC's Longhorn system*** in single precision. Longhorn hosts 96 NVIDIA Tesla V100 nodes. Each node is equipped with four GPUs with 4×16 GB GPU RAM (64 GB aggregate) and two IBM Power 9 processors with 20 cores (40 cores per node) at 2.3 GHz with 256 GB memory. Our implementation uses `PETSc` [18, 19] for linear algebra, `PETSc`'s `TAO` package for the nonlinear optimization, `CUDA` [121], `thrust` [80], `cuFFT` for FFTs [122], `niftilib` [59] for I/O, IBM Spectrum MPI [2], and the IBM XL compiler [82].

4.3.1 Preconditioning

We study different preconditioner variants. We use the datasets `na02`, `na03`, and `na10` from the NIREP repository as template images, and `na01` as reference image.

Results

We report convergence plots for a single Gauss-Newton step in Figure 4.1. We initialize the solver with `na10` as template and a reference image synthetically generated by solving the forward problem with a true registration velocity (`na10` to `na01`). The true (non-zero) velocity is used as an initial guess for the Gauss-Newton-Krylov method (i.e., we solve (2.5) at the solution of the inverse problem). This allows us to assess (*i*) the convergence at a point in the optimization landscape at which we expect the PCG to take many iterations and (*ii*) identify potential issues that may arise due to a zero-velocity approximation at a point at which the velocity is non-zero.We report results for varying grid sizes and values for β.

Observations

The proposed preconditioner leads to faster convergence (fewer iterations) and is less sensitive to a reduction in β than Inv$\mathcal{A}$. We expect the preconditioner to be mesh-independent but not β-independent. All preconditioners exhibit (close to) mesh independent behavior. Interestingly, for the considered range for β, 2LInv$\mathcal{H}_0$ is close to being β-independent; only for $\beta = 5\text{E}{-2}$ we see the performance slightly deteriorate as the mesh size increases. In general, we expect that we might have to use larger values for β for higher resolutions, since higher frequencies can occur in the images and the velocity field (coarsening can be viewed as an additional regularization).

4.3.2 Registration Performance

We study the performance of the proposed methods for the solution of the inverse registration problem. We report results for three different template images from the NIREP repository: `na02`, `na03`, and `na10`. For `na10`, we increase the resolution from 256^3 to 1024^3 (spectral prolongation). Results for the registration of the dataset `na10` to `na01` are shown

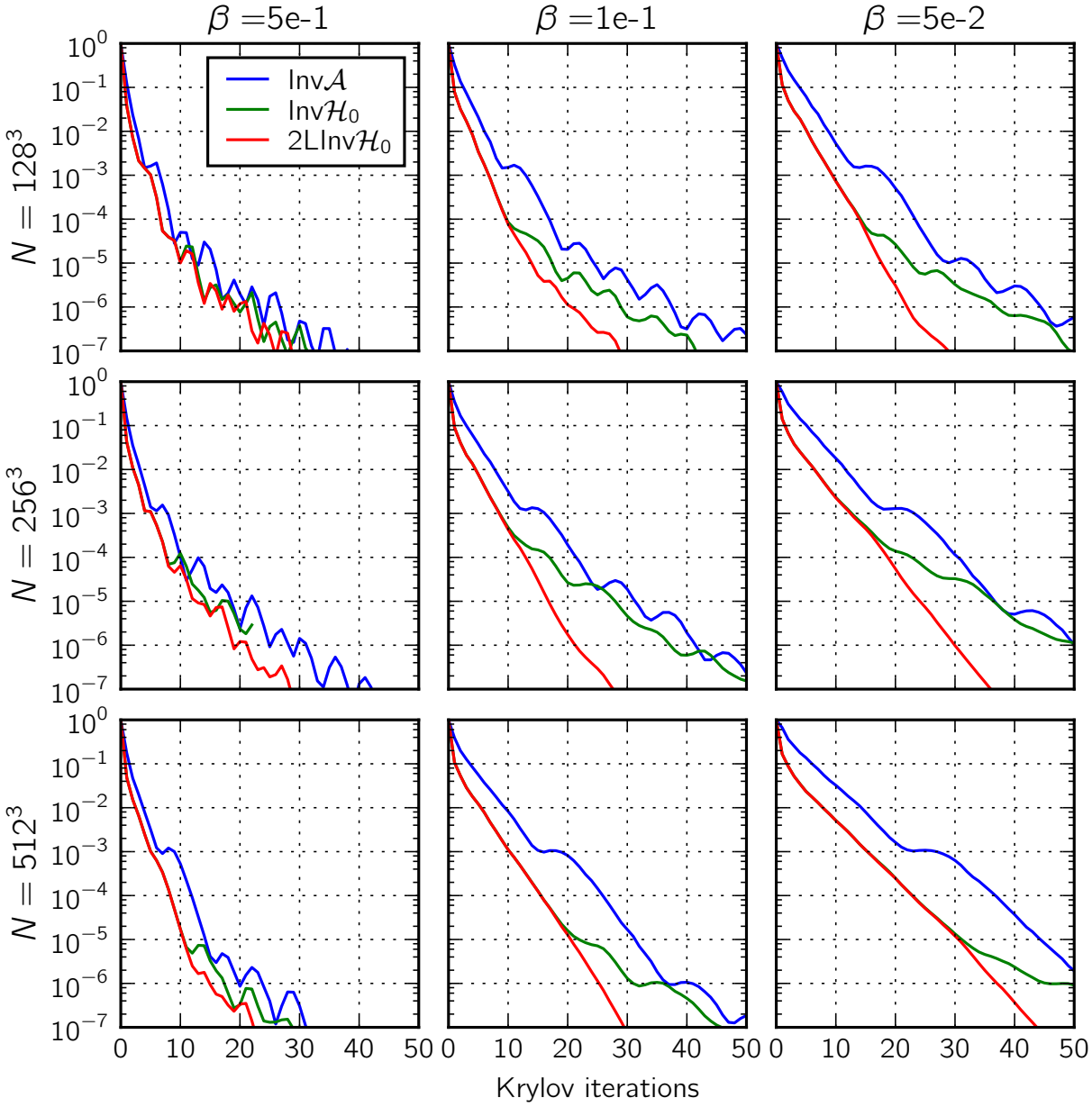


Figure 4.1: *We report the trend of the PCG residual versus PCG iterations for the benchmark preconditioner Inv$\mathcal{A}$ used in [28, 108] and the proposed preconditioner variants Inv$\mathcal{H}_0$ and 2LInv$\mathcal{H}_0$. We vary the regularization parameter β (columns; $\beta \in \{5\text{E}{-}1, 1\text{E}{-}1, 5\text{E}{-}2\}$) and the domain size N (rows; $N \in \{128^2, 256^3, 512^3\}$). We solve the problem at the true solution (see text for a description).*

in Figure 1.1. We expect the convergence behavior of the Gauss-Newton-Krylov method to be independent of the mesh size. In addition to that, we report results for the registration of two representative CLARITY volumes (dataset Cocaine 175 to Control 189; Control 189 is visualized in Figure 1.3). We consider all preconditioner variants.

Results

The results can be found in Table 4.5. We report the number of Gauss-Newton iterations, the accumulated number of PCG iterations across all Gauss-Newton iterations, the relative reduction of the mismatch, the relative reduction of the gradient, the number of applications of the inverse regularization operator, the number of applications of Inv$\mathcal{H}_0$ or 2LInv$\mathcal{H}_0$, the number of PCG iterations to invert $\mathbf{H}_0$ (in total and on average), the time spent in the core parts of the solver, and the total runtime. We visualize the runtime of the solver components in Figure 4.2.

Observations

The most important observation is that our solver converges quickly to accurate solutions. We require 14 to 22 Gauss-Newton-Krylov iterations. The number of Gauss-Newton-Krylov and PCG iterations is approximately mesh-independent. The most effective preconditioner is 2LInv$\mathcal{H}_0$. If we compare the runtime for our new version to the results reported in [28], we can observe a speedup of about 50%. The average time-to-solution for clinically relevant problems on a single GPU is ~5 s. We can reduce the runtime on a single GPU to 3.70 s, which corresponds to a speedup of 70% compared to [28] (for `na02`, 256^3) by storing the gradient of the state variable. Storing the gradient of the state variable reduces the runtime by approximately 15% (but increases the memory pressure). We can also observe that we can solve large-scale real-world imaging problems with grid sizes of 1024^3 for the NIREP data and up to 1024×768×768 for the CLARITY data on 8 nodes with 32 GPUs or one 4 nodes

Table 4.5: *Results for the registration of different NIREP and CLARITY datasets. We report results for the different preconditioners Inv$\mathcal{A}$ (**[A]**), Inv$\mathcal{H}_0$ (**[B]**), and 2LInv$\mathcal{H}_0$ (**[C]**). We use a parameter continuation scheme for β with target parameter $\beta = 5\text{E}{-4}$. The number of time steps for the semi-Lagrangian method is $N_t = 4, 8, 16$ for domain sizes $N = 256^3, 512^3, 1024^3$, respectively. All runs use linear IP and FD for 1^{st} order derivatives. For each domain size, we use the minimum number of resources possible, i.e., a single GPU for $N = 256^3$, four GPUs on a single node for $N = 512^3$, 32 GPUs on 8 nodes for $N = 1024^3$. We report from left to right: (**data**) the selected template image, (**PC**) the Hessian preconditioner method, (**GN**) the number of Gauss-Newton iterations, (**PCG**) the number of PCG iterations, (**mism.**) the relative mismatch, ($||\mathbf{g}||_{rel}$) the relative gradient norm, (**[A]**) the number of applications of Inv$\mathcal{A}$, (**[B/C]**) the number of applications of Inv$\mathcal{H}_0$/2LInv$\mathcal{H}_0$ (notice, that we use Inv$\mathcal{A}$ for large values of β in our continuation scheme), (**total**) and (**average**) the number of PCG iterations to invert $\mathbf{H}_0$ (total and average), (**PC**) the overall runtime for preconditioner, (**Obj**) objective function evaluation, (**Grad**) reduced gradient computation, (**Hess**) Hessian matvecs, and (**Total**) the total runtime of the entire solver. All runtimes are in seconds.*

setting		**solver**				**preconditioner**				**runtimes**				
		iterations		relative accuracy		applications		CG steps						
data	PC	GN	PCG	mism.	$\|\|\mathbf{g}\|\|_{\text{rel}}$	A	B|C	total	avg.	PC	Obj	Grad	Hess	Total
NIREP $N = 256^3$, $N_t = 4$, $\epsilon_{\mathcal{H}0}$ =1e-3, 1 node, 1 GPU														
na02	[A]	14	75	2.73E−2	3.09E−2	75	—	—	—	4.43E−1	2.04E−1	4.33E−1	3.82E0	6.19E0
	[B]	14	23	2.62E−2	2.82E−2	3	20	235	11.8	2.45E0	2.04E−1	4.33E−1	1.27E0	5.54E0
	[C]	14	28	2.79E−2	3.23E−2	3	25	294	11.8	1.04E0	2.05E−1	4.35E−1	1.52E0	4.44E0
na03	[A]	17	93	2.55E−2	3.11E−2	93	—	—	—	5.50E−1	2.49E−1	5.24E−1	4.69E0	7.53E0
	[B]	17	36	2.50E−2	3.04E−2	14	22	255	11.6	2.72E0	2.48E−1	5.23E−1	1.91E0	6.80E0
	[C]	17	39	2.56E−2	3.17E−2	14	25	301	12.0	1.11E0	2.49E−1	5.24E−1	2.05E0	5.39E0
na10	[A]	17	94	1.96E−2	2.94E−2	94	—	—	—	5.58E−1	2.50E−1	5.25E−1	4.76E0	7.61E0
	[B]	17	36	1.90E−2	2.81E−2	9	27	299	11.1	3.17E0	2.48E−1	5.25E−1	1.91E0	7.25E0
	[C]	17	38	1.93E−2	2.90E−2	9	29	328	11.3	1.22E0	2.49E−1	5.26E−1	2.01E0	5.45E0
NIREP $N = 512^3$, $N_t = 8$, $\epsilon_{\mathcal{H}0}$ =1e-3, 1 node, 4 GPUs														
na10	[A]	18	107	2.53E−2	3.84E−2	107	—	—	—	5.28E0	1.68E0	3.86E0	3.52E1	5.18E1
	[B]	18	37	2.66E−2	4.38E−2	10	27	307	11.4	2.19E1	1.70E0	3.89E0	1.25E1	4.55E1
	[C]	18	37	2.68E−2	4.39E−2	10	27	309	11.4	5.55E0	1.67E0	3.87E0	1.25E1	2.92E1
NIREP $N = 1024^3$, $N_t = 16$, $\epsilon_{\mathcal{H}0}$ =1e-3, 8 nodes, 32 GPUs														
na10	[A]	21	128	3.19E−2	4.41E−2	128	—	—	—	4.63E1	3.55E0	2.14E1	1.76E2	2.55E2
	[B]	22	59	2.70E−2	3.34E−2	18	41	531	13.0	2.33E2	3.79E0	2.24E1	8.08E1	3.46E2
	[C]	22	59	2.73E−2	3.77E−2	18	41	533	13.0	5.69E1	3.80E0	2.24E1	8.11E1	1.71E2
CLARITY $N = 1024\times384\times384$, $N_t = 4$, $\epsilon_{\mathcal{H}0}$ =1e-2, 1 nodes, 4 GPUs														
	[A]	13	205	2.01E−1	4.23E−2	205	—	—	—	2.12E1	8.78E−1	2.53E0	5.11E1	7.13E1
	[C]	12	75	2.02E−1	4.54E−2	4	71	1007	14.2	1.67E1	8.49E−1	2.34E0	1.89E1	4.36E1
CLARITY $N = 1024\times768\times768$, $N_t = 4$, $\epsilon_{\mathcal{H}0}$ =1e-2, 4 nodes, 16 GPUs														
	[A]	20	663	1.95E−1	5.81E−2	663	—	—	—	1.96E2	4.02E0	1.37E1	5.12E2	7.38E2
	[B]	15	52	2.03E−1	4.38E−2	6	46	648	14.1	2.28E2	1.57E0	1.09E1	4.02E1	2.86E2

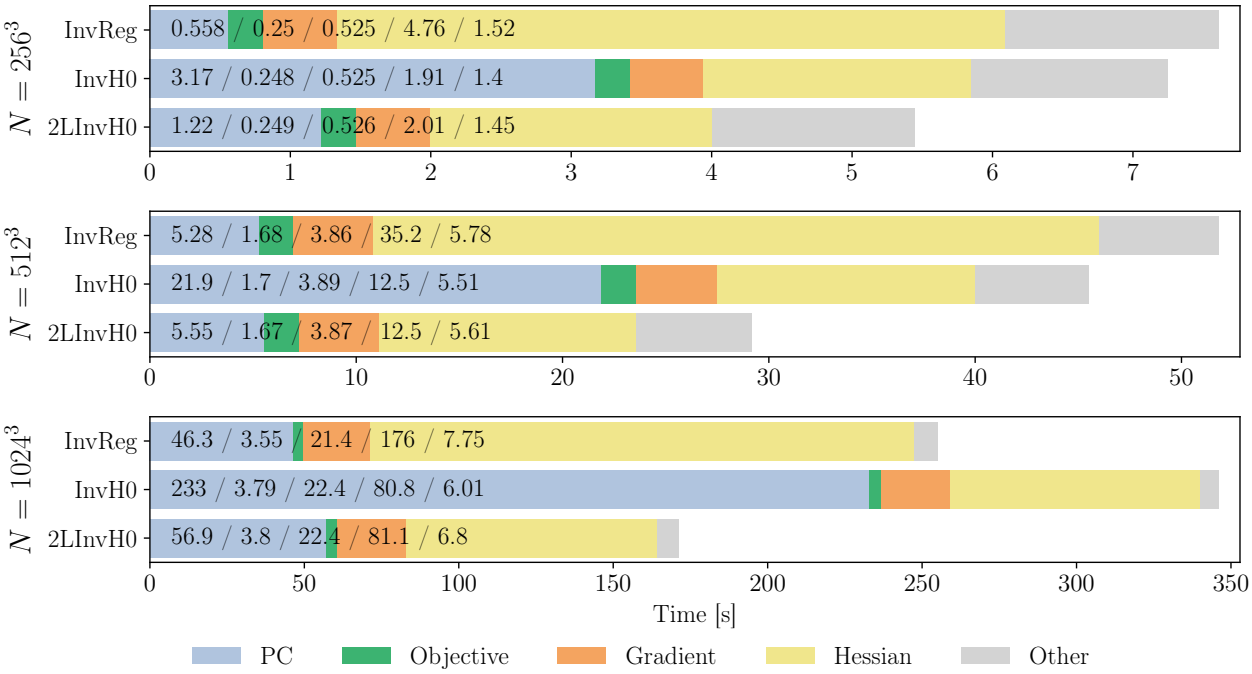


Figure 4.2: *Visualization of the allocated runtime for the results reported in Table 4.5. The color bars (times are in seconds) illustrate the amount of execution time spent in the main mathematical operators of our solver (PC: application of inverse of preconditioner; objective: evaluation of the objective functional; gradient: evaluation of gradient (includes PDE solves for state and adjoint equation); hessian: Hessian matvecs (includes PDE solves for incremental state and adjoint equation). We can observe that we spend a large fraction of our runtime on the computation of the Newton step.*

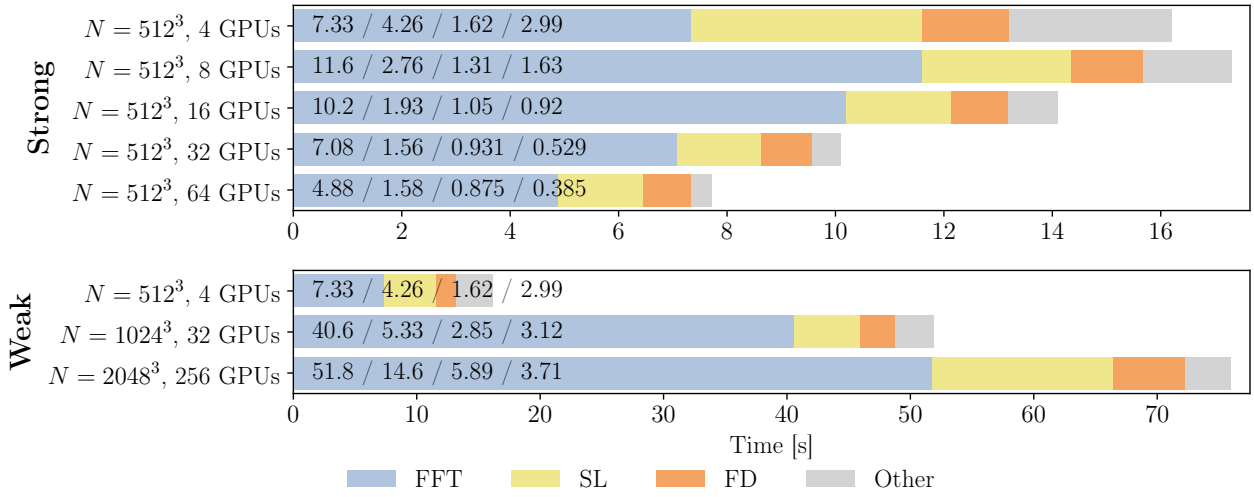


Figure 4.3: *We visualize exemplary strong (top block) and weak (bottom block) scaling results for the experiments reported in Table 4.6. For each run, we report the fraction (color bars and runtime in seconds) spent in the individual kernels. We can see that the runtime is dominated by the FFT kernel. We can also observe that almost the entire runtime of our solver is spent in the three main computational kernels—FFTs, SL, and FD. The scalability of our multi-node multi-GPU implementation is limited due to the high communication costs for small local problem sizes and load imbalance across ranks. We provide a more detailed analysis in the text.*

with 16 GPUs, respectively. In terms of registration quality, we achieve the same accuracy as reported in [28, 108]. These studies also include comparisons to other LDDMM software packages. They demonstrated that their implementation of `CLAIRE` yields results that are significantly more accurate (in terms of data mismatch) than existing methods, and that the single-node GPU version of `CLAIRE` is up to 30× faster than other available single-GPU implementations. With the present work, we are 50× faster on a single GPU.

4.3.3 Strong and Weak Scaling Results

We study weak and strong scaling for our new multi-node multi-GPU implementation. We consider the SYN dataset and use the Inv$\mathcal{A}$ preconditioner for these runs. We fix the number of Gauss-Newton iterations to 5 and the number of PCG iterations per Newton step to 10 to avoid discrepancies arising from the use of relative tolerances.

Results

We present the results in Table 4.6 and report the time-to-solution along with the time spent in individual kernels. We additionally provide the % of the execution time spent for data communication and the total memory consumption per GPU. The strong and weak scaling experiments are restricted by the slab size and available GPU memory, respectively. For the memory restrictions we refer to the analytical estimates given above. Considering the domain decomposition, we cannot use arbitrarily many GPUs per problem size since the slab size (local data volume) per GPU becomes too small for the computations to be efficient. We visualize strong scaling for $N = 512^3$ and weak scaling in Figure 4.3.

Observations

The most important observations are (i) we can solve problems of unprecedented scale (the 1024^3 and the 2048^3 problem can not be solved on a single GPU; the largest problem solved

in [28] is 384^3) and (*ii*) the scalability of our solver suffers from high communication costs for small local problem sizes. In particular, the runtime in FFTs is dominated by communication because of the required all-to-all collective. For a single GPU, we utilize the `cuFFT` 3D FFT and need no additional memory transfers. For small problem sizes (e.g., 128^3 or 256^3), the additional communication costs for strong scaling cannot be compensated by the reduced computations per rank. For all FFTs, scaling above a single node (4 ranks) increases the runtime due to off-node communication, which is the limiting factor. In Table 4.1, we considered GPU-TXTLAG to test the scalability of the semi-Lagrangian method. However, here we use GPU-TXTLIN, which has much lower computational complexity. This results in an increased percentage of communication in the overall runtime, and as we reduce the local problem size (slab width <16 voxels), this effect is further amplified. At this slab size, the communication of the query points can become non-uniform (subject to local variations in length of the characteristics). This can cause a significant load imbalance among MPI ranks and by that negatively affects the scaling because of the implicit synchronization for the next communication step (which is ghost layer sharing). The scaling performance of the FD kernel is consistent with the results in Table 4.2. For weak scaling, when switching from 512^3 on 4 GPUs to 2048^3 on 256 GPUs, the communication time increases by ~4x; the kernel execution time stays roughly the same. For the strong scaling for resolutions 512^3 and 1024^3, the communication time stays roughly the same while the kernel execution time reduces by ~2x. However, the overall time spent in FD does not scale well because of GPU memory constraints, as explained in §4.2.2.

4.4 Chapter conclusions

In this chapter, we presented a novel multi-node multi-GPU implementation for diffeomorphic registration. Our work extends the publicly available software package `CLAIRE`. `CLAIRE` relies on three main computational kernels: FFTs and FD kernels for differentiation and the

Table 4.6: *Strong and weak scaling results for* `CLAIRE` *using synthetic data. The number of Gauss–Newton iterations is fixed to 5 and we use 10 PCG steps per Gauss-Newton iteration. We consider Inv$\mathcal{A}$ as a preconditioner. We report runtimes in seconds and total memory consumtion in GB per GPU. We use a fixed regularization of $\beta = 1\text{E}{-3}$ and set $N_t = 4$ with a linear interpolation (IP) model for the semi-Lagrangian (SL) method. All 1^{st} order derivatives are computed with FDs. The parallel layout (number of GPUs) for our experiments is restricted by the local slab size (can become too small) and the available GPU memory, respectively. The 2048^3 is the largest problem we could fit on TACC's Longhorn system. We cannot use less resources for this problem due to memory restrictions.*

nodes	**#GPUs**	**FFT**		**SL**		**FD**		**overall**		
		time	% comm.	time	% comm.	time	% comm	time	% comm.	memory
$N = 128^3$										
1	1	1.03E−1	0.0	1.82E−1	0.0	6.12E−2	0.0	5.11E−1	0.0	1.11
1	2	1.74E−1	44.5	3.88E−1	69.3	1.52E−1	54.3	8.37E−1	51.3	0.95
1	4	2.35E−1	59.8	4.13E−1	76.4	1.44E−1	62.0	9.17E−1	59.5	0.79
2	8	6.95E−1	85.5	5.56E−1	83.9	2.87E−1	84.4	1.66E0	78.4	0.71
4	16	5.38E−1	90.0	6.19E−1	85.5	5.72E−1	92.1	1.87E0	82.3	0.66
$N = 256^3$										
1	1	7.74E−1	0.0	1.16E0	0.0	3.72E−1	0.0	3.32E0	0.0	5.09
1	2	7.47E−1	42.3	1.20E0	61.0	4.64E−1	34.1	2.99E0	40.5	3.18
1	4	9.84E−1	74.7	8.20E−1	66.5	3.20E−1	45.4	2.56E0	55.6	1.95
2	8	1.69E0	89.2	1.23E0	85.2	3.90E−1	71.8	3.60E0	78.9	1.29
4	16	1.96E0	91.8	1.26E0	89.4	3.70E−1	79.6	3.81E0	84.5	0.94
8	32	1.36E0	95.3	1.24E0	91.4	3.59E−1	84.0	3.15E0	86.8	0.78
$N = 512^3$										
1	4	7.33E0	74.0	4.26E0	60.6	1.62E0	32.2	1.62E1	52.5	11.2
2	8	1.16E1	90.0	2.76E0	68.0	1.31E0	56.4	1.73E1	75.5	5.84
4	16	1.02E1	94.5	1.93E0	74.5	1.05E0	70.3	1.41E1	83.9	3.32
8	32	7.08E0	94.3	1.56E0	81.3	9.31E−1	80.4	1.01E1	85.9	2.00
16	64	4.88E0	96.8	1.58E0	87.9	8.75E−1	86.9	7.72E0	89.1	1.31
$N = 1024^3$										
8	32	4.06E1	95.0	5.33E0	73.4	2.85E0	69.6	5.19E1	85.7	11.5
16	64	2.44E1	95.0	4.17E0	81.9	2.48E0	81.4	3.27E1	87.4	6.23
32	128	1.47E1	96.9	3.94E0	89.2	2.20E0	88.2	2.18E1	90.2	3.43
64	256	1.00E1	97.5	6.64E0	96.2	2.04E0	92.3	1.95E1	92.9	2.12
$N = 2048^3$										
64	256	5.18E1	93.1	1.46E1	92.4	5.89E0	88.5	7.60E1	88.1	12.5

evaluation of IP kernels in a semi-Lagrangian solver for the solution of transport equations. Our approach to port these kernels to a multi-GPU environment is highly adapted to the target architecture in various ways: (*i*) We replace FFT-based (spectral) first-order derivative evaluations used in `CLAIRE` with an 8^{th} order FD scheme for the multi-GPU version. This yields a scheme that is more accurate (for the considered resolutions and precision; not asymptotically) and, at the same time, requires substantially less communication. (*ii*) We choose texture-based Lagrange polynomial third order IP over spline IP (which had been shown to be superior on a single GPU [28]) to further reduce the communication between GPUs. (*iii*) We propose an efficient combination of `cuFFT` within nodes and a 2D slab decomposition approach across nodes, combined with an in-house developed, optimized all-to-all communication for regimes for which we could show that the available vendor MPI all-to-all [2] was sub-optimal. In addition to these kernel optimizations, we are able to substantially reduce the number of PCG iterations for computing the search direction within a Gauss–Newton–Krylov scheme and, thus, reduce the runtime by a factor of up to 2.5 compared to the prior version of `CLAIRE`. This is achieved through a new two-level (coarse grid) preconditioner based on a zero-velocity approximation of the Hessian operator, which eliminates expensive PDE solves. The entire solver is matrix-free. We optimized the memory footprint of the proposed solver. This allows us to solve larger problems on a single GPU, and to tackle problems of unprecedented scale. We ported `CLAIRE` to multi-GPU architectures as a whole, and support direct GPU-GPU communication through CUDA-aware MPI; no explicit host-to-device communication is required. The largest run reported in this study is 152× larger than the results reported for the state-of-the-art [28]. Combining all improvements, we achieved a speedup of up to 70% compared to [28] on a single GPU. To showcase the capabilities of the proposed methodology, we reported results for the registration of real imaging data for resolutions of up to 1024^3 for MR neuroimaging data (on 8 nodes with a total of 32 GPUs) and 1024×768×768 for CLARITY imaging data (on 4 nodes with a total of 16 GPUs). The achieved accuracy is equivalent to the results provided in prior work on

CLAIRE [28, 65, 105, 106, 108], and on par or superior to other state-of-the-art software for diffeomorphic registration (see [28, 108] for a comparison).

Our work applies to other transport dominated forward and inverse problems. For example, the semi-Lagrangian GPU algorithm applies to particle-in-cell and weather/climate codes. The code basis of our solver (optimization scheme, linear algebra solvers, and preconditioning) are hardware agnostic. Our three main computational kernels should translate to other GPU accelerators as long as they provide some specialized hardware support. For example, the IP kernel relies on texture memory, which needs to be supported by the hardware. Also, certain parameters will need to be retuned. Most of the kernels are written in CUDA, so—although the algorithms won't change—the implementation will have to be ported to the new GPU programming interface.

Chapter 5

Scalable image registration applications

***Remark**: The work in this chapter was done in collaboration with Malte Brunn, Miriam Mehl and Andreas Mang. The author of this dissertation contributed to model and software development, all numerical experiments on scalable image registration and analysis of results.*

In the previous chapter, we extended `CLAIRE` to support scalable image registration which can process high resolution images using multiple GPUs. We demonstrated the scalability of our solver using synthetic images with a resolution up to 2048^3 and CLARITY mouse brain images of size $768 \times 768 \times 1024$. In this chapter, we scale registration to even higher resolutions, e.g., CLARITY images of size $2816 \times 3016 \times 1162$. In our previous works [65, 102–104, 106–109], we have extensively studied the algorithmic side of image registration within the framework of `CLAIRE`. In this chapter, we pay closer attention to the quality of the registration results. We study the effect of different input parameters, including the quality and resolution of the input images, on the accuracy of the registration.

Contributions

The contributions of this chapter are as follows:

- We introduce an improved regularization parameter search and continuation scheme to automatically search for optimal regularization parameters in our solver driven by

user constraint on the properties of the registration deformation.

- We demonstrate the need for performing image registration in high resolution in order to be able to morph fine structures in the image to improve performance metrics such as the Dice coefficient. We conduct experiments to quantify the loss in registration accuracy as image resolution is decreased. Our experiments on synthetic and real imaging data demonstrate an improved registration quality as the resolution increases.

- We study the performance of our scalable image registration solver `CLAIRE` for applications in high resolution mouse brain image registration and human brain MRI registration. We perform image registration for two pairs of CLARITY mouse brain images at a resolution of $2816 \times 3016 \times 1162$ voxels. To the best of our knowledge, images of this scale have not been registered before at full resolution in under 30 min.

Related Work

The work in [93] focuses on annotating CLARITY brain images by registering them to the Allen Institute's Mouse Reference Atlas (**ARA**). They use a "masked" LDDMM approach. They also consider the registration of CLARITY-to-CLARITY brain images and compare different mismatch terms for the registrations. However, they downscale the images to a lower resolution for conducting all experiments. In [94], mutual information is used for the registration of CLARITY to the ARA dataset but at an approximately one hundred times downsampled resolution (at an original in-plane isotropic resolution 0.58 μm). The authors in [116] analyze registration performance on high resolution mouse brain images of size $2560 \times 2160 \times 633$ obtained using the CUBIC protocol [150]. They report results using different software packages including `ANTs` and `elastix`. No relationship between registration accuracy at different resolutions was reported. For their largest runs using `ANTs`, they report a wall clock time of over 200 hrs on a single compute node (2.66GHz 64bit Intel Xeon processor with 256GB RAM) while the same run with `elastix` [89] took

approximately 30hrs. The authors in [119] register high resolution images of mouse brain acquired using different imaging techniques to the ARA dataset [92]. They perform non-linear registration using `ANTs` at coarse resolution ($10\mu m$ for the ARA) and apply deformation at high resolution. In the current work, we do not downsample high resolution images but register them at the original resolution. We are able to register CLARITY images of resolution $2816 \times 3016 \times 1162$ in less then 30min using 256 GPUs. We use the Message Passing Interface (**MPI**) to parallelise our implementation. In addition to that, we study the effect of resolution on the registration quality.

Outline

Unless otherwise noted, the overall formulation and the algorithms in this chapter remain unaltered from §2.1.1. In §5.1, we discuss key solver parameters for `CLAIRE` in the context of scalable image registration and introduce a new scheme to automatically identify adequate parameters of our scheme for unseen data. We conclude with the main scalability experiments in §5.2, and present conclusions in §5.3.

5.1 Methods

The formulation and algorithms in this chapter remain unaltered from §2.1.1[9]

5.1.1 Key Solver Parameters

Here, we summarize the key parameters of `CLAIRE` and discuss their effect on the solver and their computational costs.

- β_v — **The regularization parameter for the velocity field $\boldsymbol{v}$.** Large values for β_v result in very smooth maps that are typically associated with a large image mismatch.

[9]For all the experiments performed in this chapter, we use 2LInv$\mathcal{H}_0$ (see §4.1) as the default preconditioner.

Smaller values of β_v allow complex deformations but lead the solution close to being non-diffeomorphic due to discretization issues. From a practical point of view, we are interested in computing velocities for which the determinant of the deformation map does not change sign/is strictly positive for every point inside the domain. This guarantees that the transformation is locally diffeomorphic (subject to numerical accuracy). Following [74, 102], we determine the regularization parameter β_v based on a binary search. We control the search based on a bound for the determinant of the deformation gradient. That is, we choose β_v so that the determinant of the deformation gradient is bounded below by J_{min} where $J := \det \mathbf{F}$ and bounded above by $1/J_{min}$, where $J_{min} \in (0, 1)$ is a user defined parameter. This search is expensive, since it requires a repeated solution of the inverse problem. (For each trial β_v we iterate until we meet the convergence criteria for our Gauss–Newton–Krylov solver and then use the obtained velocity as an initial guess for the next β_v.

- β_w — **The regularization parameter for the divergence of the velocity field** $w = \nabla \cdot \boldsymbol{v}$. The choice of β_w, along with β_v, is equally critical. A large value of β_w will introduce large shear deformations while keeping J close to unity and small values can result in extreme values of J and make the deformations locally non-diffeomorphic. As discussed above, in our previous work [102], we do parameter continuation in β_v and keep β_w fixed. This is sub-optimal for two reasons: *(i)* Both β_v and β_w are dependent on the resolution, so keeping β_w fixed for all resolutions can result in deformations with undesirable properties and *(ii)* Doing continuation in β_v alone does not ensure we get close enough to the set Jacobian bounds and adding continuation in β_w which also affects the Jacobian is necessary. Here we extend the continuation framework to β_w in §5.1.2.

- J_{min} — **Lower bound for the determinant of deformation gradient** J. The choice of this parameter is typically driven by dataset requirements, i.e., one has to

decide on how much volume change is acceptable. `CLAIRE` uses a default value of 0.25 [108]. Tighter bound on the Jacobian, i.e., J_{min} close to unity, will result in large β_v and β_w values leading to simple deformations and sub-par registration quality. Whereas relaxing the Jacobian bound significantly can result in very small regularization parameters and extremely complex deformations.

- n_t — **Number of time steps in the semi-Lagrangian scheme**. The semi-Lagrangian scheme is unconditionally stable and outperforms RK2 and RK4 time integration schemes in terms of runtime for a given accuracy tolerance [104]. The choice of n_t is determined based on the adjoint error, which is the error incurred in solving (2.1c) forward and then backward in time. In [104], we conducted detailed experiments for 2D image registration to find out that even for problems of clinical resolution $n_x = 256^2$, $n_t = 3$ (CFL=10), did not cause issues in solver convergence. Increasing n_t beyond a certain value will introduce artificial smoothing errors from the interpolation scheme. For our GPU implementation, which is only available in single precision (unlike the CPU implementation [108], which is available both in single and double precision), we recommend using cubic interpolation (B-splines/Lagrange polynomials) with $n_t = 4$ ($n_t = 8$ for linear interpolation) for resolutions up to $n_x = 256^3$. For higher resolutions, we use (and recommend) linear interpolation to save on computational costs and increase n_t proportionately to n_x to keep the CFL number fixed.

- **Discretization of $\boldsymbol{v}$**. We use the same spatial discretization as the input images. There exist image registration algorithms which approximate the registration deformation in a low-dimensional bandlimited space without sacrificing accuracy, resulting in dramatic savings in computational cost [170]. We have not explored this within the framework of `CLAIRE`. Notice that [170] uses higher order regularization operators, which leads to smoother velocities compared to the ones `CLAIRE` produces, therefore enabling a representation on a coarser mesh. Moreover, `CLAIRE` uses a stationary veloc-

ity field, i.e., $\boldsymbol{v}$ is constant in time. In our previous work [102], we have demonstrated that for registration between two real medical images of different subjects, stationary and time-varying velocity fields yield similar registration accuracy. More precisely, we did not observe any practically significant quantitative differences in registration accuracy for a varying number of coefficient fields in the case of time-varying velocity fields. From a computational cost perspective, using stationary velocity is significantly cheaper and has a smaller memory overhead.

5.1.2 Parameter Search Scheme

Our algorithm proceeds as follows. In the first part of the parameter search we fix β_w=$\beta_{w,init}$ ($\beta_{w,init}$ = 1e-05) and search for β_v. The registration problem is solved for a large value of β_v=$\beta_{v,init}$ so that we under-fit the data. In our experiments, we set $\beta_{v,init} = 1$. Subsequently β_v is reduced by one order of magnitude and registration problem is solved until we breach the Jacobian bounds [J_{min}, $1/J_{min}$]. When this happens, we do a binary search for β_v and we terminate the binary search when the relative change in β_v is less than 10%. We put a lower bound $\beta_{v,min}$ = 1e-05 on β_v. This lower bound is set purely to minimize computational costs. We denote the estimated value of β_v as β_v^*. In the second of the search, we do a simple reduction search for β_w by fixing β_v=β_v^* i.e. we reduce β_w by one order of magnitude until we reach J_{min}. We put a lower bound $\beta_{w,min}$ = 1e-07 on β_w in order to minimize computational costs. We take the last valid value of β_w for which the Jacobian determinant was within bounds and denote it as β_w^*. We fixed the value of $\beta_{w,init}$ = 1e-05 for all experiments and resolutions. We determine this value empirically by running image registration on a couple of image pairs at resolution $640\times880\times880$ and $160\times220\times220$ (see §5.2.4 for the images) for different values of $\beta_{w,init}$. We report these runs in Table 5.7. If we want to use specific β_v and β_w values for a registration problem, we perform parameter continuation which is exactly like the parameter search except that we do not perform binary search for β_v. This scenario appears when we do a cohort study and we use a couple of registrations to determine the

regularization parameters.

We evaluate the parameter search scheme for a real world brain images and report the performance in §5.2.2. Furthermore, we use it as the default parameter search scheme for all the experiments presented in this chapter.

5.2 Results

We test the image registration on real-world (see §5.2.4 and §5.2.5) and synthetic registration problems (see §5.2.3). The measures to analyze the registration performance are summarized in §5.2.1. We evaluate the parameter search scheme (see §5.1.2) on a set of real brain images and present the results in §5.2.2. Furthermore, we explore the following questions in the context of scalable image registration:

- **Question Q1:** Do the registration quality degrade when the registration is performed at a downsampled resolution when compared to performing registration at the original high resolution?
- **Question Q2:** How does registration perform for real, noisy and high resolution medical images of human and mouse brains?

We design experiments using real and synthetic image registrations to answer these two questions.

5.2.1 Measures of Performance

In our experiments, we evaluate the registration accuracy using one or both of the following measures:

1. **Dice Score Coefficient** D**:** Let l_0 and l_1 be the binary label maps associated with the images m_0 and m_1, respectively. Then the Dice score D between the two is given

by

$$D(l_0, l_1) = \frac{2|l_0 \cap l_1|}{|l_0| + |l_1|}, \tag{5.1}$$

where $|\cdot|$ denotes the cardinality of a set, and $\cap$ and $\cup$ are the intersection and union of two sets, respectively. We define $D(l_0, l_1)$ to be the Dice score pre-registration and $D(l(t=1), l_1)$ post-registration, where $l(t=1)$ is the label map that corresponds to the deformed template image $m(t=1)$. Furthermore, for a set of discrete labels l^i, $i = \{1, 2, \ldots, M\}$ where i corresponds to the label index, we define the volume fraction

$$\alpha^i = \frac{|l^i|}{\sum_{i=1}^{M} |l^i|}. \tag{5.2}$$

Using this definition, we compute the following statistics for the Dice coefficient: The Dice coefficient average D_a given by

$$D_a = \frac{1}{M} \sum_{i=1}^{M} D(l_0^i, l_1^i), \tag{5.3}$$

the volume weighted average of the Dice coefficient given by

$$D_{vw} = \frac{1}{\sum_{i=1}^{M} |l_1^i|} \sum_{i=1}^{M} |l_1^i| D(l_0^i, l_1^i), \tag{5.4}$$

and the inverse of the volume weighted average Dice coefficient given by

$$D_{ivw} = \frac{1}{\sum_{i=1}^{M} 1/|l_1^i|} \sum_{i=1}^{M} \frac{D(l_0^i, l_1^i)}{|l_1^i|} \tag{5.5}$$

Note that D_{vw} gives more weight to labels with higher volume fractions while D_{ivw} gives more weight to labels with smaller volume fractions.

2. **Relative Residual r:** This measure corresponds to the ratio of the image mismatch

before and after the registration. It is given by

$$r = \frac{||m(t=1) - m_1||_2^2}{||m_0 - m_1||_2^2}. \tag{5.6}$$

Along with registration accuracy, we also report the solver wall clock time as a quantitative measure of the computational performance. For each image registration, we also report the regularization parameters and the obtained minimum and maximum values of the determinant of the deformation gradient $J := \det \mathbf{F}$, i.e., the determinant of the Jacobian of the deformation map. We visually support this quantitative analysis with snapshots of the registration results. The accuracy of the registration can be visually judged from the residual image, which corresponds to the absolute value of the pointwise difference between $m(t = 1)$ and m_1. The regularity of the deformations can be assessed from the pointwise maps of the determinant of the deformation gradient.

5.2.2 Parameter Search Scheme

Aim. To evaluate the parameter search scheme on a set of real brain images and compare the registration performance with state-of-the-art `SyN` deformable registration tool in the `ANTs` toolkit.

Dataset. We use five real brain T_1-weighted MRI datasets. These images have been segmented into 149 functional brain regions using the MUSE algorithm [49]. We use these labels to evaluate registration performance in terms of volume weighted average dice score D_{vw}.

Procedure. Out of the five images, we select one image `Template27` as the reference image m_1 and register the other four images to this reference image. For the registration, we use the proposed parameter search scheme (see §5.1.2). For the Jacobian bound, we select $J_{min} = 0.1$. In the parameter search, for each trial β_v and β_w, we drive the relative gradient

Table 5.1: ***Performance of the parameter search scheme implemented in*** `CLAIRE`. *We report results for the registration of four template images to the reference image* `Template27`. *We consider the squared L_2-distance measure as image similarity. We restrict the Jacobian determinant $J \in [0.1, 10]$ for these registrations. We report the following quantities of interest: (i) optimal regularization parameters β_v^* and β_w^*, (ii) minimum J_{min} and maximum J_{max} Jacobian determinant achieved, (iii) solver wall clock time in seconds, and (iv) label volume weighted Dice average D_{vw} pre and post registration.*

Template	β_v^*	β_w^*	J_{min}	J_{max}	D_{vw}		runtime(s)	
					pre	post	search	continuation
4	7.75e-05	1.00e-04	4.53e-01	5.36e+00	5.52e-01	6.99e-01	5.90e+02	4.04e+01
16	7.89e-05	1.00e-05	2.62e-01	4.23e+00	5.50e-01	6.95e-01	4.39e+02	5.82e+01
22	1.14e-05	1.00e-04	1.19e-01	1.74e+00	5.39e-01	7.04e-01	7.05e+02	9.79e+01
31	2.83e-05	1.00e-04	2.40e-01	1.86e+00	5.26e-01	7.00e-01	6.19e+02	6.07e+01

norm $\|g\|_{rel} = \|g\|_2/\|g_0\|_2$ to 1e-02. We use linear interpolation and $n_t = 8$ time steps in the semi-Lagrangian solver. Once we have searched for adequate β_v and β_w for each image pair, we rerun the image registrations using only parameter continuation. (Note that the parameter search allows us to identify an optimal set of regularization parameters for unseen data; as we have noted before, this is expensive since it requires a repeated solution of the optimization problem, similar to identifying the regularization parameter using an L-curve. Once we have identified optimal regularization parameters (for example, for one individual image within a cohort study) we can execute the solver using a parameter continuation scheme. This continuation scheme convexifies the problems and speeds up convergence. For a baseline performance comparison, we also perform registration on the same image pairs using the `SyN` tool in `ANTs` [12]. For `ANTs`, we consider the "MeanSquares" (i.e., squared L_2-) distance measure. We run the parameter search using `CLAIRE` on a single NVIDIA V100 GPU with 16GB of memory on TACC's Longhorn supercomputer. We run `ANTs` on a single node of the TACC Frontera supercomputer (system specs: Intel Xeon Platinum 8280 ("Cascade Lake") processor with 56 cores on 2 sockets (base clock rate: 2.7GHz)). We use all 56 cores. We report the parameters used for `ANTs` in §**??**.

Results. We report the obtained estimates for β_v and β_w as well as results for registration quality in Table 5.1. In Figure 5.2, we provide a representative illustration of the obtained registration results. We report baseline registration performance using `ANTs`

Table 5.2: ***Performance of*** `ANTs`. *We report results for registration of four template images to the reference image* `Template27` *using a squared L_2-distance metric. We report the following quantities of interest (i) minimum (J_{min}) and maximum (J_{max}) determinant of the deformation gradient obtained, (ii) label volume weighted Dice average D_{vw} pre and post registration, and (iii) solver wall clock time in seconds.*

Template	J_{min}	J_{max}	D_{vw}		runtime(s)
			pre	post	
4	1.40e-01	3.10e+0	5.53e-01	6.86e-01	1.98e+02
16	2.50e-01	4.59e+0	5.51e-01	6.87e-01	2.00e+02
22	3.11e-01	9.73e+0	5.39e-01	6.62e-01	1.99e+02
31	2.07e-01	4.76e+0	5.27e-01	6.85e-01	2.10e+02

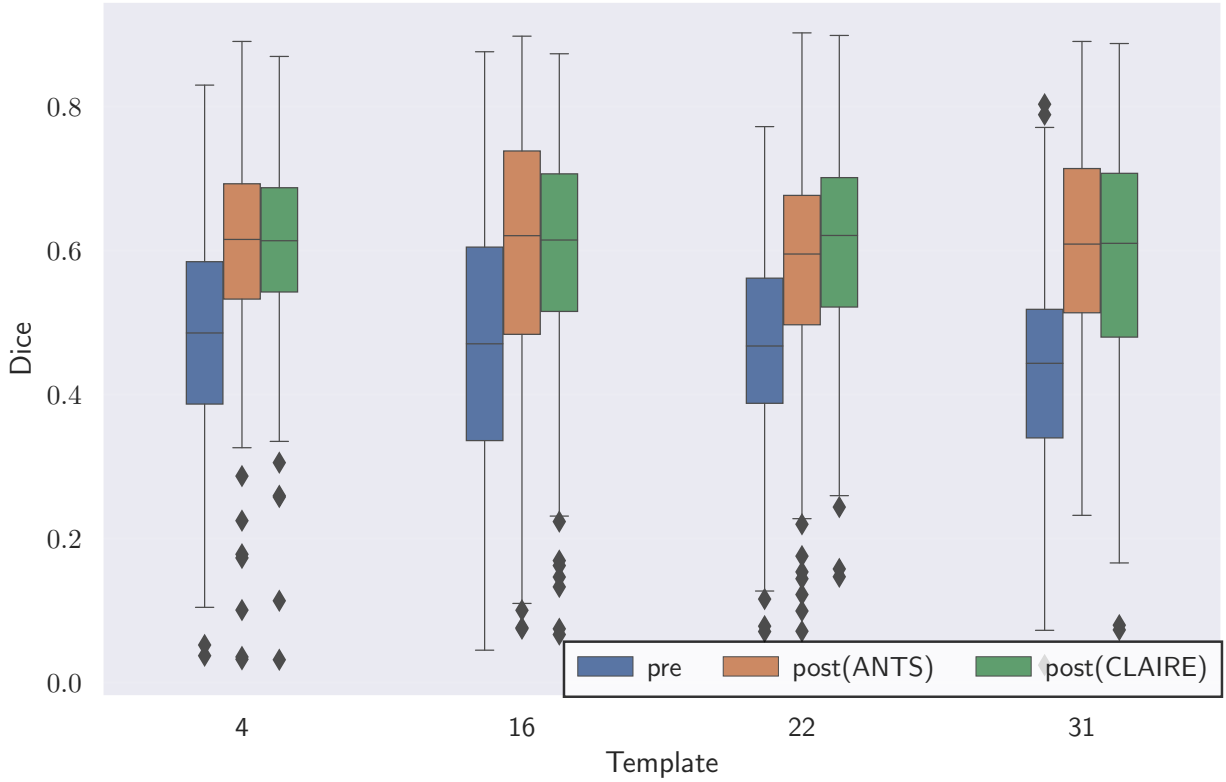


Figure 5.1: ***Comparison of Dice scores for*** `CLAIRE` ***and*** `ANTs`. *Box plots for Dice scores of the individual labels for the registration results reported for* `CLAIRE` *in Table 5.1 and* `ANTs` *in Table 5.2. The accuracy of* `CLAIRE` *is competitive with* `ANTs`.

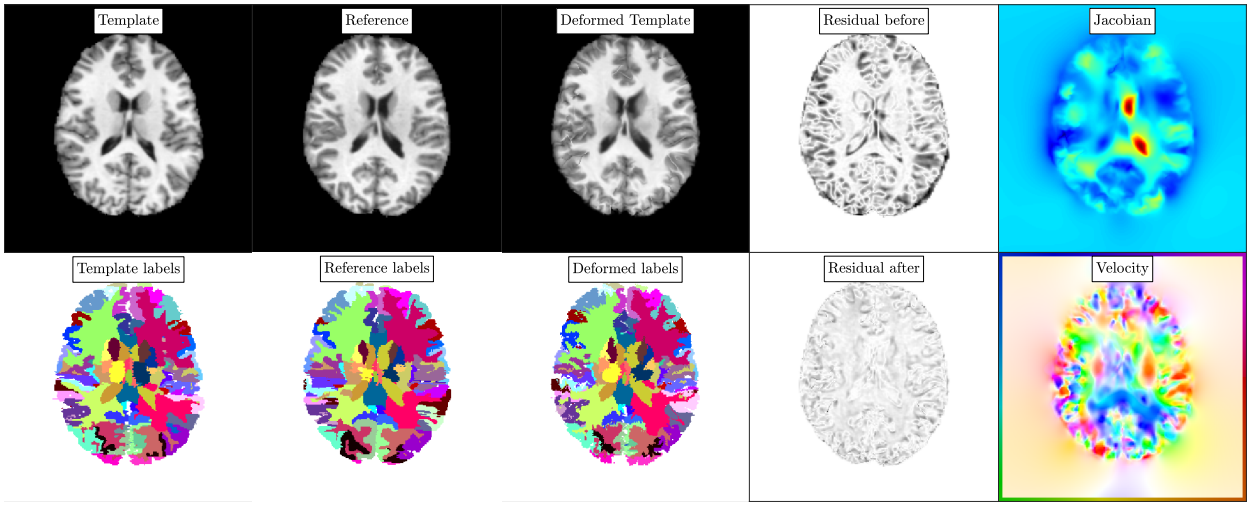


Figure 5.2: ***Exemplary regisration results using parameter search scheme implemented in*** `CLAIRE`. *We report representative registration results obtained for* `CLAIRE` *using the proposed parameter search scheme. We consider the datasets* `Template16` *(template image) and* `Template27` *(reference image). We refer to Table 5.1 and the text for details about the setup. We show (from left to right) the template, reference, deformed template image (top row) and their corresponding labels (bottom row). We also visualize the residual before and after the registration along with the determinant of the deformation gradient and an orientation map for the velocity field.*

in Table 5.2. We compare the Dice scores obtained for `CLAIRE` and `ANTs` in Figure 5.1.

Observations. `CLAIRE` allows us to precisely control the properties of the deformation without having to manually tune any parameters. The only free parameters are the Jacobian bounds, which depend on the overall workflow related to the dataset. The volume weighted Dice scores D_{vw} obtained for `CLAIRE` (see Table 5.1) are competitive to those produced by `ANTs` (see Table 5.2). The average runtime for ANTs for all the registrations reported in Table 5.2 is 201 seconds ($\approx$ 3 minutes). For `CLAIRE`, the average wall clock time in the parameter search mode is 9.8 minutes (3$\times$ slower than `ANTs`; we search for adequate regularization parameters), while in the continuation mode the runtime is 64 seconds (3$\times$ faster than `ANTs`; we apply the optimal regularization parameter and do not search for it).

5.2.3 Experiment 1A: High Resolution Synthetic Data Registration

Aim. In this experiment, we answer **Q1**. We attempt this by executing our registration algorithm on synthetic imaging data. The advantages of using such images over real datasets are as follows:

- They are noise-free, high contrast, and sharp unlike real-world images.
- There is a lack of high resolution real data because they are expensive and time-consuming to acquire. Using synthetic data we can control the resolution because the images are created using analytically known functions.
- We can control the number of discrete image intensity levels, i.e., labels. Because these labels are available as ground truth, we can use them to precisely quantify registration accuracy through the Dice coefficient, avoiding inter- and intra-observer variabilities and other issues associated with establishing ground truth labels in real imaging data.
- We create the synthetic reference image m_1 by solving (2.1c) using a given synthetic template m_0 and synthetic velocity field $\boldsymbol{v}$. By controlling the smoothness of $\boldsymbol{v}$, we can create different target images m_1 and therefore control the registration difficulty level.

By performing image registration at different resolutions, we want to check whether the registration at higher resolutions is more accurate than performing the registration at a lower resolution. In the synthetic examples which follow, we quantify the accuracy using the Dice coefficient for discrete labels before and after the registration, and compare their statistics for different resolutions. We note that we perform a search for an optimal regularization parameter for each individual dataset because *we want to obtain the best result for each pair of images*. In practical applications this is not necessary (see comments below; we also refer to [108] for a discussion).

Dataset. We create a synthetic dataset **SYN** using a linear combination of high-frequency spherical harmonics. To be precise, we define the template image $m_0(\boldsymbol{x})$ as

$$m_0(\boldsymbol{x}) = \sum_{i=1}^{10} g_i(\boldsymbol{x}), \tag{5.7a}$$

with

$$g_i(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \|\boldsymbol{x} - \hat{\boldsymbol{x}}_i\|_2 \leq |Y_l^m(\theta + \hat{\theta}_i, \phi + \hat{\phi}_i)| \\ 0, & \text{otherwise} \end{cases}, \tag{5.7b}$$

and image coordinates $\boldsymbol{x} := (x, y, z) \in (-\pi, \pi]^3$. In (5.7b) Y_l^m represents spherical harmonics of the form

$$Y_l^m(\theta, \phi) = \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}} e^{im\theta} P_l^m(\cos(\phi)) \tag{5.7c}$$

with parameters m, l, angular directions $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi]$, and associated Legendre functions P_l^m. We choose $m = 6$, $l = 8$ for our setup. $\hat{\theta}_i$ and $\hat{\phi}_i$ are random perturbations in integer multiples of $\pi/2$ and $\hat{\boldsymbol{x}}_i \in [-0.4\pi, 0.4\pi]^3$ is a random offset from the origin. The reference image $m_1(\boldsymbol{x})$ is computed by solving (2.1c) with initial condition $m_0(\boldsymbol{x})$ and velocity field $\boldsymbol{v}(\boldsymbol{x}) := (v_x(\boldsymbol{x}), v_y(\boldsymbol{x}), v_z(\boldsymbol{x}))$, $\boldsymbol{x} = (x, y, z)$, defined as

$$v_x = \sum_{k=1}^{K} \frac{1}{k^{0.5}} \cos(ky)\cos(kx), \tag{5.8a}$$

$$v_y = \sum_{k=1}^{K} \frac{1}{k^{0.5}} \sin(kz)\sin(ky), \tag{5.8b}$$

$$v_z = \sum_{k=1}^{K} \frac{1}{k^{0.5}} \cos(kx)\cos(kz), \tag{5.8c}$$

where $K = \{4, 8, 12, 16\}$. We set the template and reference base image size to $n_x = n$ where $n = (1024, 1024, 1024)$. It is important to note that m_0 and m_1 possess the discrete labels l_0^i and l_1^i, $i \in \{1, 2, \ldots, 10\}$ i.e. labels l_0^i and l_1^i have intensity i.

Procedure. We take the following steps:

1. First, we register the template image m_0 to the reference image m_1 at the base resolution n to get the velocity field v_n.

2. We transport m_0 using the velocity v_n to get the deformed template image $m(t=1)$ by solving (2.1c). Then, we compute the Dice score between $l^i(t=1)$ and l_1^i, $i \in 1, \ldots, 10$ which are discrete labels for $m(t=1)$ and m_1 respectively using (5.1).

3. Then, we downsample m_0 and m_1 using nearest neighbor interpolation to half the base resolution (for example, $n/2 = (512, 512, 512)$[10]) and register the downsampled images to get velocity $\hat{v}_{n/2}$. We upsample $\hat{v}_{n/2}$ to the base resolution n using spectral prolongation and call it $v_{n/2}$.

4. We transport m_0 using $v_{n/2}$ by solving (2.1c) to get the deformed template image $m(t=1)$ and then compute the Dice score for this new deformed template image.

5. We repeat steps 3 and 4 for resolutions $n/4$ and $n/8$ and compute their corresponding Dice scores.

For the registration, we fix the determinant of the deformation gradient to be within [5e-02, 20] and search for the regularization parameters using the proposed parameter search scheme as described above in §5.1. Another hyperparameter in our registration solver is the number of time steps n_t for the semi-Lagrangian (**SL**) scheme. We consider two cases for selecting n_t:

1. n_t **changes with resolution:** We use $n_t = 4$ time steps for the coarsest resolution $n_x = n/8$ and double n_t when we double the resolution isotropically i.e. double the number of image voxels in each dimension, in order to keep the CFL number fixed.

[10] We treat $n_x = (N_1, N_2, N_3)$ as a tuple. When we say $n_x/2$, we mean $n_x/2 = (N_1/2, N_2/2, N_3/2)$. One some occasions, for simplicity, we might denote for example $n_x = (1024, 1024, 1024)$ as $N = 1024^3$. Both have the same meaning.

Table 5.3: ***Registration performance for CLAIRE for experiment 1A, case 1 (n_t changes with resolution).*** *Comparison of registration accuracy based on the Dice score at different resolutions for the synthetic dataset* ***SYN*** *when n_t is changed along with the resolution of the data (see §5.2.3). K denotes the frequency of the synthetic velocity field in (5.8). $n = (1024, 1024, 1024)$ is the base image resolution. We fix the tolerance for the reduction of the gradient to 5e-02, which we have found to be sufficiently accurate for most image registration problems (see [108]). We use linear interpolation. The Jacobian bounds for the parameter search are $[0.05, 20]$. We increase the number of time steps n_t proportionally to the increase in spatial resolution of the data. We report β_v^* and β_w^* (the optimal regularization parameters obtained with the proposed parameter search scheme), and J_{min} and J_{max} (the minimum and maximum values for the determinant of the deformation gradient). For the Dice score, we report average Dice (D_a), the volume weighted average Dice (D_{vw}), and the inverse volume weighted average Dice (D_{ivw}), pre and post registration. We also report the wall clock time for the parameter search.*

run	K	n_x	n_t	β_v^*	β_w^*	J_{min}	J_{max}	D_a		D_{vw}		D_{ivw}		runtime(s)
								pre	post	pre	post	pre	post	search
#1	4	n	32	1.1e-05	1.0e-07	1.7e-01	7.4e+00	3.1e-01	9.2e-01	5.8e-01	9.8e-01	3.9e-02	8.5e-01	2.9e+03
#2		$n/2$	16	1.1e-05	1.0e-07	1.9e-01	7.7e+00		8.7e-01		9.7e-01		7.0e-01	6.5e+02
#3		$n/4$	8	1.1e-05	1.0e-07	2.6e-01	1.4e+01		7.9e-01		9.5e-01		5.0e-01	1.1e+02
#4		$n/8$	4	1.1e-05	1.0e-06	4.7e-01	5.6e+00		6.7e-01		9.1e-01		1.8e-01	1.5e+01
#5	8	n	32	1.1e-05	1.0e-07	5.1e-02	1.0e+01	3.2e-01	9.0e-01	5.3e-01	9.8e-01	7.4e-02	7.6e-01	2.7e+03
#6		$n/2$	16	1.1e-05	1.0e-07	1.8e-01	1.5e+01		8.5e-01		9.7e-01		6.0e-01	6.2e+02
#7		$n/4$	8	1.1e-05	1.0e-06	3.0e-01	7.8e+00		7.6e-01		9.4e-01		4.1e-01	1.0e+02
#8		$n/8$	4	2.4e-05	1.0e-06	3.8e-01	4.8e+00		6.4e-01		9.0e-01		1.7e-01	1.4e+01
#9	12	n	32	1.1e-05	1.0e-07	1.7e-01	1.2e+01	3.1e-01	9.2e-01	5.2e-01	9.8e-01	9.5e-02	8.5e-01	2.6e+03
#10		$n/2$	16	1.1e-05	1.0e-06	3.1e-01	8.9e+00		8.6e-01		9.7e-01		7.4e-01	5.4e+02
#11		$n/4$	8	1.1e-05	1.0e-06	2.9e-01	1.2e+01		7.5e-01		9.4e-01		4.5e-01	9.4e+01
#12		$n/8$	4	1.1e-05	1.0e-06	4.1e-01	9.9e+00		6.0e-01		8.9e-01		1.9e-01	1.4e+01
#13	16	n	32	1.1e-05	1.0e-07	1.6e-01	9.5e+00	2.9e-01	9.1e-01	5.1e-01	9.8e-01	9.0e-02	8.1e-01	2.4e+03
#14		$n/2$	16	1.1e-05	1.0e-07	1.7e-01	1.4e+01		8.4e-01		9.7e-01		6.0e-01	5.2e+02
#15		$n/4$	8	1.4e-05	1.0e-06	3.0e-01	8.8e+00		7.4e-01		9.4e-01		4.7e-01	9.5e+01
#16		$n/8$	4	2.7e-05	1.0e-06	3.9e-01	1.5e+01		6.1e-01		9.0e-01		2.0e-01	1.5e+01

All other solver parameters, expect for the regularization parameters, are the same at each resolution.

2. n_t **fixed with resolution:** In order to study the effect of n_t on the Dice score we keep n_t fixed for each n_x, instead of increasing n_t proportionately to n_x.

Results. In Figure 5.3, we visualize the template, reference and deformed template images for the synthetic problem constructed with $K = 4$. We report quantitative results for CLAIRE in Table 5.3 and Table 5.4, respectively. In Figure 5.4, we compare the Dice for individual labels as a function of volume fraction α. In Figure 5.5, we visualize box plots of the Dice score for the registrations reported in Table 5.3.

Observations. The most important observations are:

Table 5.4: ***Registration performance for CLAIRE for experiment 1A, case 2 (n_t fixed with resolution).*** *Comparison of registration accuracy using Dice at different resolutions for* ***SYN*** *dataset (see §5.2.3). Synthetic velocity fields of are generated using frequency $K = 8$ and $K = 16$ using (5.8). We fix the tolerance for the relative gradient to $5e-02$ which we have found to be sufficiently accurate for most image registration problems. We use linear interpolation. The Jacobian bounds for parameter search is $[0.05, 20]$. We increase the number of time steps n_t proportionately with increase in resolution. We report β_v^* and β_w^*, the regularization parameters searched by the proposed parameter search scheme, J_{min} and J_{max}, the minimum and maximum Jacobian determinant. For the Dice score, we report average Dice D_a, volume weighted average Dice D_{vw} and the inverse volume weighted average Dice D_{ivw}, pre and post the registration. We also report the wall clock time for the parameter search. The missing cases for $K = 8$ failed to finish in a reasonable time frame. We only report a couple of cases for $K = 16$ and expect a similar behavior to $K = 8$ for the rest.*

run	K	n_t	n_x	β_v^*	β_w^*	J_{min}	J_{max}	D_a		D_{vw}		D_{ivw}		runtime(s)
								pre	post	pre	post	pre	post	search
#1	8	4	$n/2$	1.4e-05	1.0e-07	9.7e-02	1.1e+01	3.2e-01	8.8e-01	5.3e-01	9.8e-01	7.4e-02	7.4e-01	3.9e+02
#2			$n/4$	1.1e-05	1.0e-07	3.8e-01	3.8e+00		6.8e-01		9.2e-01		2.1e-01	7.9e+02
#3			$n/8$	2.4e-05	1.0e-06	3.8e-01	4.8e+00		6.2e-01		8.9e-01		1.6e-01	1.4e+01
#4		8	$n/4$	1.1e-05	1.0e-06	2.9e-01	7.7e+00		7.5e-01		9.4e-01		4.1e-01	1.0e+02
#5			$n/8$	1.7e-05	1.0e-06	3.9e-01	6.4e+00		5.9e-01		8.7e-01		1.6e-01	1.6e+01
#6		16	$n/2$	1.1e-05	1.0e-07	1.8e-01	1.4e+01		8.3e-01		9.7e-01		5.2e-01	5.9e+02
#7			$n/4$	1.1e-05	1.0e-06	3.1e-01	8.2e+00		7.1e-01		9.2e-01		3.4e-01	1.2e+02
#8			$n/8$	1.1e-05	1.0e-07	5.4e-01	3.2e+00		5.6e-01		8.5e-01		1.4e-01	6.7e+01
#9		32	n	1.1e-05	1.0e-07	5.1e-02	1.0e+01		9.0e-01		9.8e-01		7.6e-01	2.7e+03
#10			$n/2$	1.1e-05	1.0e-07	1.2e-01	1.9e+01		7.8e-01		9.5e-01		4.2e-01	7.6e+02
#11			$n/4$	1.1e-05	1.0e-06	3.1e-01	1.0e+01		6.8e-01		9.0e-01		3.3e-01	1.9e+02
#12			$n/8$	1.1e-05	1.0e-07	5.2e-01	3.2e+00		5.6e-01		8.5e-01		1.4e-01	4.8e+01
#13	16	4	$n/2$	1.3e-05	1.0e-06	2.0e-01	6.9e+00	2.9e-01	8.6e-01	5.1e-01	9.7e-01	9.0e-02	7.8e-01	3.7e+02
#14		32	n	1.1e-05	1.0e-07	1.6e-01	9.5e+00		9.1e-01		9.8e-01		8.1e-01	2.4e+03

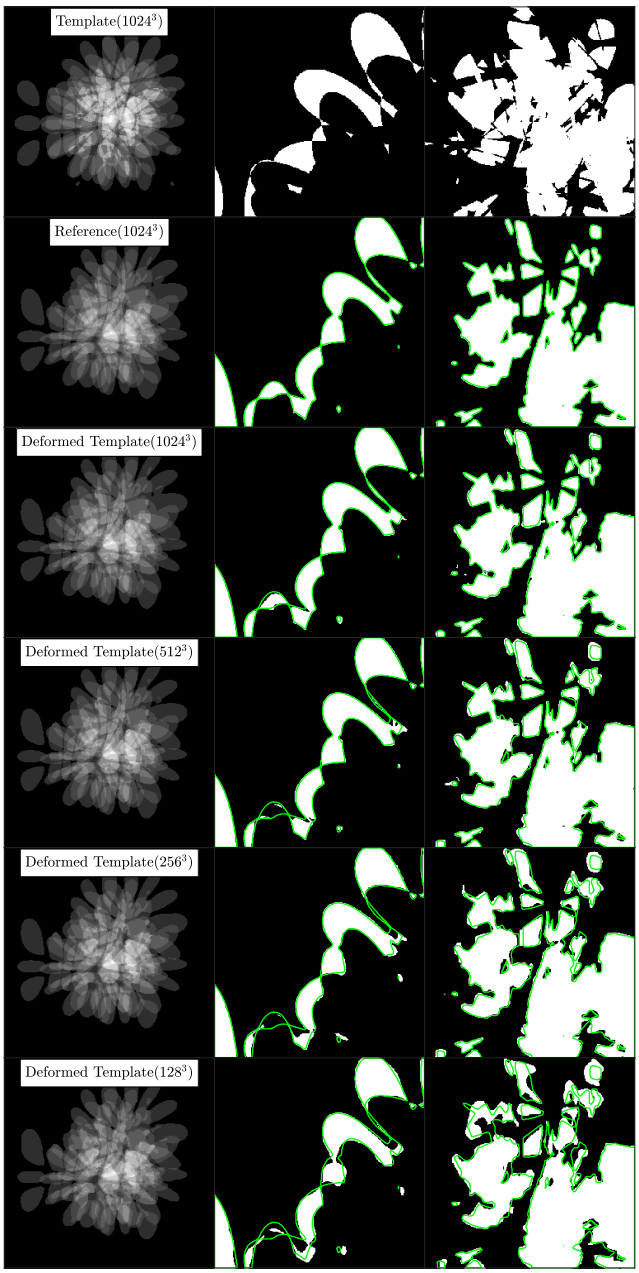


Figure 5.3: ***Visualization of registration results for experiment 1A (case 1).*** *In column 1, from top to bottom, we visualize the template, reference and deformed template images for registrations done at different resolutions. These images correspond to the runs #1-4 in Table 5.3. The value in the parenthesis in column 1 indicates the resolution at which registration was done. The visualization is done at that original resolution ($N = 1024^3$). In column 2 and 3, we visualize cropped portions of the images shown in column 1 for specific label values. In column 2 we show label 1 and in column 3 we show the union of labels with intensity value ≥ 5. We note that higher label values have smaller volumes and more fine features. We plot the label boundaries for the reference image in green to visualize the registration errors. We can observe that at lower resolutions (top to bottom) the alignment of the outlines (green lines; reference image) with the structures (white areas; deformed template image) is less accurate.*

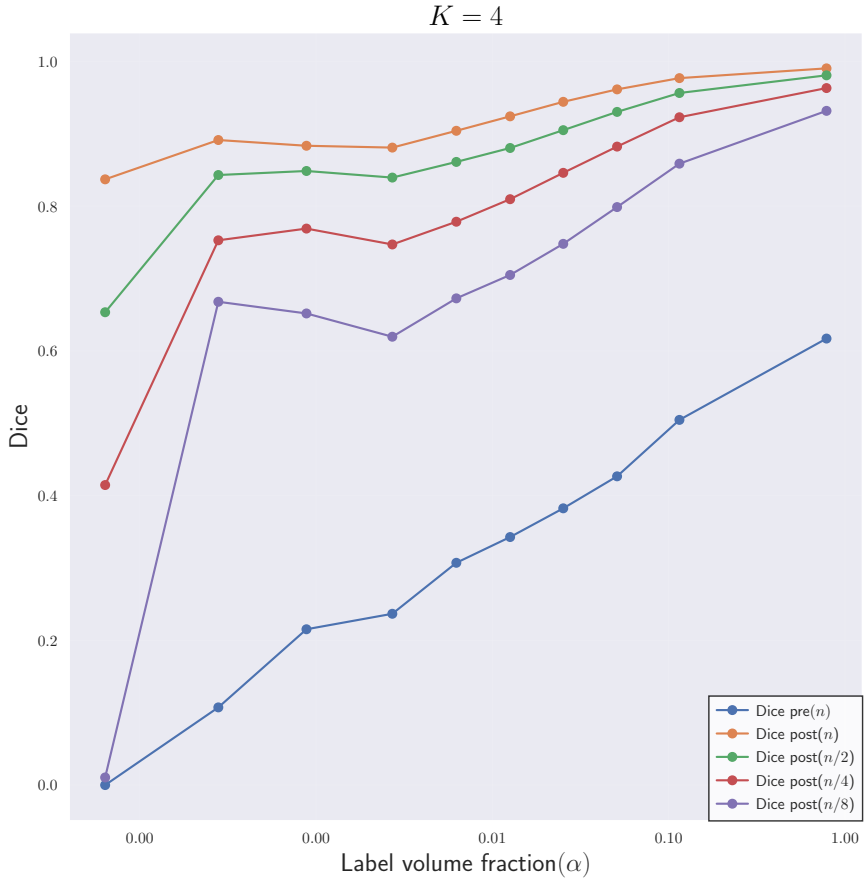


Figure 5.4: ***Quantitative results for the registration results corresponding to experiment 1A (case 1).*** *We show a plot of Dice score against label volume fraction α for each label l^i, $i = 1, \ldots, 10$ for the registration of the synthetic data sets* ***SYN*** *at different resolutions. This figure corresponds to the registration runs #1-4 in Table 5.3 for $K = 4$. This figure shows that the Dice score is worse for labels with smaller volume fractions, i.e., fine structures are matched less accurately at coarse resolutions.*

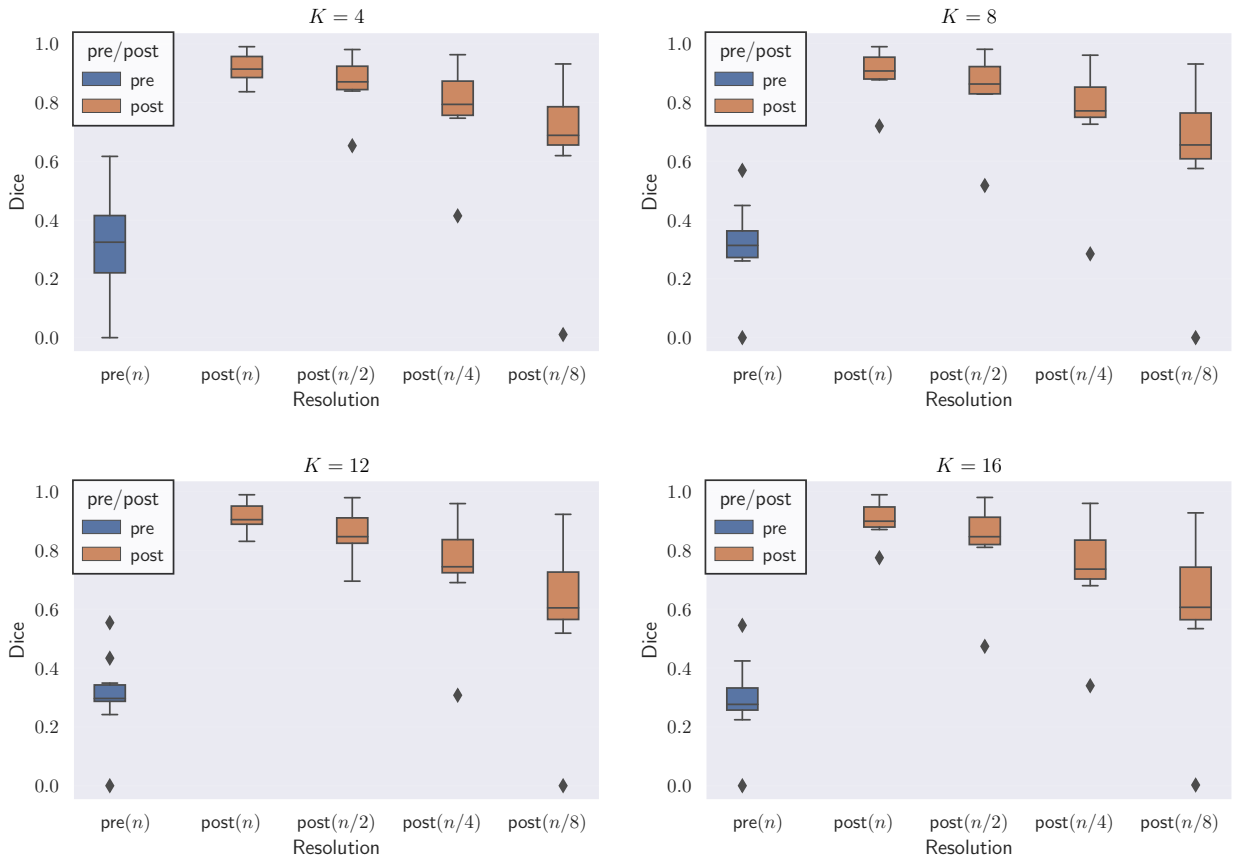


Figure 5.5: ***Quantitative results for the registration results corresponding to experiment 1A (case 1).*** *We show box plots of the Dice scores for the individual labels before and after registration for different resolutions. We consider the synthetic test problem* ***SYN****. This figure corresponds to the registration results reported in Table 5.3. We can observe that the average registration accuracy decreases as we decrease the resolution.*

1. The Dice score averages are better for registrations performed at the base resolution n with progressively worse Dice scores for registrations done at coarser resolutions.

2. The difference between Dice scores for registrations done at successively coarser resolutions for $K = 16$ (rougher velocity field) is higher than at $K = 4$ (smoother velocity field), and

3. Keeping n_t fixed for base and coarser resolutions does not affect the Dice score trend, i.e., the Dice decreases as n_x is decreased.

Regarding Dice score averages in Table 5.3, we observe that D_a, which denotes the arithmetic mean of the Dice scores of individual labels, has a difference of as much as 7% (see run #13 and #14 in Table 5.3). However, the percentage drop in volume weighted Dice average D_{vw} is smaller than D_a. This indicates that labels with higher volume are still easier to register at coarser resolutions. On the other hand, the inverse weighted Dice average D_{ivw}, which gives more weight to smaller labels, has a more pronounced effect on the Dice because smaller regions contribute to high frequency content in the image; this information is lost when the images are downsampled. We observe a 21.8% difference in D_{ivw} for the high frequency images (see run #13 and #14 in Table 5.3 for $K = 16$). As we increase K, we see that the difference in all Dice score averages between successive resolutions increases. As K increases, we get increasingly rougher velocity fields. We may not recover these velocities by registering the original images at coarser resolutions. This is because we discretize the true velocity field itself at the same resolution as the input high resolution images.

In Table 5.4, the Dice scores behave the same way even when n_t is fixed for different n_x indicating that the loss in accuracy is primarily because of the reduction in the spacial resolution (and not the temporal resolution). We also observe that for the full resolution of $n_x = n$, using $n_t < 32$ results in the solver converging at a very slow rate; the run did not finish in under 2 hrs. We attribute this slow convergence rate to the loss in numerical accuracy in the computation of the reduced gradient in (2.3). If we compare run #1 and

#9 in Table 5.4, we see that the difference in D_a is marginal in comparison to the run time cost overhead for run #9. We argue that this difference increases as K is increased and the images get less smooth (see run #13 and #14 in Table 5.4).

We use 32 GPUs for registration at $N = 1024^3$, 4 GPUs for $N = 512^3$, and a single GPU for $N = \{256^3, 128^3\}$[11]. Registration for $N = 1024^3$ takes on average 44 minutes of wall clock time. It is important to note that this includes the time spent in the search for an adequate automatic regularization parameter (i.e., we solve the inverse problem multiple times using warm starts; see §5.1.2 for details regarding the scheme). For the large scale runs that use multiple GPUs the overall runtime of the solver is dominated by communication between MPI processes [28]. Adding more resources does not necessarily reduce the runtime because of this increase in communication cost. Registrations for $n_x = 512^3$ and lower resolutions are much quicker and run in the order of 10 min or lower. In the present work, we perform the parameter search for each individual case because *we want to obtain the best result for each pair of images*. However, in practice where a medical imaging pipeline requires registrations for several similar kind of images, we suggest running the parameter search scheme on one pair of images and use the obtained regularization parameters to run the cohort registration for all images, as we have done in our previous work [108]. This strategy does not find the optimal regularization parameters for each image pair; this reduces the computational cost drastically. One downside to this strategy is that some images in the cohort will not be registered as accurately as others.

Conclusions. Our experiment with synthetic images suggest that Dice scores are better when registrations are done in the original high resolution at which labels were created. Registration accuracy is more affected for rougher velocity fields i.e. higher K. The percentage drop in Dice score is more evident for these kind of images. However, these images are synthetic and free of noise. Therefore, we cannot firmly conclude that registration

[11]Notice here that using $N = 1024^3$ or $N = 1024 \times 1024 \times 1024$ has the same meaning as saying $n_x = (1024, 1024, 1024)$

at downsampled resolutions results in degraded performance in practical applications. To strengthen our claim, we conduct another experiment on real brain MRIs in the next section.

5.2.4 Experiment 1B: High Resolution Real Data Registrations

Aim. In this experiment, we try to answer **Q1** as well as **Q2**. We attempt this by registering real human brain MRI datasets instead of synthetic images. Unlike synthetic images, these images are not noise-free. Moreover, they lack high contrast.

Datasets. We use two real brain datasets for this experiment. These datasets are:

- **MRI250** [99]: is an *in-vivo* human brain MRI image which consists of a T_1-weighted anatomical data at an isotropic spatial resolution of $250\mu m$. The original image size is $640 \times 880 \times 880$ voxels.
- **NIREP** [39]: is a standardized repository for assessing registration accuracy that contains 16 T_1-weighted MR neuroimaging datasets (`na01`–`na16`) of different individuals at an isotropic resolution of 1 mm. The original image size is $256 \times 300 \times 256$ voxels.

Procedure. We designate the 250 μm brain MRI as the template image m_0. Since we do not have access to another T_1-weighted MRI from a different subject at the original resolution of 250 μm, we use the image `na01` from the NIREP dataset as the reference image m_1. However, the acquired spatial resolution of the NIREP data is 1 mm, which is $4\times$ larger than 250 μm. Therefore, in order to generate a reference image m_1 that is 250 μm in spatial resolution, we take the following steps:

1. Upsample `na01` from $256 \times 300 \times 256$ to $640 \times 880 \times 880$ using linear interpolation.
2. Register MRI250 to the upsampled `na01` image using `CLAIRE`. We set the tolerance for the relative gradient norm g_{tol} = 1e-02. (We lower the tolerance compared to other runs to obtain a potentially more accurate registration result.) We use the

default regularization parameters β_v = 1e-02 and β_w = 1e-04. Consequently, we do not perform a parameter search to estimate an optimal regularization parameter for this registration. The reason for doing this is that we want to keep the downstream registration performance analysis, where we will use parameter search, oblivious to the process of generating the high resolution reference image. We denote the computed registration velocity by $v_{\texttt{na01}}$, where `na01` is the target NIREP image.

3. Then we transport m_0 (which corresponds to the MRI250 image) using $v_{\texttt{na01}}$ by solving (2.1c) to obtain the deformed template image $m(t = 1)$. We designate this image as the reference image, i.e, m_1 is set to $m(t = 1)$.

We use the tool `fast` from FSL [84] to segment the template image m_0 and the reference image m_1 into *WM, GM and CSF*. We use this segmentation to compute Dice scores. The remaining steps for this experiment are exactly as described in experiment 1A in §5.2.3 except that here we are registering real T_1-weighted images instead of noise-free synthetic images. It is important to note that we only downsample m_0 and m_1 and not their segmentation in order to perform the registration at smaller resolutions. The base resolution for this experiment is $n_x = n = (640, 880, 880)$. We consider $n_x = n/2$ and $n_x = n/4$ for the downsampled resolutions. We also consider the two sub-cases for selecting n_t as we did in §5.2.3. For the case where n_t changes with resolution, we use $n_t = 4$ for $n_x = n/4$, $n_t = 8$ for $n_x = n/2$ and $n_t = 16$ for $n_x = n$.

Results. We create different reference images m_1 using the above described process by selecting 10 target images `na01`–`na10` from the NIREP dataset. We report the solver parameters for all 10 cases along with the relative residual r and Dice score averages for *GM*, *WM* and *CSF* before and after the registration in Table 5.5. The relative residual r and Dice score are always computed at the base resolution $n = (640, 880, 880)$. Similar to the experiment with the **SYN** dataset in §5.2.3, we run additional registrations by keeping n_t fixed for all resolutions and report these results in Table 5.6. We visualize the image

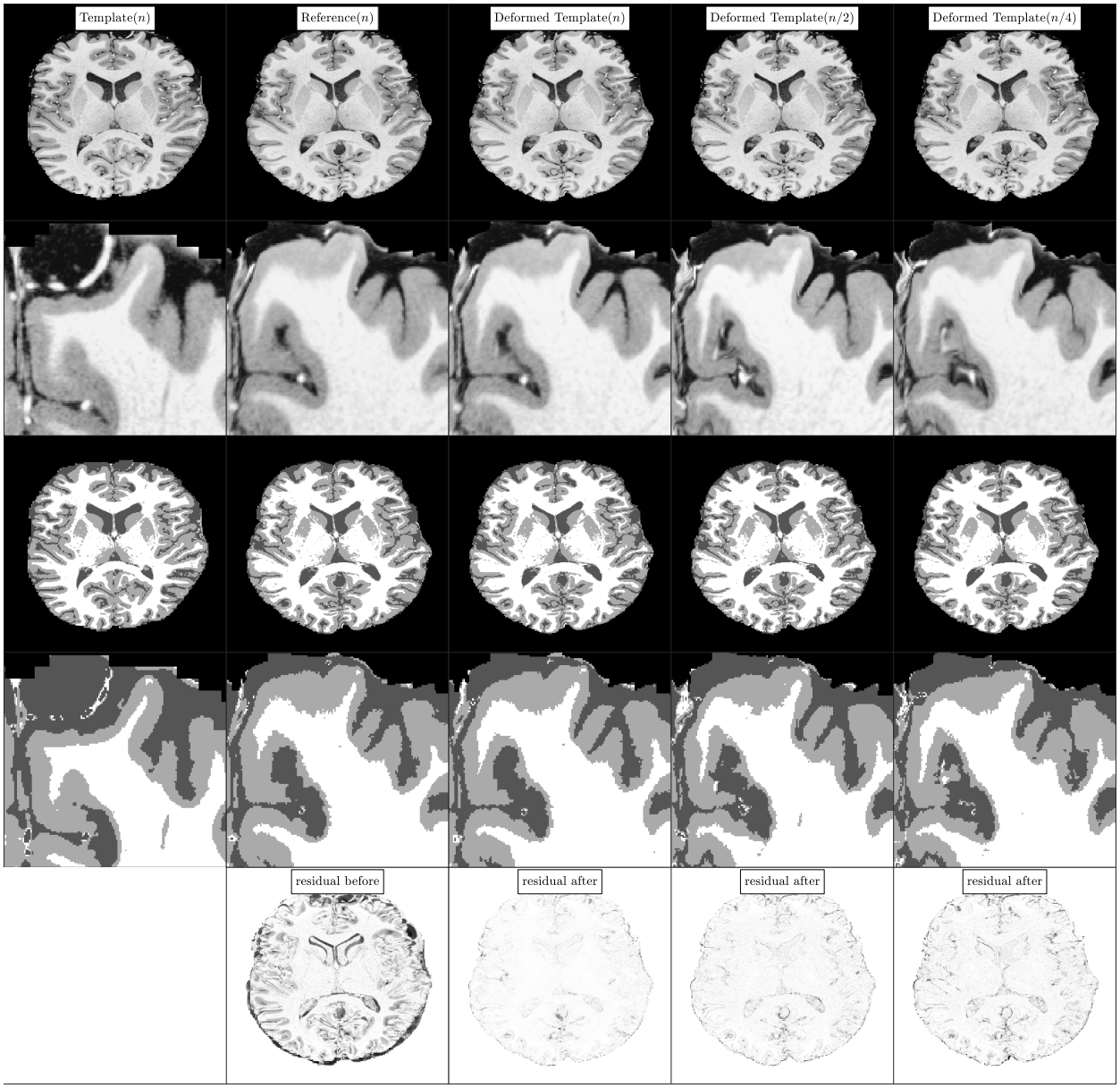


Figure 5.6: ***Illustration of registration results for the multi-resolution registration experiment on real brain images (experiment 1B; see §5.2.4).*** *The images shown here correspond to the runs #1, #2, and #3 in Table 5.5). The base resolution is* $n_x = n = (640, 880, 880)$. *In row 1, from left to right, we show the T1-weighted MRI250 datasets (template image* m_0*), the upsampled* `na01` *dataset (reference image* m_1*) from the NIREP data repository, and the deformed template images obtained from registrations at resolutions* n_x*,* $n_x/2$ *and* $n_x/4$*, respectively. In row 2, we show a cropped portions of the images from row 1. In row 3 and 4, we show the label maps consisting of white matter (WM; white), gray matter (GM; light gray) and cerebro-spinal fluid (CSF; dark gray) and their cropped versions, respectively. In row 5, we show the image residuals before and after registration with respect to each resolution level.*

Table 5.5: ***Registration performance for CLAIRE for experiment 1B, case 1 (n_t changes with resolution).*** *Comparison of registration accuracy using Dice and relative residual r at different resolutions for registration of MRI250 brain image to 10 real MRI scans from the NIREP dataset. We consider three resolution levels $n_x = \{n, n/2, n/4\}$ where $n = (640, 880, 880)$. We fix the tolerance for the relative gradient to 5e-02. We use linear interpolation. The bounds for the determinant of the deformation gradient for the parameter search are $[0.05, 20]$. We increase the number of time steps n_t proportionally to the increase in spatial resolution. We report the regularization parameters β_v^* and β_w^* obtained through the proposed parameter search scheme, the minimum and maximum determinant of the deformation gradient (J_{min} and J_{max}), the relative residual r, the average Dice D_a pre and post the registration, as well as the wall clock time for the parameter search.*

run	NIREP	n_x	n_t	β_v^*	β_w^*	J_{min}	J_{max}	r	D_a pre	D_a post	runtime(s) search
#1		n	16	1.1e-03	1.0e-05	1.8e-01	8.3e+00	2.5e-01		9.0e-01	3.1e+02
#2	na01	$n/2$	8	1.1e-05	1.0e-06	1.8e-01	6.5e+00	3.5e-01	5.5e-01	8.5e-01	3.4e+02
#3		$n/4$	4	1.1e-05	1.0e-05	2.3e-01	8.2e+00	4.5e-01		8.0e-01	3.6e+01
#4		n	16	1.1e-03	1.0e-05	7.8e-02	6.3e+00	2.5e-01		8.9e-01	3.3e+02
#5	na02	$n/2$	8	1.1e-05	1.0e-06	1.2e-01	4.3e+00	3.6e-01	5.4e-01	8.3e-01	3.0e+02
#6		$n/4$	4	1.1e-05	1.0e-06	8.1e-02	1.1e+01	4.6e-01		7.7e-01	4.0e+01
#7		n	16	1.1e-05	1.0e-07	1.1e-01	6.1e+00	3.3e-01		8.4e-01	3.2e+03
#8	na03	$n/2$	8	1.1e-05	1.0e-06	1.1e-01	1.8e+01	3.9e-01	5.1e-01	8.0e-01	2.9e+02
#9		$n/4$	4	1.1e-05	1.0e-07	1.0e-01	1.7e+01	4.7e-01		7.6e-01	4.2e+01
#10		n	16	3.1e-02	1.0e-05	1.2e-01	1.4e+01	3.7e-01		8.0e-01	1.9e+02
#11	na04	$n/2$	8	1.1e-03	1.0e-05	6.8e-02	8.9e+00	2.9e-01	5.3e-01	8.7e-01	5.1e+01
#12		$n/4$	4	1.1e-05	1.0e-05	1.0e-01	6.8e+00	4.6e-01		7.6e-01	3.7e+01
#13		n	16	1.1e-05	1.0e-05	9.5e-02	8.9e+00	3.2e-01		8.5e-01	2.8e+03
#14	na05	$n/2$	8	1.1e-05	1.0e-05	1.6e-01	1.1e+01	3.6e-01	5.3e-01	8.3e-01	2.5e+02
#15		$n/4$	4	1.1e-05	1.0e-05	1.6e-01	1.6e+01	4.5e-01		7.8e-01	3.7e+01
#16		n	16	1.1e-03	1.0e-05	7.8e-02	1.4e+01	2.5e-01		8.9e-01	3.3e+02
#17	na06	$n/2$	8	1.1e-05	1.0e-06	2.0e-01	5.6e+00	3.5e-01	5.3e-01	8.3e-01	3.0e+02
#18		$n/4$	4	1.1e-05	1.0e-05	1.6e-01	7.2e+00	4.4e-01		7.7e-01	3.6e+01
#19		n	16	1.0e-02	1.0e-05	9.5e-02	2.0e+01	3.0e-01		8.6e-01	2.4e+02
#20	na07	$n/2$	8	1.1e-05	1.0e-05	1.6e-01	1.6e+01	3.5e-01	5.3e-01	8.4e-01	3.9e+02
#21		$n/4$	4	1.1e-05	1.0e-05	1.7e-01	1.7e+01	4.5e-01		7.7e-01	3.7e+01
#22		n	16	1.1e-05	1.0e-07	1.3e-01	4.8e+00	3.1e-01		8.6e-01	2.5e+03
#23	na08	$n/2$	8	1.1e-05	1.0e-06	1.0e-01	1.3e+01	3.8e-01	5.3e-01	8.1e-01	3.0e+02
#24		$n/4$	4	1.1e-05	1.0e-06	9.4e-02	1.7e+01	4.7e-01		7.5e-01	4.2e+01
#25		n	16	1.1e-03	1.0e-05	6.3e-02	1.5e+01	2.5e-01		8.9e-01	3.5e+02
#26	na09	$n/2$	8	1.1e-05	1.0e-05	1.2e-01	5.1e+00	3.5e-01	5.3e-01	8.3e-01	2.3e+02
#27		$n/4$	4	1.1e-05	1.0e-06	9.9e-02	7.4e+00	4.5e-01		7.6e-01	4.2e+01
#28		n	16	1.1e-05	1.0e-07	1.1e-01	5.7e+00	3.2e-01		8.5e-01	2.6e+03
#29	na10	$n/2$	8	1.1e-05	1.0e-05	1.2e-01	4.5e+00	3.5e-01	5.4e-01	8.3e-01	2.7e+02
#30		$n/4$	4	1.1e-05	1.0e-06	1.0e-01	9.1e+00	4.7e-01		7.6e-01	4.1e+01

Table 5.6: ***Registration performance for `CLAIRE` for experiment 1B, case 2 (n_t fixed with resolution).*** *Comparison of registration accuracy using Dice and relative residual r for a fixed number of time steps n_t at different resolutions for the registration of the real MRI datasets MRI250 and `na01` from the NIREP repository. We consider three resolution levels $n_x = \{n, n/2, n/4\}$ where $n = (640, 880, 880)$. We fix the tolerance for the relative gradient to $5e$-02. We use linear interpolation. The bounds for the determinant of the deformation gradient for the parameter search is $[0.05, 20]$. We keep the time step n_t fixed. We report the regularization parameters β_v^* and β_w^* obtained through the proposed parameter search scheme, the minimum and maximum determinant of the deformation gradient (J_{min} and J_{max}), the relative residual r, the average Dice D_a pre and post the registration, as well as the wall clock time for the parameter search. The case with $n_x = n$ and $n_t = 4$ failed to finish in under 4 hrs.*

run	NIREP	n_t	n_x	β_v^*	β_w^*	J_{min}	J_{max}	r	D_a		runtime(s)
									pre	post	search
#1	na01	4	$n/2$	1.1e-05	1.0e-06	9.2e-02	5.2e+00	3.1e-01	5.5e-01	8.7e-01	2.7e+02
#2			$n/4$	1.1e-05	1.0e-05	2.3e-01	8.2e+00	4.5e-01		8.0e-01	3.6e+01
#3		8	n	5.6e-03	1.0e-05	1.1e-01	1.1e+01	2.4e-01		9.0e-01	2.6e+02
#4			$n/2$	1.1e-05	1.0e-06	1.9e-01	6.6e+00	3.5e-01		8.5e-01	3.1e+02
#5			$n/4$	1.1e-05	1.0e-05	2.7e-01	1.2e+01	4.7e-01		7.8e-01	4.3e+01
#6		16	n	1.1e-03	1.0e-05	1.8e-01	8.4e+00	2.5e-01		9.0e-01	3.2e+02
#7			$n/2$	1.1e-05	1.0e-05	2.6e-01	7.6e+00	3.8e-01		8.3e-01	3.6e+02
#8			$n/4$	1.1e-05	1.0e-05	2.8e-01	1.6e+01	4.8e-01		7.7e-01	5.5e+01

Table 5.7: ***Experiment 1b: effect of*** $\beta_{w,init}$ ***on registration performance for real brain images****: Comparison of registration accuracy using Dice and relative residual r for different values of $\beta_{w,init}$ at different resolutions for registration of* ***MRI250*** *brain image to* ***na01*** *and* ***na02*** *from* ***NIREP*** *dataset. We fix $\beta_{w,min} = 1e-09$. We consider $n_x = \{n, n/4\}$ where $n = (640, 880, 880)$. We fix the tolerance for the relative gradient to $5e-02$. We use linear interpolation. The Jacobian bounds for parameter search are $[0.05, 20]$. We increase the number of time steps n_t proportionately with increase in resolution. We report β_v^* and β_w^*, the regularization parameters from the parameter search scheme, J_{min} and J_{max}, the minimum and maximum Jacobian determinant the relative residual r, average Dice D_a pre and post the registration and the wall clock time for the parameter search for the registration. We highlight the best Dice scores for each resolution and for each NIREP image.*

run	NIREP	$\beta_{w,init}$	n_x	β_v^*	β_w^*	J_{min}	J_{max}	r	D_a pre	D_a post	runtime(s) search
#1	na01	1e-04	N	1.1e-05	1.0e-09	2.8e-01	2.7e+00	3.4e-01	5.5e-01	8.6e-01	5.0e+03
#2			N/4	2.3e-05	1.0e-05	3.5e-01	2.8e+00	4.6e-01		7.9e-01	3.5e+01
#3		1e-05	N	1.1e-03	1.0e-05	1.8e-01	8.3e+00	2.5e-01		9.0e-01	3.1e+02
#4			N/4	1.1e-05	1.0e-05	2.3e-01	8.2e+00	4.5e-01		8.0e-01	3.6e+01
#5		1e-06	N	2.0e-02	1.0e-06	5.2e-02	1.6e+01	3.0e-01		8.6e-01	2.1e+02
#6			N/4	1.0e-03	1.0e-06	1.2e-01	1.6e+01	3.7e-01		8.5e-01	1.2e+01
#7		1e-07	N	5.1e-02	1.0e-09	1.8e-01	1.0e+01	4.1e-01		7.9e-01	2.1e+02
#8			N/4	6.6e-03	1.0e-07	1.1e-01	1.8e+01	4.0e-01		8.2e-01	7.7e+00
#9	na02	1e-04	N	1.1e-05	1.0e-09	2.1e-01	2.7e+00	3.4e-01	5.4e-01	8.4e-01	5.7e+03
#10			N/4	1.1e-05	1.0e-06	8.7e-02	9.9e+00	4.7e-01		7.7e-01	4.0e+01
#11		1e-05	N	1.1e-03	1.0e-05	7.7e-02	6.3e+00	2.5e-01		8.9e-01	3.4e+02
#12			N/4	1.1e-05	1.0e-06	8.1e-02	1.1e+01	4.6e-01		7.7e-01	3.9e+01
#13		1e-06	N	4.1e-02	1.0e-07	1.4e-01	2.0e+01	3.8e-01		8.0e-01	2.4e+02
#14			N/4	1.1e-04	1.0e-06	8.2e-02	1.2e+01	4.1e-01		8.1e-01	1.4e+01
#15		1e-07	N	4.0e-02	1.0e-09	1.2e-01	1.8e+01	3.8e-01		8.0e-01	2.5e+02
#16			N/4	1.0e-03	1.0e-07	8.8e-02	2.0e+01	3.8e-01		8.3e-01	1.4e+01

registration results for reference image `na01` in Figure 5.6.

Observations. The most important observation is that the relative residual r increases and Dice score averages decrease for registrations done at coarser resolutions irrespective of whether we increase n_t proportionally to the resolution, see Table 5.5 or keep n_t fixed for different n_x, see Table 5.6. This observation is in line with the experiment for the synthetic dataset **SYN** in §5.2.3. Except for the case of `na04` (see runs #10 and #11 in Table 5.5), all other cases exhibit increasingly worse registration performance at coarser resolutions.

In Table 5.6, the case with $n_t = 4$ and $n_x = n$ took very long to converge (>4 hrs). For this case the CFL number is 15.66 during the inverse solve while for $n_t = 16$, the CFL number is 4. The larger CFL number for $n_t = 4$ yields a higher adjoint error in the

SL scheme. This leads to higher errors in the computation of the reduced gradient, which results in worse convergence rate of the inverse solver for $n_t = 4$. The run time overhead associated with using $n_t = 16$ against $n_t = 4$ is offset by a better convergence of the solver. We refer to [104] for a thorough study on the effect of n_t on the accuracy of the numerical computation of the reduced gradient.

5.2.5 Experiment 2: Registration of Mouse Brain CLARITY Images

Aim

The aim of this experiment is to answer both **Q1** and **Q2** by examining the performance of our scalable registration solver on ultra-high resolution mouse brain images acquired using the CLARITY imaging technique [36, 151]. The dataset in this experiment, as opposed to the previous datasets, does not provide any real metrics for its assessment other than the relative residual (nor are we aware of any segmentation software that would work on these data).

Dataset

We use the dataset from [96] which consists of 12 CLARITY mouse brains imaged using the CLARITY-Optimized Light-sheet Microscopy (**COLM**). These images have low contrast and are very noisy although they are available in ultra-high spatial resolution. The in-plane resolution is $0.585\mu\text{m} \times 0.585\mu\text{m}$ and the cross-plane resolution is 5 to 8 μm. The images are stored at eight different resolution levels with level zero being the full resolution and level seven being the lowest resolution. We use the images at resolution levels three and six in our experiments. These levels correspond to an in-plane resolution of $4.68\mu\text{m} \times 4.68\mu\text{m}$ and $37.44\mu\text{m} \times 37.44\mu\text{m}$, respectively, which translates to images of size $n = (2816, 3016)$ and $n/8 = (328, 412)$ voxels. The cross-plane resolution is constant at all levels and corresponds to 1162 voxels. We select `Control182`, `Fear197` and `Cocaine178` as the test images in our

experiments.

Procedure. *Preprocessing:* For all unprocessed images, the background intensity is non-zero. We normalize the image intensities such that they lie in the range [0,1] with the background intensity re-scaled to zero. Next, we affine register all images to `Control182` at 8× downsampled resolution using the SyN tool in ANTs. We report the parameter settings for the affine registration in the appendix. Subsequently, we zero-pad the images to ensure that periodic boundary conditions are satisfied for `CLAIRE`. After preprocessing, the base image resolution is $n_x = n$ where $n = (2816, 3016, 1162)$ and $n/8 = (328, 412, 1162)$, respectively. For these set of images, we only conduct the parameter search for a single pair of images (at both resolutions independently) and then perform the parameter continuation on the entire dataset. We only report wall clock times for the parameter continuation and not for the parameter search.

Deformable Registration: We register all images to the reference image `Control182` using `CLAIRE`. We use the proposed parameter continuation scheme. We set J_{min} to 0.05. We do this for both resolution levels. To compare the registration accuracy between each resolution level, we follow the same steps from §5.2.4. We compare the registration performance using the relative residual r. For this dataset, we do not have access to image segmentation and, therefore, we cannot evaluate accuracy using Dice scores.

Results. We report the quantitative results for the registration of the CLARITY data in Table 5.8. We showcase exemplary registration results in Figure 5.7.

Observations. The most important observation is that we can register high resolution real medical images reasonably well in under 2 hrs (see run #1 and #3 in Table 5.8). Unlike the previous experiments in §5.2.3 and §5.2.4, the reported wall clock time in Table 5.8 is for performing the parameter continuation and not the parameter search. The average time spent for the regularization parameter search for resolution $n_x = n$ is ∼2 hrs. Another observation, which is in agreement with the results reported for the experiments

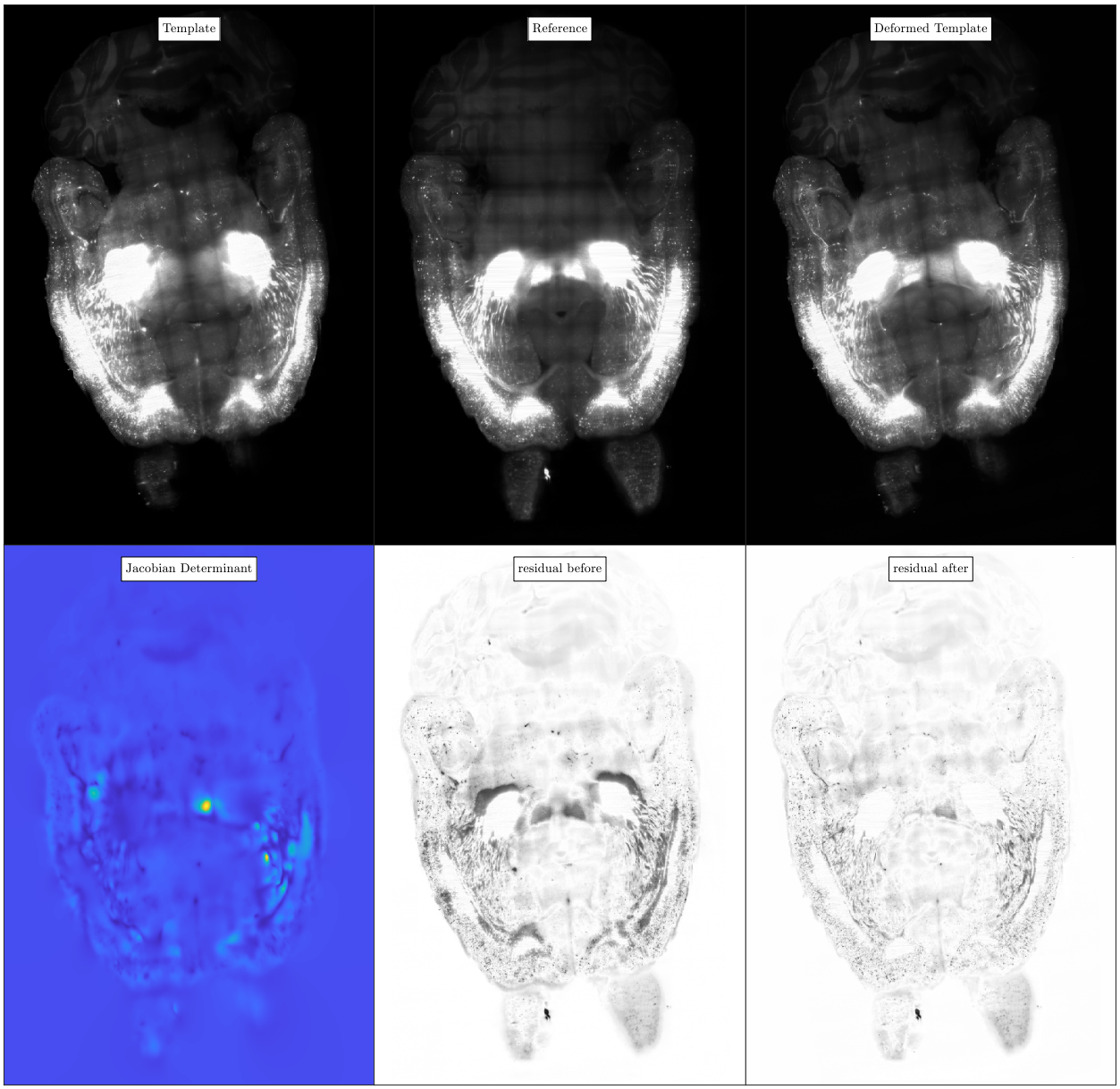


Figure 5.7: ***Illustration of the registration performance for*** `CLAIRE` ***for the CLARITY mouse brain imaging data (experiment 2).*** *We report registration results for the* `Cocaine178` *dataset registered to the to* `Control182` *dataset. In row 1 (from left to right), we have the template image* m_0 *(*`Fear197`*), the reference image* m_1 *(*`Control182`*) and the deformed template image. The resolution of the images is* $n = (2816, 3015, 1162)$*. In row 2, we show the determinant of the deformation gradient and the image residuals before and after registration.*

Table 5.8: ***Registration performance for*** `CLAIRE` ***for the CLARITY imaging data at resolutions*** $n = (2816, 3016, 1162)$ ***and*** $n/8 = (328, 412, 1162)$. `Control182` *is the fixed (reference) image. All other images selected from the CLARITY dataset are registered to* `Control182` *using a parameter continuation scheme. We fix the tolerance for the relative gradient to* 5e-02. *We use linear interpolation for the SL scheme. The bounds on the determinant of the deformation gradient for the parameter search are* $[0.05, 20]$. *We report the estimated regularization parameters* β_v^* *and* β_w^*, *the minimum and maximum values for the determinant of the deformation gradient (*J_{min} *and* J_{max}*), the relative residual* r, *as well as the wall clock time for the parameter continuation.*

run	image	#GPU	n_x	n_t	β_v^*	β_w^*	J_{min}	J_{max}	r	runtime(s)
#1	`Fear197`	256	n	16	1.1e-02	1.0e-05	5.5e-02	2.2e+01	3.4e-01	1.4e+03
#2		8	$n/8$	16	1.0e-03	1.0e-05	5.8e-02	1.5e+01	6.3e-01	9.6e+01
#3	`Cocain178`	256	n	16	1.1e-02	1.0e-05	3.1e-02	1.2e+01	4.1e-01	6.2e+03
#4		8	$n/8$	16	5.6e-03	1.0e-05	3.5e-02	4.7e+00	6.8e-01	6.1e+01

carried out in §5.2.3 and §5.2.4, is that the registration performed at downsampled resolution (see Table 5.8) results in a larger relative residual and, therefore, worse registration accuracy. We had a maximum of 256 GPUs (64 nodes, 4 GPUs per node) available to us at the TACC Longhorn supercomputer. Because of this resource constraint, our solver ran out of memory for certain parameter configurations (for example, for run #1 and #3, we could not use $n_t > 16$ time steps). Moreover, for all the runs in Table 5.8 we used the Inv$\mathcal{H}_0$ preconditioner and not 2LInv$\mathcal{H}_0$ because 2LInv$\mathcal{H}_0$ requires additional memory for the coarse grid spectral operations. mismatch

5.3 Chapter conclusions

In this publication, we apply our previously developed multi-node multi-GPU 3D image registration solver [29] to study and analyze large-scale image registration. This work builds upon our former contributions on constrained large deformation diffeomorphic image registration [102, 104, 106, 108]. In particular,

- We have presented an augmented version of the parameter search scheme previously introduced in [102]. We can compute deformations, which are guaranteed to be locally

diffeomorphic and driven by user specifications. Instead of doing a brute-force search in the parameter space, our improved scheme employs an approach in which we first fix β_w and search for β_v and subsequently alternate, if necessary. Doing so, eliminates the manual adjustment of another hyperparameter in the setup of `CLAIRE`. Using the proposed scheme, we demonstrated that `CLAIRE` provides a registration quality that is on par with results generated by ANTs, a state-of-the-art CPU image registration package.

- We are able to register CLARITY mouse brain images of unprecedented ultra-high spatial resolution ($2816 \times 3016 \times 1162$) in 23 minutes using parameter continuation. To the best of our knowledge, images of this scale have not been registered in previous work [29, 95, 96].

- We conduct detailed experiments to compare the performance of image registration at full and downsampled resolutions using synthetic and real images. We find that image registration done at higher (native) image resolution is more accurate. To quantify the accuracy we use Dice coefficients wherever image segmentation is available and relative residuals otherwise. We also do a sensitivity analysis for the overall solver accuracy with respect to the number of time steps n_t in the SL scheme. Overall, `CLAIRE` performed as expected: fully automatic parameter tuning works well and higher image resolutions results in improved image similarity compared to the registration results in lower resolution.

Acknowledgements

This work was partly supported by the National Science Foundation (DMS-1854853, DMS-2012825, CCF-1817048, CCF-1725743), the NVIDIA Corporation (NVIDIA GPU Grant Program), the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy-EXC 2075-390740016, by the U.S. Department of Energy,

Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program under Award Number DE-SC0019393; by the U.S. Air Force Office of Scientific Research award FA9550-17-1-0190; by the Portugal Foundation for Science and Technology and the UT Austin-Portugal program, and by NIH award 5R01NS042645-11A1. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily reflect the views of the DFG, AFOSR, DOE, NIH, and NSF. Computing time on the Texas Advanced Computing Center (TACC) systems was provided by an allocation from TACC and the NSF. This work was completed in part with resources provided by the Research Computing Data Core at the University of Houston.

Chapter 6

Mass effect characterization using image registration

Modern medical imaging techniques are playing an increasingly important role in understanding the anatomy and function of organ structures. Careful analysis of these images can provide valuable information which can assist in the diagnosis and prognosis of various types of diseases. Features extracted from these images can be used to study the anatomical differences between healthy and pathological populations. Furthermore, it can also identify abnormalities related to disease in an individual image scan. For example, subtle deformations in healthy tissue caused by internal hemorrhage or a cancerous tumor can cause secondary pathological effects such as mass effect (the mechanical deformation of surrounding healthy tissue due to tumor growth). Mass effect deformations are significantly different from normal anatomical variations in the brains of healthy individuals (see Figure 1.2). Identifying the location and intensity of mass effect is vital in order to make a correct diagnosis for treatment planning and therapeutic care [22]. However, it is challenging to assess the spatial structure of mass effect from a single time point image scan due to the unavailability of a healthy baseline scan without mass effect. There exist biophysical tumor modeling-based methods [146–148] which estimate the spatial structure of mass effect from a single patient scan but these are computationally expensive. This work presents a fast image registration-based statistical framework to characterize mass effect from a single patient scan. We focus primarily on the mass effect induced by Glioblastoma (GBM), the most aggressive form of

primary brain tumor.

Our method uses nonlinear deformable image registration to compute statistics of normal deformations by registering normal to normal brain images. Then, we convert the high-dimensional displacement fields from registration to a set of local features defined over an anatomical parcellation of the brain. Viewing these features as random variables, we compute different statistics to detect mass effect and then localize it to specific brain regions. Because true mass effect is unknown, we use a biophysical tumor growth model [146] to create synthetic tumor images with known ground truth mass effect for verification of our method. For verification of our method on clinical data, we use the mass effect predictions from `GLIA` [145, 148], a tumor model inversion framework based on [146] as a silver standard. For diffeomorphic image registration, we use `CLAIRE`, a fast registration software which we have developed in Chapter 3 and Chapter 4.

Contributions

In particular, the contributions of this work are as follows:

- We present a multi-template image registration algorithm to detect and classify mass effect into three classes - mild, moderate, and severe and localize it to specific brain regions.
- We evaluate our method on synthetic and clinical GBM datasets and perform the classification and localization experiments for these datasets. In particular, we tested it on 405 clinical patient scans and observed that we could solve the mass effect classification problem with 62% accuracy. Furthermore, for moderate and severe mass effect, we report a set of five brain regions with an 88% likelihood of containing at least one of the top five regions with most significant mass effect.
- One key aspect of our workflow is that it uses GLIA, a biophysical tumor model and

inversion framework, to create a synthetic tumor dataset for data augmentation during the *training* phase for mass effect detection. For inference, we only use features from deformable image registration.

Related work

In the past few years, there have been several different approaches to quantify mass effect or characterize global anatomical changes in a tissue. One of the most popular approaches has been voxel based morphometry (VBM) [8, 10, 44, 45, 110, 158]. VBM of MRI data involves registering all input images to a common atlas, extracting the gray matter region from the registered images, smoothing, and finally performing a statistical analysis using a general linear model (GLM) to localize, and make inferences about group differences in gray matter concentration. VBM has been used extensively to study group differences in cortical gray matter atrophy in Alzheimer's Disease (AD) [45,46,73,128,131], to identify structural changes in brains of subjects with bipolar disorder [159,166]. Furthermore it has been used to classify AD subjects from normal subjects and those with mild cognitive impairment (MCI) [70,136].

While VBM is based on voxel or mesoscopic level changes, deformation-based morphometry (DBM) [11] characterizes the differences in deformation fields that describe macroscopic changes in brain anatomy. These deformation fields are obtained from the nonlinear registration of the subject MRI to a template that conforms to some standard anatomical space. A DBM analysis is performed in [101] to quantify the magnitude and pattern of atrophy in patients with frontotemporal dementia. In [61, 62], DBM is applied to evaluate brain ventricular volume changes in schizophrenic subjects by performing multivariate statistical analysis on the registration deformation fields. These methods have been primarily used to localize anatomical differences for a given subject group. However, they do not identify tissue regions with anomalous structures at the level of an individual subject. Furthermore, to the best of our knowledge, voxel or deformation-based morphological methods have not

been used to study brain tumor-induced mass effect.

There have been attempts to identify image-based biomarkers for mass effect in order to perform overall survival analysis for subjects with glioblastoma [125, 143]. In [143], the authors quantify mass effect by measuring the displacement of the center of mass of lateral ventricles between a subject and an affine registered template. However, the mass effect can be very localized depending on the size and location of the tumor and not always visible through the ventricles. In [125], the authors propose a new feature called mass effect deformation heterogeneity (MEDH). This feature is derived from registering a brain tumor subject to a single template and with an image registration objective function that masks the tumor region and only compares the healthy tissue between the brain-bearing image and the healthy image. Such masking can result in erroneous registration deformations, especially if the mass effect significantly alters healthy tissue structure. In our current work, we do not use cost function masking. Instead of using a T1-weighted image for the registration, we use its semantics segmentation. We assume that GBMs mostly proliferate and diffuse in white matter and replace the tumor segmentation with white matter.

Another alternative to registration-based techniques is to use image-calibrated biophysical models [4, 81, 115, 148]. In [81], the authors formulate a PDE-constrained inverse problem to estimate tumor growth model parameters including mass effect. However, this is only a 1D study and very limited in scope since mass effect is a 3D phenomenon. A 2D study to quantify mass effect is conducted in [4], but the authors did not apply the method to actual clinical data. A statistical model for tumor-induced deformations is presented in [115] to aid deformable registration of brain tumor images. The authors present a statistical method to estimate the mass effect parameter in their tumor growth model. They decompose the registration deformation between an atlas and image of a tumor subject into two components, one representing inter-individual differences in brain shape and the other representing tumor-induced deformations. Although this approach is promising, it is regarded as a proof of concept due to limited results in clinical data. In our group,

we presented a multi-atlas-based tumor model inversion scheme in [148]. In that work, we invert all model parameters, including mass effect, by averaging information from multiple atlases to remove atlas anatomy bias. However, this method is computationally expensive since it requires solving multiple inverse problems for a given image of a tumor patient.

To the best of our knowledge, we are not aware of any other study which classifies a GBM patient based on the degree of mass effect. Our model does not require a tumor growth model for inference on new patient data and only requires a few image registrations (after generating healthy population statistics), which can be done in minutes on modern heterogeneous computing architectures.

Problem Statement

Given a T1-weighted Magnetic Resonance Image (MRI) of a brain tumor patient p and its semantic segmentation with the labels, Whole tumor (WT), White Matter (WM), Gray Matter (GM), and Cerebrospinal Fluid (CSF). We have two specific goals concerning mass effect:

1. **Detection**: we want to detect whether p has significant mass effect caused by the tumor. Quantitatively, this amounts to classifying the mass effect of the patient as mild, moderate, or severe.

2. **Localization**: If p has mass effect, then we want to localize it to specific regions of the brain. This can be used for diagnostic purposes and visually verify the classification results from the first stage.

Chapter outline

In §6.1.1, we formulate an abstract framework and motivate our methodology. In §6.1.2, we describe the statistical model. In §6.2, we describe the imaging datasets. In §6.3 and §6.4,

we discuss the detection and localization problem. In §6.5, we report the results and finally report the conclusions in §6.7.

Table 6.1: *Notation and main symbols*

Notation	*Description*
Ω	Spatial domain: $\Omega := [0, 2\pi) \subset \mathbb{R}^3$ with boundary Ω
$\boldsymbol{x}$	Spatial coordinate; $\boldsymbol{x} := (x_1, x_2, x_3)^T \in \mathbb{R}^3$
N	Number of voxels in the discretized brain image
$a(x)$	Template image
$h(x)$	Normal image
$p(x)$	Patient image
$\pi_I(a)$	Distribution of template images
$\pi_I(h)$	Distribution of normal images
$\pi_I(p)$	Distribution of patient images
a_i	i_{th} template image
n_a	Number of template images
h_j	j_{th} normal image
n_h	Number of normal images
$h_0^{(T)}$	Synthetic tumor image based on normal image h_0
r	Region of interest (ROI) in template image; $\Omega_r \subset \Omega$
n_r	Number of ROIs in template image
V_r	Volume of Ω_r measured in number of voxels
$q_l(\boldsymbol{x})$	Global feature
$\mu_r(q_l)$	Restricted mean of global feature q_l in r; $\mu_r(q_l) := \big(\int_{\Omega_r} q \, \mathrm{d}\boldsymbol{x}\big)/V_r$
f_{rl}	Regional feature in r from global q_l, e.g., $f_{rl} = \mu_r(q_l)$
n_l	Number of global features q_l based on registration displacements
$\mathcal{S}_{rl}$	Scalar abnormality score for feature f_{rl}
$\boldsymbol{\mathcal{S}}_l$	Abnormality score vector for all r using q_l; $\boldsymbol{\mathcal{S}}_l \in \mathbb{R}^{n_r}$
$[u^*, h^{(T)}] = \mathcal{T}(h)$	Create synthetic tumor in h; u^* is tumor scalar displacement field in h-space
$\boldsymbol{w}_{m_t,m_r} = \mathcal{R}(m_t, m_r)$	Image registration between moving m_t and fixed m_r
$\boldsymbol{w}_{m_t,m_r}$	Registration displacement field deforming m_t to m_r
w	Voxel-wise ℓ_2 norm of $\boldsymbol{w}$; $w = w(\boldsymbol{x}) = \|\boldsymbol{w}(\boldsymbol{x})\|_2$

6.1 Method

6.1.1 Problem formulation

We define our notation in Table 6.1. To quantify and localize mass effect, we want to build a statistical model of normal deformations through registration of normal images to normal images. To do this, we assume that we have two groups of normal images. We refer to the first group as template images a and the second group as just normal images h. We define $\pi_I(a)$ and $\pi_I(h)$ to be their probability density function (PDF) or more informally *distribution*. In principle, $\pi_I(a)$ and $\pi_I(h)$ are the same distributions because a and h are both normal images without mass effect. Then, let $\boldsymbol{w}_{a,h} = \mathcal{R}(a, h)$ be the nonlinear registration displacement field which deforms a to h. We define $\pi_w(h|a)$ to be the distribution of all the displacement fields $\boldsymbol{w}_{a,h}$ that deform a template image a to different normal images h. Similarly, for a patient image p with a tumor, let $\pi_I(p)$ be its PDF. Then $\pi_w(p|a)$ is the distribution of displacement fields from a template a to patient p. If we integrate $\pi_w(h|a)$ and $\pi_w(p|a)$ over the template a, we get

$$\pi_w(h) = \int_a \pi_w(h|a)\pi_I(a)\,\mathrm{d}a \tag{6.1}$$

$$\pi_w(p) = \int_a \pi_w(p|a)\pi_I(a)\,\mathrm{d}a \tag{6.2}$$

where $\pi_w(h)$ and $\pi_w(p)$ are the distributions of displacements from any template image a to normal image h and patient image p respectively. The first problem we consider, given a new sample $\boldsymbol{w}$, is to decide whether it comes from $\pi_w(h)$ or $\pi_w(p)$. This is essentially a binary classification problem – either mass effect is present or it is absent. If $\boldsymbol{w}$ comes from $\pi_w(h)$, then mass effect is absent otherwise it is present. Expert reviews for clinical GBM scans (see §6.2), however, suggest that there is typically some level of mass effect always present in a GBM patient. So clinically, it is not very informative to report whether a patient has mass effect or not. Therefore, we classify a new sample $\boldsymbol{w}$ into one of the three classes – mild, moderate or severe mass effect. Our approach to solve this problem is to correlate the

mass effect with the degree of *outlierness* of $\boldsymbol{w}$ with respect to $\pi_w(h)$. The simplest way to do this is to construct a kernel density estimate (KDE) of $\pi_w(h)$ and then compute the log-likelihood for $\boldsymbol{w}$ coming from $\pi_w(h)$. But $\boldsymbol{w}_{a,h} \in \mathbb{R}^{3N}$, where N is the number of voxels in the image, is a multi-dimensional vector field and this makes $\pi_w(h)$ a multi-dimensional distribution. The KDE approach, therefore, is not feasible for three reasons – *(i)* assuming that $\boldsymbol{w}_{a,h}(\boldsymbol{x})$ at different image voxels $\boldsymbol{x}$ is an i.i.d random variable, the KDE will be very noisy because the displacement at a single voxel location is susceptible to registration errors and noise in the image, *(ii)* the number of samples $\boldsymbol{w}_{a,h}$ available for KDE are limited, e.g., for 3D clinical images of size 256^3, $N = 16$M and we can only do a few thousand image registrations limited mainly by time and storage, *(iii)* KDE for high-dimensional random variables is computationally expensive. For these reasons, we compute regional features from $\boldsymbol{w}_{a,h}$ instead of operating on the high dimensional vector field itself. We define these regions as different anatomical locations in the brain e.g. right parietal lobe (see Figure 6.1). We also assume that the regional average features are spatially uncorrelated and this enables us to perform an independent univariate analysis for each region and compute a regional *outlier* score. We refer to the outlier score as abnormality score because it represents the degree of abnormal deformations in a region. Now we can use the local abnormality scores from different regions as machine learning features to train a mass effect classifier. This will address, in part, our first goal – to classify a new sample $\boldsymbol{w}$ corresponding to patient p as having a mild, moderate or severe mass effect. Moreover, access to regional abnormality scores per region also enables us to address our second goal, i.e., spatial localization of mass effect. Regions of sizeable mass effect are more likely to have higher abnormality scores, and these regions can be reported to neuroradiologists for treatment planning. In the next section, we describe our statistical mass effect model in detail.

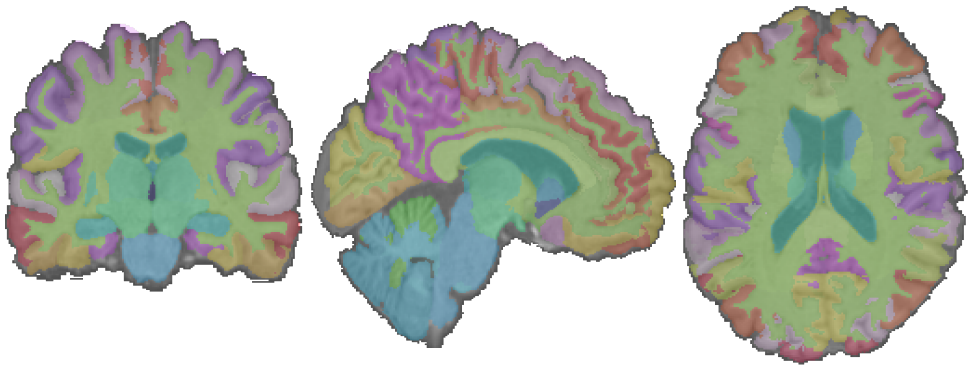


Figure 6.1: *We show an exemplar brain parcellation which we use to extract scalar regional features from multi-dimensional registration displacements and transition to a univariate analysis per region.*

6.1.2 Statistical mass effect model

In the previous section, we presented our methodology's problem formulation and motivation. We explained the reasons for the transition from multivariate analysis to a univariate analysis per region to construct the normal deformation distribution $\pi_w(h)$.

In this section, we define the overall method to construct $\pi_w(h)$ from actual image data, define global and regional features, and discuss how to compute the regional abnormality scores from the regional features. Finally, we will describe the detailed workflow to solve the detection and localization problem using the abnormality scores as features.

Before going into detail, we briefly discuss the following major steps in the overall method:

1. **Global and regional feature extraction**: In this step, we register normal images to normal images to construct $\pi_w(h)$. From these registration mappings, we derive a set of global features. From these global features, we will extract a set of regional features.

2. **Regional abnormality score computation**: Given a patient image p, we register the template images to p. This ia normal-to-abnormal registration because p has a tumor. Then we derive the global and regional features from these normal-to-abnormal

registration mappings and compare them with $\pi_w(h)$ from the previous step to compute regional abnormality scores.

3. **Classification and localization workflow**: Finally, we use the regional abnormality scores as machine learning features to solve the mass effect detection problem for patient p. For localization, we visualize the abnormality maps and report the brain regions corresponding to a few maximum abnormality scores and compare them with the tumor displacement predictions u^* obtained from `GLIA`.

In the following sections, we describe these steps in detail.

Global and regional feature extraction

This section will describe a step-by-step method to define global and regional features for brain regions r. Because we have moved to regional analysis, $\pi_w(h)$ is now a univariate distribution defined for each brain region r. The regional features characterize the univariate distribution $\pi_w(h)$ for each brain region. We need the regional features to compute regional abnormality scores for a new sample which will be further used as machine learning features for solving the mass effect classification problem and for localization of the mass effect.

As explained before, we assume that we have two sets of *normal* MRI scans. These will be used to model the PDF of registration displacement vector fields $\boldsymbol{w}$ from normal-to-normal brains. The first group is referred to as template images (these are the moving images), and the second group is just called normal images (fixed images). We solve pairwise deformable image registration problems between template images a_i, $i = 1, \ldots, n_a$ and reference images h_j, $j = 1, \ldots, n_h$ to obtain displacement vector fields $\boldsymbol{w}_{ij}$ which deform a_i to h_j

$$\boldsymbol{w}_{ij} = \mathcal{R}\left(a_i, h_j\right) \tag{6.3}$$

For simplicity, we have changed our notation from $\boldsymbol{w}_{a_i,h_j}$ to $\boldsymbol{w}_{ij}$. In our method, we use the

displacement two-norm scalar field $w_{ij} \coloneqq w_{ij}(\boldsymbol{x}) = \|\boldsymbol{w}_{ij}(\boldsymbol{x})\|_2$. We do this to simplify our analysis because we are only interested in a scalar feature value per region. In addition to w_{ij}, we consider another scalar field

$$d_{ij} \coloneqq d_{ij}(\boldsymbol{x}) = \|\boldsymbol{w}_{ij}(\boldsymbol{x})\|_2 \nabla \cdot \boldsymbol{w}_{ij}(\boldsymbol{x}) \tag{6.4}$$

d_{ij} contains information about volumetric compression or expansion of tissues in registration. Visual observation of MRI scans of GBM patients shows that a growing tumor typically *compresses* and *displaces* healthy tissues surrounding it. For example, the lateral ventricles are highly compressible tissues in the brain and a growing tumor mass near it can both compress and displace it. If the mass effect is significant, it can even lead to a midline shift in the brain. We hope that w_{ij} and d_{ij} can effectively model the displacement and compression components of mass effect respectively. We refer to w_{ij} and d_{ij} as global features. We define a generic global feature as $q_l(\boldsymbol{x})$. Therefore,

$$q_1(\boldsymbol{x}) = w_{ij}(\boldsymbol{x}) \tag{6.5}$$

$$q_2(\boldsymbol{x}) = d_{ij}(\boldsymbol{x}) \tag{6.6}$$

Following the reasoning in §6.1.1, we now transform the global multi-dimensional features q_l to regional features. First, we define brain regions r with domain $\Omega_r \subset \Omega$ in every template brain a_i (see Figure 6.1). We will discuss how these regions are acquired in §6.2. Then we define the following regional feature

$$\mu_r(q_l) = \frac{1}{V_r} \int_{\Omega_r} q_l \, \mathrm{d}\boldsymbol{x} \tag{6.7}$$

which is the restricted mean of q_l in region r. V_r is the volume of region r in voxel units. Using $\mu_r(q_l)$ as a regional feature instead of q_l results in a smoother distribution for $\pi_w(h)$. To summarize, for each pair of registrations between a_i and h_j, we have the following regional

features

$$X_{ijrl} = \mu_r(q_l) \tag{6.8}$$

where r is the region and q_l is the global feature. First, we average the feature values over the all the templates using (6.1) to reduce anatomical bias towards a single template,

$$X_{jrl} = \frac{1}{n_a} \sum_{i=1}^{n_a} X_{ijrl} \tag{6.9}$$

for $j = 1, \ldots, n_h$. For region r and global feature q_l, X_{jrl} is j^{th} sample from a univariate $\pi_w(h)$. We can now compute regional outlier scores for a patient p using the regional features $X_{jrl} \sim \pi_w(h)$.

Regional abnormality score computation

Given a patient image p, we calculate the regional abnormality scores with respect to the healthy distributions $X_{jrl} \sim \pi_w(h)$. As a first step, we register the template images a_i, $i = 1, \ldots, n_a$ to the patient brain image p to get the registration displacements $\boldsymbol{w}_i$

$$\boldsymbol{w}_i = \mathcal{R}\left(a_i, p\right) \tag{6.10}$$

Then using the regional feature extraction method presented in §6.1.2, we extract the regional features Y_{irl} for p and average them over the templates

$$Y_{rl} = \frac{1}{n_a} \sum_{i=1}^{n_a} Y_{irl} \tag{6.11}$$

Now we can compute the outlier distance of the patient regional feature Y_{rl} from the normal features $X_{jrl} \sim \pi_w(h)$. We assume that $\pi_w(h)$ is unimodal per region, then using

Gauss's inequality, we define the abnormality score as

$$\mathcal{S}_{rl} = \frac{Y_{rl} - m_{rl}}{2\tau_{rl}/\sqrt{3}} \tag{6.12}$$

where m_{rl} is the discrete mode of X_{jrl} over index j and $\tau_{rl} = \mathbb{E}_j(X_{jrl} - m_{rl})^2$. We calculate m_{rl} by constructing a histogram of the features X_{jrl} over index j and then compute the average value in the bin with the highest density. We consider Gauss's inequality because it does not require any assumptions about the shape of $\pi_w(h)$ except that it be unimodal. To summarize, $\mathcal{S}_{rl}$ denotes the scalar abnormality score in region r using global feature q_l. $\boldsymbol{\mathcal{S}}_l \in \mathbb{R}^{n_r}$ denotes the abnormality score vector for the entire brain image using global feature type q_l.

The parameters in the overall method are:

- n_a – ***number of template images*** a: We use $n_a = 12$ template images. The number and variability of template images dictate how well $\pi_w(h)$ generalizes normal brain anatomy. Higher number of template images will capture more variability in $\pi_w(h)$. In the current work, we have not explored the sensitivity analysis of the quality of abnormality scores for a different number of templates. Currently, we average features from all templates, see (6.9) and (6.11) for normal and abnormal cases respectively. However, in practice, one could select specific templates which are more suitable for a target patient p. For example, compressed lateral ventricles are among the most common visual cues for mass effect. It could perhaps be more useful if, for a target patient p, we average features only from the templates for which the ventricles volumes are higher than the ventricles in p because image registration can easily capture this compression model for ventricles. This could enable us to build a better and more localized patient-specific mass effect model. We have also not explored age and gender stratification for template selection. These factors also dictate how some brain structures look. For example, older people typically have smaller gray matter volumes. This

could induce a bias $\pi_w(h)$ and lead to incorrect results for predictions of a different age group. Therefore, building $\pi_w(h)$ conditional on the age group and gender could provide more accurate abnormality scores.

- n_h ***- number of normal images*** h: The number of normal images used for construction of $\pi_w(h)$ needs to as high as computationally feasible to capture more variability in healthy anatomy. In this work, we use $n_h = 700$ normal images. These images are of healthy individuals who do not have any brain disease.

- Ω_r ***- parcellation of regions in template images*** a: The number of regions in the brain parcellation defines the normal feature distribution. If a region is too large, then we lose accuracy in localization. If the regions are too small, the related features are too noisy. The relative size of the regions needs to be uniform; otherwise, we lose detail in some regions and capture noise in others. The brain parcellations we pick in our experiments correspond to anatomical regions and are thus unstructured. This is better than creating a structured parcellation. Because image registration aligns brain anatomy, the displacements from registration are often localized and correlated within these brain structures.

6.2 Datasets

In Table 6.2, we list the image datasets we use in our experiments. We describe them in detail below.

NRM-A – template image dataset

This dataset consists of the first set of normal images a used as moving images for registration to normal h or patient p images. We use $n_a = 12$ T1-weighted MR images. We segment these brains into white matter (WM), gray matter (GM), and cerebrospinal fluid (CSF)

Table 6.2: *We list the image datasets we used for conducting mass effect classification and localization experiments.* ***NRM-A****,* ***NRM-H*** *and* ***TUMOR-NRM-S*** *are non-overlapping sets of normal images.*

Dataset	Description	Number of images
NRM-A	Template images a	12
NRM-H	Normal images h	700
TMR-NRM-S	Normal images h_0 used for creating **TMR-S**	500
TMR-S	Synthetic tumor images h_0^T	2399
TMR-C	Real clinical tumor images	422

using the tool `fast` [171] from the FSL software package. We use this segmentation for deformable image registration. We use a brain parcellation with 149 regions for regional feature extraction for five template images made available to us from the CBICA group at the University of Pennsylvania. The parcellations for the remaining seven images were done using a multi-atlas segmentation scheme using ANTs [3]. The initial parcellation consisted of several brain regions with extremely small or large volumes that could bias the feature statistics more towards these outlier regions. We merged small regions and implemented a moment of inertia-based volume splitting method to break down large regions. We recursively split each of the large regions by a plane about which the moment of inertia is maximum. We repeat this until the region volumes have a more uniform distribution. From this new volume normalized parcellation, we have $n_r = 92$ brain regions. We show one of the template images before and after volume normalization in Figure 6.2

We also obtain brain parcellations from the Automated Anatomical Labelling(AAL) [130] atlas and repeat the volume normalization for this atlas. Then we transfer the AAL parcellation to template images using deformable image registration using ANTs. The number of regions from the AAL parcellation is $n_r = 92$. We show this template image parcellation in Figure 6.3. We conducted all downstream experiments for both the MUSE parcellation and the AAL parcellation but only reported results for the MUSE parcellation because we did not observe significant differences between them.

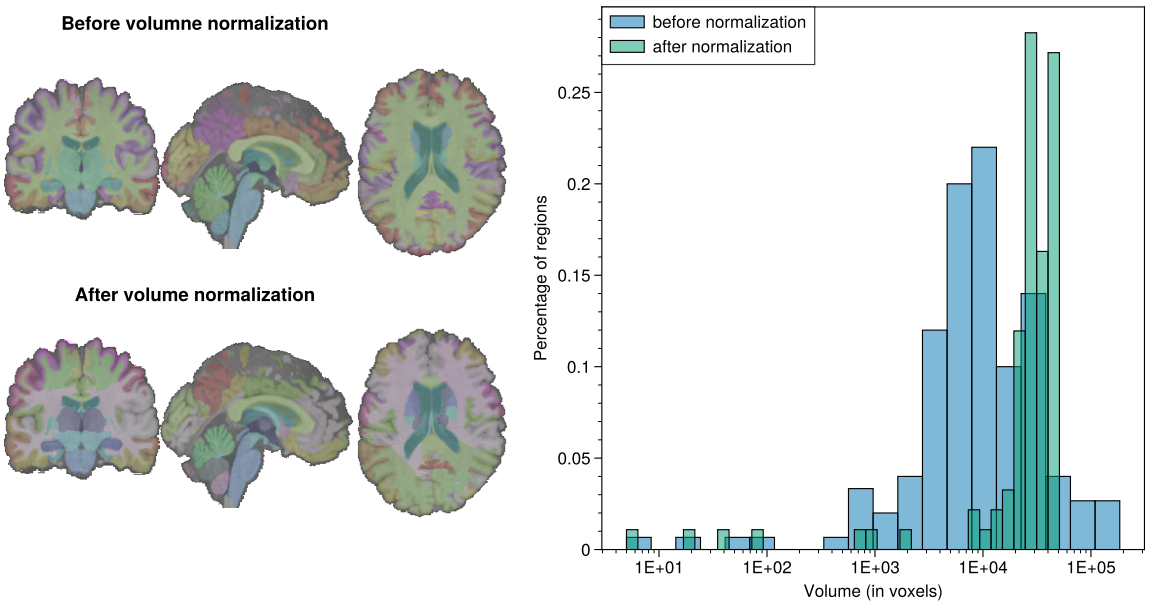


Figure 6.2: ***Template image MUSE parcellation volume normalization***. *On the left, we show a normal brain MUSE parcellation before and after post-processing to normalize the number of voxels per region. We split the larger white matter regions and combine the smaller gray matter regions. We show the volume histogram in the figure on the right.*

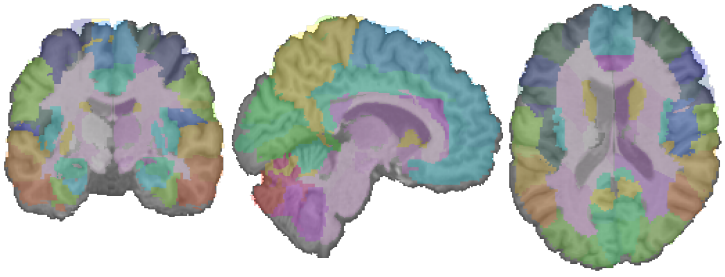


Figure 6.3: ***Automated Anatomical Labelling(AAL) template parcellation***. *Here we show the volume-normalized AAL parcellation transferred to the template image.*

NRM-H – normal image dataset

This dataset consists of the second set of normal images h, which are used to construct the distribution $\pi_w(h)$. The Human Connectome Project (HCP) [87] has 1200 brain images of young, healthy adults aged 21-35yrs. We sub-sampled 700 images from this dataset to construct $\pi_w(h)$. We segmented these images into WM, GM and CSF using `fast` [171] from FSL package. We use these segmentations for deformable image registration with the template images.

TMR-C – clinical tumor dataset

This dataset consists of clinical patient MR images with real brain tumors, which we use to test our scheme. We use the BraTS challenge 2018 dataset [16] and the Penn brain tumor dataset. We use a neural network [67] to segment the whole tumor and `fast` to segment the healthy part of the brain into WM, GM, and CSF. We use `GLIA` [145] to calibrate a biophysical tumor growth model $\mathcal{T}$ [146] and estimate tumor growth parameters which includes the prediction of tumor displacement field u^*. The tumor growth parameters are $\boldsymbol{\theta} = \{c_0, \rho, \kappa, \gamma\}$ where c_0 is the tumor seed location, ρ is the tumor reaction coefficient, κ is the tumor diffusion coefficient and γ is the tumor mass effect parameter. We use the distribution of these inverted model parameters $\hat{\boldsymbol{\theta}}$ from the BraTS 2018 dataset to construct our synthetic tumor dataset, which we discuss in the following section. We also acquired expert mass effect annotations for 58 out of the 405 images in this dataset. These annotations were done by two expert radiologists, Dr. Suyash Mohan, MD, and Dr. Seyed Ali Nabavizadeh, MD, who are Penn medicine physicians and assistant professors of radiology at the hospital of the University of Pennsylvania. The experts provided individual ratings along with a consensus rating. The ratings were mild (22 cases), moderate (29 cases), and severe (7 cases) mass effect. None of the presented cases were annotated as having no mass effect.

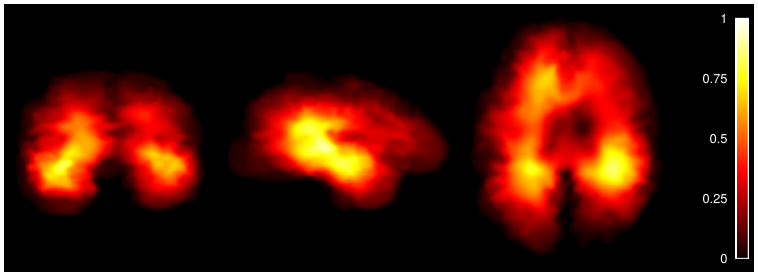


Figure 6.4: *Reconstructed tumor atlas from the inversion of the BraTS 2018 dataset using a 3D tumor inversion model [145]. The voxel intensities are equal to the probability of a tumor being found at that location. We use this atlas to sample tumor seed locations in the white matter(WM) in a healthy image h_0 to run the forward tumor growth model $\mathcal{T}$ to create synthetic tumors h_0^T.*

TMR-S – synthetic tumor dataset

We create a synthetic brain tumor dataset with known ground truth mass effect to verify our scheme. We use a single species tumor growth model $\mathcal{T}$ with mass effect [146] to create synthetic tumor images. This is the same model we used to reconstruct tumor model parameters $\hat{\boldsymbol{\theta}}$. In order to match the size, shape and mass effect distributions of the synthetic tumors with the tumors in **TMR-C**, we sample the tumor model parameters $\boldsymbol{\theta}$ from the distribution $\hat{\boldsymbol{\theta}}$ which are the inverted model parameters of BraTS dataset in **TMR-C** in §6.2. In particular, we use the following workflow to generate synthetic tumor images

1. First, we sample 150 normal images h_0 from the Alzheimers Disease Neuroimaging Initiative(ADNI) dataset. We sample an additional 350 images from the remaining HCP young adult dataset. We denote these 500 normal images h_0 images as the **NRM-TMR-S** dataset.

2. To generate a tumor in a normal image h_0, we run the forward tumor model $\mathcal{T}(h_0, \boldsymbol{\theta})$ [147].

3. We sample the tumor initial condition c_0 from a tumor atlas. This tumor atlas is a voxel-wise probability map of the reconstructed tumors of the BraTS dataset using `GLIA`(see Figure 6.4). We sample five different c_0 for each h_0. Each c_0 consists of a set of image voxel locations we refer to as tumor seeds. To sample c_0, we take the following

steps

(a) First, we randomly choose between a mono-focal and multifocal tumor. We set the probability of a tumor being multifocal to 10% based on the statistics of **TMR-C** dataset.

(b) Then, we set the first tumor seed, a single voxel location, by sampling a voxel from the tumor probabilistic atlas and then checking if it falls in the white matter (WM) region of h_0. We repeat this process until the seed falls in WM.

(c) Then, we sample five additional tumor seeds in a ball of a radius of 30 voxels around the first tumor seed.

(d) We assign random weights to each tumor seed. These weights determine the initial growth rate of the tumor. We normalize the weights to $[0, 1]$.

(e) For multifocal tumors, we sample another tumor seed outside a ball of radius 150 voxels from the first tumor seed in step b and then repeat steps c and d.

4. We sample (ρ, κ, γ) from the distribution of $\hat{\boldsymbol{\theta}}$

5. Then, using the sampled parameters, we run the forward tumor model

$$[h_0^T, u^*] = \mathcal{T}(h_0, \boldsymbol{\theta}) \tag{6.13}$$

where h_0^T is the synthetic tumor image and u^* is the voxel-wise ℓ_2 norm of tumor displacement vector field. Out of the 2500 tumor growth simulations, we keep 2399 cases for which the tumor volume and displacement ℓ_1 norm $\|u^*\|_1$ fall within the limits of **TMR-C**.

We show a few synthetic tumors in Figure 6.5 and compare the joint tumor volume and tumor mass effect distribution of **TMR-S** with **TMR-C** in Figure 6.6.

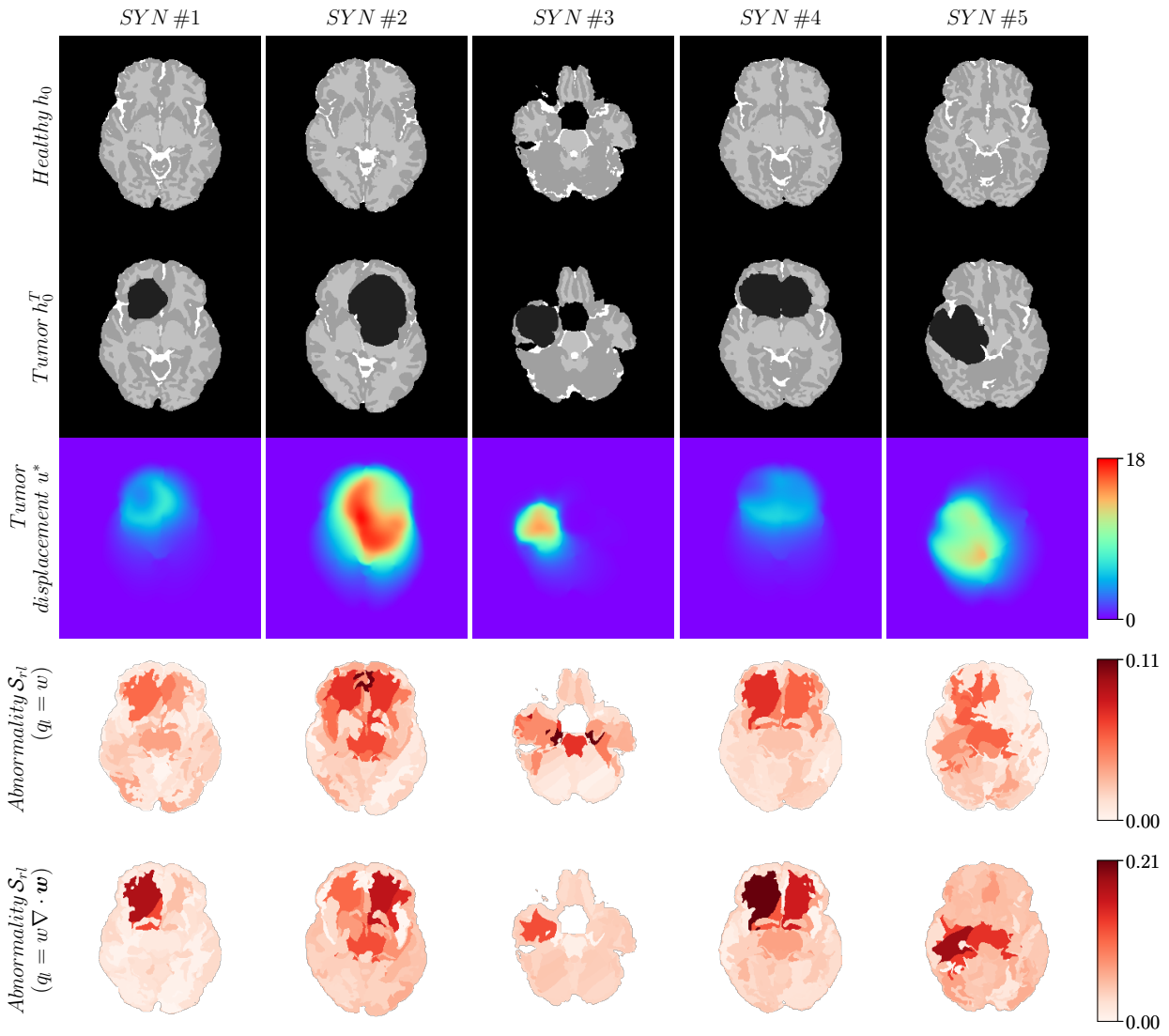


Figure 6.5: *We show five synthetic tumors SYN #1-5 and the associated tumor displacements and abnormality scores. We show the healthy subject h_0 which we use to grow the synthetic tumor, the resulting tumor image h_0^T and the displacement two-norm field u^* from the tumor growth model. The maximum displacements are 6.7, 17.6, 14, 5.7, 13.1 mm respectively. In the last two rows, we show the abnormality score maps from the statistical model for two registration global features $q_l = w$ and $q_l = w\nabla \cdot \boldsymbol{w}$.*

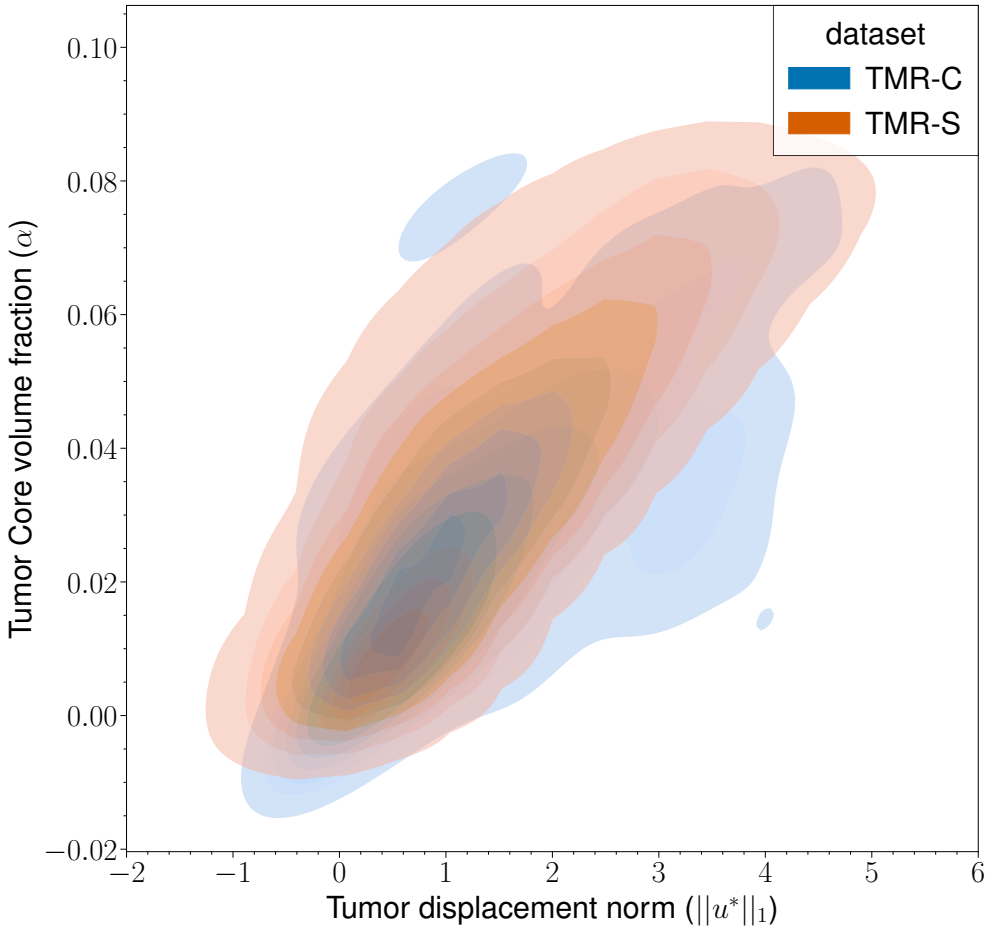


Figure 6.6: *Comparison of joint Gaussian kernel density of tumor volume fraction and tumor mass effect for* ***TMR-S*** *and* ***TMR-S*** *datasets. For* ***TMR-C****, the voxel-wise tumor displacement two-norm field* u^* *is approximated using* `GLIA` *[148]. For* ***TMR-S****,* u^* *is known from the tumor growth model* $\mathcal{T}$ *which is the same model used in* `GLIA`*. For* ***TMR-C****, the tumor volume fraction* α *is equal to the tumor core (which comprises of necrotic and enhancing tumor) volume fraction. For* ***TMR-S****, the tumor growth model* $\mathcal{T}$ *is a single species model which does not distinguish between different tumor phenotypes and so, we only consider the whole tumor volume fraction for this dataset. We do not consider tumor edema in the volume fractions. The joint densities are approximated using a Gaussian kernel, hence the negative values for* $||u^*||_1$ *and* α*.*

Image Registration:

Our analysis of mass effect is based on the statistics of mappings between MR images. These mappings are constructed using image registration. We use `CLAIRE` to evaluate the registration operator $\mathcal{R}$. In Chapters 1 and 2, we developed `CLAIRE` for single and multi-GPU applications. It is a fast 3D image registration software capable of registering two clinical images of size 256^3 in under 5 seconds. This makes `CLAIRE` suitable for running a large number of small registration problems which are needed to construct $\pi_w(h)$. We convert the T1-weighted MR images into image segmentation in our current analysis and register these. For the registrations $\mathcal{R}(a_i, h_j)$ between moving template image a_i and fixed normal image h_j, we segment both a_i and h_j into WM, GM and CSF and register these segmented images. For the registrations $\mathcal{R}(a_i, p)$ between moving template image a_i and fixed patient image p, we segment the healthy tissue in both images into WM, GM, and CSF. Since the normal images do not have a tumor, it is impossible to find point correspondence to the tumor region of the patient. For this reason, we replace the tumor label with the WM label under the assumption that most of the tumor growth takes place in the WM. The parameter configuration of `CLAIRE` for the registrations performed in this chapter is the same as for the runs in Chapter 1 for image size 256^3. All registrations performed in this study were executed on the TACC Frontera system in single precision. Frontera hosts 90 NVIDIA Quadro RTX5000 nodes. Each node is equipped with four GPUs with 4×16 GB GPU RAM (64 GB aggregate) and 2 Intel Xeon E5-2620 v4 (*Broadwell*) CPUs.

6.3 Detection problem

In §6.1.2, we detailed the method to compute regional abnormality scores for a GBM patient image p. Now using these abnormality scores, we want to solve the mass effect detection problem. This is composed of two sub-problems – *(i) classification problem*: classify p into one of the three classes – mild, moderate or severe mass effect and, *(ii) regression problem*:

estimate the tumor displacement ℓ_1 norm $\|u^*\|_1$ for p. We will treat these two problems separately.

Before we describe these two problems, we do a comparison study to check how different the abnormality scores $\mathcal{S}_{rl}$ for patients with mass effect (ME) and patients with no mass effect (NME). We also want to see how abnormality scores and tumor model displacements compare with expert annotations for **TMR-C** dataset. In light of these comparisons, we ask the following questions:

(Q1) Can we find some aggregate abnormality score using regional scores $\mathcal{S}_{rl}$ which is good at separating no mass effect and mass effect cases?

(Q2) How do the abnormality scores and tumor model displacements compare with expert annotations for mass effect?

(Q1) Can we find some aggregate abnormality score using regional scores $\mathcal{S}_{rl}$ which is good at separating no mass effect and mass effect cases?

The aim of this study is visualize the mass effect (ME) and no mass effect (NME) patient populations using some abnormality score aggregator $g(X)\colon X \mapsto g(X) \in \mathbb{R}$, where X is some subset of the regional abnormality scores $\mathcal{S}_{rl}$. We want to find $g(X)$ which maximizes the distance between π_{NME} which is the PDF of $g(X)$ when X comes from NME patients and π_{ME} which is the PDF of $g(X)$ when X comes from ME patients. Although this problem formulates like a binary classification problem, our intention here is only to visualize the distributions of NME and ME cases using $g(X)$.

Datasets: We consider the **TMR-C** and **TMR-S** datasets for the ME cases and **TMR-NRM-S** dataset for the NME cases.

Selecting features X: We select features X which are a subset of regional abnormality scores $\mathcal{S}_{rl}$ where r is the region and l corresponds to global registration feature q_l. We consider

several different combinations for X: *(i)* X is the volume weighted (VW) abnormality scores $\mathcal{S}_{rl}$, i.e., we weight $\mathcal{S}_{rl}$ with the volume fraction V_r/V of the region r. Here, V_r is the volume of r and V is the volume of the brain in voxels. Volume weighting the scores $\mathcal{S}_{rl}$ ensures that larger regions with higher scores contribute to more mass effect. *(ii)* We consider rank ordered (RO) abnormality scores $\mathcal{S}_{rl}$, i.e, we sort the scores in descending order. For *RO*, we consider sub-selecting the $\texttt{top}_k$ features which are the top 20 maximum $\mathcal{S}_{rl}$ and *(iii)* We also consider different global registration feature q_l separately, i.e. $q_l = w$ or $q_l = w\nabla \cdot \boldsymbol{w}$.

Selecting different $g(X)$: We select different candidate $g(X)$. We try $\mathsf{mean}(X)$, $\max(X)$ and $\mathsf{stddev}(X)$. $\mathsf{stddev}(X)$ computes the standard deviation of the feature vector X.

Measuring distance between π_{NME} and π_{ME}: We compute the Kullback Leibler (KL) divergence between π_{NME} and π_{ME}.

$$D_{\mathsf{KL}}(\pi_{\mathsf{NME}}(g(X))||\pi_{\mathsf{ME}}(g(X))) = \int_{-\infty}^{\infty} \pi_{\mathsf{NME}}(g(X)) \log\left(\frac{\pi_{\mathsf{NME}}(g(X))}{\pi_{\mathsf{ME}}(g(X))}\right) dg(X) \tag{6.14}$$

We pick the $g(X)$ which maximizes $D_{\mathsf{KL}}(\pi_{\mathsf{NME}}(g(X))||\pi_{\mathsf{ME}}(g(X)))$.

Results: Based on the KL-divergence values, we found that $g(X) = \mathsf{stddev}(X)$ when X is the $\texttt{top}_{20}$ VW abnormality scores $\mathcal{S}_{rl}$ for $q_l = w\nabla \cdot \boldsymbol{w}$ works best for both **TMR-S** and **TMR-C** datasets. We show the distribution for $g(X)$ in Figure 6.7 where we also sub-stratify the ME patients based on equal quantiles of the tumor displacement ℓ_1 norm $\|u^*\|_1$. In Figure 6.8, we show the scatter plot of $g(X)$ with $\|u^*\|_1$ to show the high variance in values of $g(X)$ for a single value of $\|u^*\|_1$. This causes issues in the regression problem as we see later in §6.3.2.

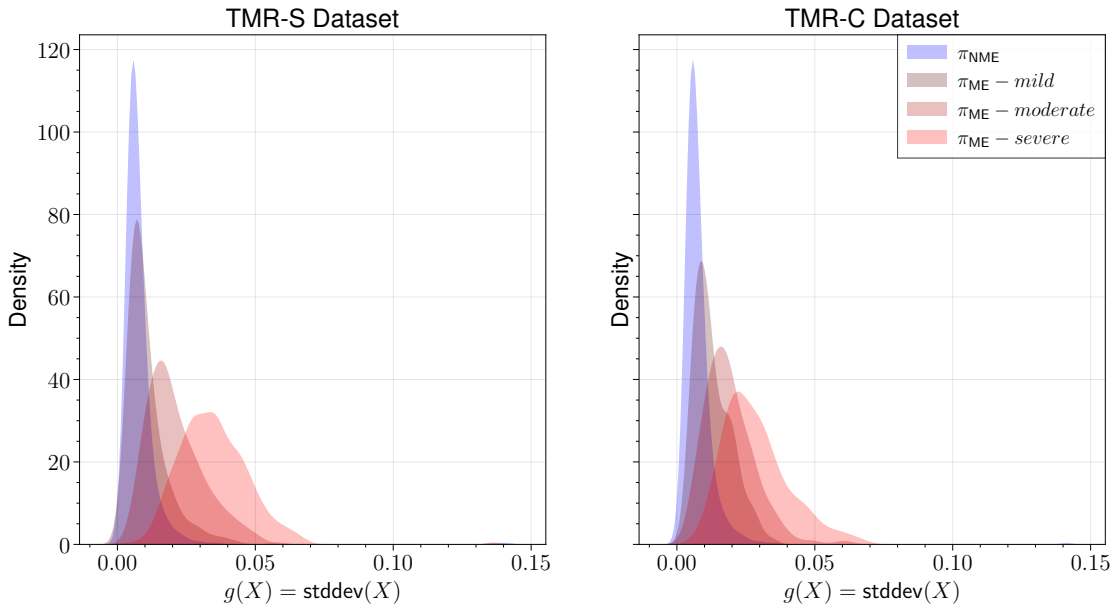


Figure 6.7: ***Best abnormality score feature aggregator:*** *We show the distribution of* $g(X) = \text{stddev}(X)$ *as an abnormality score aggregator where* $\text{stddev}(X)$ *is the standard deviation of* X*.* X *is the* $\texttt{top}_{20}$ *volume weighted (VW) abnormality scores* $\mathcal{S}_{rl}$ *for global registration feature* $q_l = w\nabla \cdot \boldsymbol{w}$*. We stratify the distributions for the ME cases into mild, moderate and severe (based on equal quantiles of* $\|u^*\|_1$*) to show the separation of* π_{NME} *and* π_{ME}*.*

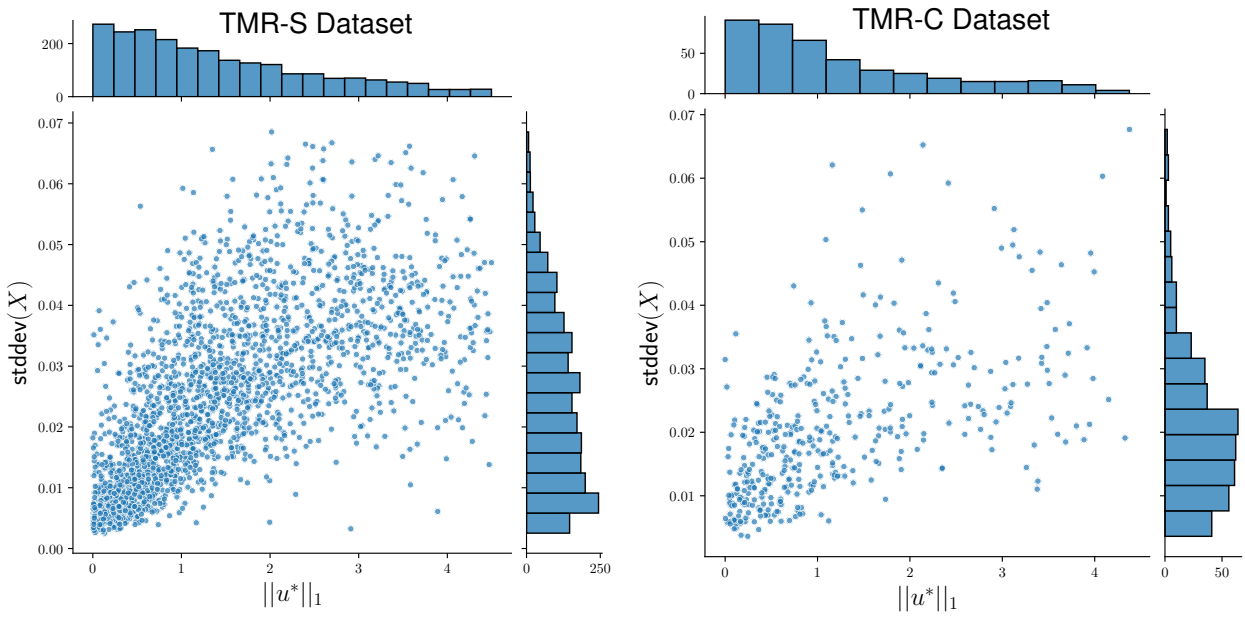


Figure 6.8: *We show the comparison of aggregate abnormality score* $\mathsf{stddev}(X)$ *with* $\|u^*\|_1$. *Here* X *is the* $\texttt{top}_{20}$ *volume weighted (VW) abnormality scores* $\mathcal{S}_{rl}$ *for global registration feature* $q_l = w\nabla \cdot \boldsymbol{w}$. *The plot demonstrates high variance for a given* $\|u^*\|_1$ *with varying degrees of aggregate abnormality scores from the statistical model.*

How do the abnormality scores and tumor model displacement compare with expert ratings for mass effect?

We want to check how the abnormality scores $\mathcal{S}_{rl}$ and tumor model displacements u^* compare with expert annotations. As mentioned before in §6.2, we acquired mass effect ratings for 58 handpicked brain tumor images from the **TMR-C** dataset from a couple of expert radiologists. The ratings are mild, moderate, and severe mass effect. We visualize three exemplar cases with these annotations in Figure 6.9.

Now, we define three different sets of features that we want to compare with the expert mass effect ratings:

Tumor volumetric features: We consider volumetric tumor features because tumor size can influence the mass effect. Larger tumors are more likely to cause more mass effects, and smaller tumors are less likely. We draw comparisons for tumor edema volume fraction λ_{ed}, tumor core (necrotic and enhancing tumor) volume fraction λ_{tc} and whole tumor volume

fraction λ_{wt} which are ratios of volumes of tumor edema, tumor core and whole tumor to the whole brain volume respectively. We show the comparison with expert ratings in Figure 6.10.

Tumor model features: We consider the following features – $\|u^*\|_\infty$, $\|u^*\|_1$ and $\|u^*\|_{1,dist}$. $\|u^*\|_{1,dist} := \int_\Omega \frac{|u^*(\boldsymbol{x})|}{\iota(\boldsymbol{x})} d\boldsymbol{x}$ is a distance weighted norm where $\iota(x)$ is the distance of a voxel $\boldsymbol{x} \in \Omega$ in the brain from the tumor core. To calculate $\iota(x)$, we extracted the tumor core surface and put 100 sample points on this surface uniformly distributed over the entire surface. Then given a voxel at $\boldsymbol{x}$, we compute the minimum euclidean distance from the sample points to $\boldsymbol{x}$. We show the comparison with expert ratings in Figure 6.11.

Aggregate abnormality score: We only compare expert ratings with the aggregate abnormality score from **Q1** which is $\mathsf{stddev}(X)$ where X is the $\mathtt{top}_{20}$ volume weighted (VW) abnormality scores $\mathcal{S}_{rl}$ for global registration feature $q_l = w\nabla \cdot \boldsymbol{w}$. We show the comparison with expert ratings in Figure 6.12.

Observations: In Figure 6.9, we see that the severe mass effect case **ABAF** has a large tumor with a significant midline shift in the brain. The mild case **AASX** on the other hand, has a tiny tumor and localized mass effect affecting only the surrounding gray matter sulci. The tumor model predicts the highest displacement for the moderate case, and the displacements are very localized around the tumor. The abnormality maps $\mathcal{S}_{rl}$ from the statistical model on the other hand are more spread out for both $q_l = w$ and $q_l = w\nabla \cdot \boldsymbol{w}$. However, we see visually that the abnormality scores are large for some gray matter and white matter regions with visible deformations around the tumor.

In Figure 6.10, we do not see a clear defining boundary which separates mass effect severity using tumor volumetric features but we do observe an increasing trend. Similarly in Figure 6.11, for the tumor model, only $\|u^*\|_\infty$ and $\|u^*\|_1$ seem to have some separation between the moderate and severe cases. Only $\|u^*\|_1$ and $\|u^*\|_{1,dist}$ seem to separate mild and moderate cases. In Figure 6.12, the abnormality score clearly separate the severe cases from the mild and moderate cases.

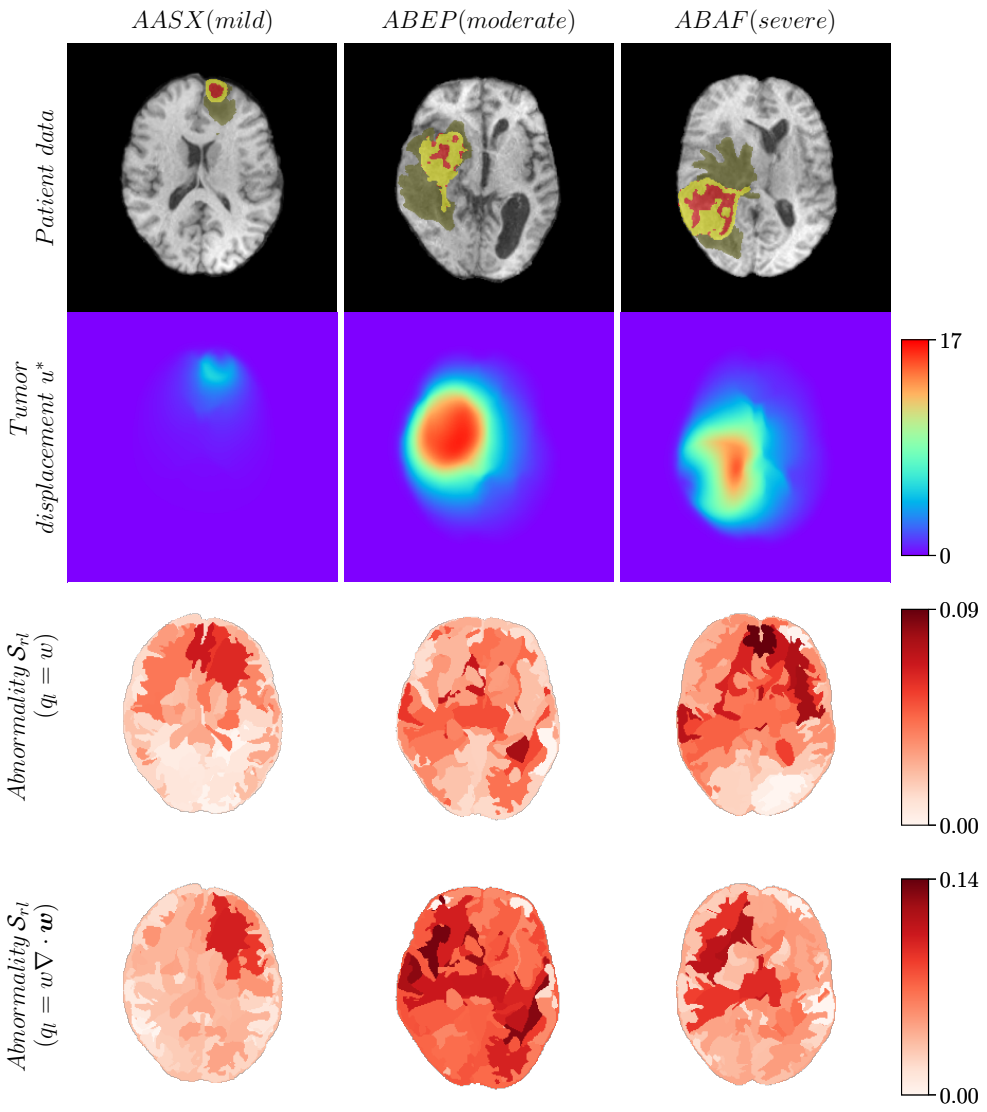


Figure 6.9: ***Visualization of mass effect with expert ratings***. *In the first row, we show the T1-weighted MR images of three patients (AASX, ABEP, ABAF) from the* ***TMR-C*** *dataset which have been annotated by experts for the mass effect severity. The tumor is segmented into Edema(green), necrotic core (red) and enhancing tumor (yellow) using a deep learning framework. The experts were only presented with the T1-weighted contrast enhanced images and not the tumor segmentation. In the second row, we show the reconstructed tumor displacement u^* from the tumor inversion model* `GLIA` *[148]. The maximum displacement from left to right are 5.4, 16.29, 15 mm respectively. In the third and fourth row, we show the abnormality score maps $\boldsymbol{\mathcal{S}}_l$ for different global registration features $q_l = w$ and $q_l = w\nabla \cdot \boldsymbol{w}$ respectively.*

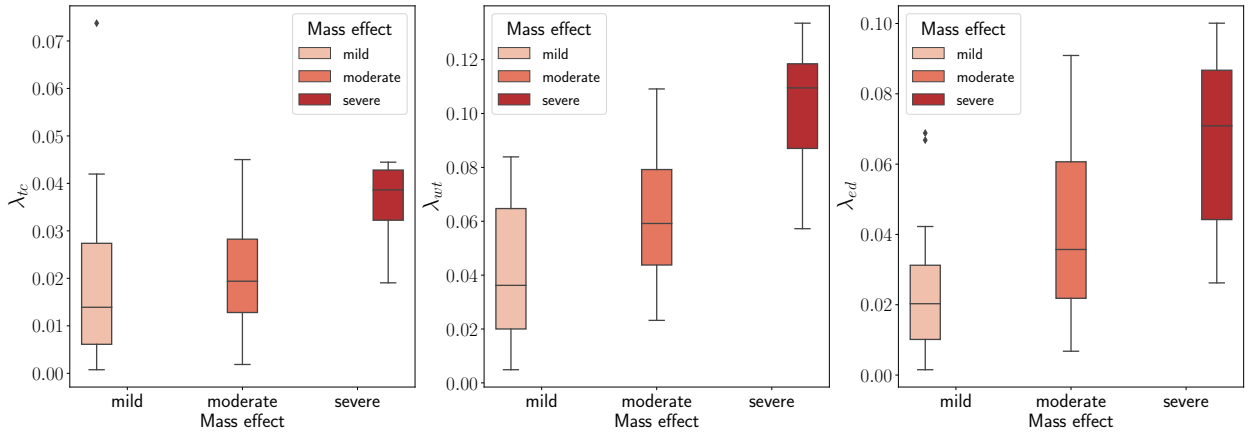


Figure 6.10: *We compare tumor volumetric features edema volume fraction λ_{ed}, tumor core volume fraction λ_{tc} and whole tumor volume fraction λ_{wt} with the expert mass effect ratings.*

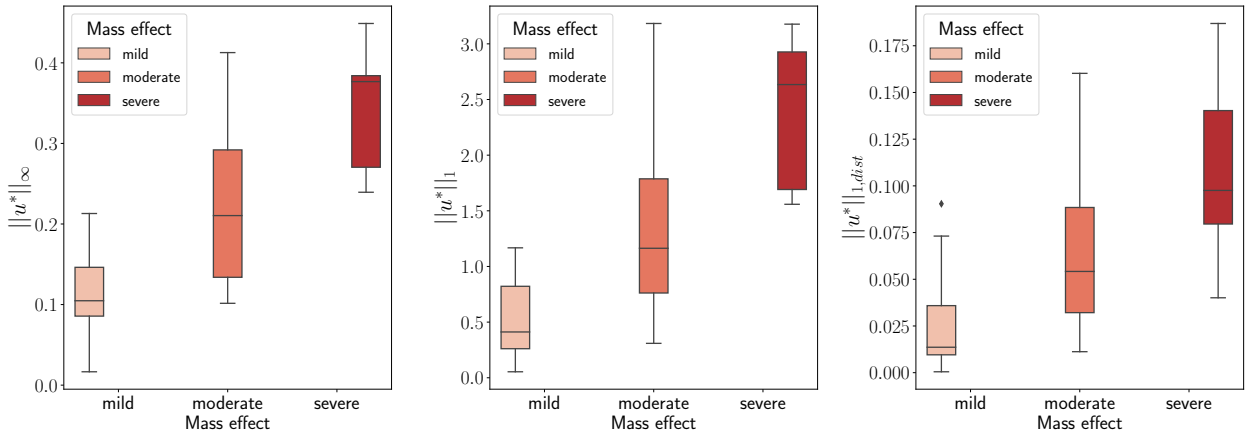


Figure 6.11: *We compare tumor model displacement features $\|u^*\|_\infty$, $\|u^*\|_1$ and $\|u^*\|_{1,dist}$ with the expert mass effect ratings.*

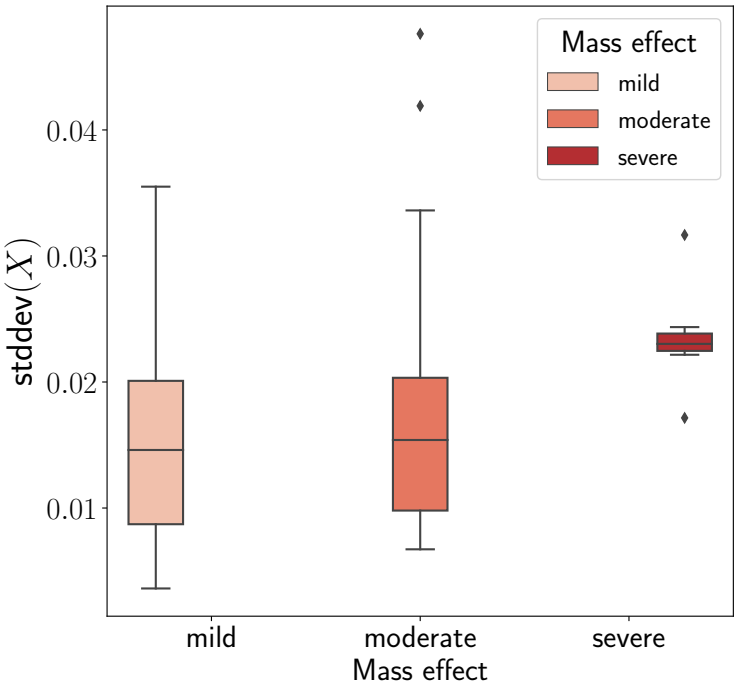


Figure 6.12: *We compare the aggregate abnormality score* stddev(X) *with the expert mass effect ratings.* X *is the* $\texttt{top}_{20}$ *volume weighted (VW) abnormality scores* $\mathcal{S}_{rl}$ *for global registration feature* $q_l = w\nabla \cdot \boldsymbol{w}$.

6.3.1 Classification problem

Here, we will solve the mass effect classification problem for a GBM patient p. We will use the abnormality scores $\mathcal{S}_{rl}$ as features and train a classifier to predict mild, moderate or severe mass effect for p.

We select **TMR-C** and **TMR-S** as our training datasets and use these to train a classification model to predict mass effect classes for images from the **TMR-C** dataset only. We also consider a binary classification problem (mild versus severe mass effect) as a more straightforward test case.

Training features: The training features are the regional abnormality scores $\mathcal{S}_{rl}$ where r is the brain region and l is the index for global registration feature q_l which can be w or $w\nabla \cdot \boldsymbol{w}$. We compute $\mathcal{S}_{rl}$ for all images in **TMR-S** and **TMR-C**. We have $n_r = 92$ regions and $n_l = 2$

global registration features. Therefore, we have $n_r n_l = 184$ regional features per image. We consider different global features q_l separately and together during training. We test whether volume weighting (VW) the scores $\mathcal{S}_{rl}$ helps in improving classification accuracy. Moreover, a key issue in training a classifier using $\mathcal{S}_{rl}$ is that these features are regional. Our training dataset is small and two tumors at different locations and having similar mass effect may activate different sets of features. This can create issues in classification. To address this issue, we rank order (RO) the scores $\mathcal{S}_{rl}$ for each patient image, i.e, we sort the features in descending order, during training and evaluation. For RO features, we also consider sub-selecting the $\texttt{top}_k$ maximum $\mathcal{S}_{rl}$ and try $k = \{20, 92, 184\}$. $k = 184$ corresponds to using the all the features $\mathcal{S}_{rl}$ for all r and l. To summarize, we have the following combinations of features that we will use during training:

- Volume weighted (VW) versus vanilla $\mathcal{S}_{rl}$
- Rank ordered (RO) versus vanilla $\mathcal{S}_{rl}$
- $\texttt{top}_{20}$, $\texttt{top}_{92}$ and $\texttt{topk}_{184}$ rank ordered $\mathcal{S}_{rl}$.

Ground truth annotations: As mentioned earlier in §6.2, we acquired expert annotations for 58 cases from the **TMR-C** dataset. A very small dataset (58 images) with annotations makes it difficult to solve and assess the classification performance. Also, the annotations were done by only two raters because of which there is some bias and uncertainty associated with these ratings and hence cannot be used for training a robust classifier. We address this issue by using `GLIA` to predict tumor displacements u^* for the **TMR-C** dataset. We then compute the ℓ_1 norm of tumor displacements, $\|u^*\|_1$ which is a scalar value and assign ground truth labels to all images in **TMR-C** using equal quantiles of $\|u^*\|_1$. For **TMR-S**, u^* is already known (see §6.2). To create annotations for **TMR-S**, we apply the equal quantile bins of $\|u^*\|_1$ obtained from **TMR-C**. We solve the classification problem for different quantiles of $\|u^*\|_1$ because the ground truth annotations are unknown. For the cases where

the chosen quantiles for $\|u^*\|_1$ are not balanced, i.e., there is a class imbalance, we upsample the under-represented class using random repetition using SMOTE [37].

Data splitting: We split **TMR-C** into training (80%, 320 images) and testing (20%, 85 images) subsets and use the entire **TMR-S** dataset for training only. We use stratified splitting based on the target classes to preserve the class distribution across splits.

Classifier model: We analyze the classification performance using a multi-layer perceptron (MLP)[12] and a random forest[13] model. These models suit well for our problem because we want to learn the nonlinear mapping from regional features to classes. We perform an exhaustive grid search for both models to find the best model hyperparameters using the nested cross-validation framework (see next paragraph on cross-validation). For MLP, we fixed the learning rate to 1e-03, which worked best in our case. We only try to find the number of hidden layers and the number of nodes in each layer, giving the best accuracy. We vary the number of hidden layers from one to three in order to learn as complex features as possible (this is only possible when we have a small number of input features, e.g., when using $\texttt{top}_{20}$ RO $\mathcal{S}_{rl}$, otherwise the model will overfit). For the random forest model, we vary the number of trees in the forest and the minimum number of samples required to be at a leaf node.

Cross-validation – using TMR-C only: In this test case, we check the classification performance by doing both training and testing only on the clinical dataset **TMR-C**. **TMR-C** is a small dataset containing only 405 images. For this reason, we do a cross-validation study to analyze the classification performance. Specifically, we use a nested cross-validation approach to assess model performance. In nested cross-validation, there are two cross-validation loops. The outer loop does standard 5-fold cross-validation while the inner loop does hyperparameter grid search using 3-fold cross-validation on the k^{th} training fold from the outer

[12] https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html
[13] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

loop. This is different from a vanilla $k-$fold cross-validation method where hyperparameter tuning and training are done on the same training data set. This could result in an optimistic estimate of the hyperparameters and overfitting. Nested cross-validation prevents this by training and doing hyperparameters on different parts of training data. Furthermore, in nested cross-validation, the model never sees the validation fold of the data from the outer loop during training. We show the nested cross-validation workflow in Figure 6.13. We measure the accuracy using f_1-score and report the mean and standard deviation of the cross-validation accuracy.

$$f_1 = \frac{2PR}{P+R} \tag{6.15}$$

$$P = \frac{T_p}{T_p + F_p} \tag{6.16}$$

$$R = \frac{T_p}{T_p + F_n} \tag{6.17}$$

where P and R are precision and recall respectively. T_p, F_p and F_n are true positives, false positives and false negative prediction rates respectively. In the ground truth annotation step, we addressed the class imbalance issue by oversampling in minority classes. If the oversampling is not done correctly, some samples could get spilled from the training to validation fold and then we would end up evaluating on the same samples we trained on. To avoid this, the oversampling step is done only within the training fold to avoid spilling of repeated samples into the validation fold.

Cross validation – using TMR-C + TMR-S: In the previous case, we only used **TMR-C** for training and testing, and because it is a small dataset, we could end up overfitting the training set, and the model may not generalize well for unseen data. To help alleviate this problem to some extent, here in this test case, we use both the large synthetic dataset **TMR-S** (2399 samples) and **TMR-C** for training and then test only on **TMR-C**. We have tried to match the tumor size and mass effect distribution of **TMR-S** with **TMR-C** (see Figure 6.6). In light of this, we hope that with a larger number of additional training samples, the model

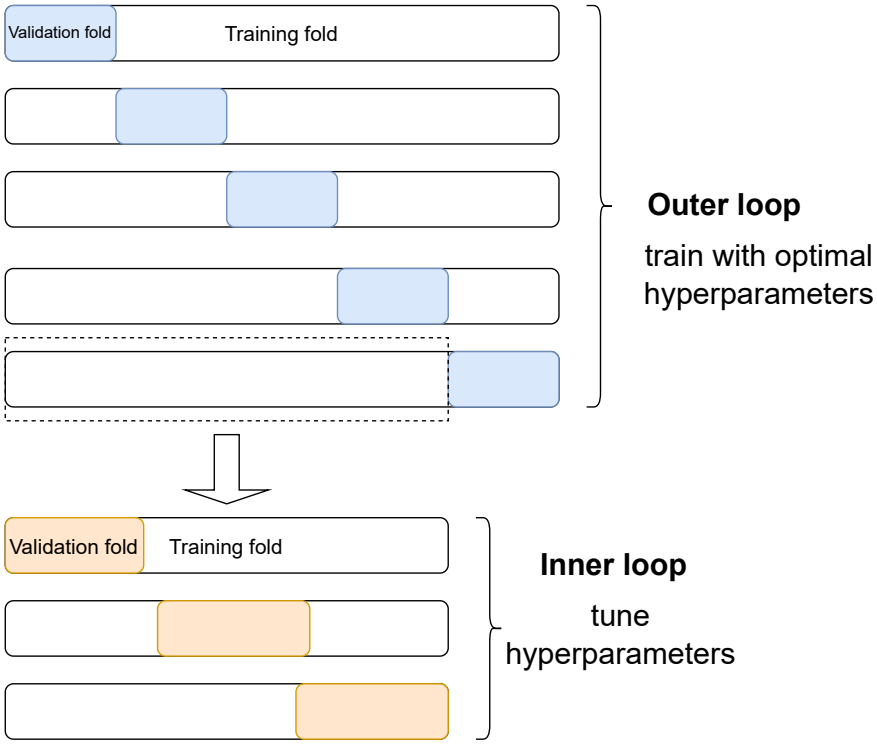


Figure 6.13: *We show the outer and inner loop to explain the nested cross-validation workflow. In the inner loop, the model cross-validates on the k^{th} training fold from outer loop to tune hyperparameters. In the outer loop, we use the tuned hyperparameters to retrain on the training fold and then evaluate the trained model on the validation fold. The validation fold of the outer loop (blue color) is never seen during training.*

can learn to generalize well for unseen data and also provide better classification accuracy. For cross-validation, we follow the same methodology as discussed in the previous paragraph except that here we append all the samples from **TMR-S** to the k^{th} outer training fold of the **TMR-C** dataset. Samples from **TMR-S** are never present in the validation fold because we only validate on **TMR-C** dataset. We use different sample weight ratio β for the samples from **TMR-C** (real) and **TMR-S** (synthetic) datasets respectively during training. We test for the following values of $\beta = \{1, 3, 7, 14\}$. $\beta = 1$ gives equal weight to real and synthetic samples and higher β gives more weight to real samples.

Finding the best classification model: We solve the classification problem for different combinations of training features, quantiles for ground truth, training datasets, and sample weights β to find out the model with the highest mean cross-validation accuracy. We then evaluate this model on the unseen test dataset samples from **TMR-C** and report the results in §6.5.1.

6.3.2 Regression problem

One issue in the classification problem was that the quantiles used to assign ground truth annotations were somewhat arbitrary without any actual clinical meaning. We do try to overcome this problem by predicting $\|u^*\|_1$ by solving the regression problem. We use the same training features, models, training data, and nested cross-validation method from the classification workflow. The error measure here is different, which we describe below.

Regression error measure: We compute the error in the regression predictions by computing the relative error for each sample. For the i^{th} validation sample, we compute the error as

$$e^i_{rel} = \frac{\left|\, \|u^*\|^i_{1,pred} - \|u^*\|^i_{1,ref} \,\right|}{\|u^*\|^i_{1,ref}} \tag{6.18}$$

For each validation fold, we compute $\mathbb{E}_i e^i_{rel}$ and report its mean and standard deviation across validation folds. We call this the CV score. We also compute $\min(e^i_{rel})$ and $\max(e^i_{rel})$

for across all the validation folds and call this the minimum and maximum relative error.

6.4 Localization problem

To solve the localization problem for a patient p we *(i)* visualize the abnormality score $\mathcal{S}_{rl}$ heatmap, and *(ii)* report a few brain regions r with maximum $\mathcal{S}_{rl}$. However, there is no way to validate the accuracy of the spatial structure of abnormality scores from our method because, in reality, the ground truth is unknown. So, we compare them with tumor displacements u^* which is a multi-dimensional field. In order to be able to compare $\mathcal{S}_{rl}$ with u^*, we extract regional features U_{rl}, $l = \{1, 2\}$ from u^* in the same way we did for registration displacements w in §6.1.2. We define these features below:

$$U_{r1} = \mu_r(u^*) \tag{6.19}$$

$$U_{r2} = \mu_r(u^* \nabla \cdot \boldsymbol{u^*}) \tag{6.20}$$

We call U_{rl} the ground truth feature. We consider only volume weighted (VW) U_{rl} and $\mathcal{S}_{rl}$. Now we can compare U_{rl} with $\mathcal{S}_{rl}$ because both of them are regional features and we can compare regions with regions for a fixed l. Let us also denote the feature vectors $\boldsymbol{\mathcal{S}}_l \in \mathbb{R}^{n_r}$ and $\mathbf{U}_l \in \mathbb{R}^{n_r}$ such that $\mathcal{S}_{rl}$ and U_{rl} are scalar entries of $\boldsymbol{\mathcal{S}}_l$ and $\mathbf{U}_l$. We will use these feature vector notations for our analysis. To study this comparison, we ask the following two questions:

(LQ1) Do the abnormality scores $\boldsymbol{\mathcal{S}}_l$ from the statistical model have a strong rank correlation with tumor model displacement regional features $\mathbf{U}_l$?

We compute the Spearman rank correlation between $\mathbf{U}_l$ and $\boldsymbol{\mathcal{S}}_l$ to answer this question. A higher correlation implies a better correspondence between the spatial structure of mass effect from our statistical model with the tumor model. We study the **TMR-S** and **TMR-C**

datasets separately. We consider different global features q_l, $l = \{1, 2\}$ independently. We perform the analysis for the MUSE parcellation only. We stratify the datasets into mild, moderate, and severe mass effect based on equal quantiles of $\|u^*\|_1$ and report correlation statistics (mean and standard deviation) for each class. We report the results in §6.5.3.

(LQ2) How do the top few features from $\mathbf{U}_l$ compare with top few features from $\boldsymbol{\mathcal{S}}_l$?

In **(LQ1)**, we checked for the feature correlations for all the brain regions from the statistical model and biophysical tumor model. These correlations only tell us how good or bad the mass effect structure correspondence between the statistical and biophysical tumor models is. It does not give us any information about which regions have more mass effect correspondence. Moreover, the analysis included all regions in the brain, including regions away from the tumor, with no mass effect. The abnormality scores in the no mass effect regions do not have any localized spatial structure.

We address the issues from **(LQ1)** in this experiment. Here, we are interested in comparing the set of regions corresponding to a few top abnormality scores from $\boldsymbol{\mathcal{S}}_l$ with the set of regions corresponding to a few top tumor displacement features from $\mathbf{U}_l$.

Top feature sets: We define the following sets of regions corresponding to a few top feature values:

- $\texttt{top}_k$ - set of brain regions corresponding to top k features
- $\texttt{nei}_1(\texttt{top}_k)$ - first order neighbors of $\texttt{top}_k$ regions.

Let A and B be two sets of regions corresponding to top feature subsets of $\boldsymbol{\mathcal{S}}_l$ and $\mathbf{U}_l$

Table 6.3: *We report the list of top feature sets for comparing the abnormal regions from our statistical model and the tumor model. A and B are two sets of regions corresponding to top feature subsets of $\boldsymbol{\mathcal{S}}_l$ and $\mathbf{U}_l$ respectively*

A	B
$\texttt{nei}_1(\texttt{top}_1)$	$\texttt{top}_1$
	$\texttt{top}_2$
$\texttt{top}_5$	$\texttt{top}_1$
	$\texttt{top}_2$
$\texttt{top}_5$	$\texttt{top}_1$
	$\texttt{top}_2$
	$\texttt{top}_3$
	$\texttt{top}_4$
	$\texttt{top}_5$

respectively, then we define the following metrics to measure mass effect localization accuracy

$$\tau_1 = \frac{|A \cap B|}{\min(|A|, |B|)} \tag{6.21}$$

$$\tau_2 = \begin{cases} 1, & \text{if } |A \cap B| > 0 \\ 0, & \text{otherwise} \end{cases} \tag{6.22}$$

τ_1 measures the set overlap of most abnormal regions from tumor model and statistical model while τ_2 checks if any of the top abnormal regions from tumor model is contained in the top abnormal regions from the statistical model. Specifically, we compare the top feature subsets shown in Table 6.3.

Like **(LQ1)**, we consider different global feature types q_l independently. We perform the analysis for the MUSE parcellation only. We stratify the datasets into mild, moderate and severe mass effect based on equal quantiles of $\|u^*\|_1$ and report statistics of τ_1 and τ_2 (mean and standard deviation) for each class.

In order to show the significance of our results (see Table 6.9), we do a theoretical calculation for the case where abnormality score vector $\boldsymbol{\mathcal{S}}_l \in \mathbb{R}^{n_r}$ is a random vector and does not possess any localized mass effect structure. First, we estimate the probability of

Table 6.4: ***Top five best mass effect classification models:*** *We report the top five best cases for binary and three-way classification from all combination runs. We consider different quantiles for* $\|u^*\|_1$ *for ground truth annotation. We considered different global registration features* q_l *for computing abnormality scores which are used as features during training. We test for different classifier models. We test for volume weighted (VW) and unweighted (VUW) abnormality scores. We check for rank unordered (NRO) and rank ordered top* k *(*$\texttt{top}_k$*) abnormality scores. We test different datasets used in training –* ***TMR-C*** *(*real*) for clinical datasets and* ***TMR-S*** *(*syn*) for synthetic dataset.* β *denotes ratio of weights of* real *to* syn *samples when* real+syn *is used for training.*

run	nclass	quantiles	q_l	model	$\texttt{top}_k$	alpha	VW	train dataset	CV score
#1	2	(0.0,0.41,1.0)	$(w,\ w\nabla\cdot\boldsymbol{w})$	MLP	184	14	True	real+syn	0.79 ± 0.05
#2	3	(0.0,0.33,0.66,1.0)	$w\nabla\cdot\boldsymbol{w}$	MLP	92	1	True	real+syn	0.61 ± 0.08

finding a single region r in the set $\texttt{top}_5 \subset \boldsymbol{\mathcal{S}}_l$. This probability is given by

$$p = \frac{5}{n_r} \tag{6.23}$$

where $|\cdot|$ is the set cardinality and $n_r = 92$ is the total number of brain regions, therefore $p = 0.16$. Then, we compute the probability that any of the regions from $\texttt{top}_k \subset \mathbf{U}_l$ fall in $\texttt{top}_5 \subset \boldsymbol{\mathcal{S}}_l$. This probability is given by

$$P_k = \sum_{i=1}^{k} {}^kC_i\ p^i(1-p)^{k-i} \tag{6.24}$$

which is simply the summation of Binomial probabilty mass function for $i = 1, \ldots, k$. We report these results in Table 6.7 and Table 6.9.

6.5 Results

This section presents results for classification, regression, and the localization problem.

6.5.1 Classification problem

We report the best classification model and the corresponding feature combinations for binary and three-way classification of mass effect in Table 6.4.

Observations: We ran the best model on the unseen test dataset (85 samples from **TMR-C** dataset) and observed 72% and 62% accuracy for binary and three-way classification, respectively. The best accuracy we get is 79% and 61% for binary and three-way classification, respectively. We must have a high recall for high mass effect cases, i.e., we do not want to wrongly classify high mass effect cases as low mass effect. The recall score for high mass effect on the test dataset is 79% and 76% for binary and three-way classification, respectively. These results are acceptable compared to a purely chance-based classification based on the class distribution of the training set. If we use the dividing quantiles for $\|u^*\|_1$ as 0.41, then for binary classification, the probability of correctly classifying the two classes are 41% and 59%. For three-way classification, the probability of choosing the correct class is 33% for the equal quantiles case.

Almost all the top-performing feature combinations used volume-weighted (VW) abnormality scores and global feature $q_l = w\nabla \cdot \boldsymbol{w}$ (w is a scalar field and $\boldsymbol{w}$ is a vector field), although we do not gain significant accuracy from using VW features. The average accuracy gain is 1% over the unweighted volume case. Using different quantiles does not seem to have a drastic effect on accuracy. The top-performing model is MLP for both binary and three-way classification. On average, MLP and random forest differ only by $1 \pm 3\%$ in accuracy across all combination runs. We compared the classification accuracy when trained with **TMR-C** (real) only and **TMR-C**+**TMR-S** (real+syn) datasets for different β. For the top-performing cases in Table 6.4, we observed $< 1\%$ accuracy improvement in binary classification using real+syn dataset. For three-way classification, we observed 10% improvement in accuracy for runs #6-8 in Table 6.4 over the real case. Over the entire parameter space, we observed that for binary classification, real+syn gives an average accuracy improvement

Table 6.5: *Regression performance for top five cases for predicting tumor displacement* $\|u^*\|_1$ *using abnormality scores* $\mathcal{S}_{rl}$ *as features. We check the regression performance for volume weighted (VW) and unweighted (UVW) features. We test for different global features* q_l*. We also check performance using top* k *rank ordered (*$\texttt{top}_k$*) features. We test different regression models and consider training using* ***TMR-C*** *(*real*) only and* ***TMR-C*****+*****TMR-S*** *(*real+syn*) datasets. We test different sample weight ratios in the* real+syn *case. We do a nested cross-validation method to judge model performance. Within a single validation fold, we first compute the average prediction relative error* $\mathbb{E}_i e^i_{rel}$ *and then report the CV score (mean and standard deviation) of this average error across all validation folds.*

							error statistics		
run	VW	q_l	model	$\texttt{top}_k$	β	dataset	$min(e^i_{rel})$	$max(e^i_{rel})$	CV score
#1	True	$w\nabla \cdot \boldsymbol{w}$	RandomForest	92	3	real	1.1e-03	3.8e+00	6.5e-01±6.6e-02
#2	False	$w\nabla \cdot \boldsymbol{w}$	MLP	92	1	real	5.0e-05	3.8e+00	6.7e-01±9.9e-02
#3	True	$w\nabla \cdot \boldsymbol{w}$	RandomForest	92	7	real	2.1e-03	4.1e+00	6.7e-01±9.2e-02
#4	False	$w\nabla \cdot \boldsymbol{w}$	MLP	92	3	real	3.5e-03	3.6e+00	6.7e-01±9.3e-02
#5	False	$w\nabla \cdot \boldsymbol{w}$	MLP	92	14	real	5.0e-05	3.9e+00	6.7e-01±9.0e-02

of $2 \pm 3\%$ over real only. For three-way classification, real+syn performs slightly better than real with an average improvement in accuracy of $5.4 \pm 7.2\%$. We did not observe a significant change in average accuracy when β is varied. Given the significant standard deviation in accuracy gain, there is no conclusive evidence that real+syn performs consistently better than real only.

6.5.2 Regression problem

We consider the same test cases as we did for the classification problem for the regression problem. We show the predictions of the best regression model in Figure 6.14.

Observations:

The top performing regression model is random forest which used $\texttt{top}_{92}$ volume weighted (VW) abnormality scores $\mathcal{S}_{rl}$ as features and $q_l = w\nabla \cdot \boldsymbol{w}$. For training, $\beta = 3$ and **TMR-C** dataset worked best. The cross-validation error is 6.5e-01 $\pm$ 6.6e-02 which is the mean and standard deviation of average relative error $\mathbb{E}_i e^i_{rel}$ across validation folds. The results suggest that we are under-predicting the target (see Figure 6.14) The average relative error in the

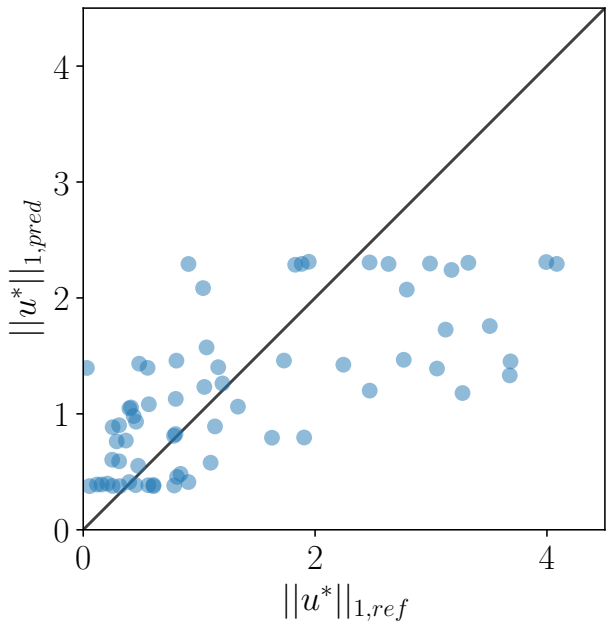
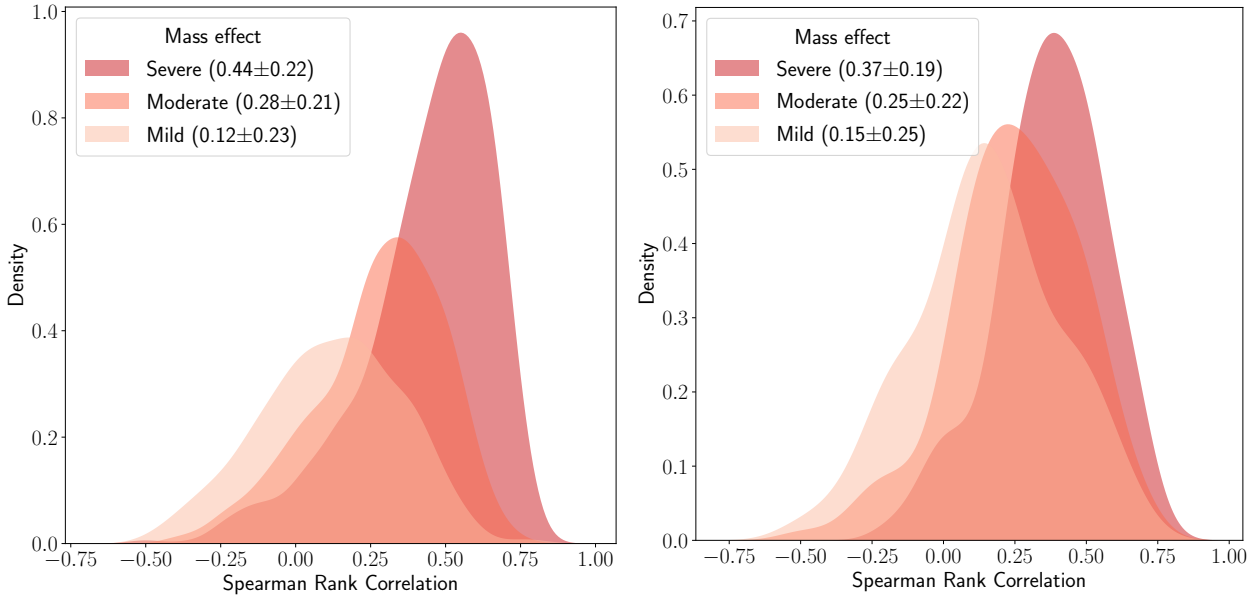


Figure 6.14: *We show the predicted tumor displacements* $\|u^*\|_{1,pred}$ *against the true values* $\|u^*\|_{1,ref}$ *for the best regression model – random forest which uses* $\texttt{top}_{92}$ *volume weighted abnormality scores* $\mathcal{S}_{rl}$ *for training. The model under-predicts for almost all samples.*

predictions for the unseen testing dataset exceeds 100%.

We observe that using volume weighted (VW) $\mathcal{S}_{rl}$ as features decreases prediction performance by up to $3\pm7\%$ on average across the entire feature space. For $\beta = 14$, real+syn performs up to 1.5% better than real only and for smaller $\beta = 1$, real data performs better by up to 2%. Overall, using real+syn does not seem to improve prediction performance, and the prediction quality suffers severely because of the small number of real samples present and the amount of noise in the data, as seen in Figure 6.8 where we can see that there is much variance in the abnormality scores for a single value of $\|u^*\|_1$.



(a) ***TMR-S*** *dataset*

(b) ***TMR-C*** *dataset*

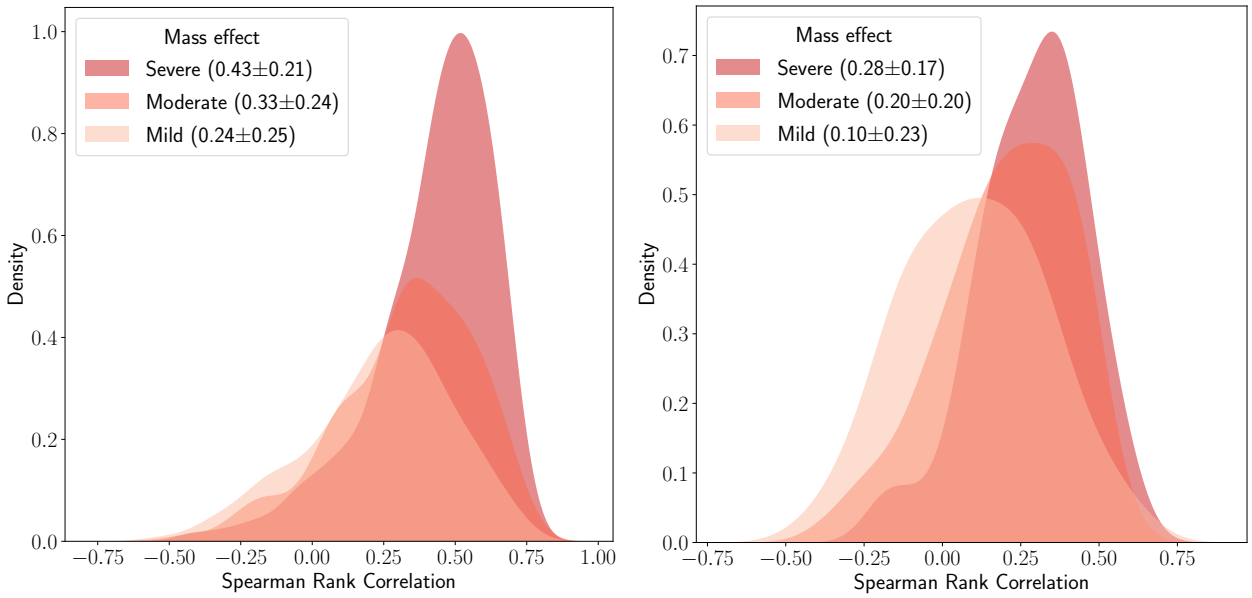
Figure 6.15: ***Experiment (LQ1):*** *We show the Spearman rank correlations for the* ***TMR-S*** *and* ***TMR-C*** *datasets between abnormality scores* $\boldsymbol{\mathcal{S}}_l$ *and tumor displacement features* $\mathbf{U}_l$ *for the global feature* q_l*. For* $\boldsymbol{\mathcal{S}}_l$*,* $q_l = w$ *and for* $\mathbf{U}_l$*,* $q_l = u^*$*. We also report the statistics (mean and standard deviation) of the correlations stratified by mass effect severity.*

6.5.3 Localization problem

(LQ1) Do the abnormality scores $\boldsymbol{\mathcal{S}}_l$ from the statistical model have a strong rank correlation with tumor model displacement regional features $\mathbf{U}_l$?

In Figure 6.16, we show the distribution and statistics of the Spearman rank correlation between $\boldsymbol{\mathcal{S}}_l$ and $\mathbf{U}_l$ for stratified by mass effect class for the **TMR-S** and **TMR-C** dataset.

Observations: In Figure 6.16, for both **TMR-S** and **TMR-C** dataset, we see a general trend that the rank correlation improves as the mass effect gets larger. This is because as the mass effect increases, it becomes easier for the statistical model to differentiate the abnormal regions from normal regions; as a result, we can localize the mass effect better. For mild mass effect cases, the abnormality scores will lack a well-defined mass effect structure. This is because the tumor does not significantly deform the brain regions. As a result, the



(a) ***TMR-S*** *dataset* **(b)** ***TMR-C*** *dataset*

Figure 6.16: ***Experiment (LQ1):*** *We show the Spearman rank correlations for the* ***TMR-S*** *and* ***TMR-C*** *datasets between abnormality scores* $\boldsymbol{\mathcal{S}}_l$ *and tumor displacement features* $\mathbf{U}_l$ *for the global feature* q_l*. For* $\boldsymbol{\mathcal{S}}_l$*,* $q_l = w\nabla \cdot \boldsymbol{w}$ *and for* $\mathbf{U}_l$*,* $q_l = u^*\nabla \cdot \boldsymbol{u^*}$*. We also report the statistics (mean and standard deviation) of the correlations stratified by mass effect severity.*

abnormality scores will be dominated by noise resulting from natural anatomical variations in the brain. Furthermore, to compute the rank correlation, we consider all the brain regions, even those which do not have mass effect. As explained earlier, the abnormality scores in normal brain regions will be noisy, decreasing the overall correlation. We address this in **(LQ2)**. The overall conclusion from this experiment is that we are able to localize severe mass effect cases (correlation $= 0.44 \pm 0.22$) better than mild mass effect (correlation $= 0.12 \pm 0.23$). We do not observe any significant differences in correlations for **TMR-S** and **TMR-C** datasets.

(LQ2) How do the top few features from $\mathbf{U}_l$ compare with top few features from $\boldsymbol{\mathcal{S}}_l$?

In Table 6.7, we report the statistics of τ_1 and τ_2 (see (6.21) in §6.4) for each mass effect class (obtained from equal quantiles of $\|u^*\|_1$) of the **TMR-S** dataset. In Table 6.9, we report the same results for the **TMR-C** dataset.

Table 6.6: ***Experiment (LQ2):*** *We compare the few top abnormality scores $\boldsymbol{\mathcal{S}}_l$ (global feature $q_l = w$) and few top ground truth features $\mathbf{U}_l$ (global feature $q_l = u^*$) for the **TMR-S** dataset. We report the average value (stratified by mass effect class) of τ_1 and τ_2 which are region comparison metrics(see (6.21)). We compare our results with the case where $\boldsymbol{\mathcal{S}}_l$ is a uniformly distributed random vector. We compare our results with the case where $\boldsymbol{\mathcal{S}}_l$ is a uniformly distributed random vector. For this case, we report the probability P_k using (6.24) for $k = 1, \ldots, 5$.*

			Mass Effect			
Metric	Abnormality score ($\boldsymbol{\mathcal{S}}_l$)	Ground Truth ($\mathbf{U}_l$)	random $\boldsymbol{\mathcal{S}}_l$ (P_k)	mild	moderate	severe
τ_1	$\texttt{nei}_1(\texttt{top}_1)$	$\texttt{top}_1$	–	0.22	0.47	0.52
		$\texttt{top}_2$	–	0.24	0.50	0.51
	$\texttt{top}_5$	$\texttt{top}_1$	–	0.20	0.48	0.56
		$\texttt{top}_2$	–	0.17	0.43	0.46
τ_2	$\texttt{top}_5$	$\texttt{top}_1$	0.05	0.20	0.48	0.56
		$\texttt{top}_2$	0.11	0.29	0.66	0.71
		$\texttt{top}_3$	0.15	0.36	0.74	0.78
		$\texttt{top}_4$	0.20	0.41	0.79	0.83
		$\texttt{top}_5$	0.24	0.43	0.82	0.87

Observations: In Table 6.7 and Table 6.9, similar to **(LQ1)**, we observe that the region

Table 6.7: ***Experiment (LQ2):*** *We compare the few top abnormality scores $\boldsymbol{\mathcal{S}}_l$ (global feature $q_l = w\nabla \cdot \boldsymbol{w}$) and few top ground truth features $\mathbf{U}_l$ (global feature $q_l = u^*\nabla \cdot \boldsymbol{u^*}$) for the **TMR-S** dataset. We report the average value (stratified by mass effect class) of τ_1 and τ_2 which are region comparison metrics(see (6.21)). We compare our results with the case where $\boldsymbol{\mathcal{S}}_l$ is a uniformly distributed random vector. We compare our results with the case where $\boldsymbol{\mathcal{S}}_l$ is a uniformly distributed random vector. For this case, we report the probability P_k using (6.24) for $k = 1, \ldots, 5$.*

			Mass Effect			
Metric	Abnormality score ($\boldsymbol{\mathcal{S}}_l$)	Ground Truth ($\mathbf{U}_l$)	random $\boldsymbol{\mathcal{S}}_l$ (P_k)	mild	moderate	severe
τ_1	$\texttt{nei}_1(\texttt{top}_1)$	$\texttt{top}_1$	–	0.26	0.48	0.47
		$\texttt{top}_2$	–	0.26	0.51	0.50
	$\texttt{top}_5$	$\texttt{top}_1$	–	0.26	0.61	0.64
		$\texttt{top}_2$	–	0.23	0.51	0.60
τ_2	$\texttt{top}_5$	$\texttt{top}_1$	0.05	0.26	0.61	0.64
		$\texttt{top}_2$	0.11	0.39	0.80	0.88
		$\texttt{top}_3$	0.15	0.49	0.87	0.93
		$\texttt{top}_4$	0.20	0.57	0.91	0.97
		$\texttt{top}_5$	0.24	0.62	0.94	0.98

comparison metrics τ_1 and τ_2 increase with increasing mass effect both for **TMR-S** and **TMR-C** dataset. τ_1 is a stricter metric than τ_2 in the sense that it will be one only when all $\texttt{top}_k$ regions from $\mathbf{U}_l$ fall in $\texttt{nei}_1(\texttt{top}_1)$ from $\boldsymbol{\mathcal{S}}_l$. For τ_2, we only require any one of $\texttt{top}_k$ regions from $\mathbf{U}_l$ to be found in $\texttt{top}_5$ from $\boldsymbol{\mathcal{S}}_l$. For this reason τ_1 has smaller population average than τ_2. The localization results for tumor cases with mass effect are significant compared to random abnormality scores without any spatial localization of mass effect. Overall, we observe that for 88% of the severe mass effect cases, we have a good correspondence between $\texttt{top}_5$ regions from $\boldsymbol{\mathcal{S}}_l$ and $\texttt{top}_5$ regions from $\mathbf{U}_l$ which is the tumor model features.

6.5.4 Clinical summary

We report a clinical summary of our results for three exemplar GBM patient cases in Figure 6.17 and Figure 6.18. In Figure 6.17, we show the patient T1-weighted MR image with overlaid tumor segmentation, the regional abnormality scores, and the boundary of the set of $\texttt{top}_5$ regions with the highest mass effect as predicted by the statistical model. We also report the probabilities of each patient having mild, moderate, or severe mass effect. In Fig-

Table 6.8: ***Experiment (LQ2):*** *We compare the few top abnormality scores $\boldsymbol{\mathcal{S}}_l$ (global feature $q_l = w$) and few top ground truth features $\mathbf{U}_l$ (global feature $q_l = u^*$) for the **TMR-C** dataset. We report the average value (stratified by mass effect class) of τ_1 and τ_2 which are region comparison metrics(see (6.21)). We compare our results with the case where $\boldsymbol{\mathcal{S}}_l$ is a uniformly distributed random vector. For this case, we report the probability P_k using (6.24) for $k = 1, \ldots, 5$.*

			Mass Effect			
Metric	Abnormality score ($\boldsymbol{\mathcal{S}}_l$)	Ground Truth ($\mathbf{U}_l$)	random $\boldsymbol{\mathcal{S}}_l$ (P_k)	mild	moderate	severe
τ_1	$\texttt{nei}_1(\texttt{top}_1)$	$\texttt{top}_1$	–	0.42	0.62	0.60
		$\texttt{top}_2$	–	0.42	0.59	0.55
	$\texttt{top}_5$	$\texttt{top}_1$	–	0.33	0.53	0.44
		$\texttt{top}_2$	–	0.29	0.44	0.34
τ_2	$\texttt{top}_5$	$\texttt{top}_1$	0.05	0.33	0.53	0.44
		$\texttt{top}_2$	0.11	0.51	0.65	0.60
		$\texttt{top}_3$	0.15	0.59	0.78	0.71
		$\texttt{top}_4$	0.20	0.64	0.82	0.79
		$\texttt{top}_5$	0.24	0.66	0.86	0.88

ure 6.18, we show the distribution of the aggregate abnormality scores and plot the patient predictions on this distribution. The overall time required to generate this kind of a summary for a new patient takes only 2 min (compared to 4 hrs using a biophysical tumor model in `GLIA`) because the most expensive part of our model is image registration which is executed using `CLAIRE` which is extremely fast. We hope that the results from this clinical summary can be used for downstream tasks such as survival prediction.

6.6 Limitations

There are several limitations to our method:

- We consider limited number of template images to construct $\pi_w(h)$. Using a higher number of templates would capture more variation in healthy brain anatomy. We use all templates to average features and do not consider patient-specific template selection.
- We do not have ground truth information about mass effect displacements for clinical datasets; hence we depend on tumor model estimations for verifying our model.
- There is a lack of sizeable clinical tumor datasets to train a robust classifier or regressor

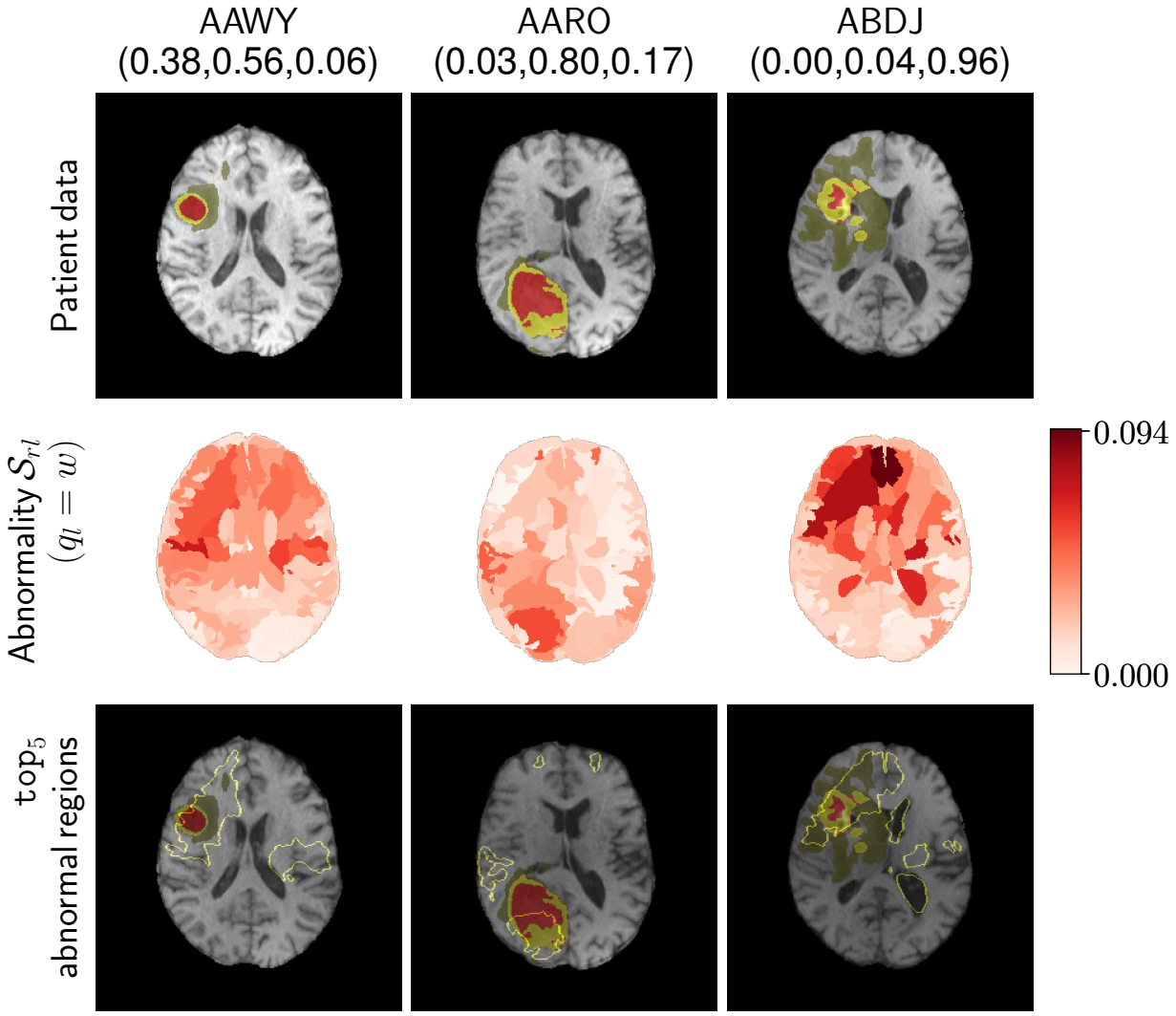


Figure 6.17: *We show part 1/2 of clinical summary of our results. We show results for three clinical patient cases (AAWY, AARO and ABDJ) from* ***TMR-C*** *dataset. In the first row, we show the patient T1-weighted MR image with the tumor segmentation. In the second row, we show the regional abnormality scores $\mathcal{S}_{rl}$ and in the last row, we show the boundaries (in yellow) of the set of* $\texttt{top}_5$ *most abnormal regions as predicted by the statistical model. We also report the mass effect class probabilities (p_{mild}, $p_{moderate}$, p_{severe}) above the first row for each patient.*

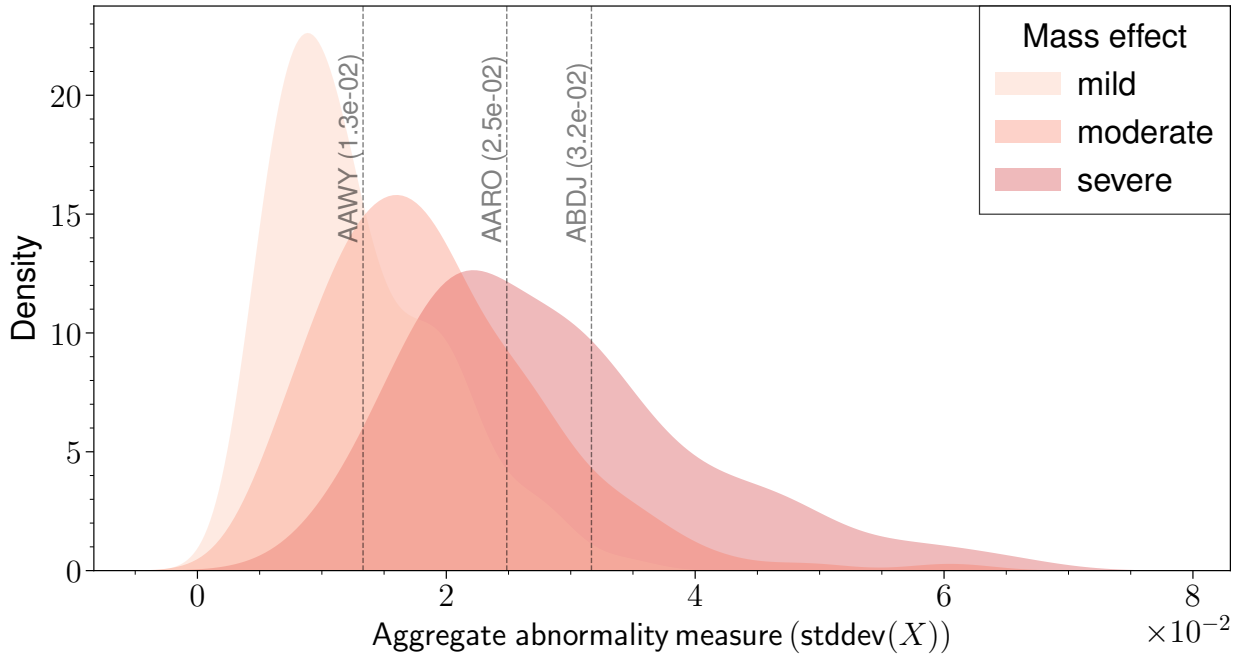


Figure 6.18: *We show part 2/2 of clinical summary of our results. We show the population distribution of the aggregate abnormality measure* stddev(X) *where* X *is the top 20 few volume weighted abnormality scores from §6.3. We plot the aggregate scores for patients (AAWY, AARO and ABDJ) from* ***TMR-C*** *dataset against this distribution. The stratification for the mass effect is based on equal quantiles of tumor displacement* $\|u^*\|_1$.

Table 6.9: ***Experiment (LQ2):*** *We compare the few top abnormality scores $\boldsymbol{\mathcal{S}}_l$ (global feature $q_l = w\nabla \cdot \boldsymbol{w}$) and few top ground truth features $\mathbf{U}_l$ (global feature $q_l = u^*\nabla \cdot \boldsymbol{u^*}$) for the **TMR-C** dataset. We report the average value (stratified by mass effect class) of τ_1 and τ_2 which are region comparison metrics(see (6.21)). We compare our results with the case where $\boldsymbol{\mathcal{S}}_l$ is a uniformly distributed random vector. For this case, we report the probability P_k using (6.24) for $k = 1, \ldots, 5$.*

			Mass Effect			
Metric	Abnormality score ($\boldsymbol{\mathcal{S}}_l$)	Ground Truth ($\mathbf{U}_l$)	random $\boldsymbol{\mathcal{S}}_l$ (P_k)	mild	moderate	severe
τ_1	$\texttt{nei}_1(\texttt{top}_1)$	$\texttt{top}_1$	–	0.52	0.65	0.68
		$\texttt{top}_2$	–	0.48	0.63	0.66
	$\texttt{top}_5$	$\texttt{top}_1$	–	0.52	0.60	0.64
		$\texttt{top}_2$	–	0.42	0.53	0.57
τ_2	$\texttt{top}_5$	$\texttt{top}_1$	0.05	0.52	0.60	0.64
		$\texttt{top}_2$	0.11	0.73	0.79	0.83
		$\texttt{top}_3$	0.15	0.79	0.90	0.94
		$\texttt{top}_4$	0.20	0.84	0.94	0.94
		$\texttt{top}_5$	0.24	0.86	0.97	0.99

for detection or prediction purposes. We only have access to a limited number of mass effect annotations from radiologists.

6.7 Chapter conclusions

In this chapter, we explored one of the many clinical applications of image registration – characterization of mass effect. In particular, we developed a statistical deformation abnormality detection model to detect and localize mass effect to specific brain regions using a single time point multi-parametric MRI scan of a GBM patient. We create a summary report for a patient, including information about tumor segmentation, a heatmap of abnormal regions, the top 5 most abnormal regions, and the probability of the patient having a mild, moderate or severe mass effect in under 3 minutes. We can classify a patient into mild, moderate, or severe mass effect with 62% accuracy and report the top 5 most abnormal regions with the highest mass effect (88% accuracy). We conducted experiments to verify the results of our method on synthetic and clinical GBM datasets. We created a sizeable synthetic tumor dataset consisting of 2400 different tumors of varying shape and size for data augmentation during training to detect mass effect. We verified the mass effect localization from the sta-

tistical method by comparing it with the predictions of an expensive tumor inversion model and achieved good accuracy. We compared our scheme's mass effect abnormality scores with expert annotations and did not observe a significant agreement. Overall, we see promising developments for a fast method that can detect deformation abnormalities not only for GBM images but also for any disease which alters brain geometry significantly, like Alzheimer's. The method is fast and only requires image registration as the primary computational tool.

Chapter 7

Conclusions and future work

To conclude this dissertation, we summarize our contributions and discuss limitations, open issues, and future work.

7.1 Conclusions

This thesis presented algorithms and software for scalable diffeomorphic image registration and its application in deformation abnormality characterization in medical images. We discussed how our framework could be used for full resolution image registration of large-scale medical images. We also discussed the development of a statistical model to estimate mass effect in GBM patients. Our contributions are as follows:

- ***Scalable diffeomorphic image registration*** We first discussed the diffeomorphic image registration problem using a Newton-Krylov method. We discussed the mathematical formulation for the registration problem and presented its optimality conditions. Solving the registration problem (especially in 3D) is very time-consuming. The main computational kernels are the FFT, used for higher-order spatial differential operators in our spectral approach, 8^{th} order finite differences for first-order derivatives, and the interpolations used in the semi-Lagrangian scheme for advection. We introduced a mixed-precision approach to computing derivatives by implementing an optimized

finite-difference kernel to compute first-order derivatives and FFTs for higher order derivatives. To improve semi-Lagrangian's performance, we discussed several options for the interpolation kernel, including different interpolating polynomials and GPU texture memory optimizations. To accelerate the registration solver, we ported the important kernels to GPU. We compared our fast single GPU registration solver with other state-of-the-art solvers and our previous distributed memory CPU solver and reported a performance improvement of 23× and 20× respectively.

- Next, we extended the single GPU registration solver to a multi-node multi-GPU setup. We introduced several improvements both algorithmically and computationally. We introduced a novel two-level preconditioner based on the zero-velocity approximation of the Hessian system arising from the second-order optimality conditions in the Gauss-Newton-Krylov method. The preconditioner reduced the runtime of the solver reduced the solver runtime by as much as 2.5× when compared to the CPU version of `CLAIRE`. We introduced several optimizations to the main kernels. We used CUDA-Aware MPI, thereby bypassing host-side communication, and used a high-speed NVlink interconnect bus. We replaced the cubic-spline interpolation, the standard on single-GPU, with a cubic-Lagrange interpolation to reduce communication between GPUs. We proposed an efficient implementation of 3D FFTs using a combination of 2D and 1D cuFFT operations in slab decomposition. Following these optimizations, we report scaling results on the Longhorn supercomputer using up to 256 GPUs. We solved a problem size 152× larger than state-of-the-art and reported a 70% improvement over a single-GPU implementation. We also presented results for clinical human brain images (1024^3 using 32 GPUs) and CLARITY murine brain images ($1024 \times 768 \times 768$ using 16 GPUs).

- Them, we showcased the scalability of `CLAIRE` on more high-resolution datasets. We presented an augmented version of the regularization parameter search scheme for

`CLAIRE`. We could compute deformations, which are guaranteed to be locally diffeomorphic and driven by user specifications. Instead of doing a brute-force search in the parameter space, our improved scheme employed an approach in which we first fix one regularization parameter and search for the other and subsequently alternate, if necessary. We eliminated the manual adjustment of another hyperparameter in the setup of CLAIRE. Using the proposed scheme, we demonstrated that CLAIRE provides a registration quality that is on par with results generated by ANTs, a state-of-the-art CPU image registration package. We registered CLARITY mouse brain images of unprecedented ultra-high spatial resolution ($2816 \times 3016 \times 1162$) in 23 minutes using parameter continuation. We conduct detailed experiments to compare image registration performance at full and downsampled resolutions using synthetic and real images. We find that image registration at higher (native) image resolution is more accurate. We also do a sensitivity analysis for the overall solver accuracy for the number of time steps in the semi-Lagrangian scheme.

- finally, we developed a statistical deformation abnormality detection and localization model using 3D image registration. In particular, we developed a statistical deformation abnormality detection model to detect and localize mass effect to specific brain regions using a single time point multi-parametric MRI scan of a GBM patient. We can classify a patient into mild, moderate, or severe mass effect with 62% accuracy. We create a clinical summary report for a patient, including information about tumor segmentation, a heatmap of abnormal regions, the top 5 most abnormal regions (with 88% accuracy), and probabilities of the patient having a mild, moderate or severe mass effect in under 3 minutes.

7.2 Future work

The multi-node multi-GPU image registration solver uses a slab decomposition to partition data for distributed memory parallelism. It is well known that slab decomposition does not scale as well as pencil decomposition as has previously been done in the CPU version of `CLAIRE`. Exploring the pencil decomposition for the GPU solver while keeping the communication in check is one area of work that needs to be explored. `CLAIRE` used PETSc for optimization tools like line search and linear algebra solvers like PCG. PETSc adds unnecessary memory footprint in the solver and makes it challenging to support larger problem sizes on a single GPU. Removing this dependency can reduce memory pressure, enable larger problem sizes per GPU and improve the scalability of the distributed memory solver.

`CLAIRE` uses L_2 distance as the image dissimilarity metric in the objective function. L_2 distance measure is very robust for registering images of similar intensities. However, it fails when different image modalities must be registered, such as the registration between a T1-weighted and T2-weighted MRI scan. Normalized cross-correlation and mutual information are two similarity metrics that are typically used in other registration codes. Implementing normalized cross-correlation in `CLAIRE` is an ongoing work.

We did an exploratory analysis of image registration for mass effect characterization in GBM patients. We used an H1-seminorm as the regularization operator in the registration scheme. We could use H2-seminorm, which gives smoother velocity fields than H1-seminorm. Using smoother velocity fields, we can get much smoother displacements which can be used to build a better statistical model for normal deformations. Furthermore, masking the registration to a specific ROI around the tumor and only predicting the deformations around the tumor can be helpful. `CLAIRE` uses a linear advection equation for the forward model. Using a linear elasticity equation as the forward problem, which models more significant modes of mass effect type displacements, could be beneficial and needs to be explored.

Bibliography

[1] Cuda toolkit dcoumentation.

[2] IBM Spectrum MPI (version 10.3.0).

[3] A reproducible evaluation of ANTs similarity metric performance in brain image registration - ScienceDirect. https://www.sciencedirect.com/science/article/pii/S1053811910012061?via%3Dihub.

[4] Daniel Abler, Philippe Büchler, and Russell C. Rockne. Towards Model-Based Characterization of Biomechanical Tumor Growth Phenotypes. In George Bebis, Takis Benos, Ken Chen, Katharina Jahn, and Ernesto Lima, editors, *Mathematical and Computational Oncology*, Lecture Notes in Computer Science, pages 75–86, Cham, 2019. Springer International Publishing.

[5] A. Alexanderian, N. Petra, G. Stadler, and O. Ghattas. A fast and scalable method for A-optimal design of experiments for infinite-dimensional Bayesian nonlinear inverse problems. *SIAM Journal on Scientific Computing*, 38(1):A243–A272, 2016.

[6] V. Arsigny, O. Commowick, X. Pennec, and N. Ayache. A Log-Euclidean framework for statistics on diffeomorphisms. In *Proc Medical Image Computing and Computer-Assisted Intervention*, volume LNCS 4190, pages 924–931, 2006.

[7] J. Ashburner. A fast diffeomorphic image registration algorithm. *NeuroImage*, 38(1):95–113, 2007.

[8] J. Ashburner and K. J. Friston. Voxel Based Morphometry. In Larry R. Squire, editor, *Encyclopedia of Neuroscience*, pages 471–477. Academic Press, Oxford, January 2009.

[9] J. Ashburner and K. J. Friston. Diffeomorphic registration using geodesic shooting and Gauss-Newton optimisation. *NeuroImage*, 55(3):954–967, 2011.

[10] John Ashburner and Karl J. Friston. Voxel-Based Morphometry—The Methods. *NeuroImage*, 11(6):805–821, June 2000.

[11] John Ashburner, Chloe Hutton, Richard Frackowiak, Ingrid Johnsrude, Cathy Price, and Karl Friston. Identifying global anatomical differences: Deformation-based morphometry. page 10.

[12] B. B. Avants, C. L. Epstein, M. Brossman, and J. C. Gee. Symmetric diffeomorphic image registration with cross-correlation: Evaluating automated labeling of elderly and neurodegenerative brain. *Medical Image Analysis*, 12(1):26–41, 2008.

[13] B. B. Avants, N. J. Tustison, G. Song, P. A. Cook, A. Klein, and J. C. Gee. A reproducible evaluation of ANTs similarity metric performance in brain image registration. *NeuroImage*, 54:2033–2044, 2011.

[14] Brian B. Avants, Nicholas J. Tustison, and Hans J. Johnson. ANTs.

[15] R. Azencott, R. Glowinski, J. He, A. Jajoo, Y. Li, A. Martynenko, R. H. W. Hoppe, S. Benzekry, and S. H. Little. Diffeomorphic matching and dynamic deformable surfaces in 3D medical imaging. *Computational Methods in Applied Mathematics*, 10(3):235–274, 2010.

[16] Spyridon Bakas, Mauricio Reyes, and et al. Identifying the best machine learning algorithms for brain tumor segmentation, progression assessment, and overall survival prediction in the BRATS challenge. *CoRR*, abs/1811.02629, 2018.

[17] G. Balakrishnan, A. Zhao, M. R. Sabuncu, J. Guttag, and A. V. Dalca. VoxelMorph: A learning framework for deformable medical image registration. *IEEE Transactions on Medical Imaging*, 2019. (in press) DOI: 10.1109/TMI.2019.2897538.

[18] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. Gropp, D. Karpeyev, D. Kaushik, M. Knepley, D. May, L. Curfman McInnes, R. Mills, T. Munson, K. Rupp, P. Sanan, B. Smith, S. Zampini, H. Zhang, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.13, Argonne National Laboratory, 2020.

[19] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. D. Gropp, D. Karpeyev, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang. PETSc and TAO webpage (PETSc version 3.12.4).

[20] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.7, Argonne National Laboratory, 2016.

[21] V. Barbu and G. Marinoschi. An optimal control approach to the optical flow problem. *Systems & Control Letters*, 87:1–9, 2016.

[22] Mustafa Mahmut Baris, Ahmet Orhan Celik, Naciye Sinem Gezer, and Emel Ada. Role of mass effect, tumor volume and peritumoral edema volume in the differential diagnosis of primary brain tumor and metastasis. *Clinical Neurology and Neurosurgery*, 148:67,Äì71, 2016.

[23] S. Bauer, S. Joshi, and K. Modin. *Diffeomorphic Density Registration*, chapter 16, pages 577–603. Elsevier, 2020.

[24] M. F. Beg, M. I. Miller, A. Trouvé, and L. Younes. Computing large deformation metric mappings via geodesic flows of diffeomorphisms. *International Journal of Computer Vision*, 61(2):139–157, 2005.

[25] A. Bone, O. Colliot, and S. Durrleman. Learning distributions of shape trajectories from longitudinal datasets: A hierarchical model on a manifold of diffeomorphisms. *arXiv e-prints*, (1803.10119), 2019.

[26] A. Bone, M. Louis, B. Martin, and S. Durrleman. Deformetrica 4: An open-source software for statistical shape analysis. In *Proc International Workshop on Shape in Medical Imaging*, volume LNCS 11167, pages 3–13, 2018.

[27] A. Borzì, K. Ito, and K. Kunisch. Optimal control formulation for determining optical flow. *SIAM Journal on Scientific Computing*, 24(3):818–847, 2002.

[28] M. Brunn, N. Himthani, G. Biros, M. Mehl, and A. Mang. Fast GPU 3D diffeomorphic image registration. *arXiv e-prints*, 2020.

[29] Malte Brunn, Naveen Himthani, George Biros, Miriam Mehl, and Andreas Mang. Multi-Node Multi-GPU Diffeomorphic Image Registration for Large-Scale Imaging Problems. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–17, November 2020.

[30] Malte Brunn, Naveen Himthani, George Biros, Miriam Mehl, and Andreas Mang. Claire: Constrained large deformation diffeomorphic image registration on parallel computing architectures. *Journal of Open Source Software*, 6(61):3038, 2021.

[31] D. Budelmann, L. Koenig, Nils Papenberg, and J. Lellmann. Fully-deformable 3D image registration in two seconds. In *Bildverarbeitung für die Medizin*, pages 302–307, 2019.

[32] T. Bui-Thanh, O. Ghattas, J. Martin, and G. Stadler. A computational framework for infinite-dimensional Bayesian inverse problems Part I: The linearized case, with application to global seismic inversion. *SIAM Journal on Scientific Computing*, 35(6):A2494–A2523, 2013.

[33] M. Burger, J. Modersitzki, and L. Ruthotto. A hyperelastic regularization energy for image registration. *SIAM Journal on Scientific Computing*, 35(1):B132–B148, 2013.

[34] Center for Imaging Science, Johns Hopkins University. LDDMM Suite.

[35] F. Champagnat and Y. Le Sant. Efficient cubic B-spline image interpolation on a GPU. *Journal of Graphics Tools*, 16(4):218–232, 2012.

[36] V. Chandrashekhar, A. Crow, J. Bogelstein, and K. Deisseroth. NEURODATA CLARITOMES.

[37] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321,Äì357, Jun 2002.

[38] K. Chen and D. A. Lorenz. Image sequence interpolation using optimal control. *Journal of Mathematical Imaging and Vision*, 41:222–238, 2011.

[39] G. E. Christensen, X. Geng, J. G. Kuhl, J. Bruss, T. J. Grabowski, I. A. Pirwani, M. W. Vannier, J. S. Allen, and H. Damasio. Introduction to the non-rigid image registration evaluation project. In *Proc Biomedical Image Registration*, volume LNCS 4057, pages 128–135, 2006.

[40] G. E. Christensen, R. D. Rabbitt, and M. I. Miller. Deformable templates using large deformation kinematics. *Image Processing, IEEE Transactions on*, 5(10):1435–1447, 1996.

[41] K. Chung, J. Wallace, S.-Y. Kim, S. Kalyanasundaram, A. S. Andalman, T. J. Davidson, J. J. Mirzabekov, K. A. Zalocusky, J. Mattis, A. K. Denisin, S. Pak, H. Bernstein, L. G. Charu Ramakrishnan, V. Gradinaru, and K. Deisseroth. Structural and molecular interrogation of intact biological systems. *Nature*, 497:332–337, 2013.

[42] N. Courty and P. Hellier. Accelerating 3D non-rigid registration using graphics hardware. *International Journal of Image and Graphics*, 8(1):81–98, 2008.

[43] W. R. Crum, T. Hartkens, and D. L. G. Hill. Non-rigid image registration: Theory and practice. *The British Journal of Radiology*, 77:S140–S153, 2004.

[44] C. Davatzikos, M. Vaillant, S. M. Resnick, J. L. Prince, S. Letovsky, and R. N. Bryan. A computerized approach for morphological analysis of the corpus callosum. *Journal of Computer Assisted Tomography*, 20(1):88–97, 1996 Jan-Feb.

[45] Christos Davatzikos, Ahmet Genc, Dongrong Xu, and Susan M. Resnick. Voxel-Based Morphometry Using the RAVENS Maps: Methods and Validation Using Simulated Longitudinal Atrophy. *NeuroImage*, 14(6):1361–1369, December 2001.

[46] Luciano de Gois Vasconcelos, Andrea Parolin Jackowski, Maira Okada de Oliveira, Yoná Mayara Ribeiro Flor, Orlando Francisco Amodeo Bueno, and Sonia Maria Dozzi Brucki. Voxel-based morphometry findings in Alzheimer's disease: Neuropsychiatric symptoms and disability correlations – preliminary results. *Clinics*, 66(6):1045–1050, June 2011.

[47] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM Journal on Numerical Analysis*, 19(2):400–408, 1982.

[48] Therese A Dolecek, Jennifer M Propp, Nancy E Stroup, and Carol Kruchko. Cbtrus statistical report: primary brain and central nervous system tumors diagnosed in the united states in 2005–2009. *Neuro-oncology*, 14(suppl_5):v1–v49, 2012.

[49] Jimit Doshi, Guray Erus, Yangming Ou, Susan M. Resnick, Ruben C. Gur, Raquel E. Gur, Theodore D. Satterthwaite, Susan Furth, and Christos Davatzikos. MUSE: MUlti-atlas region Segmentation utilizing Ensembles of registration algorithms and parameters, and locally optimal atlas selection. *NeuroImage*, 127:186–195, February 2016.

[50] P. Dupuis, U. Gernander, and M. I. Miller. Variational problems on flows of diffeomorphisms for image matching. *Quarterly of Applied Mathematics*, 56(3):587–600, 1998.

[51] A. S. Durrleman, A. Bone, M. Louis, B. Martin, P. Gori, A. Routier, M. Bacci, A. Fougier, B. Charlier, J. Glaunes, J. Fishbaugh, M. Prastawa, M. Diaz, and C. Doucet. deformetrica [commit: v4.0.0-390-ged9c1f9; libraries: python3.6; cuda9.2.88], 2019.

[52] S. Durrleman, M. Prastawa, N. Charon, J. R. Korenberg, S. Joshi, G. Gerig, and A. Trouve. Morphometry of anatomical shape complexes with dense deformations and sparse parameters. *NeuroImage*, 101:35–49, 2014.

[53] Stanley Durrleman, Alexandre Brone, Maxime Louis, Benoit Martin, Pietro Gori, A.exandre Routier, Michael Bacci, Ana Fouquier, Benjamin Charlier, Joan Glaunes, James Fishbaugh, Marcel Prastawa, Mauricio Diaz, and Cedric Doucet. deformetrica.

[54] S. C. Eisentat and H. F. Walker. Choosing the forcing terms in an inexact Newton method. *SIAM Journal on Scientific Computing*, 17(1):16–32, 1996.

[55] A. Eklund, P. Dufort, D. Forsberg, and S. M. LaConte. Medical image processing on the GPU–past, present and future. *Medical Image Analysis*, 17(8):1073–1094, 2013.

[56] N. D. Ellingwood, Y. Yin, M. Smith, and C.-L. Lin. Efficient methods for implementation of multi-level nonrigid mass-preserving image registration on GPUs and multi-threaded CPUs. *Computer Methods and Programs in Biomedicine*, 127:290–300, 2016.

[57] B. Fischer and J. Modersitzki. Ill-posed medicine – an introduction to image registration. *Inverse Problems*, 24(3):1–16, 2008.

[58] J. Fishbaugh, S. Durrleman, M. Prastawa, and G. Gerig. Geodesic shape regression with multiple geometries and sparse parameters. *Medical Image Analysis*, 39:1–17, 2017.

[59] K. Fissell and R. Reynolds. niftilib, 2019.

[60] O. Fluck, C. Vetter, W. Wein, A. Kamen, B. Preim, and R. Westermann. A survey of medical image registration on graphics hardware. *Computer Methods and Programs in Biomedicine*, 104(3):e45–e57, 2011.

[61] C. Gaser, I. Nenadic, B. R. Buchsbaum, E. A. Hazlett, and M. S. Buchsbaum. Deformation-based morphometry and its relation to conventional volumetry of brain lateral ventricles in MRI. *NeuroImage*, 13(6 Pt 1):1140–1145, June 2001.

[62] Christian Gaser, Hans-Peter Volz, Stefan Kiebel, Stefan Riehemann, and Heinrich Sauer. Detecting Structural Changes in Whole Brain Based on Nonlinear Deformations—Application to Schizophrenia Research. *NeuroImage*, 10(2):107–113, August 1999.

[63] A. Gholami and G. Biros. AccFFT, 2017.

[64] A. Gholami, A. Mang, and G. Biros. An inverse problem formulation for parameter estimation of a reaction-diffusion model of low grade gliomas. *Journal of Mathematical Biology*, 72(1):409–433, 2016.

[65] A. Gholami, A. Mang, K. Scheufele, C. Davatzikos, M. Mehl, and G. Biros. A framework for scalable biophysics-based image analysis. In *Proc ACM/IEEE Conference on Supercomputing*, pages 1–13, 2017.

[66] Amir Gholami and George Biros. AccFFT home page, 2017.

[67] Amir Gholami, Shashank Subramanian, Varun Shenoy, Naveen Himthani, Xiangyu Yue, Sicheng Zhao, Peter Jin, George Biros, and Kurt Keutzer. A novel domain adaptation framework for medical image segmentation. In *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, volume LNCS 11384, pages 289–298, 2019.

[68] D. Grzech, L. Folgoc, M. P. Heinrich, B. Khanal, J. Moll, J. A. Schnabel, B. Glocker, and B. Kainz. FastReg: Fast non-rigid registration via accelerated optimisation on the manifold of diffeomorphisms. *arXiv e-prints*, 2019.

[69] X. Gu, H. Pan, Y. Liang, R. Castillo, D. Yang, D. Choi, E. Castillo, A. Majumdar, T. Guerrero, and S. B. Jiang. Implementation and evaluation of various demons deformable image registration algorithms on a GPU. *Physics in Medicine and Biology*, 55(1):207–219, 2009.

[70] Yubraj Gupta, Kun Ho Lee, Kyu Yeong Choi, Jang Jae Lee, Byeong Chae Kim, Goo Rak Kwon, the National Research Center for Dementia, and Alzheimer's Disease Neuroimaging Initiative. Early diagnosis of Alzheimer's disease using combined features from voxel-based morphometry and cortical, subcortical, and hippocampus regions of MRI T1 brain images. *PLOS ONE*, 14(10):e0222446, October 2019.

[71] L. Ha, J. Krüger, S. Joshi, and C. T. Silva. Multiscale unbiased diffeomorphic atlas construction on multi-GPUs. In *CPU Computing Gems Emerald Edition*, chapter 48, pages 771–791. Elsevier Inc, 2011.

[72] L. K. Ha, J. Krüger, P. T. Fletcher, S. Joshi, and C. T. Silva. Fast parallel unbiased diffeomorphic atlas construction on multi-graphics processing units. In *Proc Eurographics Conference on Parallel Grphics and Visualization*, pages 41–48, 2009.

[73] Sam-Yeol Ha, Young Chul Youn, SangYun Kim, Ging-Yuek Robin Hsiung, Suk-Won Ahn, Hae-Won Shin, Kwang-Yeol Park, Tai Hwan Park, Sung-Su Kim, Baik Seok

Kee, and Oh-Sang Kwon. A voxel-based morphometric study of cortical gray matter volume changes in Alzheimer's disease with white matter hyperintensities. *Journal of Clinical Neuroscience: Official Journal of the Neurosurgical Society of Australasia*, 19(11):1506–1510, November 2012.

[74] Eldad Haber and Douglas Oldenburg. A GCV based method for nonlinear ill-posed problems. *Computational Geosciences*, 4(1):41–63, May 2000.

[75] J. V. Hajnal, D. L. G. Hill, and D. J. Hawkes, editors. *Medical Image Registration*. CRC Press, Boca Raton, Florida, US, 2001.

[76] M. Harris. Nvidia developer blog, 2019.

[77] G. L. Hart, C. Zach, and M. Niethammer. An optimal control approach for deformable registration. In *Proc IEEE Conference on Computer Vision and Pattern Recognition*, pages 9–16, 2009.

[78] M. Hernandez, M. N. Bossa, and S. Olmos. Registration of anatomical images using paths of diffeomorphisms parameterized with stationary vector field flows. *International Journal of Computer Vision*, 85(3):291–306, 2009.

[79] D. L. G. Hill, P. G. Batchelor, M. Holden, and D. J. Hawkes. Medical image registration. *Physics in Medicine and Biology*, 46:R1–R45, 2001.

[80] Jared Hoberock and Nathan Bell. Thrust, the cuda c++ template library, 2010.

[81] Cosmina Hogea, Christos Davatzikos, and George Biros. An image-driven parameter estimation problem for a reaction-diffusion glioma growth model with mass effects. *Journal of Mathematical Biology*, 56(6):793–825, June 2008.

[82] IBM. IBM XL C/C++ (version 16.1.1).

[83] Insight Software Consortium. ITKNDReg.

[84] Mark Jenkinson, Christian F. Beckmann, Timothy E. J. Behrens, Mark W. Woolrich, and Stephen M. Smith. FSL. *NeuroImage*, 62(2):782–790, August 2012.

[85] A. R. Jones, C. C. Overly, and S. M. Sunkin. The Allen Brain Atlas: 5 years and beyond. *Nature Reviews Neuroscience*, 10:821–828, 2009.

[86] S. Joshi, B. Davis, M. Jornier, and G. Gerig. Unbiased diffeomorphic atlas construction for computational anatomy. *NeuroImage*, 23(1):S151–S160, 2005.

[87] David Van Essen Kamil Ugurbil. Human Connectome Project. https://www.humanconnectome.org/study/hcp-young-adult, 2017.

[88] S.-Y. Kim, K. Chung, and K. Deisseroth. Light microscopy mapping of connections in the intact brain. *Trends in Cognitive Sciences*, 17(12):596–599, 2013.

[89] S. Klein, M. Staring, K. Murphy, M. A. Viergever, and J. P. W. Pluim. ELASTIX: A tollbox for intensity-based medical image registration. *Medical Imaging, IEEE Transactions on*, 29(1):196–205, 2010.

[90] L. Koenig, J. Ruehaak, A. Derksen, and J. Lellmann. A matrix-free approach to parallel and memory-efficient deformable image registration. *SIAM Journal on Scientific Computing*, 40(3):B858–B888, 2018.

[91] J. Krebs, H. Delingette, B. Mailhé, N. Ayache, and T. Mansi. Learning a probabilistic model for diffeomorphic registration. *IEEE Transactions on Medical Imaging*, 2019. (in press) DOI: 10.1109/TMI.2019.2897112.

[92] Leonard Kuan, Yang Li, Chris Lau, David Feng, Amy Bernard, Susan M. Sunkin, Hongkui Zeng, Chinh Dang, Michael Hawrylycz, and Lydia Ng. Neuroinformatics of the Allen Mouse Brain Connectivity Atlas. *Methods*, 73:4–17, February 2015.

[93] K. S. Kutten, N. Charon, M. I. Miller, J. T. Ratnanather, K. Deisseroth, L. Ye, and J. T. Vogelstein. A diffeomorphic approach to multimodal registration with mutual information: Applications to CLARITY mouse brain images. *ArXiv e-prints*, 2016.

[94] K. S. Kutten, N. Charon, M. I. Miller, J. T. Ratnanather, K. Deisseroth, L. Ye, and J. T. Vogelstein. A diffeomorphic approach to multimodal registration with mutual information: Applications to CLARITY mouse brain images. In *Proc Medical Image Computing and Computer-Assisted Intervention*, volume LNCS 10433, pages 275–282, 2017.

[95] Kwame S. Kutten, Nicolas Charon, Michael I. Miller, J. T. Ratnanather, Jordan Matelsky, Alexander D. Baden, Kunal Lillaney, Karl Deisseroth, Li Ye, and Joshua T. Vogelstein. A Large Deformation Diffeomorphic Approach to Registration of CLARITY Images via Mutual Information. *arXiv:1612.00356 [cs]*, August 2017.

[96] Kwame S. Kutten, Joshua T. Vogelstein, Nicolas Charon, Li Ye, Karl Deisseroth M.d, and Michael I. Miller. Deformably registering and annotating whole CLARITY brains to an atlas via masked LDDMM. In *Optics, Photonics and Digital Technologies for Imaging Applications IV*, volume 9896, pages 282–290. SPIE, April 2016.

[97] M. Lorenzi, N. Ayache, G. B. Frisoni, and X. Pennec. LCC-demons: a robust and accurate symmetric diffeomorphic registration algorithm. *NeuroImage*, 81:470–483, 2013.

[98] M. Lorenzi and X. Pennec. Geodesics, parallel transport and one-parameter subgroups for diffeomorphic image registration. *International Journal of Computer Vision*, 105(2):111–127, 2013.

[99] Falk Lüsebrink, Alessandro Sciarra, Hendrik Mattern, Renat Yakupov, and Oliver Speck. T1-weighted in vivo human whole brain MRI dataset with an ultrahigh isotropic resolution of 250 Mm. *Scientific Data*, 4(1):170032, March 2017.

[100] J. B. Antoine Maintz and M. A. Viergever. A survey of medical image registration. *Medical Image Analysis*, 2(1):1–36, 1998.

[101] Ana L. Manera, Mahsa Dadar, D. Louis Collins, and Simon Ducharme. Deformation based morphometry study of longitudinal MRI changes in behavioral variant frontotemporal dementia. *NeuroImage: Clinical*, 24:102079, January 2019.

[102] A. Mang and G. Biros. An inexact Newton–Krylov algorithm for constrained diffeomorphic image registration. *SIAM Journal on Imaging Sciences*, 8(2):1030–1069, 2015.

[103] A. Mang and G. Biros. Constrained H^1-regularization schemes for diffeomorphic image registration. *SIAM Journal on Imaging Sciences*, 9(3):1154–1194, 2016.

[104] A. Mang and G. Biros. A Semi-Lagrangian two-level preconditioned Newton–Krylov solver for constrained diffeomorphic image registration. *SIAM Journal on Scientific Computing*, 39(6):B1064–B1101, 2017.

[105] A. Mang and G. Biros. Constrained large deformation diffeomorphic image registration (CLAIRE), 2019. [Commit: v0.07-131-gbb7619e].

[106] A. Mang, A. Gholami, and G. Biros. Distributed-memory large-deformation diffeomorphic 3D image registration. In *Proc ACM/IEEE Conference on Supercomputing*, 2016.

[107] A. Mang, A. Gholami, C. Davatzikos, and G. Biros. PDE-constrained optimization in medical image analysis. *Optimization and Engineering*, 19(3):765–812, 2018. https://doi.org/10.1007/s11081-018-9390-9.

[108] A. Mang, A. Gholami, C. Davatzikos, and G. Biros. CLAIRE: a distributed-memory solver for constrained large deformation diffeomorphic image registration. *SIAM Journal on Scientific Computing*, 41(5):C548–C584, 2019.

[109] A. Mang and L. Ruthotto. A Lagrangian Gauss–Newton–Krylov solver for mass- and intensity-preserving diffeomorphic image registration. *SIAM Journal on Scientific Computing*, 39(5):B860–B885, 2017.

[110] Andrea Mechelli, Cathy J. Price, Karl J. Friston, and John Ashburner. Voxel-Based Morphometry of the Human Brain: Methods and Applications. *Current Medical Imaging Reviews*, 1(2):105–113, June 2005.

[111] M. I. Miller and L. Younes. Group actions, homeomorphism, and matching: A general framework. *International Journal of Computer Vision*, 41(1/2):61–81, 2001.

[112] M. Modat, G. R. Ridgway, Z. A. Taylor, M. Lehmann, J. Barnes, D. J. Hawkes, N. C. Fox, and S. Ourselin. Fast free-form deformation using graphics processing units. *Computer Methods and Programs in Biomedicine*, 98(3):278–284, 2010.

[113] J. Modersitzki. *Numerical methods for image registration.* Oxford University Press, New York, 2004.

[114] J. Modersitzki. *FAIR: Flexible algorithms for image registration.* SIAM, Philadelphia, Pennsylvania, US, 2009.

[115] Ashraf Mohamed, Evangelia I. Zacharaki, Dinggang Shen, and Christos Davatzikos. Deformable registration of brain tumor images via a statistical model of tumor-induced deformation. *Medical Image Analysis*, 10(5):752–763, October 2006.

[116] Abdullah Nazib, James Galloway, Clinton Fookes, and Dimitri Perrin. Performance of Registration Tools on High-Resolution 3D Brain Images. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 566–569, July 2018.

[117] Y. Nesterov. Smooth minimization of non-smooth functions. In *Mathematical Programming*, pages 127–152, 2005.

[118] neurodata. ARDENT.

[119] Christian J. Niedworok, Alexander P. Y. Brown, M. Jorge Cardoso, Pavel Osten, Sebastien Ourselin, Marc Modat, and Troy W. Margrie. aMAP is a validated pipeline for registration and segmentation of high-resolution mouse brain data. *Nature Communications*, 7(1):11879, July 2016.

[120] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, New York, US, 2006.

[121] NVIDIA. CUDA Toolkit (version 10.1).

[122] Nvidia. CUDA CUFFT Library, 2007.

[123] A. Paszke, S. Gross, S. Chintala, and G. Chanan. Tensors and dynamic neural networks in python with strong GPU acceleration, 2019.

[124] T. Polzin, M. Niethammer, M. P. Heinrich, H. Handels, and J. Modersitzki. Memory efficient LDDMM for lung CT. In *Proc Medical Image Computing and Computer-Assisted Intervention*, volume LNCS 9902, pages 28–36, 2016.

[125] Prateek Prasanna, Jhimli Mitra, Niha Beig, Ameya Nayate, Jay Patel, Soumya Ghose, Rajat Thawani, Sasan Partovi, Anant Madabhushi, and Pallavi Tiwari. Mass Effect Deformation Heterogeneity (MEDH) on Gadolinium-contrast T1-weighted MRI is associated with decreased survival in patients with right cerebral hemisphere Glioblastoma: A feasibility study. *Scientific Reports*, 9(1):1–13, February 2019.

[126] J. S. Preston. Python for computational anatomy.

[127] J. S. Preston. Python for computational anatomy, 2019. [Commit: v0.01-434-gf31ab43; Libraries: ITK4.13.2; boost1.69; FFTW3.3.6-pl2; python2.7; CUDA9.2.88].

[128] Luís Gustavo Ribeiro and Geraldo Busatto. Voxel-based morphometry in Alzheimers disease and mild cognitive impairment: Systematic review of studies addressing the frontal lobe. *Dementia & Neuropsychologia*, 10(2):104–112, 2016.

[129] L. Risser, F.-X. Vialard, R. Wolz, M. Murgasova, D. D. Holm, and D. D. Rueckert. Simultaneous multiscale registration using large deformation diffeomorphic metric mapping. *Medical Imaging, IEEE Transactions on*, 30(10):1746–1759, 2011.

[130] Edmund T. Rolls, Chu-Chung Huang, Ching-Po Lin, Jianfeng Feng, and Marc Joliot. Automated anatomical labelling atlas 3. *NeuroImage*, 206:116189, February 2020.

[131] R. Rossi, M. Pievani, T. Järvenpää, C. Testa, M. Koskenvuo, I. Räihä, J. Kaprio, G. B. Frisoni, J. O. Rinne, and M. P. Laakso. Voxel-based morphometry study on monozygotic twins discordant for Alzheimer's disease. *Acta Neurologica Scandinavica*, 133(6):427–433, June 2016.

[132] D. Rueckert, L. I. Sonoda, C. Hayes, D. L. G. Hill, M. O. Leach, and D. J. Hawkes. Non-rigid registration using free-form deformations: Application to breast MR images. *Medical Imaging, IEEE Transactions on*, 18(8):712–721, 1999.

[133] D. Ruijters. GPU accelerated pre-filtered cubic B-spline interpolation using CUDA, 2019.

[134] D. Ruijters, B.M. ter Haar Romeny, and P. Suetens. Efficient gpu-based texture interpolation using uniform b-splines. *Journal of Graphics Tools*, 13(4):61–69, 2008.

[135] D. Ruijters and P. Thévenaz. GPU prefilter for accurate cubic B-spline interpolation. *The Computer Journal*, 55(1):15–20, 2012.

[136] S. Saravanakumar and P. Thangaraj. A voxel based morphometry approach for identifying Alzheimer from MRI images. *Cluster Computing*, 22(6):14081–14089, November 2019.

[137] J. Shackleford, N. Kandasamy, and G. Sharp. On developing B-spline registration algorithms for multi-core processors. *Physics in Medicine and Biology*, 55(21):6329–6351, 2010.

[138] D. P. Shamonin, E. E. Bron, B. P. F. Lelieveldt, M. Smits, S. Klein, and M. Staring. Fast parallel image registration on CPU and GPU for diagnostic classification of Alzheimer's disease. *Frontiers in Neuroinformatics*, 7(50):1–15, 2014.

[139] R. Shams, P. Sadeghi, R. A. Kennedy, and R. I. Hartley. A survey of medical image registration on multicore and the GPU. *Signal Processing Magazine, IEEE*, 27(2):50–60, 2010.

[140] C. Sigg and M. Hadwiger. Fast third-order texture filtering. pages 313–329, February 2005.

[141] S. Sommer. Accelerating multi-scale flows for LDDKBM diffeomorphic registration. In *Proc IEEE International Conference on Computer Visions Workshops*, pages 499–505, 2011.

[142] A. Sotiras, C. Davatzikos, and N. Paragios. Deformable medical image registration: A survey. *Medical Imaging, IEEE Transactions on*, 32(7):1153–1190, 2013.

[143] Tyler C. Steed, Jeffrey M. Treiber, Michael G. Brandel, Kunal S. Patel, Anders M. Dale, Bob S. Carter, and Clark C. Chen. Quantification of glioblastoma mass effect by lateral ventricle displacement. *Scientific Reports*, 8(1):1–8, February 2018.

[144] Roger Stupp, Warren P Mason, Martin J Van Den Bent, Michael Weller, Barbara Fisher, Martin JB Taphoorn, Karl Belanger, Alba A Brandes, Christine Marosi, Ulrich Bogdahn, et al. Radiotherapy plus concomitant and adjuvant temozolomide for glioblastoma. *New England journal of medicine*, 352(10):987–996, 2005.

[145] S. Subramanian, K. Scheufele, M. Mehl, and G. Biros. Where did the tumor start? an inverse solver with sparse localization for tumor growth models. *ArXiv e-prints*, july 2019.

[146] Shashank Subramanian, Amir Gholami, and George Biros. Simulation of glioblastoma growth using a 3D multispecies tumor model with mass effect. *Journal of Mathematical Biology*, May 2019.

[147] Shashank Subramanian, Amir Gholami, and George Biros. Simulation of glioblastoma growth using a 3D multispecies tumor model with mass effect. *Journal of Mathematical Biology*, 79(3):941–967, August 2019.

[148] Shashank Subramanian, Klaudius Scheufele, Naveen Himthani, and George Biros. Multiatlas Calibration of Biophysical Brain Tumor Growth Models with Mass Effect. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2020*, Lecture Notes in Computer Science, pages 551–560, Cham, 2020. Springer International Publishing.

[149] Ganesh Sundaramoorthi and Anthony Yezzi. Variational pdes for acceleration on manifolds and application to diffeomorphisms. In *Proc Advances in Neural Information Processing Systems 31*, pages 3793–3803, 2018.

[150] Etsuo A. Susaki, Kazuki Tainaka, Dimitri Perrin, Hiroko Yukinaga, Akihiro Kuno, and Hiroki R. Ueda. Advanced CUBIC protocols for whole-brain and whole-body clearing and imaging. *Nature Protocols*, 10(11):1709–1727, November 2015.

[151] Raju Tomer, Li Ye, Brian Hsueh, and Karl Deisseroth. Advanced CLARITY for rapid and high-resolution imaging of intact tissues. *Nature protocols*, 9(7):1682–1697, 2014.

[152] A. Trouvé. Diffeomorphism groups and pattern matching in image analysis. *International Journal of Computer Vision*, 28(3):213–221, 1998.

[153] P. Valero-Lara. A GPU approach for accelerating 3D deformable registration (DARTEL) on brain biomedical images. In *Proc European MPI Users' Group Meeting*, pages 187–192, 2013.

[154] P. Valero-Lara. Multi-GPU acceleration of DARTEL (early detection of Alzheimer). In *Proc IEEE International Conference on Cluster Computing*, pages 346–354, 2014.

[155] T. Vercauteren, X. Pennec, A. Perchant, and N. Ayache. Diffeomorphic demons: Efficient non-parametric image registration. *NeuroImage*, 45(1):S61–S72, 2009.

[156] F.-X. Vialard, L. Risser, D. Rueckert, and C. J. Cotter. Diffeomorphic 3D image registration via geodesic shooting using an efficient adjoint calculation. *International Journal of Computer Vision*, 97:229–241, 2012.

[157] J. T. Vogelstein, E. Perlman, B. Falk, A. Baden, W. Gray Roncal, V. Chandrashekhar, F. Collman, S. Seshamani, J. L. Patsolic, K. Lillaney, M. Kazhdan, R. Hider, D. Pryor, J. Matelsky, T. Gion, P. Manavalan, B. Wester, M. Chevillet, E. T. Trautman, K. Khairy, E. Bridgeford, D. M. Kleissas, D. J. Tward, A. K. Crow, B. Hsueh, M. A. Wright, M. I. Miller, S. J. Smith, R. J. Vogelstein, K. Deisseroth, and R. Burns. A community-developed open-source computational ecosystem for big neuro data. *Nature Methods*, 11(15):846–847, 2018.

[158] Jennifer L. Whitwell. Voxel-Based Morphometry: An Automated Technique for Assessing Structural Changes in the Brain. *Journal of Neuroscience*, 29(31):9661–9664, August 2009.

[159] Marko Wilke, Robert A. Kowatch, Melissa P. DelBello, Neil P. Mills, and Scott K. Holland. Voxel-based morphometry in adolescents with bipolar disorder: First results. *Psychiatry Research*, 131(1):57–69, May 2004.

[160] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: An insightful visual performance model for multicore architectures. *Commun. ACM*, 52(4):65–76, April 2009.

[161] X. Yang, R. Kwitt, and M. Niethammer. Fast predictive image registration. In 48-57, editor, *Proc International Workshop on Deep Learning in Medical Image Analysis*, volume LNCS 10008, pages 48–57, 2016.

[162] X. Yang, R. Kwitt, M. Styner, and M. Niethammer. Quicksilver: Fast predictive image registration—A deep learning approach. *NeuroImage*, 158:378–396, 2017.

[163] L. Younes. Jacobi fields in groups of diffeomorphisms and applications. *Quarterly of Applied Mathematics*, 650(1):113–134, 2007.

[164] L. Younes. *Shapes and diffeomorphisms.* Springer, 2010.

[165] L. Younes, F. Arrate, and M. I. Miller. Evolutions equations in computational anatomy. *NeuroImage*, 45:S40–S50, 2009.

[166] Hosam Abozaid Yousef, Yasser Mohamed Bader-Eldein ElSerogy, Sherif Mohamed Abdelal, and Shaza Ragab Abdel-Rahman. Voxel-based morphometry in patients with mood disorder bipolar I mania in comparison to normal controls. *Egyptian Journal of Radiology and Nuclear Medicine*, 51(1):9, January 2020.

[167] M. Zhang and P. T. Fletcher. Fast diffeomorphic image registration via Fourier-approximated Lie algebras. *International Journal of Computer Vision*, pages 1–13, 2018.

[168] M. Zhang and P.T. Fletcher. Finite-dimensional lie algebras for fast diffeomorphic image registration. In *Proc Information Processing in Medical Imaging*, pages 249–260. Springer International Publishing, 2015.

[169] M. Zhang, W. M. Wells, and P. Golland. Probabilistic modeling of anatomical variability using a low dimensional parameterization of diffeomorphisms. *Medical Image Analysis*, 41:55–62, 2017.

[170] Miaomiao Zhang and P. Thomas Fletcher. Fast Diffeomorphic Image Registration via Fourier-Approximated Lie Algebras. *International Journal of Computer Vision*, 127(1):61–73, January 2019.

[171] Y. Zhang, M. Brady, and S. Smith. Segmentation of brain MR images through a hidden Markov random field model and the expectation-maximization algorithm. *IEEE transactions on medical imaging*, 20(1):45–57, January 2001.