



Méthodes de reconstruction avancées pour l'imagerie par résonance magnétique à haute résolution

Mohamed Aziz Zeroual

► To cite this version:

Mohamed Aziz Zeroual. Méthodes de reconstruction avancées pour l'imagerie par résonance magnétique à haute résolution. Medical Imaging. Université de Lorraine, 2025. English. ⟨NNT : 2025LORR0167⟩. ⟨tel-05517356⟩

HAL Id: tel-05517356

<https://theses.hal.science/tel-05517356v1>

Submitted on 18 Feb 2026

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



HAL Authorization



**UNIVERSITÉ
DE LORRAINE**

**BIBLIOTHÈQUES
UNIVERSITAIRES**

AVERTISSEMENT

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur. Ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document.

D'autre part, toute contrefaçon, plagiat, reproduction illicite encourt une poursuite pénale.

Contact bibliothèque : ddoc-theses-contact@univ-lorraine.fr
(Cette adresse ne permet pas de contacter les auteurs)

LIENS

Code de la Propriété Intellectuelle. articles L 122. 4

Code de la Propriété Intellectuelle. articles L 335.2- L 335.10

http://www.cfcopies.com/V2/leg/leg_droi.php

Advanced Reconstruction Methods for High-Resolution Magnetic Resonance Imaging

*Méthodes de reconstruction avancées pour
l'imagerie par résonance magnétique à haute
résolution*

THÈSE

présentée et soutenue publiquement le 02 Décembre 2025

pour l'obtention du titre

DOCTEUR DE L'UNIVERSITE DE LORRAINE

(Mention : Automatique, Traitement du Signal et des Images, Génie Informatique)

par

Mohamed Aziz ZEROUAL

Composition du jury

<i>Président :</i>	Dr. Hélène Ratiney	Directrice de recherche, CNRS, CREATIS, Université Claude Bernard Lyon 1
<i>Rapporteurs :</i>	Pr. Aurélien Bustin	Professeur Junior, IHU LIRYC, Université de Bordeaux
	Pr. David Atkinson	Professor, University College Lon- don, Department of Imaging, UK
<i>Directeur de thèse :</i>	Dr. Freddy Odille	Directeur de Recherche, INSERM, IADI, Université de Lorraine, Nancy
	Dr. Pierre-André Vuissoz	Ingénieur de Recherche, IADI, Uni- versité de Lorraine, Nancy

Acknowledgments

At the end of this journey, I wish to express my deepest gratitude to all those who, in one way or another, have contributed to this achievement and supported me throughout these intense years, filled with challenges, learning, and personal growth.

First and foremost, I owe my sincere thanks to my thesis supervisors. Freddy, your guidance, scientific expertise, and encouragement have been invaluable throughout the course of this work. I am deeply grateful for the trust you placed in me since day one, for your availability, and for the discussions that have shaped the direction of this work. Pierre-André, thank you for the insightful comments and invaluable suggestions throughout the preparation of this thesis. your ability to identify critical details and provide precise feedback has greatly improved the clarity and rigor of my work. I am especially grateful for the generous sharing of your scientific knowledge, which deepened my understanding of MRI.

I would like to sincerely thank the IADI laboratory, precisely Jacques Felblinger, for welcoming me during these years of research. The laboratory provided not only the facilities and resources necessary to carry out my work, but also a stimulating environment where collaboration and exchange were always encouraged. I am grateful for the opportunities to participate in seminars, discussions, and meetings that allowed me to broaden my scientific perspective and benefit from the expertise of others. The daily interactions with colleagues in the laboratory created a supportive and constructive atmosphere that greatly facilitated the progress of this thesis. Being part of this community has been an enriching experience, both professionally and personally, and I am thankful to the entire laboratory, in addition to the CIC-IT team.

I would also like to thank our project collaborators, in particular Natalia Dudysheva, Vincent Gras and Franck Mauconduit from the CEA NeuroSpin team, for their contributions and fruitful discussions. Their expertise have been highly beneficial to the advancement of this work, and I am grateful for the opportunity to collaborate with them within the MOSAR project.

During these years of doctoral research, I was fortunate to be surrounded not only by colleagues, but by what I now consider a new family. Each of you has contributed in your own way to making this journey more enjoyable, and memorable. I would like to take this opportunity to thank you individually:

To my office friends, I owe a very special thank you:

Guillaume (Le nouveau Papa), my office mate for the entire three years, and the very first person I met when I arrived. I am truly grateful for your presence and for the countless hours we spent side by side.

Catherine, thank you for your kindness and your constant willingness to help. Take care of Auggie.

Ikram, My compatriot, thank you for your support and companionship throughout this period. Having someone to share both cultural references and professional challenges with was always a source of comfort and strength.

Diego, Amigo, your friendship, humor, and spirit brought a lot of joy into the office.

Coralie, Thank you for bringing with you an immense charge of positive energy that never failed to lift the spirits of those around you.

Jérémy and Paul, I am grateful for the friendly atmosphere you helped create in our group.

Aylin, I am grateful for the good times we shared during the last few months and for getting me into cooking !

Walid, thank you for the unforgettable vacations that will remain among the best memories of my life. From our trips to Albania and Montenegro, to our time in Strasbourg, each journey was filled with laughter, discovery, and friendship.

To my best friends, my brothers, Anas, Othmane, Simo and Tareq, I owe a very special thank you. Othmane, for the hours we spent talking about every possible subject, and for being there by my side during the most difficult moments. Your presence and support have been priceless. Anas, my brother in every sense of the word, who has shared with me my passion since 2019, who has witnessed so many versions of me, and who never stopped believing in me, even when I doubted myself. Your trust and friendship have carried me to this point, and I will always be grateful to both of you for walking this journey with me. Tareq and Simo, thank you for the countless conversations, the motivation during difficult moments, and the joy you brought throughout these years.

Above all, my deepest gratitude goes to my parents, Omar and Nadia who gave me everything. You have stood by me through every consequence, trusted me from day zero, and offered me the best possible education and values to build my life upon. None of this would have been possible without your sacrifices, your patience, and your unconditional love. To my sisters Kenza and Meriem and my brother Amine, thank you for trusting me throughout my entire life, for supporting me in every step of the way, and for always believing in me. I feel truly blessed and proud to have such a family, whose presence, encouragement, and love have been the strongest foundation of this journey.

Finally, I would like to dedicate this thesis to the memory of my grandmother, who passed away last year. Her love, wisdom, and strength have always been a source of inspiration to me, and her presence continues to guide me even in her absence. This work is, in many ways, a tribute to everything she represented in my life.

Abstract

Ultra-high-field (UHF) Magnetic Resonance Imaging (MRI) offers submillimetric resolution and enhances tissue contrast, which is particularly promising for assessing the small epileptogenic lesions in pediatric patients with drug-resistant epilepsy. Early identification of such lesions is complicated from a clinical perspective, as surgical outcomes are strongly related to both the age at intervention and the precise localization of the epileptogenic zone. Therefore, long acquisition times and the high sensitivity of UHF MRI to patient motion may generate severe artifacts that compromise image quality.

The computational clinical feasibility of iterative reconstruction frameworks was investigated. Using the roofline performance model, and taking the Generalized Reconstruction by Inversion of Coupled Systems (GRICS) software as an example of such frameworks, the performance of several parallel implementations of GRICS was quantified on modern CPU and GPU platforms. Hybrid Open Multi Processing (OpenMP) / Message Passing Interface (MPI) parallelization provided important acceleration but remained bandwidth-limited. A dedicated CUDA implementation was developed, exploiting Single Instruction Multiple Threads (SIMT) parallelism and optimized memory access with cuFFT/cuBLAS libraries. On NVIDIA A100 GPUs, the CUDA solution achieved 9–14 $\times$ speedup compared with CPU baselines, drastically reducing reconstruction times and approaching clinical feasibility and applicability.

Moreover, the integration of motion-compensated reconstruction into clinically relevant sequence designs (MPRAGE, MP2RAGE) at 7T was explored. Systematic evaluations in phantom and in vivo datasets assessed the impact of pseudorandom sampling strategy, navigator configuration, and motion model selection on reconstruction quality. Quantitative sharpness indices and statistical analyses confirmed that Shuffle Lines sampling yields the best reconstructions. Another result concerned the non-rigid motion model consideration in brain MRI, since most of the state of the art works about motion correction in brain MRI mainly focus on rigid modeling. We proved that non-rigid motion modeling can, in some cases, yield better reconstruction quality and motion compensation.

In this thesis, we demonstrate that rigid and non-rigid motion modeling, randomized acquisition strategies, and GPU-accelerated implementations enable robust motion correction for high-resolution MRI at 7T. By addressing both physics-based modeling and computational efficiency, the proposed framework directly considers the clinical need for reliable, high-resolution pediatric neuro-imaging.

Résumé de la Thèse

Introduction

L'imagerie par résonance magnétique (IRM) occupe une place centrale dans le diagnostic médical grâce à sa capacité de fournir des images anatomiques et fonctionnelles non invasives avec un excellent contraste des tissus mous. Depuis plusieurs décennies, les progrès technologiques des champs magnétiques statiques, des antennes radiofréquence et des techniques de reconstruction ont permis d'améliorer simultanément la résolution spatiale, la vitesse d'acquisition et la robustesse des séquences. Dans ce contexte, l'avènement des IRM à très haut champ ($\geq 7T$) marque une étape importante, offrant un gain significatif en rapport signal sur bruit (SNR) et ouvrant la voie à des acquisitions *in-vivo* submillimétriques.

L'IRM à 7T permet ainsi de révéler des détails micro-structuraux du cerveau, d'améliorer la sensibilité aux contrastes T_1 et T_2 , et d'explorer des biomarqueurs plus fins pour l'étude des pathologies neurologiques. Ces avancées présentent un intérêt majeur pour la recherche fondamentale et clinique, notamment en neurodégénérescence, en oncologie et dans l'étude de la connectivité cérébrale. Cependant, cette montée en champ ne se fait pas sans contrepartie. L'augmentation de la fréquence de Larmor entraîne des inhomogénéités accrues du champ radiofréquence, des variations spatiales du signal et des effets diélectriques. Le temps de relaxation longitudinal (T_1) tend à s'allonger, tandis que le temps de relaxation transverse effectif (T_2^*) diminue, modifiant le contraste habituel des tissus et compliquant l'optimisation des séquences. Par ailleurs, les contraintes liées au débit d'absorption spécifique (SAR) deviennent plus impactantes, limitant la puissance RF utilisable et imposant des compromis dans la conception des impulsions RF.

Au-delà de ces considérations physiques, un problème particulièrement critique en imagerie cérébrale haute résolution est celui du mouvement. Même de faibles déplacements, qu'ils soient rigides (translations et rotations de la tête) ou non rigides (déformations induites par la respiration, l'activité cardiaque ou des pulsations physiologiques), compromettent la cohérence des données acquises dans l'espace de Fourier. Ces mouvements se traduisent par une perte de résolution effective, l'apparition d'artéfacts de flou, de ghosting ou de distorsion, et limitent directement la fiabilité des mesures quantitatives. Plus la résolution spatiale envisagée est élevée, plus la sensibilité aux mouvements devient critique. À 7T, où l'objectif est souvent de descendre à une résolution submillimétrique, même des micromouvements de très faible amplitude non perceptibles peuvent altérer considérablement la qualité de l'image reconstruite.

Pour répondre à ces défis, différentes approches ont été proposées. Certaines stratégies

consistent à limiter l'impact du mouvement dès l'acquisition, par l'utilisation de navigateurs, de séquences accélérées ou de schémas de randomisation de l'ordre d'acquisition. D'autres approches, dites rétrospectives, visent à corriger le mouvement a posteriori au moment de la reconstruction. Dans ce domaine, les méthodes dites « model-based », qui intègrent explicitement le mouvement dans le modèle physique de l'acquisition, apparaissent particulièrement prometteuses car elles permettent de restaurer la cohérence des données en exploitant la physique sous-jacente. La méthode *Generalized Reconstruction by Inversion of Coupled Systems* (GRICS) a montré sa capacité à corriger à la fois les mouvements rigides et non rigides en les intégrant directement dans l'opérateur d'encodage, et par la suite dans le pipeline de reconstruction.

Ainsi, l'IRM à très haut champ représente à la fois une opportunité et un défi : elle permet d'explorer de nouveaux contrastes et d'accéder à une résolution inégalée, mais impose en retour de développer des stratégies de correction du mouvement robustes et pertinentes. La présente thèse traite cette problématique, en combinant conception de séquences, modélisation physique du mouvement, calcul haute performance et évaluation quantitative afin de démontrer la faisabilité et l'intérêt clinique d'une reconstruction compensée en mouvement pour l'IRM cérébrale à 7T.

Objectif de la Thèse

L'objectif principal de ce travail est de développer, optimiser et évaluer une méthode de reconstruction compensée en mouvement pour l'IRM cérébrale à 7T, en s'appuyant sur l'approche GRICS. Cette méthode itérative intègre explicitement le mouvement dans l'opérateur d'encodage et permet de corriger le mouvement qui a eu lieu lors de la reconstruction, qu'il soit de nature rigide ou non rigide. Pour aboutir à ceci, plusieurs axes complémentaires ont été investigués :

- établir un formalisme mathématique rigoureux du modèle GRICS, couvrant à la fois les mouvements rigides et non rigides.
- analyser et optimiser les performances de la reconstruction GRICS sur des architectures CPU et GPU modernes, afin de réduire les temps de calcul et de quantifier l'applicabilité clinique de la méthode.
- modifier le code source de séquences IRM (MPRAGE et MP2RAGE) pour intégrer des schémas de randomisation de l'ordre d'acquisition. L'objectif étant de disperser d'une façon uniforme l'effet du mouvement sur l'ensemble des états de mouvement considérés.
- évaluer et comparer de façon quantitative l'efficacité de GRICS par rapport à des méthodes end-to-end concurrentes, notamment des approches basées sur de l'apprentissage profond (MC-NET).

Ainsi, cette thèse se situe entre modélisation physique et intégration du mouvement, calcul haute performance et conception de séquences, traitant les aspects algorithmiques

et techniques de la reconstruction en IRM. Elle vise à démontrer qu’une reconstruction compensée en mouvement, robuste et rapide, est possible à très haut champ, ce qui ouvre la voie à une exploitation clinique et scientifique du potentiel de l’IRM cérébrale à 7T.

Organisation de la Thèse

Cette thèse est structurée en dix chapitres principaux.

- **Chapitre 1** introduit les fondements physiques et techniques de l’IRM, en rappelant les équations de Bloch, les phénomènes de relaxation, l’encodage spatial, les séquences de base (SE, GRE), les séquences avancées (MPRAGE, MP2RAGE), les spécificités de l’IRM à très haut champ et le principe de l’imagerie parallèle.
- **Chapitre 2** présente la reconstruction IRM comme un problème inverse. Il formalise le modèle direct et le problème inverse, discute les méthodes de régularisation et les algorithmes d’optimisation pour résoudre ce problème.
- **Chapitre 3** traite la notion du mouvement en IRM. Les différents types de mouvement (rigide et non rigide) et leurs effets sur les images sont décrits, ainsi que les approches de correction de mouvement, prospectives, rétrospectives et plus récemment basées sur l’apprentissage profond, tout en présentant les limites de ces dernières.
- **Chapitre 4** est consacré au calcul haute performance en IRM. Les lois de Moore, Amdahl, Gustafson et Wirth y sont rappelées, les caractéristiques et spécificités des architectures CPU et GPU décrites. On présente également les outils de profilage de performance d’un programme, essentiels pour comprendre le comportement d’une application dans une architecture.
- **Chapitre 5** établit le formalisme mathématique de GRICS pour la reconstruction corrigée en mouvement. Les modèles rigide et non rigide sont présentés et décrits mathématiquement. Le modèle du Roofline est introduit. Cet outil permet de quantifier la performance d’un programme donné dans une architecture bien précise.
- **Chapitre 6** introduit le cadre de la programmation parallèle, décrit les implémentations parallèles de GRICS sur CPU et Matlab–GPU. Les performances sont évaluées via le modèle de Roofline et des optimisations hybrides MPI/OpenMP sont évaluées, ce qui a permis d’atteindre des temps de reconstruction de l’ordre de quelques minutes.
- **Chapitre 7** introduit la programmation parallèle en GPU et présente l’implémentation CUDA de GRICS. Les performances atteignent jusqu’à 9-14× d’accélération par rapport aux versions CPU, sans compromettre la qualité de la reconstruction.
- **Chapitre 8** est consacré à la modification du code source des séquences IRM (MPRAGE et MP2RAGE), en intégrant des stratégies de randomisation de l’ordre d’acquisition des données de l’espace- k .

- **Chapitre 9** analyse les reconstructions compensées en mouvement à 7T. Les modèles rigide et non rigide de GRICS sont comparés à une approche d'apprentissage profond (MC-NET). L'évaluation par indice de netteté (Sharpness Index) met en évidence la supériorité de GRICS, tout en mettant en avant l'apport que peut porter la modélisation non rigide dans l'amélioration de la qualité d'IRM cérébrale à 7T.
- **Chapitre 10** conclut la thèse en synthétisant les résultats obtenus et en discutant les perspectives futures.

Chapitre 1 : Fondements physiques et techniques de l'IRM

Le premier chapitre de cette thèse est consacré au rappel des bases physiques et mathématiques nécessaires à la compréhension de la formation de l'image en IRM. Il présente d'abord les principes fondamentaux de l'imagerie par résonance magnétique (IRM), en détaillant les notions de magnétisation nucléaire et leurs interactions avec les champs magnétiques appliqués. La description s'appuie sur les équations de Bloch, modélisant l'évolution temporelle du vecteur de magnétisation sous l'effet d'un champ magnétique $\mathbf{B}(t) = (B_x(t), B_y(t), B_0 + \Delta B_z(t))$, la magnétisation nucléaire est notée $\mathbf{M}(t) = (M_x(t), M_y(t), M_z(t))$.

$$\begin{aligned}\frac{dM_x(t)}{dt} &= \gamma(\mathbf{M}(t) \times \mathbf{B}(t))_x - \frac{M_x(t)}{T_2}, \\ \frac{dM_y(t)}{dt} &= \gamma(\mathbf{M}(t) \times \mathbf{B}(t))_y - \frac{M_y(t)}{T_2}, \\ \frac{dM_z(t)}{dt} &= \gamma(\mathbf{M}(t) \times \mathbf{B}(t))_z - \frac{M_z(t) - M_0}{T_1},\end{aligned}$$

Les phénomènes de relaxation sont ensuite détaillés : la relaxation longitudinale (T_1) et la relaxation transverse (T_2) constituent les bases de la génération du contraste en IRM. Les équations différentielles associées permettent de caractériser la régénération exponentielle de la composante longitudinale $M_z(t)$ et la décroissance de la composante transverse $M_{xy}(t)$ selon les expressions suivantes :

$$M_z(t) = M_0 (1 - e^{-t/T_1}), \quad M_{xy}(t) = M_{xy}(0) e^{-t/T_2}.$$

La localisation spatiale de l'information est possible par l'application de gradients linéaires de champ magnétique, qui permettent d'encoder la fréquence et la phase du signal. La construction de l'espace de Fourier (l'espace k) est décrite. Différents schémas d'échantillonnage du k -space sont présentés, du traditionnel parcours cartésien aux trajectoires radiales ou spirales, chacun possédant des avantages et des limitations, reliées notamment à la robustesse du schéma d'acquisition au mouvement. L'IRM à très haut champ ($B_0 \geq 7$ T) offre un gain significatif en rapport signal sur bruit, permettant d'accéder à une résolution submillimétrique et à des contrastes particuliers. Ce bénéfice s'accompagne toutefois de limitations majeures : augmentation des inhomogénéités RF

(B_1), allongement des temps de relaxation T_1 , réduction de T_2^* et contraintes liées au débit d'absorption spécifique (SAR). L'exploitation optimale du 7 T nécessite donc des stratégies adaptées de séquence et de transmission parallèle, mais surtout de correction de mouvement. Le chapitre aborde également les séquences d'impulsions classiques : Spin Echo (SE) et Gradient Echo (GRE), qui constituent les séquences de base de la plupart des protocoles IRM. Enfin, les séquences spécifiques MPRAGE et MP2RAGE, particulièrement adaptées à l'imagerie anatomique haute résolution à 7T, sont introduites. Ces séquences combinent des inversions de magnétisation et des trains d'échos pour générer un contraste T_1 pondéré stable tout en réduisant les biais liés à l'inhomogénéité du champ B_1 . Le chapitre se conclut par une introduction aux techniques d'imagerie parallèle, telles que SENSE et GRAPPA, qui exploitent les sensibilités spatiales des antennes multicanaux afin d'accélérer les acquisitions. Ces méthodes, bien qu'efficaces, engendrent des artéfacts de repliement et nécessitent une estimation précise des profils de sensibilité des canaux de réception.

En résumé, ce premier chapitre établit les bases théoriques de l'IRM : magnétisation, relaxation, équations de Bloch, encodage spatial, trajectoires d'échantillonnage et séquences d'acquisition et concepts de l'imagerie parallèle.

Chapitre 2 : La reconstruction IRM comme problème inverse

Le deuxième chapitre introduit la reconstruction IRM comme étant un problème inverse. La formation d'une image IRM est modélisée comme la résolution d'un système linéaire reliant les données mesurées dans l'espace de Fourier (k -space) au signal issu de la distribution des spins dans l'espace image. Cette modélisation correspond à un problème direct (*forward problem*), dont la résolution inverse constitue la base mathématique de la reconstruction d'image. Le modèle linéaire de l'acquisition peut s'écrire :

$$s = \mathbf{E}\rho + \mathbf{n},$$

où s est le vecteur des mesures de l'espace k , $\mathbf{E}$ étant l'opérateur d'encodage, ρ l'image reconstruite et $\mathbf{n}$ un terme de bruit. Le problème de reconstruction IRM est ainsi un problème inverse, généralement mal posé en cas de sous-échantillonnage des données. Pour stabiliser la solution, il est nécessaire d'introduire des contraintes supplémentaires sous forme de régularisation. La formulation générale s'écrit :

$$\hat{\rho} = \arg \min_{\rho} \|\mathbf{E}\rho - s\|_2^2 + \lambda \mathcal{R}(\rho),$$

où le premier terme est un terme de fidélité aux données acquises, et $\mathcal{R}(\rho)$ représente un terme de pénalisation (norme L_2 , variation totale, sparsité, etc.). Le paramètre λ règle alors le compromis entre fidélité et régularisation. Différents schémas numériques sont détaillés : - Les méthodes de descente de gradient et de gradient conjugué, adaptées aux grands systèmes creux, sont couramment utilisées pour résoudre le problème de minimisation au sens des moindres carrés. - Les approches plus avancées exploitent

l'optimisation non lisse, permettant d'incorporer des régularisations comme la variation totale ou les contraintes de parcimonie en Compressed Sensing. - La reconstruction « model-based » repose sur l'intégration explicite de modèles physiques (mouvement, parallélisation, champs inhomogènes) afin de coupler la physique de l'acquisition au processus de reconstruction.

Ce chapitre établit le cadre mathématique et algorithmique de la reconstruction IRM comme problème inverse. Il présente la formulation générale, les méthodes de régularisation et les principaux algorithmes d'optimisation. Ces notions constituent la base des développements de la thèse, centrés sur la compensation du mouvement et l'intégration de modèles physiques dans des schémas de reconstruction itérative.

Chapitre 3 : Le mouvement en IRM

Le chapitre 3 traite l'impact du mouvement en IRM et les approches développées pour son estimation et sa compensation. Deux types de mouvement sont distingués : le mouvement rigide, correspondant à des translations et rotations, et le mouvement non rigide, qui inclut les déformations locales liées aux processus physiologiques, respiration, activité cardiaque, péristaltisme et micromouvements physiologiques, chacun étant à l'origine d'artéfacts spécifiques (flou, ghosting, distorsion). L'effet global du mouvement en IRM se traduit par une dégradation de la qualité diagnostique des images.

Les méthodes de correction du mouvement en IRM sont regroupées en deux grandes familles. Les approches prospectives cherchent à mesurer et corriger le mouvement pendant l'acquisition, grâce à des navigateurs ou des systèmes de suivi en temps réel. Les approches rétrospectives, plus flexibles, exploitent des données auxiliaires ou des modèles mathématiques pour corriger a posteriori les artéfacts. Parmi ces dernières, les méthodes basées sur un modèle physique (model-based MoCo) intègrent explicitement le mouvement dans la reconstruction en considérant ses effets dans l'opérateur d'encodage. En parallèle, les méthodes récentes basées sur l'apprentissage profond sont présentées : correction dans le domaine image, hybridation image/ k -space et estimation explicite de champs de déformation. Ces réseaux, incluant des transformeurs spatiaux et des architectures récurrentes, permettent d'apprendre directement la compensation à partir de données simulées ou réelles, mais souffrent encore de limites de généralisabilité entre différents types de jeux de données, ainsi que d'exigences en annotations de données.

En conclusion, ce chapitre met en évidence la diversité des sources de mouvement en IRM et leurs conséquences sur la qualité d'image. Il montre également l'évolution des méthodes de compensation, depuis les approches classiques fondées sur des navigateurs et des modèles physiques jusqu'aux approches d'intelligence artificielle.

Chapitre 4 : Calcul haute performance en IRM

Le quatrième chapitre aborde le rôle du calcul haute performance (HPC) dans le contexte de la reconstruction IRM. Trois lois fondamentales du calcul intensif sont rappelées : la loi de Moore, qui décrit la croissance exponentielle du nombre de transistors dans les

processeurs ; la loi d'Amdahl, qui établit la limite d'accélération due à la partie séquentielle d'un algorithme ; la loi de Gustafson, qui nuance cette vision en montrant que l'augmentation de la taille des problèmes permet de mieux exploiter le parallélisme ; et la loi de Wirth, qui met en relation l'interaction logiciel-matériel. Ces concepts servent de cadre pour analyser les contraintes et opportunités offertes par les architectures CPU/GPU modernes.

Le chapitre détaille ensuite l'architecture des processeurs actuels, notamment la hiérarchie mémoire (latences, bandes passantes, caches) et l'organisation architecturale des composantes de calcul. Dans le cas du calcul sur CPU, les possibilités de parallélisation par multi-threading et vectorisation sont discutées, mais les limites de bande passante mémoire demeurent un facteur limitant critique pour les applications IRM. Les GPU, en revanche, reposent sur un paradigme massivement parallèle basé sur le modèle SIMT (Single Instruction, Multiple Threads), particulièrement efficace pour les opérations de type FFT et les convolutions, fréquemment utilisées dans la reconstruction IRM. Enfin, le chapitre introduit les outils de profilage permettant de caractériser les performances effectives d'un programme sur CPU et GPU. Ces analyses sont essentielles pour identifier les goulets d'étranglement (mémoire, calcul, communication inter-processus) et par la suite guider les optimisations de code possibles.

En conclusion, ce chapitre établit les bases nécessaires à l'utilisation du calcul haute performance en IRM. Il montre que les lois du calcul, l'architecture des processeurs, et les outils de profilage constituent des éléments indispensables pour concevoir des implémentations efficaces et préparer l'optimisation des algorithmes de reconstruction développés dans les chapitres suivants.

Chapitre 5 : Théorie et formalisme mathématique

Le chapitre 5 introduit le formalisme théorique et mathématique qui décrit les méthodes de reconstruction utilisées dans ce travail. Il s'ouvre avec la présentation du principe général de la méthode *Generalized Reconstruction by Inversion of Coupled Systems* (GRICS), une approche de reconstruction itérative intégrant explicitement le mouvement dans l'opérateur d'encodage. Le formalisme de GRICS est détaillé pour deux types de mouvements. Dans le cas rigide, les déplacements sont décrits par des translations et des rotations appliquées au signal mesuré, ce qui permet d'adapter analytiquement les équations d'encodage. Dans le cas non rigide, le mouvement est représenté par des champs de déformation spatiaux et temporels, incorporés directement dans la matrice d'encodage, ce qui généralise la méthode à des mouvements non rigides complexes, liés aux déformations physiologiques.

Ensuite, le chapitre présente le *modèle de Roofline*, outil central pour l'analyse de performance des algorithmes de reconstruction. Sa formulation mathématique relie la performance d'un code (FLOP/s) à l'intensité arithmétique, définie comme le rapport entre le nombre d'opérations effectuées et le volume de données transférées. Le modèle de base permet de déterminer si une implémentation est limitée par la puissance de calcul ou par la bande passante mémoire. Des extensions du modèle sont ensuite discutées, intégrant notamment la hiérarchie mémoire, les coûts de communication et l'impact énergétique,

afin d’obtenir une vision plus fine des limites matérielles et des marges d’optimisation possibles.

En conclusion, ce chapitre combine le formalisme de GRICS, adapté aux mouvements rigides et non rigides, et le modèle de Roofline, utilisé pour évaluer la complexité et les performances des implémentations. Ces deux cadres – la reconstruction physique et l’analyse de performance – constituent les piliers sur lesquels reposent les contributions pratiques présentées dans les chapitres suivants.

Chapitre 6 : Implémentations parallèles CPU et Matlab–GPU de GRICS

Le chapitre 6 est consacré à l’implémentation parallèle du noyau de reconstruction GRICS sur différentes architectures de calcul. Trois approches principales sont explorées : une version OpenMP, une version hybride MPI/OpenMP et une implémentation préliminaire sous Matlab–GPU. L’évaluation est réalisée à l’aide du modèle de roofline sur deux processeurs CPU (Intel Xeon Gold et AMD EPYC) et deux GPU (NVIDIA Quadro RTX 5000 et NVIDIA A100).

Les méthodes expérimentales incluent la description des jeux de données (données IRM mammaires et cérébrales à 3T), l’utilisation de plusieurs machines de test (CPU multi-cœurs et GPU), et l’évaluation par le modèle du roofline. Différents indicateurs sont analysés : intensité arithmétique, empreinte mémoire, complexités de calcul et de communication. Des optimisations spécifiques sont introduites, notamment au niveau de la compilation et de la gestion de l’affinité des threads. Les analyses des modèles du roofline montrent que, sur CPU, le code est *memory-bound*. Sur Xeon Gold, la performance est de 16 GFLOP/s, soit une efficacité de 28 % par rapport à la performance théorique maximale de 56 GFLOP/s. Sur AMD EPYC, les 80 GFLOP/s atteints traduisent une efficacité de 52 %, rendue possible par la bande passante DRAM élevée (221 GB/s) et une meilleure réutilisation des données en mémoire cache. Sur GPU, le code est au contraire *compute-bound*, mais reste très éloigné du plafond : 5 GFLOP/s sur RTX 5000 et 16 GFLOP/s sur A100. L’utilisation de la mémoire est optimale (90% et 89% de la bande passante), mais la faible occupation des multiprocesseurs de flux (SMs) est à l’origine des faibles performances. La taille du niveau de cache L2 de l’A100 améliore la réutilisation des données, mais la sous-utilisation des SMs demeure le principal goulot. Les optimisations CPU confirment l’importance du *thread binding* : sur EPYC, les gains atteignent 9-17% avec OpenMP. La meilleure configuration hybride MPI/OpenMP, avec $n = N_t/2$ processus MPI et $m = N_c/2$ threads OpenMP, fournit des gains de 62-68 % selon les jeux de données (N_c et N_t sont respectivement le nombre de bobines et d’états de mouvement). Grâce à ces optimisations, le temps total de reconstruction est compris entre 80 et 102 secondes pour des volumes 3D haute résolution, soit inférieur au temps typique d’acquisition, ouvrant la possibilité d’une application clinique.

En conclusion, ce chapitre montre que GRICS bénéficie de la parallélisation CPU et GPU. Les CPU haut débit mémoire (EPYC) offrent de bonnes performances pour les algorithmes de reconstruction itératifs, tandis que les GPU révèlent un potentiel promet-

teur mais encore sous-exploité, motivant une implémentation CUDA dédiée, présentée au chapitre suivant.

Chapitre 7 : Implémentation CUDA de GRICS

Le chapitre 7 présente l’implémentation native de GRICS en CUDA testée sur le GPU NVIDIA A100. Les opérations matricielles décrites au niveau de l’application de l’opérateur ($E^H E$) ont été parallélisées en exploitant la gestion des threads et des blocks, avec recours aux bibliothèques cuBLAS et cuFFT. L’implémentation CUDA a été testée sur 6 jeux de données, 3 jeux de données issues d’IRM mammaire à 3T acquis avec une 3D GRE, et 3 jeux de données issues d’IRM cérébrales à 7T, acquises avec une 3D MPRAGE.

Les résultats expérimentaux sur les jeux de données montrent une accélération comprise entre 9.7 et 13.9 par rapport à la version CPU hybride MPI/OpenMP. Comparée à la version Matlab–GPU (≈ 16 GFLOP/s), l’implémentation CUDA atteint ≈ 250 GFLOP/s, soit un gain de $15.6\times$, et se rapproche encore plus des limites prédictives du modèle de Roofline. L’analyse confirme que l’opérateur global $E^H E$ est limité par la bande passante mémoire (bandwidth-bound), malgré une excellente utilisation de la mémoire HBM ($\approx 2,94$ FLOP/Byte). La qualité des images reconstruites est préservée : le NRMSE reste compris entre 10^{-6} et 10^{-4} , bien en dessous du seuil de tolérance du noyau de reconstruction de GRICS $\epsilon_r = 10^{-3}$. Les images reconstruites à partir des jeux de données en mouvement rigide (cerveau, 7T) et non rigide (sein, 3T) montrent une suppression efficace des artéfacts de mouvement.

En conclusion, l’implémentation CUDA exploite les capacités mémoire des GPU modernes et améliore considérablement les performances sans perte de qualité d’image. Les gains observés ouvrent la voie à des optimisations futures (réduction de précision en FP16, meilleure occupation des SMs et utilisation des Tensor Cores) afin de réduire davantage l’écart avec la limite théorique prédite par le modèle du Roofline.

Chapitre 8 : design des séquences MPRAGE/MP2RAGE

Le Chapitre 8 est consacré à la modification de séquences MPRAGE et MP2RAGE en intégrant des stratégies de randomisation de l’ordre d’acquisition afin d’atténuer la sévérité des artéfacts de mouvement. Trois méthodes pseudo-aléatoires ont été implémentées : Shuffle Lines (SL), Shuffle All (SA) et Elliptical Shuffling (ES). La randomisation, réalisée à l’aide de l’algorithme de Fisher–Yates, permet de disperser les erreurs de phase et de déplacement uniformément dans l’espace k , créant ainsi une sorte d’incohérence.

Des navigateurs (ligne centrale de l’espace k) ont été insérés au début de chaque RAGE shot, incluant dans certaines acquisitions une insertion des navigateurs dans les 3 directions spatiales. Les signaux navigateurs sont ensuite corrigés par recalage non rigide et analysés par ACP, fournissant ainsi les signaux de mouvement. Pour limiter la complexité du problème GRICS, un clustering réduit le nombre d’états de mouvement à douze représentatifs, contenant l’essentiel de l’énergie du k -space. Dans le Chapitre suivant, on teste la pertinence de l’ensemble des méthodes de randomisation proposées

avec la méthode de reconstruction GRICS.

Chapitre 9 : Reconstruction IRM compensée en mouvement mouvement à 7T

Le chapitre 9 évalue l'efficacité de GRICS pour la reconstruction compensée en mouvement en IRM cérébrale à 7T, en comparaison avec une méthode d'apprentissage profond (MC-NET). Dix jeux de données in vivo (MPRAGE/MP2RAGE, 0.6–1 mm isotropique avec les schémas de randomisation présentés dans le Chapitre 8) et plusieurs acquisitions fantômes ont été analysés. L'évaluation quantitative repose sur l'indice de netteté (SI), moyenné par coupe et comparé au test de Wilcoxon.

Les expériences sur fantômes montrent que la méthode Shuffle Lines préserve l'uniformité et la fidélité structurale, tandis que Shuffle All et Elliptical Shuffling introduisent des modulations d'intensité et du ringing. In vivo, GRICS améliore significativement la netteté par rapport aux images non corrigées, que ce soit avec un modèle de mouvement rigide ou non rigide. Les deux modèles présentent des performances globalement comparables, mais le modèle non rigide apporte des gains supplémentaires de 2-4 % du SI dans certains cas, notamment ceux à très haute résolution ou slab-selective. Concernant les navigateurs, l'utilisation d'une ou deux lignes centrales (k_x, k_y) améliore les reconstructions, tandis que l'ajout d'un troisième navigateur (k_z) dégrade parfois la qualité de l'image. La comparaison avec MC-NET montre que le réseau améliore la netteté lorsque la reconstruction converge, mais échoue dans 40% des cas (fluctuations d'intensité importantes). Même en cas de succès, MC-NET reste 5-10 % inférieur à GRICS, en particulier en présence de mouvements non rigides. Ces limites sont liées au fait que MC-NET a été entraîné sur des données MPRAGE 3T et ne généralise pas aux acquisitions 7T.

En conclusion, les résultats confirment que l'association d'un schéma Shuffle Lines, de navigateurs adaptés et d'un modèle de mouvement non rigide/rigide dans GRICS constitue une stratégie robuste pour l'imagerie cérébrale haute résolution à 7T. De plus, un résultat intéressant, du point de vue de l'IRM cérébrale, est la contribution que peut apporter la modélisation non rigide à la correction du mouvement.

Chapitre 10 : Conclusions et Futures Perspectives

Ce chapitre conclut la thèse en rappelant la problématique principale abordée : la dégradation de la qualité d'image en IRM à très haut champ (7T) due aux mouvements du patient lors d'acquisitions longues et à haute résolution. En combinant des approches de reconstruction itératives compensant le mouvement (GRICS), des implémentations de calcul haute performance et des stratégies de parcours aléatoire de l'espace k , la faisabilité d'une imagerie cérébrale robuste au mouvement à résolution submillimétrique a été démontrée, que ce soit avec un modèle de mouvement rigide ou non rigide.

Contributions

Etude des performances des méthodes de reconstruction itérative sur CPU et GPU

- Évaluation des performances de GRICS sur CPU et GPU en identifiant les limitations liées au calcul ou à la mémoire dans chaque architecture.
- Développement de méthodologies reproductibles pour comparer les performances des algorithmes de reconstruction IRM.

Implémentations haute performance sur CPU et GPU

- Déploiement d'implémentations parallèles sur architectures multi-cœurs (OpenMP, MPI) et GPU (CUDA) du noyau de reconstruction de GRICS.
- Accélération significative sur GPU (jusqu'à $\times 14$ sur NVIDIA A100) permettant une réduction du temps de reconstruction et une faisabilité clinique envisageable.

Intégration de stratégies d'acquisition aléatoires dans les séquences IRM

- Modification des séquences MPRAGE et MP2RAGE pour incorporer des stratégies de randomisation de l'espace k (Shuffle Lines, Shuffle All, Elliptical Shuffling).
- Réduction et homogénéisation des effets du mouvement grâce à une distribution uniforme des artéfacts sur l'ensemble des états de mouvement.

Évaluations quantitatives et qualitatives de GRICS pour l'IRM cérébrale à 7T

- Études sur fantômes et *in vivo* avec différents modèles de mouvement et stratégies de randomisation.
- GRICS a montré des performances supérieures à une approche d'apprentissage profond (MC-Net) en termes d'indice de netteté et de réduction des artéfacts de mouvement.
- L'introduction du modèle non rigide a amélioré la qualité de reconstruction et la définition des structures anatomiques cérébrales, ce qui pousse vers la considération de la modélisation non rigide dans le cadre de la reconstruction IRM cérébrale compensée en mouvement.

Perspectives Futures

Optimisation des protocoles d'imagerie

- Développement de protocoles dédiés avec instructions explicites aux volontaires de bouger pendant l'acquisition afin d'induire et d'identifier les événements de mouvement, ceci permettra une meilleure évaluation des performances.

Estimation des cartes de sensibilité

- Proposition d'utiliser les cartes de champ B_1 pour estimer directement les sensibilités des antennes plutôt que d'exploiter uniquement les lignes centrales de l'espace- k .
- Cette approche pourrait réduire le repliment résultant de l'imagerie parallèle et améliorer la qualité de l'image reconstruite.

Utilisation de la matrice de diffusion (matrice S)

- Exploration de la matrice de diffusion (matrice S) comme alternative aux navigateurs basés sur l'acquisition des lignes centrales de l'espace- k .
- La matrice S encode les interactions émission-réception, modulées par le mouvement du sujet. Grâce à sa sensibilité élevée au mouvement, elle pourrait produire un a priori plus robuste sur celui-ci.

Contents

Acknowledgments	i
Abstract	iii
Résumé de la Thèse	v
Table of Contents	xxii
List of Abbreviations	xxiii
List of Figures	xxx
List of Tables	xxxii
General Introduction	1
Motivation	2
Thesis Organization	3
List of Publications & Conferences	5
1 Background	7
1.1 Physical Fundamentals of MRI	7
1.1.1 A brief history of MRI	7
1.1.2 Nuclear Magnetization	7
1.1.3 Interactions between Magnetization and Externally Applied Mag- netic Fields	8
1.1.4 The Bloch Equations	8
1.1.5 Theory of Relaxation	10
1.1.5.1 Longitudinal relaxation	10
1.1.5.2 Transverse relaxation	11
1.1.6 Spatial Encoding and Image Reconstruction	11
1.1.6.1 Linear Magnetic Field Gradients	12
1.1.6.2 Slice Detection	13
1.1.6.3 Frequency Encoding	13
1.1.6.4 Phase Encoding	14
1.1.6.5 The Construction of k -space	14
1.1.6.6 Signal Formation and Detection	15

1.1.6.7	Image Reconstruction	17
1.1.7	Ultra-High-Field MRI	17
1.1.8	Pulse Sequences	19
1.1.8.1	Spin Echo	19
1.1.8.2	Gradient Echo	19
1.1.8.3	MPRAGE and MP2RAGE sequences	20
1.1.9	Fourier Space Sampling Trajectories	23
1.2	Parallel Imaging	24
1.2.1	Techniques for Parallel Imaging	27
1.2.1.1	Generalized autocalibrating partially parallel acquisitions - GRAPPA	27
1.2.1.2	Sensitivity Encoding	29
1.2.1.3	Sensitivity Maps Computation	30
1.2.1.4	Conjugate Gradient SENSE CG-SENSE	31
1.2.2	Artifacts in Parallel Imaging	32
1.3	Synopsis	33
2	MRI Reconstruction as an Inverse Problem	35
2.1	Forward and Inverse Problems	35
2.2	Solution of the MR reconstruction problem	36
2.3	Prior Incorporation and Regularization types	37
2.4	Least squares reconstruction	39
2.4.1	The Gradient Descent GD	39
2.4.2	The Conjugate Gradient CG	40
2.5	Model-based reconstruction	40
2.5.1	Smooth Optimization	41
2.5.2	Non-smooth Optimization	41
2.6	Synopsis	42
3	Motion in MRI	43
3.1	Rigid Vs. Non-Rigid Motion	43
3.2	Motion types	45
3.2.1	Physiological motion	45
3.2.2	Respiratory motion	45
3.2.3	Cardiac Motion	46
3.2.4	Peristaltic Motion	47
3.2.5	Head Motion	48
3.3	Effect of motion in MRI	49
3.4	Motion estimation and compensation	49
3.4.1	Prospective VS. Retrospective MoCo	49
3.4.2	Model-based MoCo.	52
3.4.3	Deep Learning based MoCo	53
3.4.3.1	Image-domain methods	53
3.4.3.2	k -space and hybrid-domain methods	53
3.4.3.3	Motion field estimation	54

3.4.3.4	Limitations of DL based MoCo methods	54
3.5	Synopsis	54
4	High Performance Computing in MRI	57
4.1	Foundation of High Performance Computing	57
4.2	Laws of Computing	59
4.2.1	Moore's Law	59
4.2.2	Amdahl's Law	60
4.2.3	Gustafson's Law	60
4.2.4	Wirth's Law	61
4.3	Modern Processor Architecture	61
4.3.1	The Memory System of Modern Processors	61
4.3.2	The Computing System of Modern Processors	63
4.4	CPU MRI	63
4.5	GPU MRI	65
4.6	Performance Profiling	66
4.6.1	GPU Profiling	67
4.6.2	CPU Profiling	67
4.7	Synopsis	67
5	Theory and Mathematical Formalism	69
5.1	Joint Optimization of image and motion	69
5.1.1	Rigid motion modeling	70
5.1.1.1	Case of a translational motion	70
5.1.1.2	Case of a rotational motion	71
5.1.2	Non-Rigid motion modeling	71
5.2	The Generalized Reconstruction by Inversion of Coupled Systems Method .	72
5.3	The Roofline Model	74
5.3.1	Mathematical Formulation	74
5.3.2	The basic roofline model Extensions	75
5.4	Synopsis	76
6	CPU/ Matlab-GPU Parallel Implementations of GRICS	77
6.1	Parallel Programming	77
6.1.1	Open Multi Processing parallelization	78
6.1.2	Message Passing Interface parallelization	79
6.2	Overview of the Reconstruction Kernel - RK	81
6.3	Parallel Implementations of the RK	81
6.3.1	Pure OpenMP Implementation	81
6.3.2	Hybrid MPI & OpenMP Implementation	82
6.3.3	MATLAB-GPU Implementation	82
6.4	Methods	83
6.4.1	Data Acquisition	83
6.4.2	Performance Analysis	83
6.4.2.1	The test Machines	83

6.4.2.2	The Empirical Roofline Toolkit	84
6.4.2.3	Application Characterization	85
6.4.2.4	Memory Footprint	88
6.4.2.5	Computational and Communication Complexities	88
6.4.3	Optimization techniques on CPUs	89
6.4.3.1	Compilation-level optimization	89
6.4.3.2	Threads Binding and affinity	90
6.5	Results	90
6.5.1	The roofline analysis of the RK on CPUs	90
6.5.2	The roofline analysis of the RK on GPUs	92
6.5.3	Optimization Results on CPUs	93
6.6	Discussion	95
6.7	Conclusion	99
7	CUDA Implementation of GRICS	101
7.1	Compute Unified Device Architecture parallelization	101
7.1.1	CUDA Kernel	101
7.1.2	Threads and Blocks Management	102
7.1.3	Single Instruction Multiple Thread SIMT	102
7.1.4	Useful Libraries	104
7.1.4.1	cuBLAS	104
7.1.4.2	cuFFT	104
7.2	Methods	104
7.2.1	CUDA Implementation of the RK	104
7.2.2	Data Acquisition	105
7.2.3	The test machines	105
7.2.4	Performance Profiling	106
7.3	Results	107
7.4	Discussion	111
7.5	Conclusion	111
8	MRI Sequence Design: MPRAGE & MP2RAGE	113
8.1	k -space Sampling Randomization Strategies	113
8.2	The pseudo-randomization methods implemented : ReorderInfo Class	114
8.3	k -space Center Line Navigator and Motion Eigenmodes extraction	117
8.4	Motion Clustering and Motion States Determination	118
9	Motion Compensated Reconstruction for brain MRI at 7T	121
9.1	Data Acquisition	121
9.2	Comparison with a Deep-Learning based solution: MC-NET	122
9.3	Quantitative Image Quality Metrics : The Sharpness Index - SI	123
9.4	Results	124
9.4.1	Phantom Evaluation : Impact of the sampling scheme	124
9.4.2	Impact of motion model	126
9.4.3	Impact of Navigators	128

CONTENTS

9.4.4	Quantitative Assessment and Comparison With MC-NET	128
9.5	Discussion	134
9.6	Conclusion	136
10	Conclusion and Future Works	137
10.1	Contributions	137
10.2	Future Work	138
	Bibliography	141
	Abstract	155
	Résumé	155

List of Abbreviations

AF	Acceleration Factor
CG	Conjugate Gradient
CPU	Central Processing Unit
CSF	Cerebrospinal Fluid
CUDA	Compute Unified Device Architecture
cuBLAS	CUDA Basic Linear Algebra Subroutines
cuFFT	CUDA Fast Fourier Transform library
EPI	Echo Planar Imaging
ERT	Empirical Roofline Toolkit
FA	Flip Angle
FE	Frequency Encoding
FLASH	Fast Low Angle Shot
FLOPS	Floating Point Operations per Second
FOV	Field of View
FP16/32/64	Half, Single, Double precision floating-point
FFT	Fast Fourier Transform
FFTW	Fastest Fourier Transform in the West
GD	Gradient Descent
GRAPPA	Generalized Auto-calibrating Partially Parallel Acquisition
GRE	Gradient Echo
GRICS	Generalized Reconstruction by Inversion of Coupled Systems
GPU	Graphics Processing Unit
GRASE	Gradient and Spin Echo
HBM	High Bandwidth Memory
HPC	High-Performance Computing
iPAT	Integrated Parallel Acquisition Technique
L1/L2	Level-1 / Level-2 cache
LS	Least Squares
MC-NET	Motion Correction Network
MoCo	Motion Correction
MPI	Message Passing Interface
MPRAGE	Magnetization-Prepared Rapid Gradient Echo
MP2RAGE	Magnetization-Prepared 2 Rapid Gradient Echoes
MRI	Magnetic Resonance Imaging
OpenMP	Open Multi-Processing

PE	Phase Encoding
PI	Parallel Imaging
RF	Radiofrequency
SENSE	Sensitivity Encoding
SI	Sharpness Index
SIMT	Single Instruction, Multiple Data
SIMT	Single Instruction, Multiple Threads
SNR	Signal-to-Noise Ratio
SM	Streaming Multiprocessor
SS	Slab-Selective
TE	Echo Time
TI	Inversion Times
TR	Repetition Time
TV	Total Variation
UHF	Ultra-High Field

List of Figures

1.1	Exponential regrowth of the longitudinal magnetization $M_z(t)$ after (a) a 90° pulse and (b) a 180° inversion pulse.	10
1.2	Exponential decay of the transverse magnetization due to T_2 relaxation. The blue curve highlights the decay of the transverse component $M_y'(t)$ after an excitation with a flip angle α . (b) Immediately after excitation, the transverse magnetization vectors are in phase, yielding a coherent net vector. The loss of phase coherence and the corresponding net transverse magnetization is illustrated, after one excitation (b), after one (c) or several (d) T_2 periods.	12
1.3	Illustration of a Cartesian k -space acquisition strategy. In the first step (a), both phase and frequency gradients are applied to shift the encoding origin from the center of k -space (red dot) to its starting corner. Once the desired k -space position is reached, the frequency encoding (readout) gradient is activated to acquire the first line of data. In the next repetition, a different combination of phase encoding and frequency gradients is applied to target the next line in k -space. This sequence of operations is iteratively repeated, line by line, until the entire k -space is covered (b).	15
1.4	A basic spin echo sequence begins with an initial RF excitation pulse, followed by signal decay due to dephasing. Subsequently, a 180° refocusing pulse is applied, which reverses the dephasing effects. As a result, a spin echo forms and reaches its maximum amplitude at the echo time (TE). . .	19
1.5	A basic gradient echo sequence begins with the application of a low flip angle RF pulse, denoted α , typically less than 90° in fast imaging protocols such as the FLASH (Fast Low Angle Shot) sequence. Following this excitation, a dephasing readout gradient G_x is applied to induce initial dephasing of the transverse magnetization. This is immediately followed by a rephasing gradient of opposite polarity to refocus the spins.	20
1.6	Schematic illustration of the MPRAGE sequence, where TI denotes the inversion time : the delay between the center of the non-selective 180° inversion pulse and the acquisition of the central k -space line within the RAGE readout train. The repetition time between successive excitations is indicated by TR , while α_1 represents the excitation flip angle applied during readout. TD corresponds to the relaxation delay between the end of a RAGE readout train and the onset of the subsequent inversion, completing one full MPRAGE repetition time.	21

1.7	Schematic illustration of the MP2RAGE sequence. The inversion times TI_1 and TI_2 are defined as the delays from the midpoint of the inversion pulse to the acquisition of the central k -space line in the respective readout blocks. The flip angles α_1 and α_2 correspond to the excitation angles used in the first and second readout modules, respectively.	22
1.8	Examples of 2D k -space trajectories used for spatial encoding: (A) Cartesian, (B) Echo-Planar Imaging (EPI), (C) Radial and (D) spiral.	24
1.9	A phantom image collected with various receiver coils Top: Reference phantom image. Middle and bottom: Magnitude and phase images as seen by four different receiver coils of an array. Each coil's unique spatial sensitivity (both in magnitude and phase) enables the reconstruction of undersampled data in Parallel Imaging. When acquiring images with the considered coils, an additional element of spatial localization is added to complement the gradient-based encoding, which is the localized sensitivity information. In magnitude, we can see that areas next to the coil in question appears brighter than those who are not.	25
1.10	Sampling and the resulting aliasing patterns for several acceleration factors. Top: k -space sampling patterns for fully sampled ($R = 1$), $R = 2$, and $R = 3$ accelerated acquisitions. Bottom: Corresponding phantom images, showing the aliasing artifacts as the acceleration factor R increases in each case. The acceleration factor stands for the scan time reduction relative to the fully-sampled case.	26
1.11	Sample k -space acquisition scheme and GRAPPA kernel for Cartesian GRAPPA with $R = 2$	29
1.12	SENSE equation of an acceleration factor R of 2. the aliased image is the summation of two image pixels weighted by the corresponding coil sensitivity at each point.	30
1.13	Diagram of the CG SENSE Method. The operator $E^H E$ is broken down into Coil sensitivity multiplication C_i , Fourier transform F , re-sampling S , gridding (if non-Cartesian sampling), inverse Fourier transform IFT , and coil combination steps.	32
3.2	Example of a real respiratory motion signal, illustrating the quasi-periodic nature of respiration with cycle-to-cycle variability in amplitude and frequency.	45
3.1	Simulated Diaphragm motion patterns during different breathing conditions showing normalized intensity over a period of 45 s. Top row: Normal resting breathing at 12 breaths per minute with regular amplitude and frequency. Middle row: Slow deep breathing at 8 breaths per minute demonstrating increased amplitude and prolonged respiratory cycles typical of relaxed or meditative breathing. Bottom row: Fast shallow breathing at 16 breaths per minute exhibiting reduced amplitude and shortened cycles characteristic of mild exertion or anxiety states. Negative values indicate upward diaphragm movement during expiration, while positive values represent downward movement during inspiration.	46

3.3	Electrocardiogram trace showing three consecutive cardiac cycles at 75 beats per minute with corresponding cardiac phases. The first row displays the ECG waveform with labeled P, Q, R, S, and T waves, where red shading indicates systolic phases (ventricular contraction from QRS onset to T wave end) and green shading indicates diastolic phases (ventricular relaxation and filling). The second row presents a binary representation of the cardiac cycle phases, illustrating the temporal relationship between electrical activity and mechanical cardiac events. Systole occupies approximately 37% of each cardiac cycle (300ms), while diastole comprises 63% (500ms at this heart rate), demonstrating the heart's longer filling period compared to ejection period.	47
3.4	Schematic representation of peristaltic motion as encountered in abdominal MRI. The intestinal wall (blue) exhibits traveling contractile waves (red dashed lines) that propagate along the lumen, resulting in dynamic changes in tube diameter. Boluses of content (orange) are displaced distally as a consequence of these contractions. Such cyclic deformation and transport introduce non-rigid motion patterns in the abdominal region, which can lead to signal inconsistencies, phase errors, and blurring in MR acquisitions.	48
3.5	Representation of gradients update to compensate for head rotations: initially, the frequency-encoding requires only one gradient G_z . To compensate for the head rotation occurred on the yz -plane, both the PE G_y and the FE G_z gradients are adjusted to align the FOV with the new had pose.	51
4.1	Comparison between CPU and GPU architectures. A CPU is contains a small number of powerful cores, each containing arithmetic logic units (ALUs), control units, and private L1 caches, complemented by larger shared L2/L3 caches and system DRAM. A GPU integrates many streaming multiprocessors (SMs), each with lightweight control units, ALUs, and L1 caches, organized for massive parallel execution. The GPU relies on high-throughput memory hierarchies (L2 cache and device DRAM). Data exchange between CPU and GPU occurs through the PCI-Express interconnect.	64
5.1	The basic roofline model	75
5.2	Example of the hierarchical roofline model.	76

6.1	OpenMP follows a fork-join execution model. At the start of a parallel region, the master thread forks a team of worker threads; each thread executes a subset of the region's code while sharing the same address space. When the region ends, an implicit barrier forces all threads to synchronize, after which they join back into a single thread context. The worker threads are then either terminated or parked by the runtime until the next parallel region begins. Because OpenMP allows the application, environment variables, or the runtime to pick the team size dynamically, the number of active threads can change from one parallel region to the next, letting the program adapt to workload and hardware availability.	78
6.2	OpenMP parallelization: multiple threads compute portions of the output array by directly accessing shared memory.	79
6.3	Illustration of MPI parallelization: each process operates independently with its own local memory. Communication is explicit and takes place via message passing (Send/Recv) between processes.	80
6.4	MPI parallelization: each process works independently on its local data, with results optionally gathered after computation. No shared memory exists between processes.	81
6.5	Modeling of a two-socket, two-NUMA-node architecture. LLC stands for the last-level cache, which is shared between all cores of the same socket/NUMA node. MC is the memory controller, and DRAM is the main memory.	85
6.6	Modeling of NVIDIA Quadro RTX 5000 GPU.	87
6.7	Modeling of NVIDIA A100 GPU.	87
6.8	Roofline analysis of GRICS on Intel Xeon Gold.	91
6.9	Roofline analysis of GRICS on AMD EPYC.	92
6.10	Computational and memory complexity of the reconstruction kernel as a function of elapsed time per iteration (on AMD EPYC CPU). The cyan and blue dashed lines in the plot represent the linear regression fits for computational and memory complexities, respectively. Time per iteration refers to the elapsed time of the reconstruction kernel per conjugate gradient iteration.	93
6.11	Roofline analysis of $E^H E$ on NVIDIA Quadro RTX 5000.	94
6.12	Roofline analysis of $E^H E$ on NVIDIA A100.	95
6.13	Performance throughputs on RTX 5000 and A100. On GPUs, the main memory system is used optimally, and the achieved throughputs are 90% and 89% for RTX 5000 and A100, respectively. The first difference concerns the L2 cache throughput, 22% for RTX 5000 and 92% for A100. However, the main hardware limitation is the SM utilization; the throughputs are low on both architectures.	96
6.14	Elapsed time of GRICS for different MPI processes on AMD EPYC; the number of OpenMP threads was fixed to 20. The reported times are the mean of 5 executions. The maximum standard deviations were 0.96, 0.14, 0.39, and 0.13 for D_1 , D_2 , D_3 , and D_4 , respectively. We highlighted the best times per dataset in red.	98

7.1	CUDA parallelization: the host CPU launches a GPU kernel. Each thread computes the sum of one (or more) elements independently in GPU memory. Threads are organized in blocks and access global memory directly. . .	101
7.2	Threads and Block in CUDA	103
7.3	Comparison of the elapsed time per CG iteration for the three implementations.	108
7.4	Roofline analysis of the implemented CUDA kernels on NVIDIA A100. Dataset D_1	108
7.5	Comparison of GRICS reconstructions with the three implementations for dataset D_2 . (A) Uncorrected reconstruction. (B): The CPU-only solution. (C): The Matlab-GPU solution. (D): The CUDA solution. The three solutions markedly reduced the motion artifacts visible in the blue and orange squares.	109
7.6	Comparison of GRICS reconstructions with the three implementations for dataset D_4 . (A) Uncorrected reconstruction. (B): The CPU-only solution. (C): The Matlab-GPU solution. (D): The CUDA solution. Improvements of the reconstruction quality were noticed on the frontal (yellow square) and the parietal (green square) lobes.	110
8.1	k -space acquisition reordering. The color gradient stands for the acquisition order.	118
8.2	Pipeline for motion states determination. (Step 1) The continuous motion signal illustrating physiological motion during MR acquisition. (Step 2) Quantization of the signal into 32 discrete amplitude levels. (Step 3) Computation of Fourier-space energy per quantized level, and highlighting the highest N_t'' , here 12, levels. (Step 4) Describing the motion signal with only N_t'' motion states.	119
9.1	Phantom evaluation of the three sampling strategies. For each sampling scheme, the right image is the same as the left image but with magnified window level, to highlight subtle artifacts. A: Without shuffling. B: Shuffle All (SA). C: Shuffle Lines (SL).	125
9.2	Phantom MP2RAGE (iPAT=3) reconstructed with different k -space randomization strategies, from both 0.9 mm isotropic acquisition (top row) and 0.6 mm isotropic slab-selective acquisition (bottom row). (A): Shuffle Lines (SL). (B) Shuffle All (SA).	125
9.3	GRICS Reconstructions for dataset D_1 . A) Uncorrected reconstruction. B) Motion Corrected reconstruction with rigid model. C) Motion Corrected reconstruction with non-rigid model.	126
9.4	GRICS reconstructions for dataset D_8 . A: Rigid correction. B: Non rigid correction. Top row: MP2RAGE images at TI_1 . Bottom row: MP2RAGE images at TI_2	127

LIST OF FIGURES

9.5	GRICS reconstructions for dataset D_7 . A : Uncorrected reconstruction. B : Rigid correction with 1 navigator. C : Non-Rigid correction with 1 navigator. D : Rigid correction with 2 navigators. E : Non-Rigid correction with 2 navigators. F : Rigid correction with 3 navigators. G : Non-Rigid correction with 3 navigators.	129
9.6	The impact of the navigator orientation on the reconstructed image.	130
9.7	An example of an MC-NET failed reconstruction(B). (A) shows the uncorrected reconstruction. (C) is the Rigid GRICS reconstruction.	131
9.8	GRICS vs MC-NET. A: GRICS without MoCo. B: GRICS with MoCo. C: MC-NET. Datasets D_1 and D_2	132
9.9	GRICS reconstructions for dataset D_{11} . (A) Reconstruction performed using sensitivity maps derived from the B_1^- field maps. (B) Reconstruction performed using sensitivity maps estimated from the central lines of k -space.	135

List of Tables

1.1	Comparison between MPRAGE and MP2RAGE sequences.	22
4.1	Typical memory access latencies across technologies.	62
6.1	The four test datasets.	84
6.2	The characteristics of the CPUs.	84
6.3	The characteristics of the GPUs. SM stands for Streaming Multiprocessor. SMs are akin to CPU cores.	86
6.4	NVIDIA Nsight compute metrics. FP stands for Floating Point. In the following, M_{prec} will stand for <code>sm__sass_thread_inst_executed_op_fp_prec_pred_on.sum</code> , where $prec \in \{64, 32, 16\}$, M_{TC} stands for <code>sm__inst_executed_pipe_tensor.sum</code> , and M_{m_l} for the m_l memory level metric, where $m_l \in \{L1, L2, HBM\}$	86
6.5	Memory footprint of Algorithm 5. $M = \max(NNZ(S_{t_i}))$ stands for the maximum number of non-zero elements per S_{t_i} row. The elements of operator T are pre-computed. N_{ps} stands for the number of physiological signals used to derive the spatial transformation matrix during acquisition. s_f refers to the size of the elements. In GRICS, we work with single-precision float, so $s_f = 4$ bytes (32 bits). FFT is conducted using FFTW implementation, so we do not need explicit storage of the FFT operator F	88
6.6	Elapsed time of GRICS with different OpenMP configurations on CPUs.	97
7.1	N_x, N_y, N_z are the number of samples in frequency encoding, phase-encoding, and slice directions, respectively. N_c is the number of coils in the coil array. N_t is the number of motion states.	106
7.2	The main characteristics of the test machines. SM stands for streaming multi-processor (akin to CPU cores). FP stands for floating point. The bandwidth of the memory levels and the peak FP performance were estimated empirically with the Empirical Roofline Toolkit ERT, which provides more realistic benchmarking of a workstation.	106
7.3	NVIDIA Nsight Compute metrics. FP stands for floating point.	107
9.1	The datasets of the study. SS stands for Slab-Selective. A_F is the GRAPPA acceleration factor. N_x, N_y and N_z stand respectively for the number of samples in frequency encoding direction, phase-encoding direction in the image plane, and in the direction of slices.	122

9.2	Phantom acquisitions used to evaluate the effect of k -space sampling order on image quality.	122
9.3	Wilcoxon signed-rank test results comparing Sharpness Index (SI) across reconstruction methods. N indicates the number of datasets included in the test. Significant differences ($p < 0.05$) are highlighted in bold. Since MC-NET provided interpretable reconstruction in only 6 datasets, we did not perform a statistical analysis involving this method (NA).	131
9.4	Sharpness Index (Higher is better) & Motion Model. A_D is the peak-to-peak maximal displacement field in mm. NA*: Not Available: excluded from analysis due to failed reconstruction	133

General Introduction

Motivation

Magnetic Resonance Imaging (MRI) has become an indispensable imaging modality in neuroscience and clinical practice; it provides high-resolution, non-invasive access to brain anatomy and function. The advent of ultra-high-field (UHF) systems, notably 7 Tesla scanners, has enabled new perspectives by significantly increasing the signal-to-noise ratio (SNR) and improving tissue contrast. These advances are particularly important for detecting subtle cortical malformations and epileptogenic lesions, where submillimetric precision directly impacts surgical planning.

However, realizing the full potential of UHF MRI faces two major obstacles: patient motion and computational complexity. First, long acquisition times at high resolution increase the impact of motion. Even submillimetric displacements can propagate across k -space, and yield severe artifacts, mainly blurring, ghosting, and loss of fine anatomical details. This challenge is even more present in vulnerable populations—pediatric patients, young subjects, or patients with neurodegenerative diseases—where immobility during prolonged scans may be difficult.

Second, methods to correct for motion have been constrained by the limits of computational power. Conventional reconstruction pipelines do not optimally integrate accurate physical motion models within clinically acceptable times. Non-rigid motion correction, while theoretically more powerful, has often been dismissed as impractical due to its heavy computational demand.

Recent progress at the intersection of motion compensated reconstruction and High-Performance Computing (HPC) offers an opportunity to overcome these limitations. Frameworks such as the Generalized Reconstruction by Inversion of Coupled Systems (GRICS) allow the incorporation of physiological signals and deformation fields directly into the reconstruction process. Moreover, the exponential increase of parallel computing resources—multicore CPUs, GPU accelerators, optimized numerical libraries—now enables reconstructions that were previously computationally intensive. GPU implementations in particular have demonstrated speedups of one order of magnitude or more, bringing advanced iterative reconstructions closer to clinical practice.

Moreover, sequence design innovations complement algorithmic MRI reconstruction advances. Techniques such as pseudo-random k -space sampling improve incoherence and mitigate motion sensitivity, while maintaining compliance with gradient system hardware and physiological constraints. The complex interaction between sequence design, robust motion modeling, and efficient parallel implementations defines the next frontier

of motion-robust MRI at 7T.

This thesis is motivated by the clinical and computational necessity to develop motion-compensated reconstruction strategies, that are practically deployable in a clinical context. By combining rigid and non-rigid motion modeling, optimized randomized scanning of Fourier space, and GPU-accelerated implementations, the work aims to demonstrate that motion-robust, high-resolution imaging at 7T is both technically feasible and clinically relevant.

Thesis Organization

This thesis is structured as follows:

- **Chapter 1** introduces the physical principles of Magnetic Resonance Imaging (MRI), including nuclear magnetization, relaxation mechanisms, and spatial encoding. Particular attention is given to ultra-high-field MRI (7T) and the main acquisition sequences (MPRAGE, MP2RAGE) for brain imaging.
- **Chapter 2** formulates MRI reconstruction as an inverse problem, presenting the mathematical framework of iterative reconstruction, regularization strategies, and optimization algorithms relevant to motion-compensated reconstruction.
- **Chapter 3** explores the sources and types of motion in MRI, including rigid and non-rigid motion, and reviews existing motion estimation and compensation techniques.
- **Chapter 4** introduces the fundamentals of high-performance computing (HPC), including processor architectures, parallelization strategies, and performance profiling tools.
- **Chapter 5** details the Generalized Reconstruction by Inversion of Coupled Systems (GRICS) method and the mathematical modeling of rigid and non-rigid motion. It also presents the roofline performance model used to quantify performance and guide optimization paths.
- **Chapter 6** describes CPU and Matlab–GPU implementations of GRICS, including OpenMP and MPI parallelization strategies, with a detailed analysis of complexities, performance bottlenecks and CPU optimization results.
- **Chapter 7** develops a CUDA implementation of GRICS on modern GPUs, using optimized libraries and GPU-specific memory architectures to achieve clinically acceptable reconstruction times.
- **Chapter 8** investigates MRI sequence design aspects, including k -space randomization strategies, navigator-based motion estimation, and energy-based clustering methods for the determination of motion states.

-
- **Chapter 9** presents an evaluation of motion-compensated reconstruction in brain MRI at 7T, including phantom experiments, in vivo studies, quantitative sharpness metrics, and a comparison with deep-learning-based solutions.
 - **Chapter 10** concludes the thesis by summarizing the main contributions, assessing their clinical and computational impact and discussing future perspectives.

List of Publications & Conferences

Research Articles

1. **Zeroual, M.A.**; Dudysheva, N.; Gras, V.; Mauconduit, F.; Isaieva, K.; Vuissoz, P.-A.; Odille, F. Motion-Compensated Reconstruction for brain MRI at 7T using Rigid and Non-Rigid Motion Models. In Peer Review in *Magnetic Resonance Materials in Physics, Biology and Medicine*.
2. **Zeroual, M.A.**; Dudysheva, N.; Gras, V.; Mauconduit, F.; Isaieva, K.; Vuissoz, P.-A.; Odille, F. Feasibility of Implementing Motion-Compensated Magnetic Resonance Imaging Reconstruction on Graphics Processing Units Using Compute Unified Device Architecture. *Appl. Sci.* 2025, 15, 5840. <https://doi.org/10.3390/app15115840>
3. **Zeroual, M.A.**; Isaieva, K.; Vuissoz, P.-A.; Odille, F. Performance Study of an MRI Motion-Compensated Reconstruction Program on Intel CPUs, AMD EPYC CPUs, and NVIDIA GPUs. *Appl. Sci.* 2024, 14, 9663. <https://doi.org/10.3390/app14219663>
4. Isaieva, K.; Meullenet, C.; Vuissoz, P.-A. Fauvel, M.; Nohava, L.; Laistler, E.; **Zeroual, M.A.**; Henrot, P.; Felblinger, J.; Odille, F. Feasibility of online non-rigid motion correction for high-resolution supine breast MRI. *Magn. Reson. Med.* 2023, 90, 2130–2143. <https://doi.org/10.1002/mrm.29768>.

Conference Proceedings

1. **Zeroual, M.A.**; Dudysheva, N.; Gras, V.; Mauconduit, F.; Isaieva, K.; Vuissoz, P.-A.; Odille, F. Feasibility of Implementing the Generalized Reconstruction by Inversion of Coupled Systems Method on GPUs Using CUDA. In *International Conference of the IEEE Engineering in Biology and Medicine Society (EMBS) 2025*. Copenhagen, Denmark. (Poster)
2. **Zeroual, M.A.**; Dudysheva, N.; Gras, V.; Mauconduit, F.; Isaieva, K.; Vuissoz, P.-A.; Odille, F. Motion-Compensated Reconstruction for brain MRI at 7T: Comparison of Rigid and Non-Rigid Motion Models. In *the International Society of Magnetic Resonance in Medicine (ISMRM) Motion Correction Workshop*. 2024. Quebec, Canada. (Poster)

3. **Zeroual, M.A.**; Isaieva, K.; Vuissoz, P.-A.; Odille, F. Performance Study and Optimization of an MRI motion-compensated Reconstruction Program. In *Proceedings of 32nd Annual Meeting of International Society of Magnetic Resonance in Medicine (ISMRM)*. Singapore. 2024 (Digital Poster)
4. **Zeroual, M.A.**; Isaieva, K.; Vuissoz, P.-A.; Odille, F. High Performance Computing applied to the study of MRI reconstruction codes. In *Proceedings of French Society of Magnetic Resonance in Biology and Medicine (SFRMBM)*. Paris, France. 2023. (Poster)

Other Communications

1. Dudysheva, N.; **Zeroual, M.A.**; Vuissoz, P.-A.; Odille, F.; Mauconduit, F; Massire, A; Gras, V. Mitigating motion for pediatric imaging at 7 Tesla with DISORDER reordering. In SIEMENS IDEA club users meeting, Saint-Malo, France. 2025. (Oral Presentation)

Chapter 1

Background

1.1 Physical Fundamentals of MRI

1.1.1 A brief history of MRI

Magnetic Resonance Imaging (MRI) is based on the phenomenon of Nuclear Magnetic Resonance (NMR), which was first introduced by Isidor I. Rabi in the 1930s [1] and confirmed in 1938 [2]. The physicist Evgenii K. Zavoiskii performed the first experiments on NMR in 1941. However, due to difficulties in achieving sufficient magnetic field homogeneity, his results were not reproducible and remained unrecognized for several years. Then, Felix Bloch and Edward M. Purcell took the demonstration to the next step, by demonstrating independently the NMR phenomenon in liquids and solids in 1946 [3, 4]. These works were the pillars of the NMR spectroscopy. Further research works were required before the translation of the idea to the image formation direction. Herman Y. Carr was the first to report the use of gradient fields to encode the spatial location of magnetization in NMR experiments in 1952 [5]. Using this spatial encoding perspective, Lauterbur created a two-dimensional image, and introduced the notion of image formation used in MRI scanners [6]. Then, Peter Mansfield outlined the mathematical framework of the newly formed imaging methodology [7].

In this chapter, we describe the physical fundamentals of MRI, data acquisition techniques, the reconstruction problem in the MR context, the k -space sampling strategies and the parallel imaging principles. The main aim is to understand the process that links the MR signal to the image formation.

1.1.2 Nuclear Magnetization

MRI signals arise from the nuclear magnetization of spins. Although inherently a quantum mechanical phenomenon, a classical description of magnetization is generally adopted [8]. In this framework, the magnetization of an ensemble of spins within a voxel is represented by a net magnetization vector $\mathbf{M}$. The magnitude of $\mathbf{M}$ is determined by the externally applied static magnetic field $\mathbf{B}$, also referred to as the polarizing magnetic field.

1.1.3 Interactions between Magnetization and Externally Applied Magnetic Fields

In the presence of an externally applied magnetic field

$$\mathbf{B} = [B_x, B_y, B_z]^T,$$

the magnetization vector $\mathbf{M}$ experiences changes that drive it away from equilibrium. The first effect is a rotation of $\mathbf{M}$ around the axis defined by $\mathbf{B}$ (generally the z -axis), a motion known as *precession*. The precession frequency is determined by the *gyromagnetic ratio* γ (MHz/T), which is specific to each nucleus. Hydrogen (^1H) is the primary nucleus studied in MRI due to its high abundance in biological tissues. The precessional motion can be mathematically described by the vector equation:

$$\frac{d\mathbf{M}}{dt} = \gamma \mathbf{M} \times \mathbf{B}. \quad (1.1)$$

Precession generates magnetization components in all three spatial directions. Over time, these components relax back to their equilibrium values. The transverse components M_x and M_y decay to zero with a characteristic time constant T_2 (*transverse relaxation time*), whereas the longitudinal component M_z recovers towards the equilibrium magnetization M_0 with a time constant T_1 (*longitudinal relaxation time*).

1.1.4 The Bloch Equations

The NMR phenomenon is described on the macroscopic level by a set of first-order differential equations named the *Bloch equations*. Given $\mathbf{M}(t) = (M_x(t), M_y(t), M_z(t))$ as the nuclear magnetization and $\mathbf{B}(t) = (B_x(t), B_y(t), B_0 + \Delta B_z(t))$ as the magnetic field

$$\begin{aligned} \frac{dM_x(t)}{dt} &= \gamma(\mathbf{M}(t) \times \mathbf{B}(t))_x - \frac{M_x(t)}{T_2}, \\ \frac{dM_y(t)}{dt} &= \gamma(\mathbf{M}(t) \times \mathbf{B}(t))_y - \frac{M_y(t)}{T_2}, \\ \frac{dM_z(t)}{dt} &= \gamma(\mathbf{M}(t) \times \mathbf{B}(t))_z - \frac{M_z(t) - M_0}{T_1}, \end{aligned} \quad (1.2)$$

where γ is the gyromagnetic ratio, T_2 is the spin-spin (transverse) relaxation constant and T_1 stands for the spin-lattice (longitudinal) relaxation constant. The Bloch equations can also be written using matrix formalism, after expanding the cross product term: $\mathbf{M}(t) \times \mathbf{B}(t) = (M_y B_z - M_z B_y)\hat{\mathbf{x}} - (M_x B_z - M_z B_x)\hat{\mathbf{y}} + (M_x B_y - M_y B_x)\hat{\mathbf{z}}$, which will lead to

$$\frac{d}{dt} \begin{pmatrix} M_x \\ M_y \\ M_z \end{pmatrix} = \begin{pmatrix} -\frac{1}{T_2} & \gamma B_z & -\gamma B_y \\ -\gamma B_z & -\frac{1}{T_2} & \gamma B_x \\ \gamma B_y & -\gamma B_x & -\frac{1}{T_1} \end{pmatrix} \begin{pmatrix} M_x \\ M_y \\ M_z \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \frac{M_0}{T_1} \end{pmatrix}. \quad (1.3)$$

Equation (1.2) can also be expressed using complex notation, $M_+ = M_x + iM_y$, $B_+ = B_x + iB_y$, $B_- = B_+^*$, $M_- = M_+^*$, which leads to

$$\begin{cases} \frac{dM_+(t)}{dt} = -i\gamma(M_+(t)B_z(t) - M_z(t)B_+(t)) - \frac{M_+(t)}{T_2}, \\ \frac{dM_z(t)}{dt} = \frac{i\gamma}{2}(M_+(t)B_-(t) - M_-(t)B_+(t)) - \frac{M_z(t) - M_0}{T_1}, \end{cases} \quad (1.4)$$

Moreover, the Bloch equations can also be simplified by expressing them in a rotating frame of reference. Assume that (x, y, z) is the Cartesian coordinate system of the laboratory frame. $(x', y', z') = (x', y', z)$ is a Cartesian coordinate system that is rotating around the z -axis with an angular frequency of ω_{rf} . The new magnetization reads, using Eq. (1.4) as $M'_z(t) = M_z(t)$ and $M'_x(t) = e^{i\omega_{rf}t}M_+(t)$. In order to find the new equations of motion within the rotating frame of reference, we first compute the derivative on both sides of the equation:

$$\frac{dM'_+(t)}{dt} = e^{i\omega t} \frac{dM_+(t)}{dt} + i\omega e^{i\omega t} M_+(t) = e^{i\omega t} \frac{dM_+(t)}{dt} + i\omega M'_+(t) \quad (1.5)$$

Substituting the second equation of the system (1.4) into the last expression, we get

$$\frac{dM'_+(t)}{dt} = e^{i\omega t} \left(-i\gamma(M_+(t)B_z(t) - M_z(t)B_+(t)) - \frac{M_+(t)}{T_2} \right) + i\omega M'_+(t) \quad (1.6)$$

Since $B_z(t) = B_0 + \Delta B_z(t)$, as previously noted, we can simplify this into a more compact format

$$\frac{dM'_+(t)}{dt} = i(\omega - \omega_0)M'_+(t) - i\gamma\Delta B_z(t)M'_+(t) + i\gamma B'_+(t)M_z(t) - \frac{M'_+(t)}{T_2} \quad (1.7)$$

Hence, if a purely transverse time-varying magnetic field is considered ($\Delta B_z = 0$), the equation reads

$$\frac{dM'_+(t)}{dt} = i(\omega - \omega_0)M'_+(t) + i\gamma B'_+(t)M_z(t) - \frac{M'_+(t)}{T_2} \quad (1.8)$$

Moreover, if the transverse time-varying magnetic field is circularly polarized (i.e. $B_x(t) + iB_y(t) = B_1 \cos(\omega t) - iB_1 \sin(\omega t)$), the above equation leads to

$$\frac{d}{dt} \begin{pmatrix} M'_x \\ M'_y \\ M'_z \end{pmatrix} = \begin{pmatrix} -\frac{1}{T_2} & -\Delta\omega & \gamma B_1 \sin(\omega t) \\ -\gamma B_z & -\frac{1}{T_2} & \gamma B_1 \cos(\omega t) \\ -\gamma B_1 \sin(\omega t) & -\gamma B_1 \cos(\omega t) & -\frac{1}{T_1} \end{pmatrix} \begin{pmatrix} M'_x \\ M'_y \\ M'_z \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \frac{M_0}{T_1} \end{pmatrix} \quad (1.9)$$

which can be seen as the Eq. (1.3) in the rotating frame of reference. The rotating frame of reference is particularly useful when studying the impact of RF pulses on the magnetization.

1.1.5 Theory of Relaxation

1.1.5.1 Longitudinal relaxation

In the recovery to the thermal equilibrium process, the system of spins dissipates energy to the surrounding lattice. The rate of this recovery will depend on two factors: the amount of energy in the spin system, and a sample-specific constant T_1 . The dynamics of longitudinal magnetization, including the recovery process, is described by the third equation of Eq 1.2

$$\frac{dM_z(t)}{dt} = \gamma(\mathbf{M}(t) \times \mathbf{B}(t))_z - \frac{M_z(t) - M_0}{T_1} \quad (1.10)$$

After the application of an RF pulse of a flip angle α the above equation gives the analytical expression of M_z , as (see Figure 1.1)

$$M_z(t) = M_0 \cos(\alpha) + M_0(1 - \cos(\alpha)) \left(1 - e^{-\frac{t}{T_1}}\right). \quad (1.11)$$

Hence, if $\alpha = 90^\circ$, the longitudinal magnetization M_z reads

$$M_z(t) = M_0 \left(1 - e^{-\frac{t}{T_1}}\right). \quad (1.12)$$

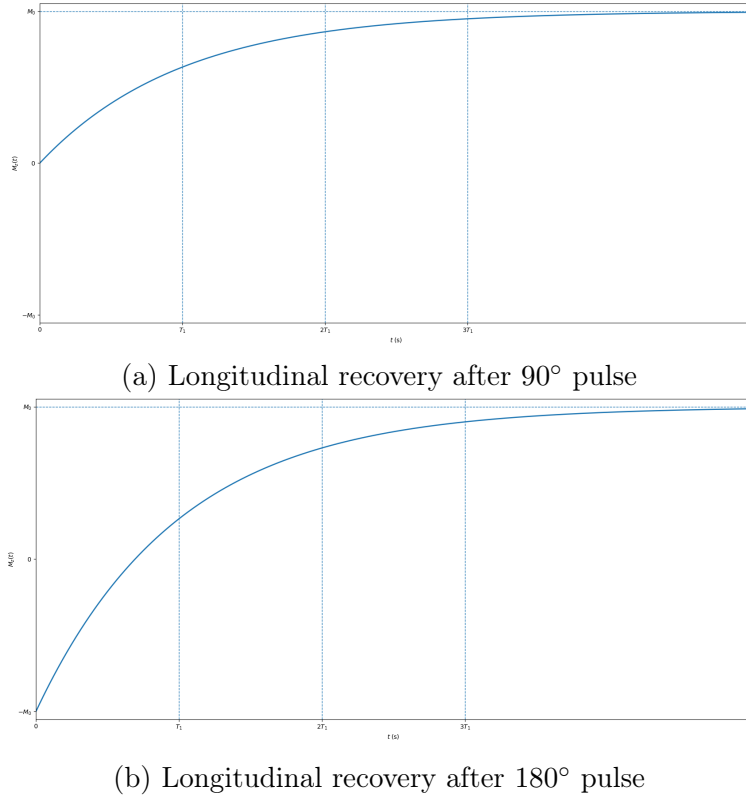


Figure 1.1: Exponential regrowth of the longitudinal magnetization $M_z(t)$ after (a) a 90° pulse and (b) a 180° inversion pulse.

1.1.5.2 Transverse relaxation

The principle of the transverse magnetization decay is different from the T_1 relaxation process. The transverse magnetization decays is caused by losing the *phase coherence* among the magnetic moments. Hence the net magnetization decays over time. This magnetization tends to go back to the thermal equilibrium value of zero with a precise time constant T_2 :

$$\begin{aligned}\frac{dM_x(t)}{dt} &= \gamma(\mathbf{M}(t) \times \mathbf{B}(t))_x - \frac{M_x(t)}{T_2}, \\ \frac{dM_y(t)}{dt} &= \gamma(\mathbf{M}(t) \times \mathbf{B}(t))_y - \frac{M_y(t)}{T_2}\end{aligned}\tag{1.13}$$

The loss of the phase coherence is mainly due to 2 phenomenons : the first one is the "real" T_2 decay, a direct consequence of the spin-spin interaction. The second one is the spatial inhomogeneities and disparities of the magnetic field strength across the sample. These effects create a decline in phase coherence, a phenomenon that is described by a specific relaxation time T_2' (see Figure 1.2). Hence, the effective relaxation time that characterizes the decay of the transverse magnetization is expressed in terms of T_2 (signal loss component) and T_2' . We introduce T_2^* :

$$\frac{1}{T_2^*} = \frac{1}{T_2} + \frac{1}{T_2'}.\tag{1.14}$$

1.1.6 Spatial Encoding and Image Reconstruction

In 1973, Paul Lauterbur introduced a method of NMR imaging, using linear static magnetic field gradients in combination with a projection reconstruction approach [6]. In parallel, considerable advances had already been made by Richard Ernst, who in 1966 introduced Fourier transform spectroscopy in NMR [9] and later, together with Anderson, described in 1974 the general method of 2D Fourier transform NMR [10]. They suggested that NMR experiments could be extended to an additional spatial dimension by repeating acquisitions under different phase encoding. The major contribution in spatial encoding came with the work of Paul Lauterbur and Peter Mansfield, who introduced the use of spatially-varying linear magnetic field gradients [6, 7]. Linear magnetic field gradients enable the Larmor frequency to vary linearly with spatial position, allowing the use of FT techniques to reconstruct images from the acquired signal in k -space. The first MR image of a human thorax was acquired in 1977 by Raymond Damadian using a whole-body scanner; the method relied on a specific method of spatial encoding [11]. This approach, known as *field-focusing*, consisted of acquiring a single voxel at a time by shimming the static magnetic field at a specific point and performing an NMR experiment. To scan an entire object, the subject had to be repositioned point by point, resulting in a total scan time of about five hours for a single slice of the human thorax containing just 106 voxels.

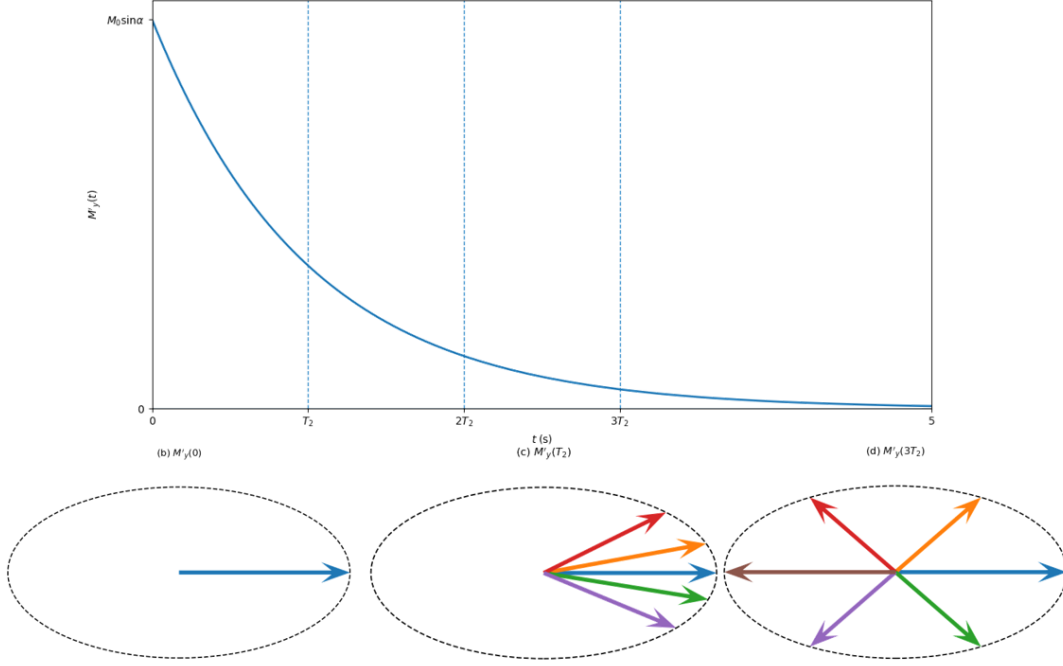


Figure 1.2: Exponential decay of the transverse magnetization due to T_2 relaxation. The blue curve highlights the decay of the transverse component $M_y'(t)$ after an excitation with a flip angle α . (b) Immediately after excitation, the transverse magnetization vectors are in phase, yielding a coherent net vector. The loss of phase coherence and the corresponding net transverse magnetization is illustrated, after one excitation (b), after one (c) or several (d) T_2 periods.

1.1.6.1 Linear Magnetic Field Gradients

In modern MRI systems, dedicated gradient coils generate linear magnetic field gradients described by the vector field:

$$\mathbf{G} = \frac{\partial B_z}{\partial x} \hat{x} + \frac{\partial B_z}{\partial y} \hat{y} + \frac{\partial B_z}{\partial z} \hat{z} \equiv G_x \hat{x} + G_y \hat{y} + G_z \hat{z} \quad (1.15)$$

This produces a magnetic field of the form $\mathbf{B} = (G_x x + G_y y + G_z z + B_0) \hat{z}$, where the angular frequency of precession is given by $\omega = \gamma(B_0 + \mathbf{G} \cdot \mathbf{r})$. In practice, the x and y components of the magnetic field are generally nonzero ($B_x \neq 0, B_y \neq 0$), resulting in *concomitant magnetic fields*, as dictated by Gauss's law for magnetism ($\nabla \cdot \mathbf{B} = 0$). These concomitant magnetic fields can be interpreted as additional, non-linear field components along the main field direction $\hat{z}$. Minimizing the impact of concomitant fields is critical in certain MRI pulse sequences and techniques to reduce image artifacts. Gradient fields are used to apply precise temporal variations to the magnetization, enabling spatial encoding and signal localization. The accumulated phase of the spins at time t is given by

$$\phi(t) = \omega(x, y, z) \cdot t = \gamma B_0 t + \gamma \left[\int_0^t \mathbf{G}(\tau) d\tau \right] \cdot \mathbf{r} = \omega_0 t + \mathbf{k}(t) \cdot \mathbf{r} \quad (1.16)$$

where $\mathbf{k}(t) = \gamma \int_0^t \mathbf{G}(\tau) d\tau$ defines the trajectory in k -space. In MRI, gradients serve multiple functions, including slice selection, spatial encoding, frequency encoding (readout gradients), and phase encoding. They can be used for some specialized tasks such as spoiling (eliminating unwanted transverse magnetization).

1.1.6.2 Slice Detection

In conventional MRI, RF pulses are either selective or non-selective. Selective RF pulses are typically played simultaneously with a slice-selection gradient, allowing for spatial localization of the excitation. Non-selective pulses, often used in 3D pulse sequences, do not employ a selection gradient and excite the entire sample volume. Selective pulses, by comparison, are commonly used in 2D imaging to excite specific slices. At low flip angles, the resulting excitation profile closely matches the FT of the RF pulse envelope. This holds true when T_2 relaxation during excitation is neglected. Adopting the rotating frame of reference as the default, the equation governing the evolution of the transverse magnetization component $M_+(t)$ can be written as:

$$\frac{dM_+(t)}{dt} = i\Delta\omega M_+(t) + i\gamma B_1(t)M_z(t) \quad (1.17)$$

Given the initial condition $M_+(0) = 0$, Eq 1.17 admits the following solution:

$$M_+(t) = i\gamma e^{-i\Delta\omega t} \int_0^t M_z(\tau) B_1(\tau) e^{i\Delta\omega\tau} d\tau \approx i\gamma e^{-i\Delta\omega t} M_0 \int_0^t B_1(\tau) e^{i\Delta\omega\tau} d\tau \quad (1.18)$$

where we have approximated $M_z(t) \approx M_0$ (valid for small flip angles). Taking the magnitude yields:

$$|M_+(t)| \approx \gamma M_0 \left| \int_0^t B_1(\tau) e^{i\Delta\omega\tau} d\tau \right| \quad (1.19)$$

and the flip angle α is related by:

$$\sin(\alpha) = \pm \frac{|M_+(t)|}{M_0} \approx \gamma \left| \int_0^t B_1(\tau) e^{i\Delta\omega\tau} d\tau \right| \approx \alpha(\Delta\omega) \approx \pm \gamma \left| \int_0^t B_1(\tau) e^{i\Delta\omega\tau} d\tau \right| \quad (1.20)$$

For small flip angles, $\sin(\alpha) \approx \alpha$, $M_y(t) = M_0 \sin(\alpha)$, the slice profile is then determined using the Fourier transform of the RF pulse. Since the Fourier transform of a rectangular pulse is a sinc function, sinc-shaped pulses are commonly employed to achieve well-defined selective excitation profiles. Conversely, non-selective (“hard”) pulses are designed to uniformly excite a broad frequency range, but at the cost of large bandwidth requirements and shorter durations τ_p . Block-shaped non-selective pulses are typically used in non-selective excitation, whereas sinc-shaped pulses are favored for slice-selective excitation due to their well-defined spatial selectivity.

1.1.6.3 Frequency Encoding

The application of a magnetic field gradient introduces a controlled spatial variation in the magnetic field strength along a specific axis, conventionally the x -axis. As a result, the

Larmor frequency of the nuclear spins becomes a linear function of their spatial position along this axis. This frequency variation causes spins located at different x -positions to precess at different rates. By exploiting this spatial-frequency relationship, one can localize the origin of the NMR signal along the x -direction—a process known as frequency encoding. The gradient applied during this step is referred to as the readout gradient. At the isocenter of the gradient coil ($x = 0$), the contribution of the gradient field is null, and only the static main field $\mathbf{B}_0$ determines the spin dynamics. Data acquisition occurs during the application of this readout gradient, allowing spatial encoding of the signal along the x -axis.

1.1.6.4 Phase Encoding

In addition to slice-selection and frequency encoding, a third gradient—applied orthogonally to both—is introduced by convention along the y -axis to achieve spatial encoding in the phase-encoding direction. This gradient momentarily alters the magnetic field across the y -axis, causing spins in different positions to experience shifts in their precession frequencies. Once the gradient is turned off, the spins tend to the original Larmor frequencies. However, they retain a spatially-dependent phase shift, which is directly proportional to the amplitude and duration of the applied gradient. This encoding is referred to as the phase encoding. Unlike frequency encoding, signal acquisition does not occur during the application of the phase-encoding gradient.

1.1.6.5 The Construction of k -space

The spatial encoding gradients described earlier are used to construct a raw data matrix in the k -space. The gradients $G_x(t)$ and $G_y(t)$ enable the acquisition of signal components along the k_x and k_y axes, respectively. The corresponding k -space coordinates are defined as:

$$k_x = \frac{\gamma}{2\pi} \int_0^{t_x} G_x(t) dt \quad (1.21)$$

$$k_y = \frac{\gamma}{2\pi} \int_0^{t_y} G_y(t) dt \quad (1.22)$$

where t_x and t_y denote the durations over which the readout and phase encoding gradients are applied. Using these definitions, the accumulated phase of a spin located at position (x, y) relative to the center of the gradient field (isocenter) can be expressed as:

$$\phi(x, y) = 2\pi(xk_x + yk_y) = \gamma \left(\int_0^{t_x} G_x(t) dt + \int_0^{t_y} G_y(t) dt \right) \quad (1.23)$$

To fully sample the k -space domain, the encoding process must sequentially acquire data at multiple (k_x, k_y) coordinates. This is typically performed using a Cartesian sampling scheme. In this framework, the gradients are incrementally adjusted to traverse the k -space grid row by row. In order to meet the Nyquist-Shannon sampling criterion and avoid aliasing, the sampling intervals in k -space, denoted Δk_x and Δk_y , must be inversely

proportional to the respective fields of view (FOV) along the x and y directions:

$$\Delta k_x = \frac{1}{FOV_x} \quad (1.24)$$

$$\Delta k_y = \frac{1}{FOV_y} \quad (1.25)$$

The acquisition protocol typically begins with the simultaneous application of both phase and frequency encoding gradients, resulting in the sampling of one line in k -space. In subsequent repetitions, the frequency encoding gradient remains unchanged, while the amplitude of the phase encoding gradient is varied to access different k_y positions. This process is iterated until the entire k -space grid has been covered. Notably, signal acquisition occurs during the application of the frequency encoding gradient, following the deactivation of the phase encoding gradient.

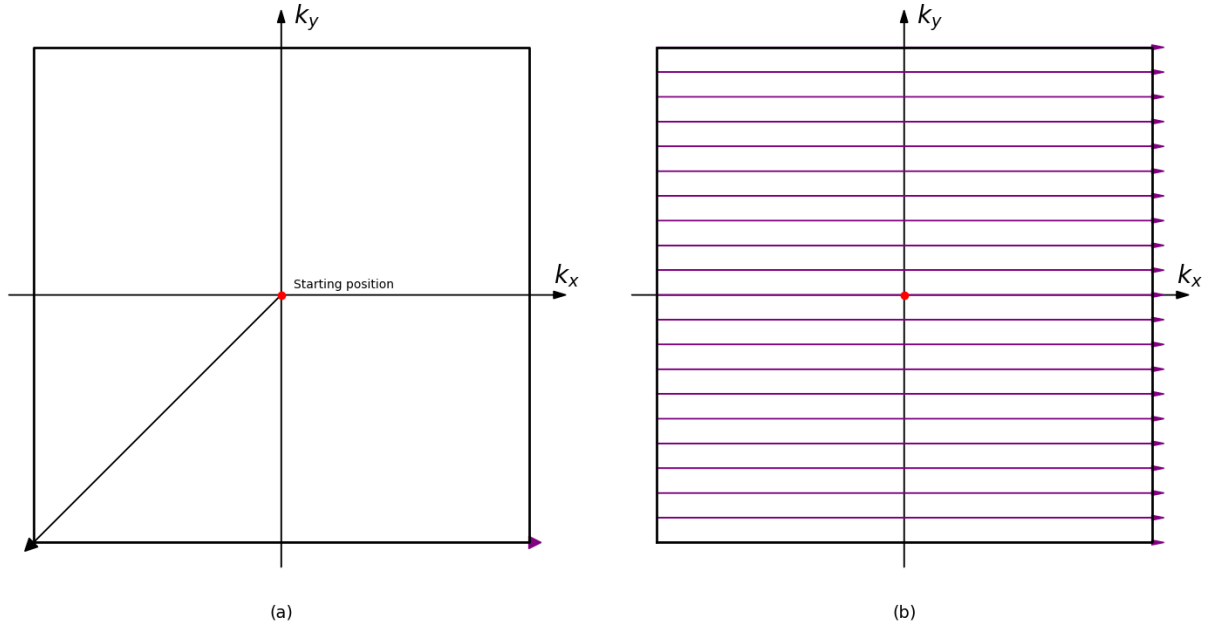


Figure 1.3: Illustration of a Cartesian k -space acquisition strategy. In the first step (a), both phase and frequency gradients are applied to shift the encoding origin from the center of k -space (red dot) to its starting corner. Once the desired k -space position is reached, the frequency encoding (readout) gradient is activated to acquire the first line of data. In the next repetition, a different combination of phase encoding and frequency gradients is applied to target the next line in k -space. This sequence of operations is iteratively repeated, line by line, until the entire k -space is covered (b).

1.1.6.6 Signal Formation and Detection

Once the net magnetization of proton spins in a sample has been tipped into the transverse (xy -) plane by RF excitation, Faraday's Law predicts that placing a receiver coil near the

sample will induce a time-varying magnetic flux within the coil. The induced flux is given by

$$\Phi(t) = \oint_V \mathbf{B}_1^r(\mathbf{r}) \cdot \mathbf{M}(\mathbf{r}, t) d\mathbf{r} \quad (1.26)$$

where $\mathbf{B}_1^r$ represents the spatial sensitivity profile of the receive coil, and $\mathbf{M}$ is the local magnetization. The detected signal is proportional to the electromotive force (EMF) generated in the coil, which by Faraday's law corresponds to the time derivative of the magnetic flux:

$$V(t) = -\frac{\partial \Phi(t)}{\partial t} = -\frac{\partial}{\partial t} \oint_V \mathbf{B}_1^r(\mathbf{r}) \cdot \mathbf{M}(\mathbf{r}, t) d\mathbf{r} \quad (1.27)$$

$$= -\int \left[B_{1x}^r(\mathbf{r}) \frac{\partial M_x(\mathbf{r}, t)}{\partial t} + B_{1y}^r(\mathbf{r}) \frac{\partial M_y(\mathbf{r}, t)}{\partial t} \right] d\mathbf{r} \quad (1.28)$$

Here, the z -component is neglected, as it changes much more slowly than the transverse components and contributes negligibly to the detected signal. By employing complex notation, the detected voltage can be rewritten as

$$V(t) = -\int \text{Re} \left[B_{1,+}^r(\mathbf{r}) \frac{\partial M_-(\mathbf{r}, t)}{\partial t} \right] d\mathbf{r} \quad (1.29)$$

where M_- and $B_{1,+}^r$ are defined by

$$M_-(\mathbf{r}, t) = |M_+(\mathbf{r}, 0)| e^{-i[\omega(\mathbf{r})t - \phi_r(\mathbf{r})]} e^{-t/T_2(\mathbf{r})}, \quad B_{1,+}^r(\mathbf{r}) = |B_{1,+}^r(\mathbf{r})| e^{i\phi_r(\mathbf{r})} \quad (1.30)$$

with $\phi_r(\mathbf{r})$ being the reception phase and $\omega(\mathbf{r})$ the local Larmor frequency (including any phase offset from RF excitation). Taking the time derivative yields

$$\frac{\partial M_-(\mathbf{r}, t)}{\partial t} = |M_+(\mathbf{r}, 0)| e^{-i[\omega(\mathbf{r})t - \phi_r(\mathbf{r})]} e^{-t/T_2(\mathbf{r})} \left(-\frac{1}{T_2(\mathbf{r})} - i\omega(\mathbf{r}) \right) \quad (1.31)$$

For typical MRI conditions where $\omega(\mathbf{r}) \gg 1/T_2(\mathbf{r})$, the relaxation term is negligible compared to the precessional frequency, and we can approximate:

$$\frac{\partial M_-(\mathbf{r}, t)}{\partial t} \approx -i\omega(\mathbf{r}) |M_+(\mathbf{r}, 0)| e^{-i[\omega(\mathbf{r})t - \phi_r(\mathbf{r})]} e^{-t/T_2(\mathbf{r})} \quad (1.32)$$

Substituting back, the detected voltage simplifies to

$$V(t) = \omega_0 \int |B_{1,+}^r(\mathbf{r})| |M_+(\mathbf{r}, 0)| e^{-t/T_2(\mathbf{r})} \sin[\omega(\mathbf{r})t - \phi_r(\mathbf{r}) + \phi_r(\mathbf{r})] d\mathbf{r} \quad (1.33)$$

or, equivalently,

$$V(t) = \omega_0 \int B_{1,+}^r(\mathbf{r})^* M_-(\mathbf{r}, 0) e^{-t/T_2(\mathbf{r})} e^{-i\omega(\mathbf{r})t} d\mathbf{r} \quad (1.34)$$

Detecting $V(t)$ in practice involves two primary challenges. First, the detected signal oscillates at the Larmor frequency, which is very high ($\omega(\mathbf{r}) = \omega_0 + \Delta\omega(\mathbf{r})$), necessitating high-speed electronics. Second, the direction of rotation of the transverse magnetization (clockwise or counterclockwise) is not readily distinguishable due to phase ambiguities.

To address these challenges, signal demodulation and quadrature detection are applied. Demodulation removes the high-frequency component, while quadrature detection splits the signal into two orthogonal branches. One branch is multiplied by $2 \cos(\omega_0 t)$, and the other by $2 \sin(\omega_0 t)$, introducing a 90° phase shift between the two channels. After mixing, low-pass filtering, and recombination, the result is a complex signal $S(t) = S_R(t) + iS_I(t)$. The final demodulated and quadrature-detected MR signal can thus be written as

$$S(t) = \omega_0 \int |B_{1,+}^r(\mathbf{r})| |M_+(\mathbf{r}, 0)| e^{-t/T_2(\mathbf{r})} e^{-i[\Delta\omega(\mathbf{r})t - \phi_r(\mathbf{r}) + \phi_r(\mathbf{r}) - \frac{\pi}{2}]} d\mathbf{r} \quad (1.35)$$

or, more concisely,

$$S(t) = \omega_0 e^{i\pi/2} \int B_{1,+}^r(\mathbf{r})^* M_-(\mathbf{r}, 0) e^{-t/T_2(\mathbf{r})} e^{-i\Delta\omega(\mathbf{r})t} d\mathbf{r} \quad (1.36)$$

1.1.6.7 Image Reconstruction

Since the frequency variation $\Delta\omega(\mathbf{r})$ is generated by the gradient (G_x, G_y, G_z) . The net phase accumulation equation leads to:

$$S(t) \propto \int [B_{1,+}^-(\mathbf{r})^* M_-(\mathbf{r}, 0) e^{-t/T_2(\mathbf{r})}] e^{-i\mathbf{k}(t) \cdot \mathbf{r}} d\mathbf{r} \quad (1.37)$$

The above equation is typically a Fourier transform. Hence, and by computing the IFT of the signal after a sampling on a Cartesian grid, the image intensity I reads:

$$I(\mathbf{r}) = \int e^{i\mathbf{k} \cdot \mathbf{r}} \left[\int B_{1,+}^-(\mathbf{r})^* M_-(\mathbf{r}, 0) e^{-t/T_2(\mathbf{r})} e^{-i\mathbf{k}(t) \cdot \mathbf{r}} d\mathbf{r} \right] d\mathbf{k} \quad (1.38)$$

Eq (1.37) describes the MR signal acquisition process by expressing the detected signal S as a spatial integral of the transverse magnetization weighted by receive sensitivity, relaxation terms, and the spatial phase modulation induced by the gradient fields. Each acquired data sample corresponds to a Fourier component of the object's magnetization distribution at a specific location in k -space. Eq (1.37) and (1.38) form the fundamental mathematical formulation of the MR reconstruction problem. The first defines the encoding of spatial information into measurable frequency-domain data, while the second describes the inversion of this encoding to recover the image. Image reconstruction will be discussed in detail in Chapter 2.

1.1.7 Ultra-High-Field MRI

Ultra-high-field (UHF) MRI, typically defined as static magnetic fields of $B_0 \geq 7$ T, has emerged as a transformative technology for neuroimaging and other biomedical applications. While 7T systems have become the standard for human UHF-MRI, recent advances have extended the frontier to even higher field strengths, such as 9.4 T, which are now operational in a number of research facilities [12], 10.5T [13], or even 11.7T [14]. The primary motivation for moving to higher B_0 lies in the intrinsic gain in SNR. Under idealized conditions, SNR is expected to scale at least linearly with field strength,

$\text{SNR} \propto B_0^1$ [15, 16]. However, recent experimental work using standardized acquisition protocols and state-of-the-art, tight-fitting multi-channel receive arrays has demonstrated a *supralinear* SNR dependence, with $\text{SNR} \propto B_0^{1.65}$ across the cerebrum between 3 T and 9.4 T [12]. Quantitatively, Pohmann *et al.* reported an SNR gain of 3.10 ± 0.20 from 3 T to 7 T, and a further gain of 1.76 ± 0.13 from 7 T to 9.4 T. This substantial SNR advantage can be leveraged to achieve higher spatial resolution, improved tissue contrast, or shorter acquisition times, thereby enhancing sensitivity to subtle anatomical and functional features. Nevertheless, translating these theoretical gains into routine practice is non-trivial due to several UHF-specific challenges:

1. **RF Field (B_1) Inhomogeneity.** At higher Larmor frequencies, the RF wavelength in tissue becomes comparable to, or smaller than, head dimensions, leading to destructive interference and transmit inefficiency. Pohmann *et al.* [12] reported that the relative standard deviation of the B_1 field in the cerebrum doubled from 0.11 ± 0.01 at 3T to 0.22 ± 0.003 at 7T, and further increased to 0.25 ± 0.006 at 9.4T. Therefore, advanced RF shimming, parallel transmit arrays, and tailored pulse designs are required to mitigate these effects.
2. **Relaxation Time Modulation.** The longitudinal relaxation time T_1 of white matter increases with B_0 (947.4 ± 66.1 ms at 3T; 1334.2 ± 21.7 ms at 7T; 1426.7 ± 51.8 ms at 9.4T), whereas the apparent transverse relaxation time T_2^* decreases substantially (e.g., in gray matter: 59.9 ± 3.0 ms at 3T vs. 23.8 ± 1.0 ms at 9.4T). These trends require field-dependent optimization of echo times, flip angles, and contrast mechanisms.
3. **Parallel Imaging Performance.** Although theory predicts improved g-factors at higher fields due to increased coil element sensitivity localization, experimental results in [12] showed no systematic improvement from 3 T to 9.4 T. This highlights that coil geometry, element placement, and inter-element coupling often dominate parallel imaging efficiency.
4. **Engineering and Safety Constraints.** RF coil design at UHF must reconcile efficient transmit field generation with specific absorption rate (SAR) limits. Multi-element transmit arrays, SAR-optimized pulse sequences, and hardware capable of delivering high peak power are essential for safe and effective operation.

Several contrast mechanisms, such as susceptibility-weighted imaging and quantitative phase imaging, may inherently benefit from UHF-specific properties (e.g., shorter T_2^*), whereas spin-echo-based contrasts often require modification to compensate for signal losses. The substantial SNR advantage of UHF-MRI can only be fully exploited through integrated advances in hardware engineering, pulse sequence design, and advanced reconstruction algorithms.

1.1.8 Pulse Sequences

1.1.8.1 Spin Echo

The discovery of the spin echo phenomenon by Erwin Hahn in 1949 [17] marked a milestone in the history of NMR, enabling its application in imaging. A spin echo is generated through the application of two RF pulses, typically a 90° excitation pulse followed by a 180° refocusing pulse. Following the initial 90° pulse, the transverse magnetization vectors begin to dephase due to magnetic field inhomogeneities, leading to the decay of the free induction decay (FID) signal. At this stage, the 180° refocusing pulse is applied, inverting the spin vectors. This causes the spins to rephase, with the previously slower precessing spins now moving ahead of the faster ones, thereby producing a refocused signal known as the Spin Echo. The maximum signal is observed at the **echo time (TE)**, which corresponds to the center of the echo. Figure 1.4 illustrates this technique using a pulse sequence diagram. Although the signal subsequently decays again, additional 180° refocusing pulses can be applied to generate multiple echoes—an approach discussed in a later section.

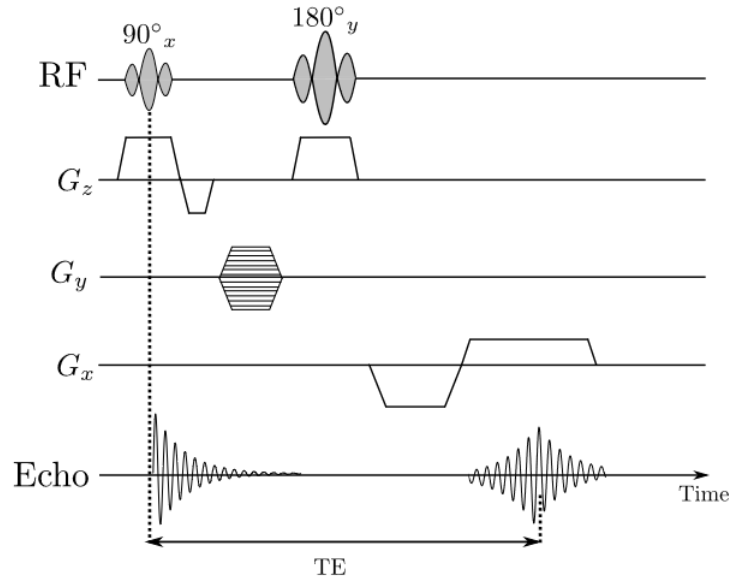


Figure 1.4: A basic spin echo sequence begins with an initial RF excitation pulse, followed by signal decay due to dephasing. Subsequently, a 180° refocusing pulse is applied, which reverses the dephasing effects. As a result, a spin echo forms and reaches its maximum amplitude at the echo time (TE).

1.1.8.2 Gradient Echo

The natural decay of the free induction decay (FID) signal, imposes a limited time window for spatial encoding. To recover usable signal within this timeframe, a gradient of opposite polarity can be applied to rephase the spins and regenerate the signal. In gradient echo acquisition, the sequence begins with the application of a strong, negatively polarized

gradient—referred to as the dephasing gradient. This gradient accelerates the dephasing of spins by inducing a broader spread in their resonant frequencies, leading to a faster decay of the FID. Immediately following this, a rephasing gradient of positive polarity is applied. This rephaser compensates for the earlier dephasing, progressively re-aligning the spins. When full rephasing occurs, a detectable signal termed the gradient echo is produced. Notably, the rephasing gradient is designed to have half the zero-order moment of the dephasing gradient, such that maximum signal amplitude is achieved when the areas (time $\times$ amplitude) under the two gradients are equal.

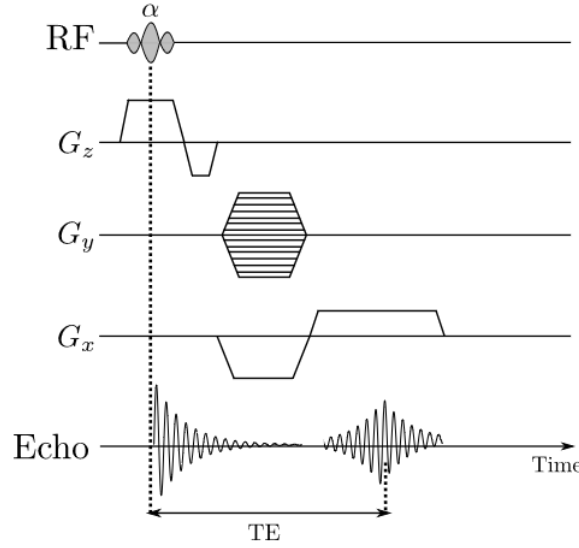


Figure 1.5: A basic gradient echo sequence begins with the application of a low flip angle RF pulse, denoted α , typically less than 90° in fast imaging protocols such as the FLASH (Fast Low Angle Shot) sequence. Following this excitation, a dephasing readout gradient G_x is applied to induce initial dephasing of the transverse magnetization. This is immediately followed by a rephasing gradient of opposite polarity to refocus the spins.

1.1.8.3 MPRAGE and MP2RAGE sequences

3D Magnetization-Prepared Rapid Gradient Echo (MPRAGE) is one of the most widely used MRI sequences for high-resolution anatomical imaging of the brain. First introduced by Mugler and Brookeman in 1990 [18], it combines an inversion recovery preparation with a segmented 3D gradient-echo readout, enabling strong T_1 -weighting, high SNR, and efficient volumetric coverage. An evolution of this technique, the Magnetization-Prepared 2 Rapid Gradient Echo (MP2RAGE) sequence, was proposed by Marques et al. in 2010 [19] to address the limitations of MPRAGE at high magnetic fields, particularly the pronounced sensitivity to B_1^+ inhomogeneities and bias field effects. MP2RAGE acquires two gradient-echo volumes at different inversion times (TI_1 and TI_2) within the same repetition time (TR). These are combined to generate a uniform image with reduced sensitivity to reception profile inhomogeneities.

In MPRAGE, Figure 1.6, an initial 180° inversion pulse is followed by a delay time (inversion time, TI) to allow partial recovery of the longitudinal magnetization, which optimizes tissue contrast. The subsequent gradient-echo readout train is segmented and fills k -space over multiple inversion cycles. The TI is typically chosen to null the cerebrospinal fluid (CSF) signal at 3 T ($TI \approx 1100\text{--}1300$ ms), maximizing gray–white matter contrast.

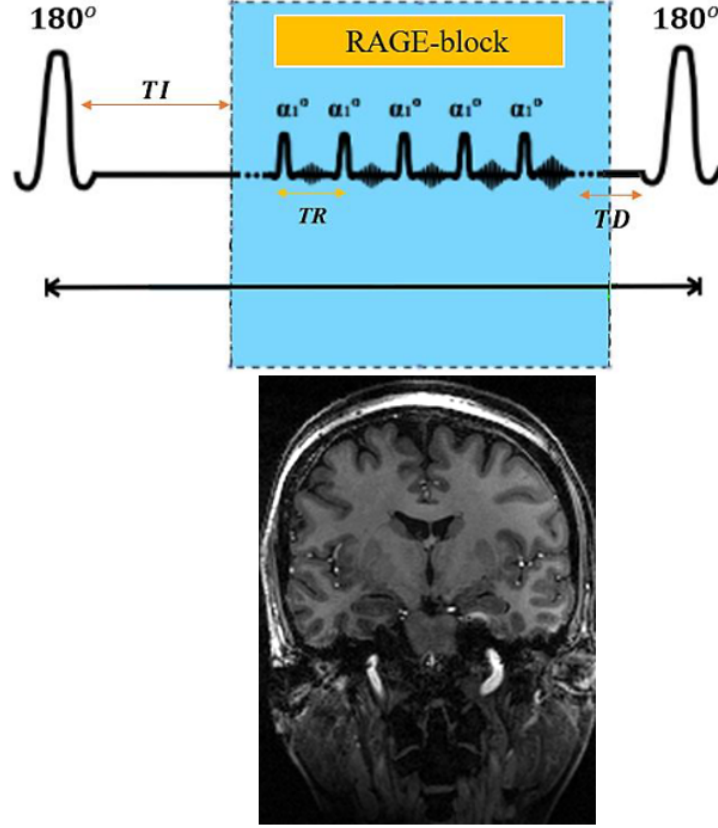


Figure 1.6: Schematic illustration of the MPRAGE sequence, where TI denotes the inversion time : the delay between the center of the non-selective 180° inversion pulse and the acquisition of the central k -space line within the RAGE readout train. The repetition time between successive excitations is indicated by TR , while α_1 represents the excitation flip angle applied during readout. TD corresponds to the relaxation delay between the end of a RAGE readout train and the onset of the subsequent inversion, completing one full MPRAGE repetition time.

In MP2RAGE, Figure 1.7, the readout is split into two gradient-echo acquisitions at different inversion times. The first acquisition (TI_1) is performed when the magnetization is still largely negative (early recovery), while the second (TI_2) occurs after further recovery toward equilibrium. The two datasets are combined according to the following equation:

$$S_{\text{UNI}} = \frac{\text{Re}(S_{\text{TI1}} \cdot S_{\text{TI2}}^*)}{|S_{\text{TI1}}|^2 + |S_{\text{TI2}}|^2} \quad (1.39)$$

yielding a *uniform* image largely free from B_1 inhomogeneity and proton density bias [19].

MPRAGE is the reference anatomical sequence in neuroimaging, extensively used in structural morphometry, cortical thickness analysis, and surgical planning. MP2RAGE provides improved segmentation performance, quantitative T_1 estimation, and reduced bias-field artifacts, which is more suitable for high-field imaging (7T) and studies requiring reproducible tissue parameter quantification [20, 21] (Table 1.1).

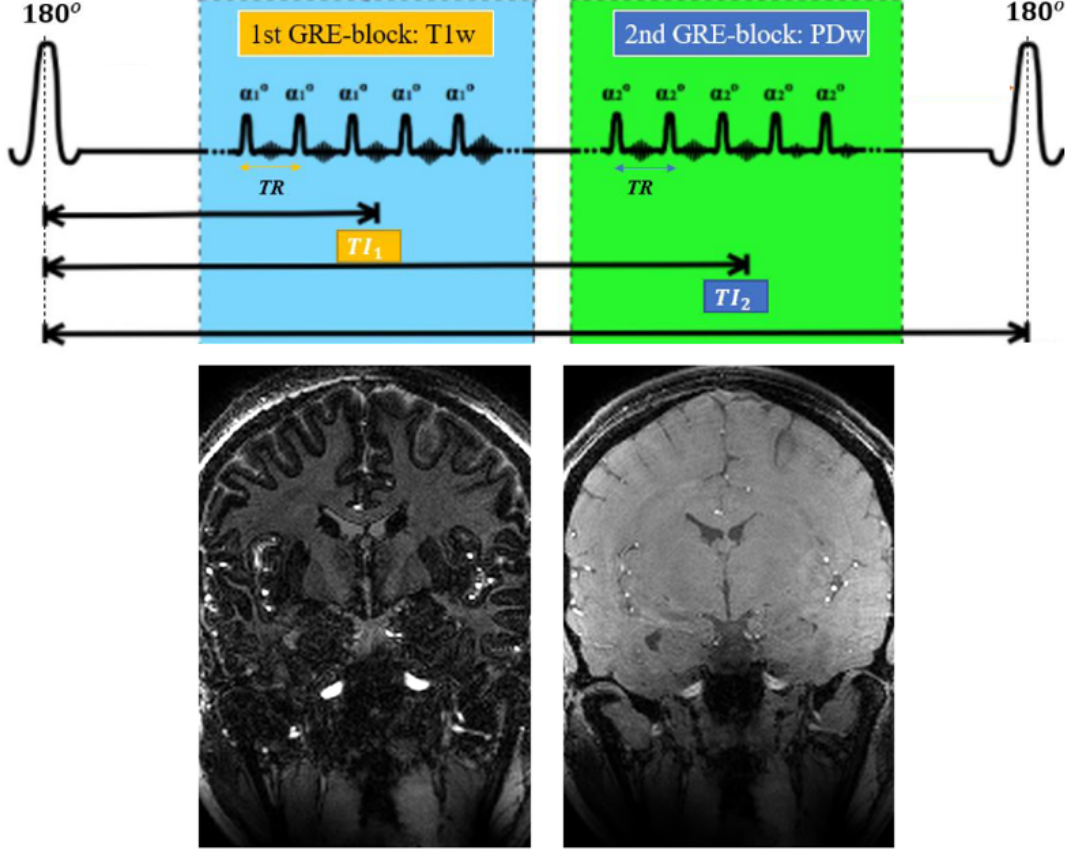


Figure 1.7: Schematic illustration of the MP2RAGE sequence. The inversion times TI_1 and TI_2 are defined as the delays from the midpoint of the inversion pulse to the acquisition of the central k -space line in the respective readout blocks. The flip angles α_1 and α_2 correspond to the excitation angles used in the first and second readout modules, respectively.

Table 1.1: Comparison between MPRAGE and MP2RAGE sequences.

Feature	MPRAGE	MP2RAGE
Number of images per TR	1	2 (TI_1 , TI_2)
Bias-field resilience	Poor	Excellent
Quantitative T_1 mapping	No	Yes
Acquisition time	Shorter	Slightly longer

1.1.9 Fourier Space Sampling Trajectories

Up to this point, we have discussed pulse sequences that sample k -space along rectilinear trajectories using frequency-encoding and phase-encoding gradients. Such sampling schemes are commonly referred to as Cartesian sampling, in contrast to non-Cartesian MRI techniques. Cartesian MRI allows for various profile orders, particularly in 3D pulse sequences, where the second phase-encoding (slice-encoding) dimension can be traversed using a wide range of ordering schemes—including spiral, radial, or elliptical patterns. Furthermore, 3D Cartesian MRI enables the omission of phase-encoding lines at the corners of k -space—an approach known as an elliptical shutter—to reduce scan time without compromising image resolution or SNR. In this method, the skipped k -space corners are simply zero-filled prior to performing the 3D FT.

Another subtype of Cartesian sampling is Echo-Planar Imaging (EPI). EPI distinguishes itself from conventional Cartesian trajectories by sampling multiple lines of k -space within a single repetition time (TR). During EPI acquisition, the frequency-encoding gradient is rapidly alternated—reversing direction from one line to the next—to minimize dead time and achieve ultra-fast imaging. EPI sequences can be implemented as single-shot, where the entire k -space is acquired after a single RF excitation, or as multi-shot, where k -space is sampled over several excitations and TRs. The EPI factor refers to the number of k -space lines acquired per TR. EPI is often described as the gradient-echo counterpart to turbo spin-echo (TSE), and the two can be combined to form a hybrid sequence known as GRAdient And Spin Echo (GRASE). As with TSE, both EPI and GRASE are frequently paired with additional modules to enable rapid imaging, such as inversion recovery EPI (IR-EPI).

The two most widely used non-Cartesian sampling trajectories in MRI are spiral and radial sampling. Similar to EPI, spiral trajectories can be implemented in either single-shot or multi-shot modes. In spiral MRI, off-resonance effects manifest mainly as image blurring—rather than geometric distortions, because the point-spread function is isotropic. Non-Cartesian MRI acquisitions require that the collected k -space data be interpolated, or "gridded," onto a Cartesian grid before performing the FT. Alternatively, a non-uniform fast Fourier transform (NUFFT) [22] algorithm can be used to directly reconstruct images from non-Cartesian data.

In contrast to spiral sampling, radial sampling is inherently a multi-shot technique, typically acquiring one spoke (from $-k_{\max}$ to $+k_{\max}$) per T_R . Multi-echo radial MRI is also feasible by repeatedly sampling the same spoke at different echo times. Both spiral and, especially, radial trajectories allow for extremely short echo times, since they follow a center-out k -space trajectory. A particular subclass of radial trajectories is ultra-short echo time (UTE) MRI. A 3D UTE radial sequence often includes a first half-echo (from $k = 0$ to $k = +k_{\max}$), followed by a second full echo (from $k = +k_{\max}$ to $k = -k_{\max}$). Unlike standard radial approaches, UTE sampling begins while the readout gradient is still ramping up, which minimizes the first TE.

In 3D UTE sequences, the radial spokes are distributed isotropically throughout a 3D sphere in k -space, forming what is commonly known as a kooshball trajectory. This trajectory is particularly robust against motion, respiratory, and undersampling artifacts. Notably, kooshball trajectories support aggressive undersampling and can be combined

with partial k -space reconstruction techniques in order to accelerate image acquisition. Figure 1.8 describes the discussed sampling patterns.

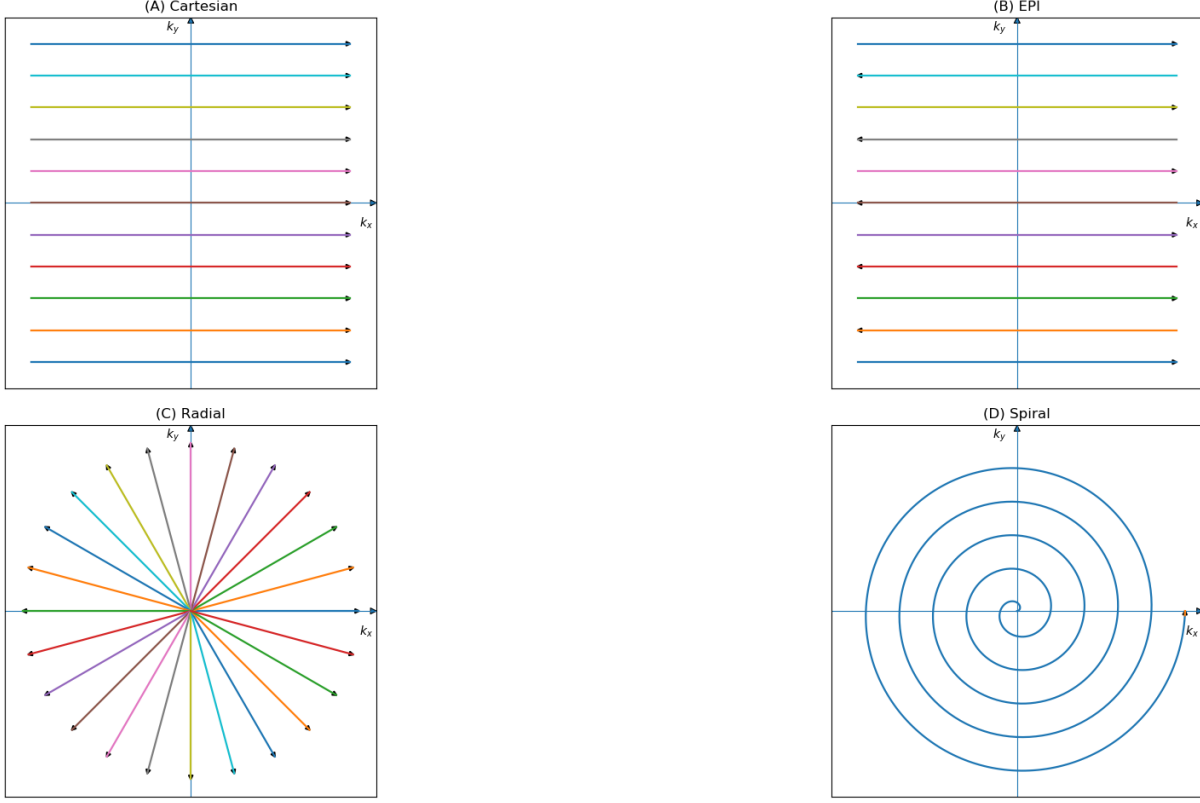


Figure 1.8: Examples of 2D k -space trajectories used for spatial encoding: (A) Cartesian, (B) Echo-Planar Imaging (EPI), (C) Radial and (D) spiral.

1.2 Parallel Imaging

Parallel Imaging is an image reconstruction technique that uses the spatial information extracted from an array of receiver coils to compensate for aliasing artifacts. Parallel imaging principles rely typically on the undersampled k -space data, while aliasing artifacts are resolved with the sensitivity profiles of the receiver coils. The MR acquisition is a relatively long process. Parallel Imaging has been proposed to address this limitation; it aims at increasing the speed of the MRI exam without compromising the spatial resolution. In addition to patient comfort, the motion-related issues longer scans may degrade image quality. A typical MRI clinical protocol takes between 30 minutes and one hour, hence Parallel Imaging offers an alternative to remain on the clinical requirements, by enabling a considerable reduction of the acquisition time. Usually, when talking about Parallel Imaging, four main methods are cited: Sensitivity Encoding (SENSE) [23], Generalized Auto-calibrating Partially Parallel Acquisition (GRAPPA) [24], iTerative Self-consistent Parallel Imaging Reconstruction (SPIRiT) [25], and Eigenvalue SPIRiT (ESPIRiT) [26]. In MRI, data collection is done on Fourier space using an array of receiver coils in order to

increase the SNR. Taking c_1 as the very first coil-element of the array, c_1 is only sensitive to the part of the body close to it. Moreover, and as a consequence of the local sensitivity patterns of each coil-element, the spatial location of signals can also be determined; the array coils provide localized coil information. In Parallel Imaging, aliasing artifacts are resolved using this information.

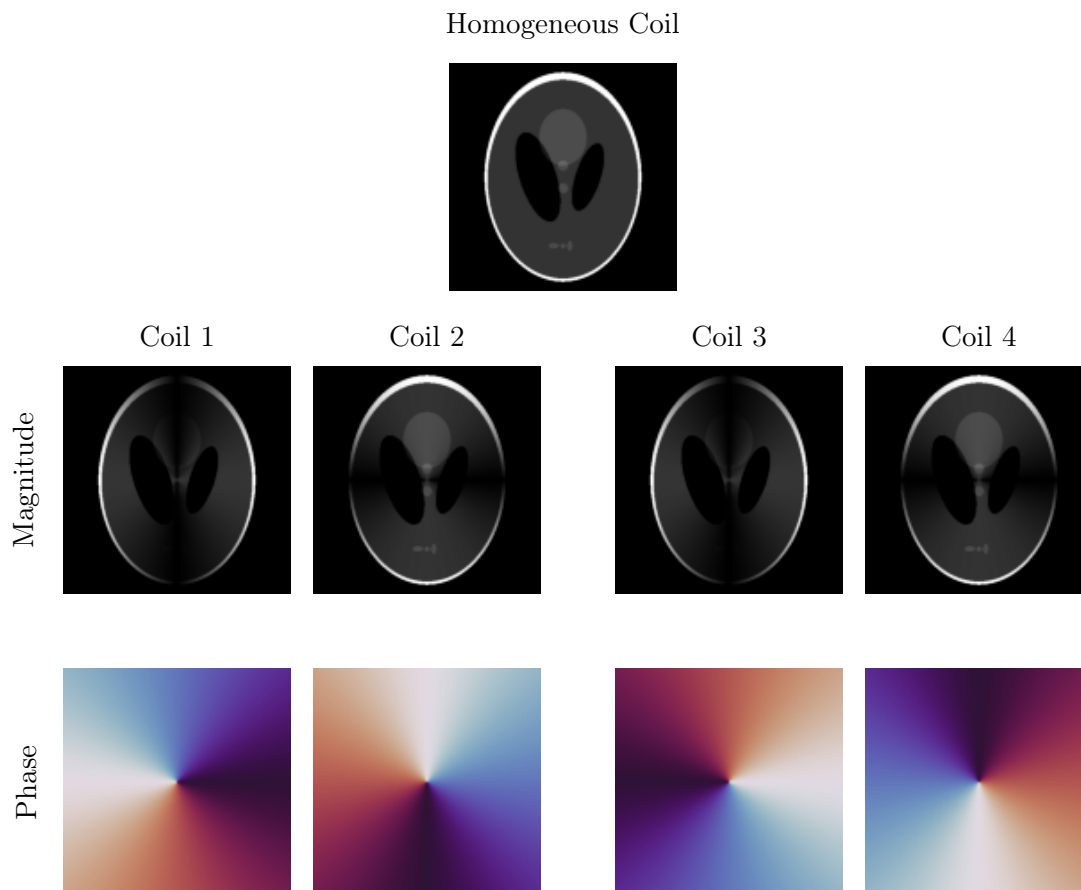


Figure 1.9: A phantom image collected with various receiver coils **Top:** Reference phantom image. **Middle and bottom:** Magnitude and phase images as seen by four different receiver coils of an array. Each coil’s unique spatial sensitivity (both in magnitude and phase) enables the reconstruction of undersampled data in Parallel Imaging. When acquiring images with the considered coils, an additional element of spatial localization is added to complement the gradient-based encoding, which is the localized sensitivity information. In magnitude, we can see that areas next to the coil in question appears brighter than those who are not.

A pillar concept in this context is the sensitivity profile of a given receiver coil. The sensitivity profile can be expressed using the *coil sensitivity maps*. The latter are complex-valued and depend on several factors, mainly the design, the shape of coils, and the placement of each element of the array. As a consequence, these sensitivities may considerably vary from one subject to another. Advanced Parallel Imaging techniques estimate the

coil sensitivity information and incorporate it into the reconstruction pipeline. The MR signal from a single receiver coil is expressed as follows

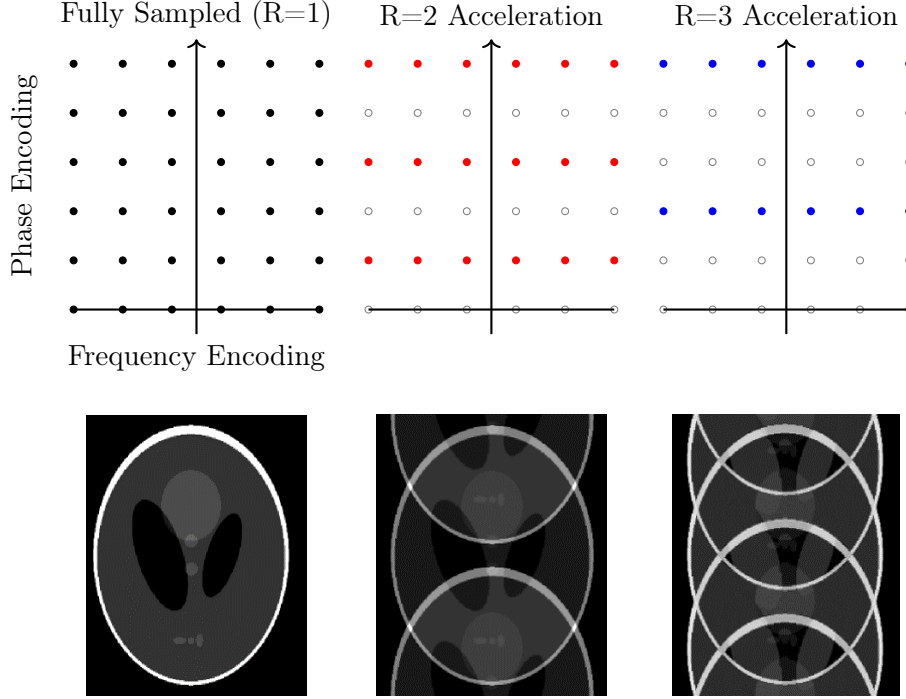


Figure 1.10: Sampling and the resulting aliasing patterns for several acceleration factors. **Top:** k -space sampling patterns for fully sampled ($R = 1$), $R = 2$, and $R = 3$ accelerated acquisitions. **Bottom:** Corresponding phantom images, showing the aliasing artifacts as the acceleration factor R increases in each case. The acceleration factor stands for the scan time reduction relative to the fully-sampled case.

The MR signal from a single receiver coil is expressed as follows

$$S_j(\mathbf{k}) = \int_V C_j(\mathbf{r}) M_{xy}(\mathbf{r}) e^{-2\pi i \mathbf{k} \cdot \mathbf{r}} d\mathbf{r} \quad (1.40)$$

where r is the spatial position vector, $C_j(\mathbf{r})$ is the spatially coil sensitivity for the coil j , M_{xy} is the spatially varying magnetization, k stands for the k -space position. In a discretized matrix notation, the signal is expressed:

$$s = \mathbf{SFC}x = \mathbf{E}x \quad (1.41)$$

where $\mathbf{S}$ is the sampling operator, $\mathbf{F}$ is the Fourier transform operator and $\mathbf{C}$ is the sensitivity maps operator. The three operators form $\mathbf{E}$, the encoding operator, used in a large number of parallel imaging reconstruction tasks. In multiple coil acquisition, each coil contains only a part of the image; a combination of the image per coil is then required to correctly reconstruct the image.

In parallel imaging, the acceleration factor stands for the degree of undersampling the Fourier space with respect to the fully sampled space, as defined by the Nyquist-Shannon

sampling theorem. In Cartesian acquisition, which is the frequent case in clinical routine, undersampling is typically performed in the phase encoding direction. For example, when undersampling is performed with an acceleration factor of $R = 2$, we skip one every two lines. When the factor is equal to $R = 3$, 2 out of every 3 consecutive lines are skipped in the acquisition process, see Figure 1.10. The direct consequence of undersampling is the decrease of the image field-of-view. As an example, when $R = 2$, the field of view decreases by a factor of 2, and pixels that are originally half a field-of-view apart in the unaccelerated image fall on top of one another, which creates the aliasing artifact. The trade-off between the temporal resolution, image quality, and ultimate SNR limits the range of values for the acceleration factor [27]. Generally, an acceleration factor of 2 is used in the clinical routine, ensuring sufficiently high-quality images. In applications where motion information and temporal variations are clinically required, such as perfusion scans, an acceleration factor of 3 and 4 is more common.

The main objective is to have a high-fidelity between the reconstructed image and the desired one using parallel imaging. As described in the following subsection, missing data are estimated from acquired data using a system of equations, depending on the considered method. Therefore, all these methods rely on the calibration information, and use it to resolve this system of equations. The amount and quality of calibration data play a major contribution in the reconstruction quality.

1.2.1 Techniques for Parallel Imaging

SENSE [23] and GRAPPA [24] are the two most commonly used parallel imaging techniques either in a research or a clinical context. GRAPPA is a k -space based technique that operates on coil information from a fully-sampled calibration space and performs the reconstruction of the missing points in the Fourier space. SENSE is an image-domain technique that uses the coil sensitivity maps to solve for unaliased pixel values.

1.2.1.1 Generalized autocalibrating partially parallel acquisitions - GRAPPA

GRAPPA is based on estimating the unsampled k -space data by using the relationships between neighboring points [24]. A small fully sampled part of the dataset is required for calibration purposes, named as the autocalibration signal. From the calibration data, a GRAPPA weight set can be computed to determine the relationship between neighboring points in Fourier space. Then, this weight set is applied to the undersampled data to estimate the missing points. A Fourier transform is then performed to generate the unaliased image from the estimated k -space data.

The GRAPPA kernel is the core implementation of the GRAPPA acceleration method. It is a recurring pattern of neighboring points in k -space. The kernel contains source and target points, such as the source points are collected, and the target samples must be reconstructed. In the calibration space ACS, the source and target points are known; hence the relationship between these points can be established. The center of k -space is generally used as the ACS; it contains high signal data, which reduces the noise impact on GRAPPA weights. Let N_c be the number of receiver coils. R is the acceleration factor. For each coil, the measured k -space data is denoted $S^c(k_x, k_y)$, where $c \in \{1, \dots, N_c\}$.

For a missing k -space sample at (k_x, k_y) in coil c , GRAPPA estimates its value as a linear combination of neighboring acquired samples from all coils with the following equation:

$$S_{\text{miss}}^c(k_x, k_y) = \sum_{k=1}^{N_c} \sum_{(m,n) \in \mathcal{N}} w_{c,k}^{m,n} \cdot S_{\text{acq}}^k(k_x + m, k_y + n) \quad (1.42)$$

where: $S_{\text{miss}}^c(k_x, k_y)$ is the missing value to reconstruct in coil c , $S_{\text{acq}}^k(k_x + m, k_y + n)$ is a neighboring acquired k -space value in coil k , offset by (m, n) , $\mathcal{N}$ is the set of points defining the **GRAPPA kernel**, and $w_{c,k}^{m,n}$ are the complex kernel weights, to be estimated from ACS. For each possible application of the kernel in the ACS, the above equation can be written in matrix notation as a linear system:

$$y_i = \mathbf{w}^\top \mathbf{x}_i \quad (1.43)$$

where: y_i is the target value (center point of the kernel, for a given coil), $\mathbf{x}_i$ is a vector containing all the neighboring data points (from all coils), $\mathbf{w}$ is the vector of kernel weights. By collecting such examples from the ACS region, an overdetermined system is built:

$$\mathbf{Y} = \mathbf{X}\mathbf{w} \quad (1.44)$$

where $\mathbf{Y}$ is a column vector of target values, $\mathbf{X}$ is a matrix whose rows are the neighbor vectors $\mathbf{x}_i$, and $\mathbf{w}$ is the vector of unknown weights. The weights are estimated via least-squares fitting:

$$\mathbf{w}^* = (\mathbf{X}^H \mathbf{X})^{-1} \mathbf{X}^H \mathbf{Y} \quad (1.45)$$

Once the kernel weights $\mathbf{w}$ are determined, they are applied throughout the undersampled region of k -space to estimate the remaining unsampled points:

- For each missing point and each coil, the acquired neighbor values from all coils are collected into a vector $\mathbf{x}_j$.
- The missing value is estimated as $y_j = \mathbf{w}^\top \mathbf{x}_j$.

The process will be repeated for all the elements of the undersampled space.

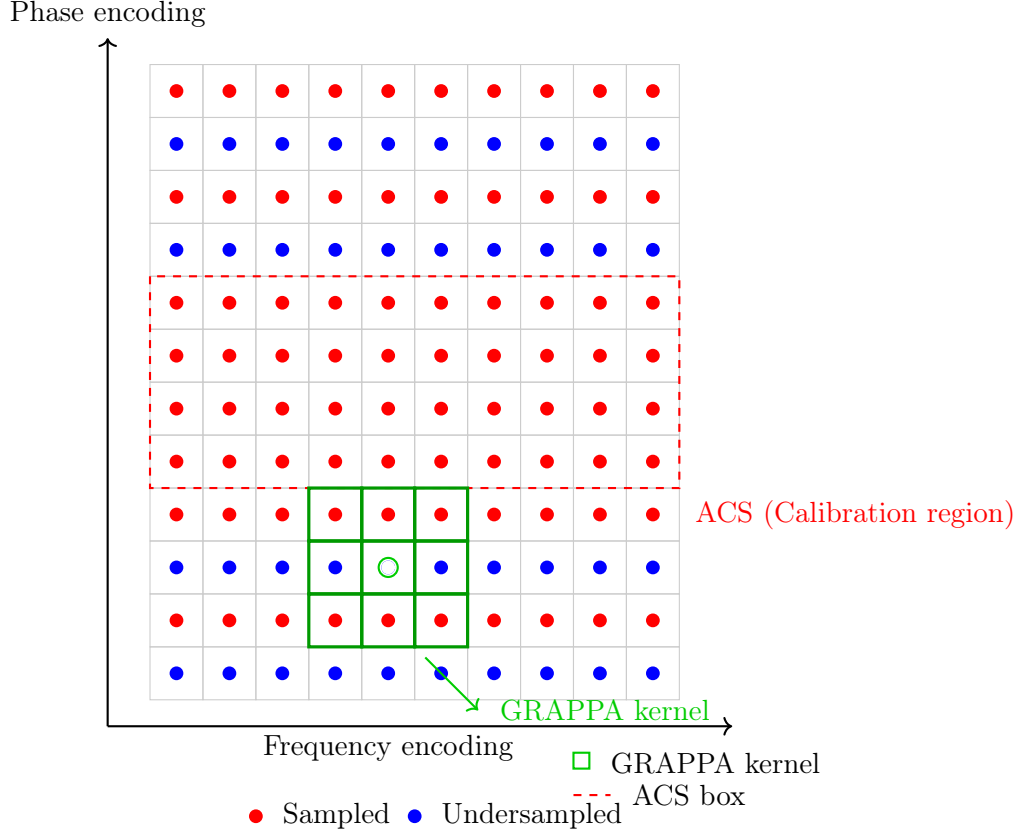


Figure 1.11: Sample k -space acquisition scheme and GRAPPA kernel for Cartesian GRAPPA with $R = 2$

1.2.1.2 Sensitivity Encoding

SENSE is one of the earliest suggested methods in PI. The method operates in the image domain and uses the following principle : each point in the aliased image is the sum of several points in the unaliased image weighted by the local coil sensitivity value at each point. Let's assume a case for which we have an acquisition with three coils with an acceleration factor R of 3. The set of equations for each pixel in the image is:

$$\begin{bmatrix} y_a \\ y_b \\ y_c \end{bmatrix} = \begin{bmatrix} c_{a1} & c_{a2} & c_{a3} \\ c_{b1} & c_{b2} & c_{b3} \\ c_{c1} & c_{c2} & c_{c3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad (1.46)$$

where x_1, x_2, x_3 are the pixels in the unaliased image, to be recovered. y_a, y_b and y_c are the aliased pixels in coils a, b and c respectively. The c_{ij} elements are the coil sensitivity value for the coil indicated by $i \in \{a, b, c\}$ and the location defined by $j \in \{1, 2, 3\}$. In matrix notation, the equation above reads:

$$\mathbf{y} = \mathbf{C}_p \mathbf{x}, \quad (1.47)$$

where $\mathbf{x}$ is the unknown vector of R unaliased image pixels, $\mathbf{C}_p$ is an $N_c \times R$ matrix whose elements are the coil sensitivity values for the considered set of pixels, and $\mathbf{y}$ is a vector of

length N_c containing the aliased pixels for each coil. Since the values of the coil sensitivity maps can be estimated, and the pixels of the aliased image are known, it is possible to determine the unaliased pixels by the following equation

$$\hat{\mathbf{x}} = \mathbf{C}_p^\dagger \mathbf{y} \quad (1.48)$$

where $\mathbf{C}_p^\dagger = (\mathbf{C}_p^H \mathbf{C}_p)^{-1} \mathbf{C}_p^H$. By determining $\mathbf{x}$, a set of aliased points will be placed in the appropriate positions (1,2,3) in the previous example. To reconstruct the full image, a similar system of equations should be solved for each set of aliased pixels.

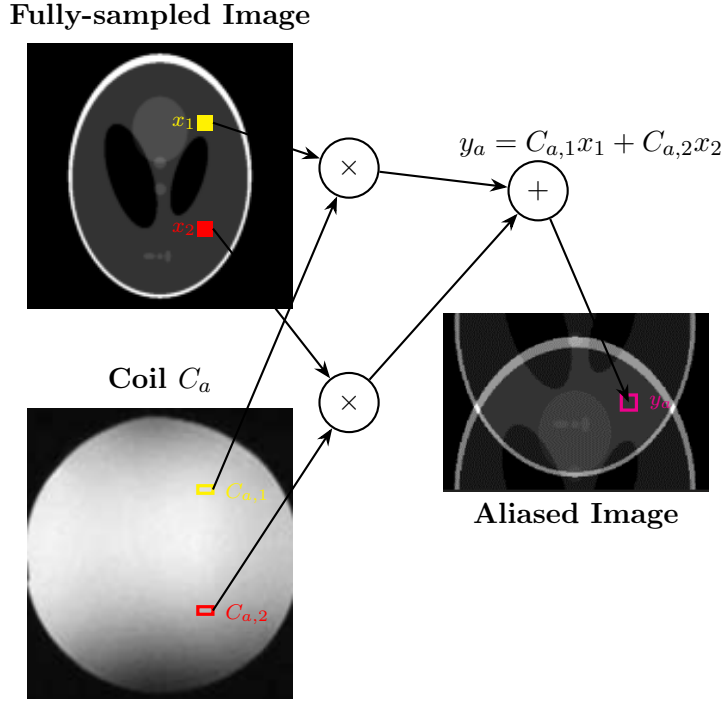


Figure 1.12: SENSE equation of an acceleration factor R of 2. the aliased image is the summation of two image pixels weighted by the corresponding coil sensitivity at each point.

An intuitive method to estimate coil sensitivity maps is presented in the original SENSE paper [23]. Images of the same object can be obtained using a coil with a homogeneous sensitivity profile, and with the receiver coils array used in the parallel imaging. The images obtained with the latter are divided by the body coil image, which leads to an estimate for the coils' sensitivities at each pixel. [26], [28] are other methods for sensitivity maps estimation. The SENSE method is limited to uniformly sampled Cartesian data. For more complicated sampling schemes, the CG-SENSE method is employed [29].

1.2.1.3 Sensitivity Maps Computation

In practice, the SENSE algorithm requires prior knowledge of the coil sensitivity maps. A first fundamental point is that coil sensitivity is not an intrinsic property of the antenna. Instead, it depends on the coil loading conditions, since the quality factor of the coil varies

with the presence of the human body or any other object in proximity. Hence, a patient-specific calibration is necessary for every scan. This calibration is typically performed at the beginning of the scan by acquiring a low-resolution image covering the full volume of interest with the receive coil array. One approach to estimate coil sensitivities is to acquire two low-resolution datasets requiring only a few central lines of k -space. The first dataset is obtained with a volume (body) coil providing a spatially homogeneous reference, and the second with the surface coil array. The sensitivity of each surface coil is then defined pixel-by-pixel as the ratio of the surface coil image to the homogeneous reference image. However, this method is susceptible to errors in cases where the patient moves between the two acquisitions or is in different respiratory phases.

An alternative method relies on a single low-resolution acquisition with the surface coil array only. The homogeneous reference image is then constructed by combining the surface coil images through the root-Sum-Of-Squares of the reconstructed coil magnitudes [30]. In this case, the low-resolution images are generated using a subset of central k -space lines. A binary mask is applied before computing the pixel-wise ratio in order to prevent division by zero. The resulting sensitivity map is a complex-valued quantity, where the magnitude reflects the sensitivity amplitude and the phase encodes the spatial phase distribution.

A third solution consists of embedding the calibration data—namely the central lines of k -space—directly into the accelerated acquisition. Such acquisitions are referred to as auto-calibrated. In this framework, the coil sensitivity maps can be estimated from the calibration lines using the same principles as above, thereby eliminating the need for separate calibration scans.

1.2.1.4 Conjugate Gradient SENSE CG-SENSE

CG SENSE [29] is a generalization of the SENSE. The method is designed to handle data collected with nonuniform Cartesian and non-Cartesian schemes. In the Cartesian framework, and when data are undersampled with a regular pattern, the aliasing artifacts can be easily modeled; they can be predicted. However, with non-Cartesian sampling (spiral, radial) and nonuniform Cartesian schemes, the aliasing artifacts are much more complex, since each pixel can compromise every other pixel in the aliased image. To solve the problem using the SENSE method on undersampled non-Cartesian/nonuniform Cartesian data, let's start with the discretized signal model introduced previously:

$$\mathbf{s} = \mathbf{E}\mathbf{x} \quad (1.49)$$

Here, the generalized encoding operator $\mathbf{E}$ is a multichannel version of the encoding matrix described in the previous section. For the case of non-Cartesian data, $\mathbf{E}$ corresponds to a multichannel version of the NUFFT operations: including coil multiplication, FT, and nonuniform sampling. The pseudo-inverse of $\mathbf{E}$ should be computed to determine the solution $\hat{\mathbf{x}}$:

$$\hat{\mathbf{x}} = (\mathbf{E}^H \mathbf{E})^{-1} \mathbf{E}^H \mathbf{s}, \quad (1.50)$$

$\mathbf{E}$ is a very large operator, the direct inversion of it would be computationally intensive. This can be avoided by multiplying both sides of Eq 1.50 by $\mathbf{E}^H \mathbf{E}$:

$$\mathbf{E}^H \mathbf{E} \mathbf{x} = \mathbf{E}^H \mathbf{s} \quad (1.51)$$

CG SENSE uses the CG solver to iteratively solve this equation. Although CG SENSE is more computationally efficient than the classic CG (Figure 1.13), it requires the computation of multiple NUFFT operations in the case of non-Cartesian trajectories, which may render the problem relatively computationally intense.

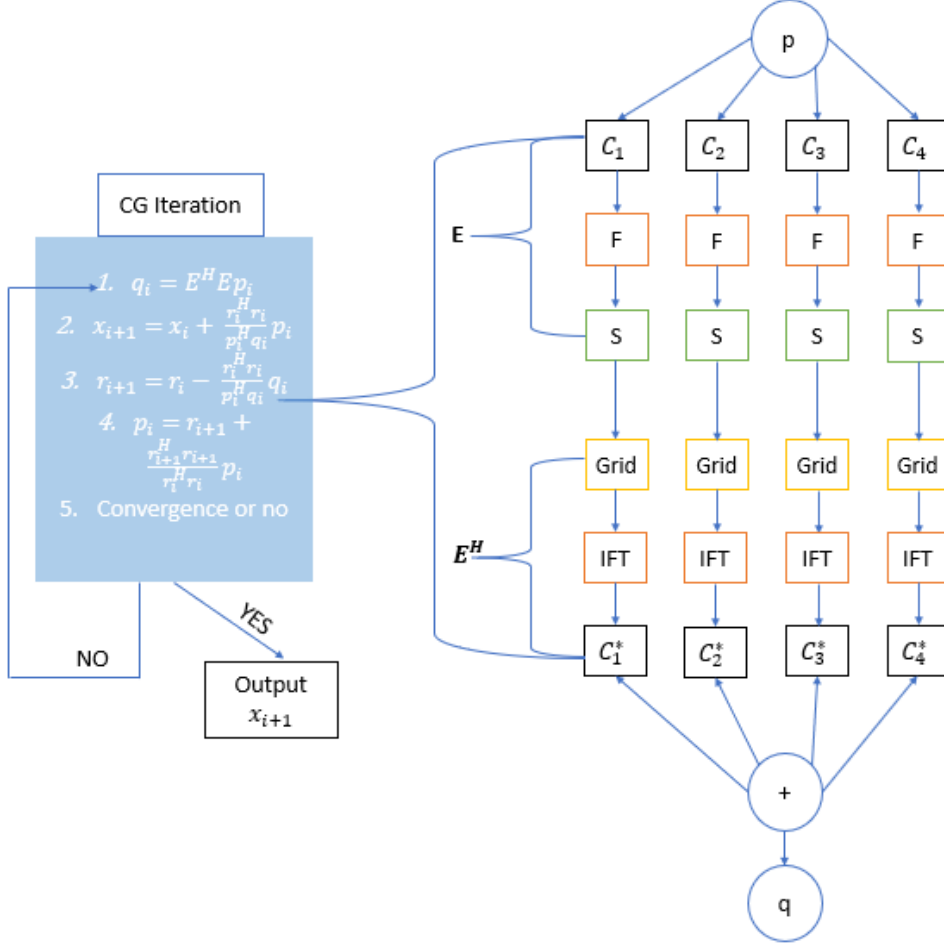


Figure 1.13: Diagram of the CG SENSE Method. The operator $E^H E$ is broken down into Coil sensitivity multiplication C_i , Fourier transform F , re-sampling S , gridding (if non-Cartesian sampling), inverse Fourier transform IFT , and coil combination steps.

Another benefit of the CG SENSE method is the facility of incorporating the regularization terms within the CG iteration.

1.2.2 Artifacts in Parallel Imaging

There are several artifacts that may arise if the parallel imaging technique does not properly resolve the aliasing due to undersampling. One of these artifacts is known as Noise enhancement, which stands for a remarkable decrease of SNR in a specific region of the image. Noise enhancement often occurs when a number of pixels overlap, and the coil sensitivity maps contain less distinct values. Noise enhancement can be significantly reduced by decreasing the acceleration factor [31]. Residual aliasing artifacts happen when

the SENSE coil sensitivity maps or the computation of GRAPPA weights are inappropriately estimated [32]. This suboptimal estimation can be a consequence of patient motion during the ACS estimation or the coil sensitivity maps computation and the MR scan, or if the number of lines in the calibration region is relatively small. For both artifacts, singularity in the coil sensitivity or the GRAPPA kernel operator may introduce errors in the reconstruction process. Increasing the amount of lines in the calibration region may considerably reduce the residual aliasing artifacts [31].

1.3 Synopsis

In this chapter, we have reviewed the fundamental physical principles of MRI, from nuclear magnetization to relaxation theory, spatial encoding, and pulse sequence design, with a particular focus on UHF MRI. This exploration emphasizes that the advantages of 7T systems—higher SNR, enhanced tissue contrast, and the possibility of submillimetric resolution—come at the cost of being much more sensitive to patient motion. Therefore, while UHF MRI represents an invaluable asset for clinical neuroscience, its full potential cannot be achieved without addressing the problem of motion. This observation establishes the necessity of advanced motion compensated reconstruction frameworks. The problem of reconstruction is then described in detail in the next chapter.

Chapter 2

MRI Reconstruction as an Inverse Problem

2.1 Forward and Inverse Problems

A forward problem predicts the effects of a given system based on their causes. This is the opposite of an inverse problem, where we know the observed measurements of a given phenomenon and the aim is to determine its causes. The difficulty of solving an inverse problem is particularly due to its well-posedness. A given problem P is said to be well-posed if it satisfies the three Hadamard conditions:

1. Solution Existence: a solution must exist according to the underlying mathematical model.
2. Solution Uniqueness: a set of k -space measurements should lead to a single solution in the image space domain.
3. Solution Stability: Small perturbations in acquired data should cause small perturbations in the image solution.

The first condition states that P , in order to be solved, must have a solution S . The second condition states P admits two solutions S and S' exist, then $S = S'$. The third condition refers to the stability criterion, which states that small perturbations of the acquired data should produce small perturbations to the solution S . If a problem P_1 does not satisfy one of the three conditions, P_1 is said to be ill-posed. Generally, inversion problems are considered as ill-posed; a reformulation of the problem by taking into account additional requirements and constraints is needed to derive the solution.

Most medical image reconstruction problems (i.e. MR reconstruction) are formulated as inverse problems, where the aim is to reconstruct the unknown image (the transverse macroscopic magnetization) from the sampled k -space measurements (the electrical potential induced in the receiver coils). In MR reconstruction, if the forward model fails to capture what happens in the acquisition pipeline, then a solution may not exist. Since ideally, the purpose is to reconstruct a continuous distribution of the imaged object's magnetization from a discrete number of k -space measurements, it is evident that the

problem admits an infinite set of solutions. In fact, when insufficient k -space samples are acquired, the Fourier transform-based reconstruction $M(x) = \mathcal{F}^{-1}\{S(k)\}$ fails to satisfy the uniqueness condition, as infinitely many image arrangements remain consistent with the limited data. More critically, the problem violates the stability condition due to the inherent sensitivity of the Fourier transform to high-frequency perturbations. This instability is mathematically characterized by the Riemann-Lebesgue lemma, where $\lim_{k \rightarrow \infty} \mathcal{F}\{\sin(kx\pi)\} = 0$, demonstrating that while high-frequency oscillations in k -space produce negligible changes $\Delta S(k)$ in the acquired data, they can generate arbitrarily large variations $\Delta M(x)$ in the reconstructed image through noise amplification. Hence, small measurement errors or noise in high-frequency k -space components are translated into significant reconstruction artifacts, which violate the third Hadamard criterion.

In order to resolve the ill-posed nature of the MR reconstruction problem, it can be reformulated by incorporating regularization terms and additional constraints.

2.2 Solution of the MR reconstruction problem

The linear inverse problem of MR reconstructions aims at minimizing a least squares cost function:

$$\hat{\mathbf{x}} = \operatorname{argmin}_{\mathbf{x}} f(\mathbf{x}) = \operatorname{argmin}_{\mathbf{x}} \|\mathbf{s} - \mathbf{E}\mathbf{x}\|_2^2. \quad (2.1)$$

Where $\mathbf{s}$ is the acquired k -space data, $\mathbf{E}$ is the encoding operator, $\mathbf{x}$ is the image. The solution $\hat{\mathbf{x}}$ is a minimum argument of $f(\mathbf{x})$, its gradient vanishes, which means

$$\nabla f = \left[\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_N} \right]^T = 0. \quad (2.2)$$

First, let's rewrite the cost function by using the hermitian (conjugate) transpose. Einstein notation will be used to describe the matrix product (assuming the matrix product $C = AB$, the C element corresponding to the i -th row and k -th column C_k^i verifies $C_k^i = A_j^i B_k^j$, where the repeated index stands for a summation) f can be expressed as follows (using the product rule)

$$f(x) = (\mathbf{s} - \mathbf{E}\mathbf{x})^H (\mathbf{s} - \mathbf{E}\mathbf{x}) = (s_j - x_i E_j^i)(s^j - E_i^j x^i). \quad (2.3)$$

Now, let's compute the partial derivative of the cost function f with respect to the n -th entry of x (using the product rule).

$$\frac{\partial f}{\partial x_n} = \left(\frac{\partial s_j}{\partial x_n} - \frac{\partial x_i E_j^i}{\partial x_n} \right) (s^j - E_i^j x^i) + (s_j - x_i E_j^i) \left(\frac{\partial s^j}{\partial x_n} - \frac{\partial E_i^j x^i}{\partial x_n} \right). \quad (2.4)$$

Moreover, and using the fact that for (m, n) , $\frac{\partial x_m}{\partial x_n} = \delta_{mn}$, where δ_{mn} is the Kronecker symbol.

$$\frac{\partial f}{\partial x_n} = -E_j^n s^j + E_j^n E_i^j x^i - s_j E_n^j + x_i E_j^i E_n^j = -2E_j^n (s^j - E_i^j x^i). \quad (2.5)$$

In matrix-vector product notation, and generalizing the above equation over every entry of the vector x , the gradient of the cost function f reads

$$\nabla f = -2\mathbf{E}^H (\mathbf{s} - \mathbf{E}\mathbf{x}). \quad (2.6)$$

Let $\hat{x}$ be the vector for which we have $\nabla f = 0$, $\hat{x}$ verifies

$$-2\mathbf{E}^H(s - \mathbf{E}\hat{x}) = 0. \quad (2.7)$$

Or in other terms,

$$\hat{x} = (\mathbf{E}^H \mathbf{E})^{-1} \mathbf{E}^H s = \mathbf{E}^\dagger s, \quad (2.8)$$

where $\mathbf{E}^\dagger$ is the Moore-Penrose generalized inverse. $\hat{x}$ defines the solution of the MR reconstruction problem. If the acquisition is fully sampled, there are as many data points in the vector s as unknowns in $\mathbf{x}$ and the noise level is adequate, then generally, a well-posed problem is expected. If the data is undersampled, there are fewer points in s than elements in $\mathbf{x}$; infinitely many solutions to the problem exist, the second Hadamard condition is then violated, and the problem may not produce a correct reconstruction.

The computation of $\mathbf{E}^\dagger$ involves the inversion of the matrix $\mathbf{E}^H \mathbf{E}$, which is computationally intensive. Generally, the direct inversion of the matrix is not computationally reasonable. However, the cost function f is convex and differentiable, so one way to solve the system is to iteratively search for a local minimum. One of the most efficient optimization methods to solve the problem of the type $Ax = b$ is the Conjugate Gradient method. The method will be detailed later.

A final point concerns the condition number of the forward model matrix, noted $C(E)$. It presents a theoretical prediction on the impact of small perturbations in the acquired data on the solution of the inverse problem $\hat{x}$. $C(E)$ assesses the third condition of Hadamard by providing a mathematical measure of the stability of the solution. $C(E)$ states the next relation of relative error propagation from the measured signal s to the image x .

$$\begin{cases} \frac{\|x - \hat{x}\|_2}{\|x\|_2} \leq C(E) \frac{\|\epsilon\|_2}{\|s\|_2}, \\ C(E) = \frac{\tau_{max}}{\tau_{min}} \end{cases} \quad (2.9)$$

where τ_{min} and τ_{max} are, respectively, the minimum and maximum singular values of the operator E . The matrix is considered well-conditioned if $C(E) \approx 1$. In this case, the solution $\hat{x}$ is said to be stable to small perturbations of the acquired signal. Based on the definition of $C(E)$, it is clear that $C(E) \geq 1$. As $C(E)$ increases, the system becomes severely ill-conditioned, and the stability condition no longer holds. If E is singular, the problem becomes undetermined. However, one way to solve the ill-posed reconstruction problem is by incorporating additional data, adding constraints, or applying regularization.

2.3 Prior Incorporation and Regularization types

In order to accelerate the MR acquisition process, undersampling is a strategy that considerably reduces the acquisition time of the MR exam; however, it is a direct cause of the ill-posedness of the MR reconstruction problem. In clinical routine, undersampling is inevitable in order to provide an interpretable image within a relatively short time. Hence, the MR reconstruction problem should be reformulated to leverage its ill-posed

nature. This can be accomplished by adding additional information to the system for two purposes:

- Isolation: where multiple solutions are encountered.
- Stabilization: where the solutions are unstable.

The MR problem is now described using a constrained cost function. The solution $\hat{x}$ reads

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} ||s - \mathbf{E}\mathbf{x}||_2^2 + \lambda R(\mathbf{x}) \quad (2.10)$$

where the first term refers to data consistency to guaranty consistency between the acquired data and the forward model, $R(x)$ is the mathematical element that adds prior information about the image to be reconstructed. λ is the regularization parameter, and it controls the trade-off between the optimization constraint and the data consistency. By setting x_{ref} as a reference solution obtained from a previous acquisition, the solution $\hat{x}$ is enforced to be similar to x_{ref} in the least-squares sense. Following the same logic as in subsection 2.2, $\hat{x}$ is formulated as the solution of the following constrained optimization problem:

$$\begin{cases} \hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} ||R(\mathbf{x})|| = \underset{\mathbf{x}}{\operatorname{argmin}} ||\mathbf{x} - \mathbf{x}_{ref}||, \\ ||s - \mathbf{E}\mathbf{x}||_2^2 < \epsilon \end{cases} \quad (2.11)$$

Eq 2.11 can be formulated as an unconstrained regularized problem; the problem reads

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} ||s - \mathbf{E}\mathbf{x}||_2^2 + \lambda ||\mathbf{x} - \mathbf{x}_{ref}||_2^2 \quad (2.12)$$

the regularization term is represented by the quantity $||x - x_{ref}||_2^2$. This regularization is called the Tikhonov regularization [33]. Choosing a regularizer in the sense of the L_2 norm is beneficial; this choice leads to a linear problem in the variable x with a closed-form solution. Using the same logic to find the expression in Eq 2.8, the solution of the Tikhonov regularization problem is

$$\hat{\mathbf{x}} = (\mathbf{E}^H \mathbf{E} + \lambda \mathbf{I})^{-1} (\mathbf{E}^H s + \lambda x_{ref}) \quad (2.13)$$

The choice of the parameter λ is crucial in achieving a good balance between both terms of equation 2.10, allowing for noise and aliasing reduction without compromising fidelity to the ground truth image. The Tikhonov regularization can be extended to any linear operator L . The regularization term will read then $R(x) = ||Lx - x_{ref}||_2^2$. The problem and its solution are presented by the following equation:

$$\begin{cases} \hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} ||s - \mathbf{E}\mathbf{x}||_2^2 + \lambda ||\mathbf{L}\mathbf{x} - \mathbf{x}_{ref}||_2^2 \\ \hat{\mathbf{x}} = (\mathbf{E}^H \mathbf{E} + \lambda \mathbf{L}^H \mathbf{L})^{-1} (\mathbf{E}^H s + \lambda \mathbf{L}^H \mathbf{x}_{ref}) \end{cases} \quad (2.14)$$

The regularization term may also involve an L_1 norm. The L_1 norm enforces the prior information that the vector x has only a few nonzero elements, which means that x should be sparse. The problem, under an L_1 regularization is expressed as follows

$$\hat{x} = \underset{\mathbf{x}}{\operatorname{argmin}} \psi(\mathbf{x}) = \underset{\mathbf{x}}{\operatorname{argmin}} ||s - \mathbf{E}\mathbf{x}||_2^2 + \lambda ||\Phi \mathbf{x}||_1, \quad (2.15)$$

Φ is a sparsifying transform (a wavelet transform for example). Φ is a linear operator that projects the non-sparse image x into another domain where the quantity $\Phi \mathbf{x}$ is sparse. The computation of the gradient of $\psi(\mathbf{x})$ involves the computation of $\frac{\partial |x_i|}{\partial x_n}$; the latter quantity is not well defined. When approaching 0^- and 0^+ , the derivative of $|x|$ is not the same. To solve this derivation problem, one approach is to consider $x \approx \sqrt{x^2 + \epsilon}$. With the Kronecker notation, we have : $\frac{\partial |x_i|}{\partial x_n} = \delta_{in} \frac{x_i}{|x_n|}$. With the quantity $\|\Phi \mathbf{x}\|_1 = |\Phi|_i^j x^i$, we can derive the following expression for the partial derivative of the L_1 norm:

$$\frac{\partial |\Phi_i^j x^i|}{\partial x_n} = \Phi_j^i \delta_{in} \frac{\Phi_i^j x^i}{|\Phi_n^j x^n|} = \frac{\Phi_j^n \Phi_n^j x^n}{\sqrt{x_n \Phi_j^n \Phi_n^j x^n + \epsilon}}. \quad (2.16)$$

With this, we can derive the gradient of the cost function ψ

$$\frac{\partial \psi}{\partial x_n} = -2E_j^n (s^j - E_i^j x^i) 2\lambda \frac{\Phi_j^n \Phi_n^j x^n}{\sqrt{x_n \Phi_j^n \Phi_n^j x^n + \epsilon}} \quad (2.17)$$

$$\nabla \psi = -2\mathbf{E}^H (\mathbf{s} - \mathbf{E}\mathbf{x}) + 2\lambda \Phi^H \mathbf{K} \Phi \mathbf{x}, \quad (2.18)$$

$K_n^n = (x_n \Phi_j^n \Phi_n^j x^n + \epsilon)^{-1/2}$ is a diagonal matrix. The L_1 regularization creates a non-linear problem with respect to $\mathbf{x}$. This type of regularization is used notably in Compressed Sensing applications [34, 35].

2.4 Least squares reconstruction

Let's consider the problem of reconstructing an image $\mathbf{x} \in \mathbb{C}^n$ from a given set of noisy measurements $\mathbf{s} \in \mathbb{C}^m$. As was presented in the previous subsection, computing the solution $\hat{x}$ involves the computation of the operator $E^H E = \sum_{c=1}^{N_c} C_c^H F^H S^T S F C_c$, where N_c is the number of coils used in the acquisition. The computation of this operator along with its inversion can be computationally demanding. Hence, iterative methods, such as gradient descent or the Conjugate Gradient, are used in order to solve the least squares problem.

2.4.1 The Gradient Descent GD

The idea of the GD is to iteratively compute the gradient of the least squares objective at the current estimate, then to update the estimate in the negative direction of the gradient with a defined step size. At iteration $n \geq 1$ the GD update is expressed as follows

$$x_n = x_{n-1} - \gamma_n E^H (E x_{n-1} - s), \quad (2.19)$$

where γ_n is the step size at the actual iteration n . Let's assume a fixed step size γ and an initial vector $x_0 = 0$, x_n reads

$$x_n = (I - \gamma E^H E) x_{n-1} + \gamma E^H s = \gamma \sum_{p=0}^{n-1} (I - \gamma E^H E)^p E^H s. \quad (2.20)$$

Under the condition that the spectral norm $\|I - \gamma E^H E\| < 1$, the geometric series $\sum_p (I - \gamma E^H E)^p$ converges to $1/\gamma E^H E$. The convergence of the series implies a condition on the step size γ .

$$0 < \gamma \leq \frac{2}{\lambda_{max}}, \quad (2.21)$$

where λ_{max} is the largest eigenvalue of the matrix $E^H E$. In this case, the gradient descent converges to the solution in the least squares sense. The convergence of the GD method depends on the scaling of the matrix E , and the choice of the step size γ .

2.4.2 The Conjugate Gradient CG

The CG method [36] is a popular approach for solving linear systems when the matrix is positive definite and symmetric. A CG iteration consists of updating the estimate in a direction that is conjugate to the update directions in all the previous iterations. The pseudo code of the Conjugate Gradient for solving a linear system of the type $A(x) = y$ reads

Algorithm 1 Conjugate gradient method for solving $\mathbf{y} = \mathbf{A}(\mathbf{x})$

```

1: input: initial estimate  $\mathbf{x}^0$  and residual  $\mathbf{r}_0 = \mathbf{A}(\mathbf{x}_0) - \mathbf{y}$ 
2:  $t = 0$ 
3:  $\mathbf{p}_0 = \mathbf{r}_0$ 
4: repeat
5:  $\alpha_t = \frac{\|\mathbf{r}_t\|_2^2}{\mathbf{p}_t^T \mathbf{A}(\mathbf{p}_t)}$ 
6:  $\mathbf{x}_{t+1} = \mathbf{x}_t + \alpha_t \mathbf{p}_t$ 
7:  $\mathbf{r}_{t+1} = \mathbf{r}_t - \alpha_t \mathbf{A}(\mathbf{p}_t)$ 
8: if  $\|\mathbf{r}_t\|_2^2 \leq \text{threshold}$  then
9:   break
10: end if
11:  $\beta_t = \frac{\|\mathbf{r}_{t+1}\|_2^2}{\|\mathbf{r}_t\|_2^2}$ 
12:  $\mathbf{p}_{t+1} = \mathbf{r}_{t+1} + \beta_t \mathbf{p}_t$ 
13:  $t \leftarrow t + 1$ 
14: until convergence or termination criteria satisfied
```

2.5 Model-based reconstruction

Generally, the MR reconstruction problem can be formulated as follows:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} f(\mathbf{x}), \quad f(\mathbf{x}) = g(\mathbf{x}) + h(\mathbf{x}), \quad (2.22)$$

where g is the term that enforces data fidelity and consistency, h is related to the prior knowledge. In this section, we will discuss optimization methods that are used to solve the problems of form 2.16. We first introduce smooth optimization methods applicable to differentiable functions, followed by non-smooth optimization methods where the regularizer is not differentiable.

2.5.1 Smooth Optimization

As discussed in section 2.3, Tikhonov Regularization is one of the most well-known approaches for stabilizing the ill-posed inverse problems [37, 38] with a quadratic regularizer.

$$g(\mathbf{x}) = \frac{1}{2} \|\mathbf{s} - \mathbf{E}\mathbf{x}\|_2^2 \quad \text{and} \quad h(\mathbf{x}) = \frac{\lambda}{2} \|\Phi\mathbf{x}\|_2^2, \quad (2.23)$$

where $\lambda > 0$ is a regularization parameter, Φ is a sparsifying transform. Then, the solution using Tikhonov regularization reads

$$\hat{\mathbf{x}} = (\mathbf{E}^H \mathbf{E} + \lambda \Phi^H \Phi)^{-1} (\mathbf{E}^H \mathbf{s}). \quad (2.24)$$

$\hat{\mathbf{x}}$ can be computed using the Conjugate Gradient (CG) method.

Generally, the regularizer h corresponds to a non-quadratic differentiable function. The Accelerated Gradient Method (AGM) is widely used to minimize such functions. The t iteration of the AGM [39] is expressed as follows

$$\begin{cases} \mathbf{x}^t = \mathbf{z}^{t-1} - \gamma_t \nabla f(\mathbf{z}^{t-1}), \\ \mathbf{z}^t = \mathbf{x}^t + \frac{(q_{t-1}-1)}{q_t} (\mathbf{x}^t - \mathbf{z}^{t-1}) \end{cases} \quad (2.25)$$

The system is initialized with $\mathbf{x}^0 = \mathbf{z}^0 \in \mathbb{C}^\kappa$, γ_t is the step parameter. q_t are the overrelaxation parameters. If $\forall t \quad q_t = 1$, AGM reduces to the previously discussed gradient descent. In order to accelerate the algorithm's convergence, a classical choice for the parameters q_t is the following set:

$$q_t = \frac{1}{2} \left(1 + \sqrt{1 + 4q_{t-1}^2} \right), \quad q_0 = 1 \quad (2.26)$$

For the step size γ_t , it is either fixed to a small value γ for all the iterations, or adapted via backtracking line search [40].

2.5.2 Non-smooth Optimization

In MRI, we encounter some non-smooth regularizers. Some well-known examples include the Total Variation TV [41, 42] and Hessian Schatten-norm [43, 44]. For example, the TV-regularized MRI problem reads

$$g(\mathbf{x}) = \frac{1}{2} \|\mathbf{s} - \mathbf{E}\mathbf{x}\|_2^2 \quad \text{and} \quad h(\mathbf{x}) = \lambda \|\Phi\mathbf{x}\|_1, \quad (2.27)$$

where Φ represents finite differences calculated along each dimension of $\mathbf{x}$. In order to work around the non-differentiability of h , the Proximal Gradient Method (PGM) [45, 46] and the Alternating Direction Method of Multipliers (ADMM) [47, 48] are widely used in this context. The pillar concept lies in the proximal operator [49], defined as

$$\text{prox}_{\lambda h}(\mathbf{z}) = \underset{\mathbf{x} \in \mathbb{C}^n}{\text{argmin}} \left\{ \frac{1}{2} \|\mathbf{x} - \mathbf{z}\|_2^2 + \lambda h(\mathbf{x}) \right\}, \quad \lambda > 0, \quad \mathbf{z} \in \mathbb{C}^n. \quad (2.28)$$

Algorithm 2 PGM

```

1: input: initial values  $\mathbf{x}^0 = \mathbf{z}^0$  and step-size  $\gamma > 0$ 
2: for  $t = 1, 2, \dots$  do
3:    $\mathbf{u}^t \leftarrow \mathbf{z}^{t-1} - \gamma \nabla g(\mathbf{z}^{t-1})$ 
4:    $\mathbf{x}^t \leftarrow \text{prox}_{\gamma h}(\mathbf{u}^t)$ 
5:    $\mathbf{z}^t \leftarrow \mathbf{x}^t + \left(\frac{q_t-1}{q_t}\right)(\mathbf{x}^t - \mathbf{x}^{t-1})$ 
6: end for

```

The projection onto a subset $\mathcal{X} \subseteq \mathbb{C}^n$ is the proximal operator for $h(\cdot) = \mathbb{K}_{\mathcal{X}}(\cdot)$, where $\mathbb{K}_{\mathcal{X}}(\cdot)$ is the indicator function. For every $\mathbf{z} \in \mathbb{C}^n$, the existence and uniqueness of the proximal operator is guaranteed when the function h is proper, closed, and convex [50].

Algorithm 3 ADMM

```

1: input: initial value  $\mathbf{x}^0$  and penalty parameter  $\gamma > 0$ 
2: for  $t = 1, 2, 3, \dots$  do
3:    $\mathbf{u}^t \leftarrow \text{prox}_{\gamma g}(\mathbf{x}^{t-1} + \mathbf{z}^{t-1})$ 
4:    $\mathbf{x}^t \leftarrow \text{prox}_{\gamma h}(\mathbf{u}^t - \mathbf{z}^{t-1})$ 
5:    $\mathbf{z}^t \leftarrow \mathbf{z}^{t-1} + \mathbf{x}^t - \mathbf{u}^t$ 
6: end for

```

The ADMM algorithm proposes a reformulation of the MR reconstruction problem as an equivalent constrained optimization problem. The constrained formulation reads

$$\text{minimize } g(\mathbf{z}) + h(\mathbf{x}) \text{ subject to } \mathbf{z} = \mathbf{x} \quad (2.29)$$

This problem can be solved using the Augmented Lagrangian (AL) [51], with the dual variable s . The optimization is then done in an alternating manner over $\mathbf{x}$, $\mathbf{z}$ and s .

2.6 Synopsis

The inverse problem formalism of the reconstruction allows us to incorporate prior information and physical models directly into the reconstruction pipeline. This chapter revisited some essential notions that will help us to understand how the GRICS reconstruction works, mainly the Least squares formulation, the Tikhonov regularization, and the Conjugate Gradient solver. The next chapter addresses how motion can be defined in the MRI context and how its different types can be accurately modeled.

Chapter 3

Motion in MRI

Patient motion during Magnetic Resonance Imaging (MRI) is a major obstacle to obtaining high-quality images [52]. In MRI, the reconstructed image is obtained by performing the FT of the acquired k -space data. When considering the discrete FT, each image pixel is formed from a weighted sum of all points in k -space. Consequently, any corruption or inconsistency in the acquired data (such as that caused by patient motion) propagates through every pixel and manifests as blurring or ghosting artifacts. These motion artifacts can arise from a variety of sources, including cardiac, respiratory, and patient motion (for instance, head motion in brain MRI). To achieve high-quality reconstruction, adequate spatio-temporal information must be acquired and incorporated into the reconstruction pipeline. In particular, selecting an appropriate model, either for rigid motion (translations and rotations) or non-rigid motion (localized deformations), is critical for effective motion compensation. With the introduction of ultra-high field MRI systems (7T and beyond), even small or subtle patient movements become more impactful. Hence, correcting these motions is essential to fully exploit the increased spatial resolution and improved signal-to-noise ratio offered by higher magnetic fields.

3.1 Rigid Vs. Non-Rigid Motion

In MRI, patient movement is classified as rigid or non-rigid, and this classification determines how motion corrupts the acquired k -space and how it should be corrected. Rigid motion considers the object as a solid body whose position in time is fully described by six Degrees Of Freedom (DOFs); this assumption holds well for Brain MRI because the skull constrains intra-cranial tissue. Rigid correction therefore aims at estimating those six parameters using external cameras, embedded navigators or, more recently, single shot deep networks—and inserting them into the signal model so that corrupted k -space lines can be phase-re-modulated or re-gridded before a conventional Fourier reconstruction. Robust retrospective implementations have already shown that such six-DoF pose estimates can delete head-motion artifacts in 3T MPRAGE to near-stationary image quality. Non-rigid motion, by contrast, arises whenever tissue elasticity comes into play—respiratory motion of the diaphragm and abdominal organs, the periodic deformation of the myocardium, or fetal movements. Because every voxel may follow a different

trajectory, correction demands the estimation of a dense displacement field and its incorporation into a physics-based reconstruction loop. Although the non-rigid based algorithms are more computationally demanding and subject to over-fitting if regularization is inappropriate, they are important for assessing cardio-thoracic, abdominal and fetal imaging. The standard Fourier-encoding model for static objects is

$$s(\mathbf{k}) = \int \rho(\mathbf{x}) e^{-i2\pi \mathbf{k} \cdot \mathbf{x}} d\mathbf{x}, \quad (3.1)$$

where $s(\mathbf{k})$ denotes the complex k -space data acquired at spatial frequency $\mathbf{k}$ and ρ is the spin-density distribution. If the object moves, a point $\mathbf{x}$ is carried to a new spatial position $\Phi_t(\mathbf{x})$ at time t , so the signal becomes

$$s_t(\mathbf{k}) = \int \rho(\Phi_t^{-1}(\mathbf{x}')) e^{-i2\pi \mathbf{k} \cdot \mathbf{x}'} d\mathbf{x}'. \quad (3.2)$$

Rigid motion. For a *rigid body* the deformation is $\Phi_t(\mathbf{x}) = \mathbf{R}_t \mathbf{x} + \mathbf{t}_t$, with $\mathbf{R}_t$ a rotation matrix and $\mathbf{t}_t$ the translational component. Substituting into (3.2) and exploiting the orthogonality of $\mathbf{R}_t$ yields

$$s_t^{\text{rigid}}(\mathbf{k}) = e^{-i2\pi \mathbf{k} \cdot \mathbf{t}_t} s(\mathbf{R}_t^\top \mathbf{k}), \quad (3.3)$$

i.e. a global phase term from the translation and a rotation of the sampling trajectory. Because these distortions are spatially invariant, they can be removed retrospectively (re-gridding, phase demodulation) or prospectively (gradient updates) once the six pose parameters $\{\mathbf{R}_t, \mathbf{t}_t\}$ are known.

Non-rigid motion. When tissue deformation is significant, each voxel follows its own trajectory, mathematically written

$$\Phi_t(\mathbf{x}) = \mathbf{x} + \mathbf{u}(\mathbf{x}, t) \quad (3.4)$$

with $\mathbf{u}$ a dense displacement field. Equation (3.2) then becomes

$$s_t^{\text{non-rigid}}(\mathbf{k}) = \int \rho(\mathbf{x}) e^{-i2\pi \mathbf{k} \cdot \mathbf{x}} e^{-i2\pi \mathbf{k} \cdot \mathbf{u}(\mathbf{x}, t)} d\mathbf{x}, \quad (3.5)$$

introducing *spatially varying* phase errors that couple every k -space datum with every spatial location. Correcting non-rigid motion is therefore a non-linear, ill-posed inverse problem that normally alternates

- Estimation of $\mathbf{u}(\mathbf{x}, t)$ (via image registration or physics-informed neural networks).
- reconstruction of the image ρ .

3.2 Motion types

3.2.1 Physiological motion

Macroscopic motion in the human body primarily arises from physiological processes such as respiration, cardiac activity, and blood circulation, as well as from the motion of the stomach and intestines during digestion. In the following sections, these sources of motion are discussed individually, focusing on their temporal characteristics and potential impact on imaging. Some general features of physiological motion that are particularly relevant to MRI include:

- Pseudo-periodicity: the motion pattern is approximatively the same in the time, rendering the prediction and the motion compensation easier.
- Pathology impact: The motion of an organ could be impacted by the changes in morphology or function, due to a specific pathology. A difference in terms of motion could be observed with comparison to the healthy subjects.

3.2.2 Respiratory motion

Respiratory motion is coupled with the movement of several surrounding organs: the chest expands and contracts, the diaphragm shifts in the superior-inferior direction, the liver partially follows this movement too, while the kidneys move primarily in the superior-inferior direction with a slight anterior-posterior component. Even the heart exhibits some motion as a result of respiration. Among physiological motions, breathing generally exhibits the simplest temporal pattern but the largest amplitude of displacement. The motion of the diaphragm can be modeled by a periodic, primarily sinusoidal movement in the superior-inferior direction. The displacement of the diaphragm $d(t)$ reads mathematically as follows:

$$d(t) = A \sin(2\pi ft + \phi) + d_0, \quad (3.6)$$

where A is the amplitude of the diaphragm motion, (10-20 mm for healthy adults), f is the breathing frequency (about 0.2–0.4 Hz for 12–24 breaths per minute), ϕ is the phase, and d_0 is the baseline position of the diaphragm. Examples of a simulated diaphragm motion pattern and a real respiratory motion signal are shown in Fig 3.1, Fig 3.2 respectively:

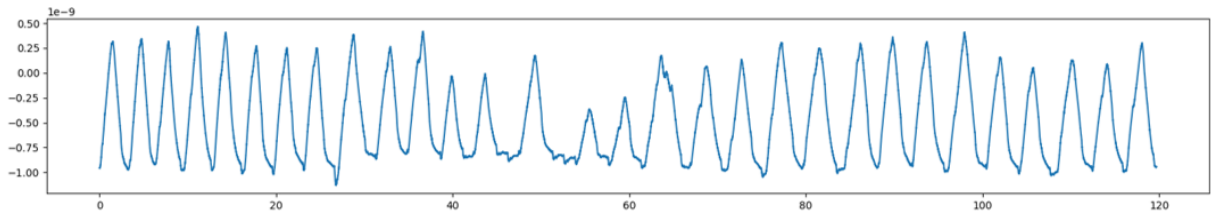


Figure 3.2: Example of a real respiratory motion signal, illustrating the quasi-periodic nature of respiration with cycle-to-cycle variability in amplitude and frequency.

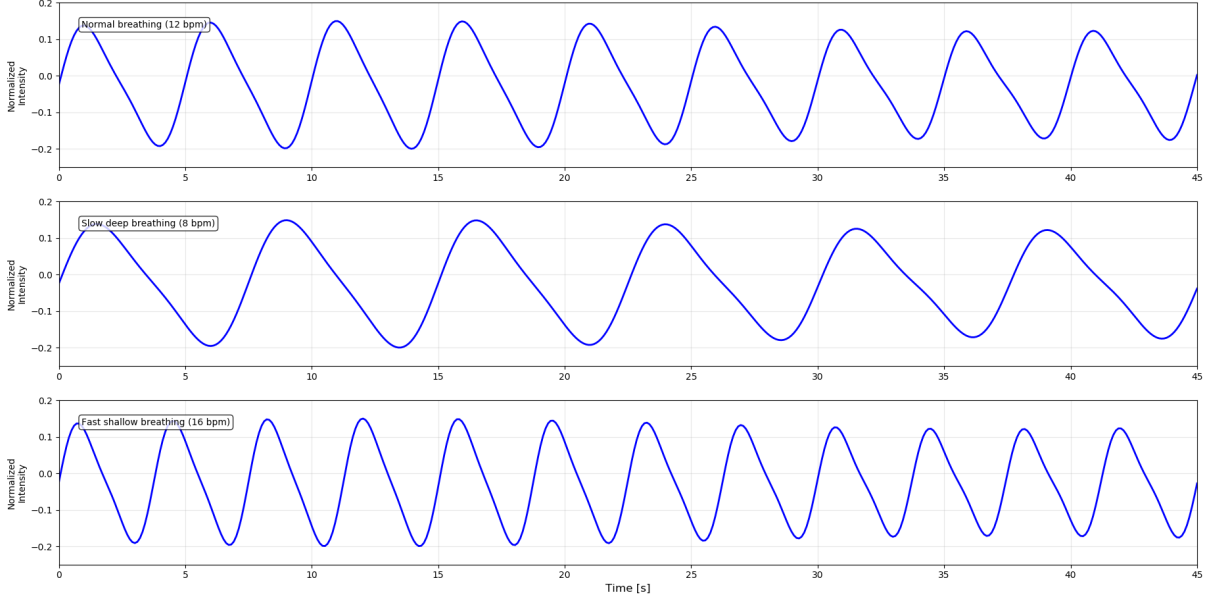


Figure 3.1: Simulated Diaphragm motion patterns during different breathing conditions showing normalized intensity over a period of 45 s. Top row: Normal resting breathing at 12 breaths per minute with regular amplitude and frequency. Middle row: Slow deep breathing at 8 breaths per minute demonstrating increased amplitude and prolonged respiratory cycles typical of relaxed or meditative breathing. Bottom row: Fast shallow breathing at 16 breaths per minute exhibiting reduced amplitude and shortened cycles characteristic of mild exertion or anxiety states. Negative values indicate upward diaphragm movement during expiration, while positive values represent downward movement during inspiration.

In advanced simulation processes, respiratory motion is described as a set of finite elements with muscle activation dynamics [53]:

$$M \frac{d^2 u}{dt^2} + C \frac{du}{dt} + Ku = F_{\text{muscle}}(t) \quad (3.7)$$

where M , C and K are the mass, damping, and stiffness matrices, u is the displacement, and $F_{\text{muscle}}(t)$ is the muscle contraction time-varying force.

Breathing motion is relatively slow compared to the timescale of signal (echo) generation and acquisition—namely, the basic pulse sequence or TR interval. As a result, its influence can often be neglected during gradient application, between spatial encoding and echo collection. However, it may cause issues between two phase encoding steps, particularly in slow spin-echo sequences with long TRs. Moreover, respiratory motion almost always poses a problem when evaluating motion between successive images.

3.2.3 Cardiac Motion

Cardiac activity is complex. It comprises mechanical, electrical, and blood pumping aspects. In particular, an electrical wave propagates through the heart; this propagation

induces a contraction of the muscle, specifically the myocardium. It also causes the blood pumping. The electrical signal is measured by ECG (ElectroCardioGraphy). A typical schematic illustration of an electrocardiogram is shown in the figure below. The R wave indicates the beginning of the systole, which is typically characterized by the heart contraction and the blood ejection into arteries, whereas the flatter interval corresponds to diastole, which is the relaxation of the myocardium. During both systole and diastole, there are periods when the heart remains nearly motionless—known as quiescent intervals. These intervals are typically longer during diastole (150–200 ms) than during systole (60–80 ms), although significant inter-subject variability exists.

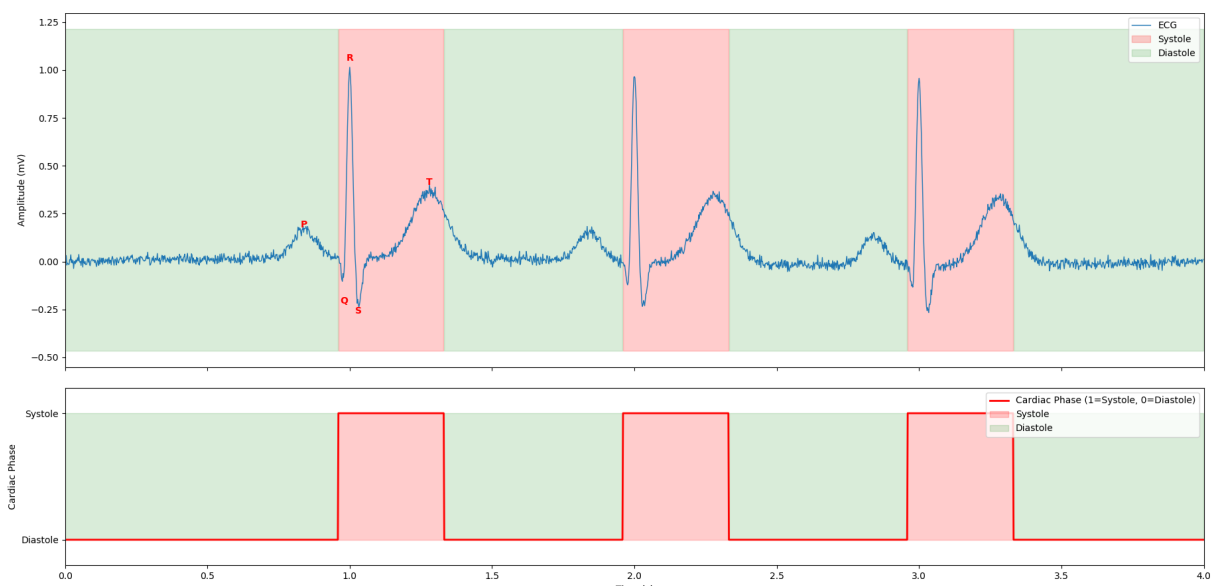


Figure 3.3: Electrocardiogram trace showing three consecutive cardiac cycles at 75 beats per minute with corresponding cardiac phases. The first row displays the ECG waveform with labeled P, Q, R, S, and T waves, where red shading indicates systolic phases (ventricular contraction from QRS onset to T wave end) and green shading indicates diastolic phases (ventricular relaxation and filling). The second row presents a binary representation of the cardiac cycle phases, illustrating the temporal relationship between electrical activity and mechanical cardiac events. Systole occupies approximately 37% of each cardiac cycle (300ms), while diastole comprises 63% (500ms at this heart rate), demonstrating the heart’s longer filling period compared to ejection period.

3.2.4 Peristaltic Motion

Peristaltic motion involves the bowels and certain other abdominal organs. It consists of radially symmetrical muscle contractions and relaxations that propagate as waves along the muscular walls. Unlike respiratory or cardiac motion, it remains less studied and utilized in MRI. Imaging these regions is challenging due to numerous air–tissue and air–fluid interfaces, which cause significant magnetic susceptibility differences. The bowels, in particular, contain many such interfaces.

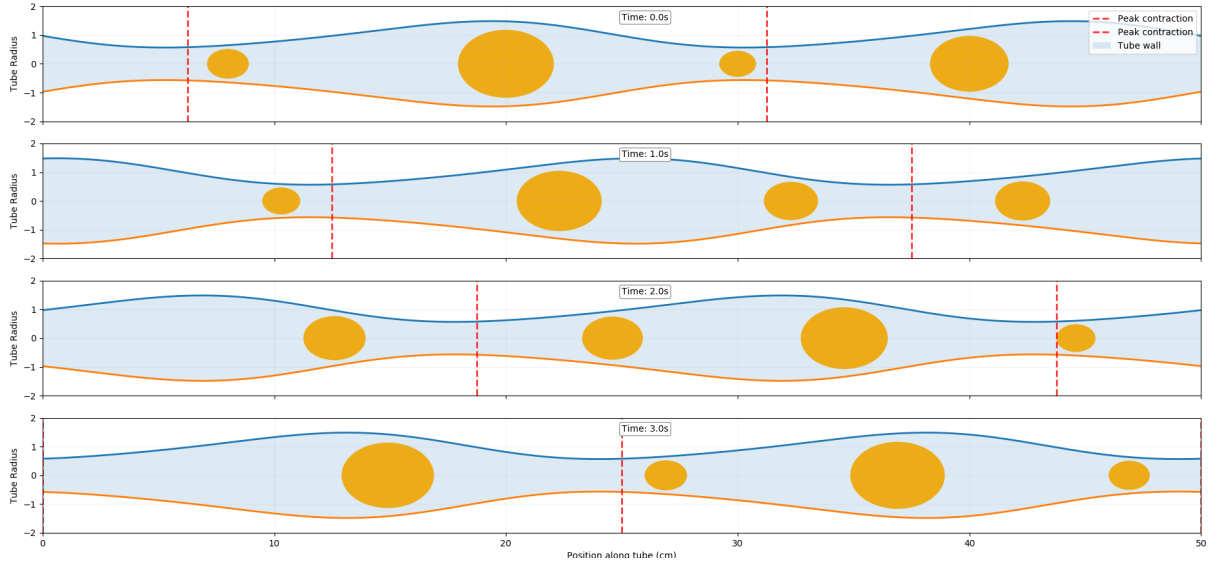


Figure 3.4: Schematic representation of peristaltic motion as encountered in abdominal MRI. The intestinal wall (blue) exhibits traveling contractile waves (red dashed lines) that propagate along the lumen, resulting in dynamic changes in tube diameter. Boluses of content (orange) are displaced distally as a consequence of these contractions. Such cyclic deformation and transport introduce non-rigid motion patterns in the abdominal region, which can lead to signal inconsistencies, phase errors, and blurring in MR acquisitions.

3.2.5 Head Motion

Brain MRI is particularly sensitive to both voluntary and involuntary movements originating from the musculature of the head and neck. Voluntary movements include swallowing, jaw clenching, mastication, eye blinks, and subtle head repositioning, all of which may introduce significant artifacts in brain imaging [54]. Swallowing, for instance, produces rapid displacements of the tongue, pharyngeal muscles, and larynx, which propagate to the base of the skull and can potentially corrupt image quality [55]. Even small voluntary adjustments of the head during long acquisitions can result in misregistration across k -space segments, and may lead to severe ghosting or blurring. Involuntary motion sources are equally relevant. Respiration induces low-frequency, quasi-periodic motion of the head and neck that can propagate through the cerebrospinal fluid (CSF) and intracranial vasculature [56]. Cardiac pulsation, transmitted through large arteries (e.g., carotid and vertebral), leads to micromovements of brain tissue at higher frequencies, on the order of the heart rate [57]. Moreover, facial and cervical muscle tone can produce micro-tremors even in resting subjects, adding stochastic, non-periodic perturbations [58]. These effects illustrate that motion in head MRI is not limited to bulk rigid displacement but is also subject to a wide range of biomechanical processes, ranging from controlled voluntary actions to involuntary motion.

3.3 Effect of motion in MRI

The motion of the imaged object occurs independently of the MRI data acquisition process; it can take place at any point during the execution of a pulse sequence. Hence, the severity of k -space data corruption depends on which specific segment of the sequence is affected. Central k -space encodes the low spatial frequencies of the imaged object, which primarily determine image contrast and global signal intensity. Hence, motion occurring during the acquisition of central lines leads to severe corruption of image contrast, often producing ghosting or global intensity modulation [59]. On the other hand, peripheral k -space contains the high spatial frequencies that define fine anatomical detail and sharpness. Motion affecting these regions mainly results in image blurring and loss of resolution rather than strong ghosting artifacts [54]. A theoretical framework to assess motion effects typically begins by a mathematical modeling of the motion, incorporating this model into the MR signal equation, and then attempting to reconstruct the image. However, this approach is based on the assumption that the motion is known—a condition that is rarely met in real-life clinical applications.

3.4 Motion estimation and compensation

Patient motion remains one of the most challenging problems in MRI. MR image formation relies on the assumption that the object being imaged remains stationary during the acquisition of k -space. This stationarity hypothesis ensures that each acquired data point corresponds to a consistent spatial encoding of the object, which makes the Fourier relationship between k -space and image space valid [59]. When this assumption is violated by motion, the correspondence between k -space samples and the underlying anatomy is disrupted. As a result, k -space is no longer a coherent representation of a static object but instead a mixture of signals originating from different positions at different times. The consequences of this violation depend on the type of motion and the time in which it occurred during acquisition. Bulk rigid-body motion during acquisition produces inconsistencies that manifest as ghosting, blurring, or misregistration in the reconstructed images [54]. Non-rigid physiological motion, such as respiration or pulsation, leads to temporal modulation of k -space data, creating periodic artifacts and contrast variations [60]. These effects are particularly severe for long readouts, high-resolution scans, and sequences acquired at ultra-high field strength, where even millimeter-scale motion can degrade the SNR. Because of this diversity and complexity, robust imaging requires both accurate motion estimation and effective motion compensation techniques.

3.4.1 Prospective VS. Retrospective MoCo

Motion-compensated reconstruction approaches have gained considerable attention as potential solutions to MRI motion artifacts. Prospective approaches adapt the acquisition on the fly, whereas retrospective techniques compensate for motion during or after reconstruction. A comprehensive review of real-time neuro-motion-correction techniques, such as spanning optical tracking and navigator-based updates, was presented by Ma-

cLaren *et al.* [60]. Optical systems update gradients and RF phases from externally tracked head pose with sub-degree accuracy; Moiré-phase-tracking markers mounted on a dental mouthpiece remain a useful approach for structural imaging at 7T [61]. Comparable precision has been achieved with integrated in-bore stereocameras that obviate subject-mounted assemblies [62]. Scanner-internal navigators provide another alternative: volumetric EPI navigators (vNavs) [63], spherical navigators [64], and fat-selective 3-D navigators (FatNavs) [65] can be interleaved within MPRAGE or MP2RAGE sequences and drive field-of-view updates with latencies below 100 ms. These navigators insert additional acquisition windows into the pulse sequence, during which low-resolution volumetric or spatially selective data are acquired to estimate subject motion. The effective latency therefore consists of two components: (i) the acquisition window necessary for navigator sampling, and (ii) the processing time required to estimate motion parameters. Although low-latency motion estimation and bidirectional sequence-host communication schemes have been implemented, such methods still require either significant modifications of the pulse sequences or extra hardware which may be difficult to fit within narrow head coils. The prospective methods do not correct for concomitant changes in B_0 field homogeneity or B_1 transmit-receive profiles caused by repositioning within the magnet [54]. Therefore, prospective MoCo has been successfully integrated into clinical imaging pipelines. Siemens Healthineers, for example, provides the FREEZEIt package for abdominal and thoracic imaging, which combines breath-hold strategies with navigator-based synchronization to improve robustness against motion. Moreover, BioMatrix Respiratory Sensors and Navigator Echo Techniques are now used for respiratory synchronization across the Siemens MRI platforms.

The principle of prospective motion correction is to maintain a constant FOV relative to the moving subject throughout the scan. This is achieved by continuously acquiring tracking data, which provide the real-time position and orientation of the imaging object, and transmitting this information to the MRI system during data acquisition. The scanner then dynamically adjusts the RF pulses and gradient waveforms to actively compensate for both rotational and translational motion. More specifically, and using a simplified mathematical model, the original gradient waveform $g(t)$ is transformed into $g'(t)$ according to the relation [60]:

$$g'(t) = A(t) \cdot g(t) \quad (3.8)$$

where $A(t)$ is the time-dependent transformation matrix encoding the detected motion.

Retrospective correction may be implemented without any sequence modifications such as reordering of the sampling scheme - or with minor modifications - and resolves motion inconsistencies during reconstruction. Applications of such techniques to brain MRI have mainly focused on rigid motion correction. Atkinson *et al.* [66] proposed an auto-focus motion-correction method for MRI based on minimizing an entropy focus criterion. Motion correction for arbitrary rigid motion was achieved using only the image data. Subsequent work integrated compressed sensing [35, 34, 67, 68] and parallel imaging [23, 24, 29], while DISORDER [69] randomizes k -space ordering to distribute artifacts and exploits the resulting redundancy for rigid-body recovery. Recent work has extended classical model-based pipelines with data-driven and physics-informed learning, markedly

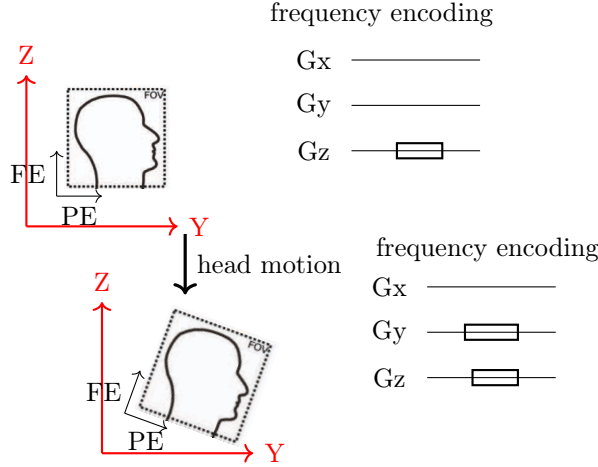


Figure 3.5: Representation of gradients update to compensate for head rotations: initially, the frequency-encoding requires only one gradient G_z . To compensate for the head rotation occurred on the yz -plane, both the PE G_y and the FE G_z gradients are adjusted to align the FOV with the new had pose.

improving the accuracy and speed of retrospective neuro-motion correction. Pirkel *et al.* [70] introduced a residual-CNN framework that learns only the motion-induced error after an initial navigator correction, enabling whole-brain multi-parametric reconstructions in under a minute while preserving quantitative accuracy. Implicit neural-representation strategies push this idea further: NeSVoR jointly estimates slice-wise rigid motion and super-resolution by optimizing a continuous neural field of the target anatomy, reducing reconstruction time by up to an order of magnitude compared with classical slice-to-volume registration [71]. Fast rigid-body estimation can also be obtained directly from k -space: MoPED uses a DenseNet regressor fed with central-band patches to predict motion parameters that are then enforced in a data-consistent solver, achieving full head-motion correction in ~ 30 s without external navigators [72]. To address through-plane and local deformations, Eichhorn *et al.* proposed PHIMO [73], a physics-informed unrolled network that first flags corrupted lines and then performs motion-aware reconstruction, reducing acquisition time by over 40 % relative to redundancy-based approaches. Retrospective motion correction methods leverage the properties of the Fourier transform to address motion-induced artifacts in k -space data. When bulk motion occurs during acquisition, it introduces both phase and magnitude distortions into the MR signal.

According to the Fourier shift theorem, translational motion in the object space manifests as a linear phase shift in k -space. Hence, these translations can be compensated by applying an appropriate phase correction, τ , to each k -space point. In this case, the corrupted signal reads:

$$\mathbf{S}'(k_x, k_y, k_z) = \mathbf{S}(k_x, k_y, k_z) \cdot e^{-2i\pi(k_x\tau_x + k_y\tau_y + k_z\tau_z)} \quad (3.9)$$

where $\mathbf{S}'$ is the corrected signal, where k_x , k_y and k_z are the coordinates of each k -space point and τ_x , τ_y and τ_z are the translational motion parameters. Using the properties of the Fourier transform, a 2D rotation in the image domain (I), centered at the origin,

corresponds to a rotation in the k -space. This relationship can be written as:

$$I(x \cos \theta - y \sin \theta, x \sin \theta + y \cos \theta) \longleftrightarrow S(k_x \cos \theta - k_y \sin \theta, k_x \sin \theta + k_y \cos \theta) \quad (3.10)$$

As a result, applying a rotation matrix $\mathbf{R}$ in the image domain induces a corresponding rotation in k -space, affecting both the phase and magnitude of the Fourier data. In contrast, translations (represented by the vector $\mathbf{T}$) affect only the phase of the k -space signal, as described above, where $\mathbf{k} = (k_x, k_y, k_z)$ denotes the k -space sampling trajectory:

$$S'(\mathbf{k}) = S(\mathbf{R} \cdot \mathbf{k}) \cdot e^{2\pi i \mathbf{R}\mathbf{k} \cdot \mathbf{T}}. \quad (3.11)$$

The relationships above assume that the rotation is performed about the origin of the image domain. In practice, however, head motion may not occur exactly around the image center. A rotation about an arbitrary pivot point $\mathbf{p}$ can be expressed as

$$\mathbf{x}' = \mathbf{R}(\theta)(\mathbf{x} - \mathbf{p}) + \mathbf{p}, \quad (3.12)$$

where $\mathbf{R}(\theta)$ is the 2D/3D rotation matrix and $\mathbf{p}$ is the rotation center. Substituting this transformation into the Fourier equation, the corresponding k -space signal becomes

$$S'(\mathbf{k}) = \int \rho(\mathbf{x}) e^{-i2\pi \mathbf{k} \cdot (\mathbf{R}(\theta)(\mathbf{x} - \mathbf{p}) + \mathbf{p})} d\mathbf{x}. \quad (3.13)$$

This expression reveals that off-center rotations result in two distinct effects: (i) a rotation of the spatial frequency coordinates through the multiplication $\mathbf{R}(\theta)\mathbf{k}$, and (ii) an additional linear phase modulation $e^{-i2\pi \mathbf{k} \cdot \mathbf{p}}$ induced by the translational term. Hence, these rotations cannot be considered as a pure k -space reorientation but also generate phase inconsistencies across the dataset. As mentioned by Atkinson and Hill [74], these inconsistencies lead to irregular and locally undersampled regions of k -space, particularly near the periphery of the FOV.

3.4.2 Model-based MoCo.

Unlike prospective or retrospective methods, which either prevent or correct artifacts after acquisition, model-based motion correction directly integrates motion into the forward MRI model. The basic principle is to consider the acquired k -space data m as the outcome of an encoding operator E that jointly accounts for the image, the coil sensitivities, and a motion operator that describes the spatial transformations occurring during acquisition. The problem is mathematically formulated as follows:

$$\min_{\rho, \alpha} \|m - E(\alpha)\rho\|_2^2 + \lambda R(\rho) + \mu Q(\alpha), \quad (3.14)$$

where $R(\rho)$ is a regularization on the image. $Q(\alpha)$ is a regularization on the motion parameters, λ, μ are regularization parameters. This joint estimation framework offers several advantages. By explicitly embedding motion into the encoding process, model-based methods preserve k -space consistency, avoid temporal blurring from simple averaging, and allow non-rigid deformations to be modeled with high flexibility. The most representative

approach is the GRICS method (Generalized Reconstruction by Inversion of Coupled Systems), introduced by Odille et al. [75], which incorporates motion fields estimated from external motion signals into the reconstruction process. Similarly, compressed sensing strategies have been extended to motion-resolved reconstructions, such as XD-GRASP (eXtra-Dimensional Golden-angle RAdial Sparse Parallel) [76], which exploits respiratory and cardiac self-navigation to sort data into motion states and reconstruct them jointly.

More recent developments include blind motion-compensated reconstructions, where motion is estimated directly from the corrupted k -space data without the need for external sensors or navigators [77]. These approaches typically employ iterative optimization, alternating between motion estimation and image reconstruction, and benefit from modern parallel computing architectures to reduce the computational demand. The main limitation of model-based methods lies in their complexity and computational burden, as the joint parameter space of image and motion can be extremely high-dimensional. Nonetheless, they represent one of the most accurate solutions to the motion problem in MRI, since they treat motion as an intrinsic part of the image formation model.

3.4.3 Deep Learning based MoCo

In recent years, the rapid progress of the deep learning research field has directly impacted MRI reconstruction and artifact correction, including motion-induced artifacts. Unlike traditional methods that rely on explicit modeling of motion or iterative reconstruction schemes, deep learning approaches employ large datasets to determine mappings between motion-corrupted and motion-free images, or to directly estimate motion fields from data. These methods can be subdivided into three families: image-domain correction, k -space domain correction, and motion field estimation networks. A comprehensive overview of the DL-based MoCo can be found in [78].

3.4.3.1 Image-domain methods

Image-based approaches treat the problem similarly to denoising or deblurring, mapping corrupted images to corrected ones through Convolutional Neural Networks (CNNs) [79], Encoder-Decoder U-Nets [80], or recurrent and transformer-based models [81]. Skip connections and cascaded structures help preserve anatomical details, while residual learning strategies focus on estimating motion artifact maps rather than the clean image itself [79]. Prior-assisted methods provide complementary information from multi-contrast acquisitions, temporal dynamics, or neighboring slices in order to improve correction fidelity [80]. The accessibility of image-based frameworks and the abundance of magnitude-only data have driven the popularity of this class of methods.

3.4.3.2 k -space and hybrid-domain methods

k -space-based strategies incorporate motion modeling directly into the reconstruction pipeline, which results in stronger physical conditioning. Hybrid frameworks replace specific components of classical iterative reconstructions with learned modules. For instance, CNNs have been employed for motion parameter estimation [82], as initialization of the

motion-corrected image [83, 84], or as part of the loss function in auto-focusing schemes [85]. Other works employ CNNs to detect motion-affected k -space lines and weight them in CS reconstructions [86]. For pseudo-periodic motion such as respiration, learned registration networks estimate DVFs that are subsequently integrated into model-based reconstructions [87]. Fully learning-based k -space approaches have also emerged, where networks jointly perform MoCo and reconstruction, often via unrolled architectures alternating between denoisers and data consistency layers [88]. These methods are particularly suited to dynamic cardiac or abdominal imaging, where temporal redundancies can be exploited to compensate for motion [89].

3.4.3.3 Motion field estimation

Another class of deep learning methods focuses on explicitly estimating motion fields. Instead of producing corrected images directly, these networks predict deformation maps (rigid or non-rigid) that align motion-corrupted frames to a reference. Optical flow networks, recurrent architectures, and spatial transformer networks (STNs) have been adapted for this purpose. Küstner et al. [90] reviewed the application of such models to cardiac and abdominal imaging, where non-rigid motion dominates. The estimated motion fields can then be integrated into conventional reconstruction frameworks, yielding motion-compensated reconstructions.

3.4.3.4 Limitations of DL based MoCo methods

Deep learning methods offer several advantages. First, they achieve fast inference, enabling near real-time correction. Second, they can capture highly non-linear relationships between corrupted and corrected images, making them particularly suitable for complex, non-rigid motion. Third, once trained, they require no additional scan time or sequence modification. However, challenges remain. The performance of supervised approaches depends heavily on the availability of ground-truth motion-free data, which is often a limitation. Synthetic motion simulations can be used for training, but may not fully represent in vivo motion statistics. Furthermore, generalization across different organs, field strengths, and vendors remains a major concern. Finally, the black-box nature of deep networks poses interpretability and reliability issues for clinical applicability.

3.5 Synopsis

The review of motion in MRI presented in this chapter demonstrates that both rigid and non-rigid motions significantly degrade image quality, particularly in high-resolution brain imaging at UHF. Rigid head shifts and rotations alone can already corrupt the consistency of k -space data, but physiological deformations due to respiration and cardiac pulsation further complicate the motion correction process by introducing complex and spatially varying distortions. Most existing brain MRI MoCo methods, however, are limited to rigid modeling, neglecting the impact of non-rigid movement. This observation justifies the decision to use GRICS with both rigid and non-rigid modeling motion. Having identified in the general introduction the need for computationally demanding solutions, this

thesis next addresses the computational aspects of MRI MoCo frameworks through the introduction of High Performance Computing (HPC), which is the focus of the following chapter.

Chapter 4

High Performance Computing in MRI

4.1 Foundation of High Performance Computing

Computational science has become a pillar of modern research, providing scientists with the capability to test hypotheses rapidly with a relatively reduced cost. Although each scientific discipline faces its own specific challenges, the computational problems they encounter often share common patterns. The domain dedicated to addressing these challenges and optimizing computational tools across scientific fields is High Performance Computing (HPC). HPC focuses on the design, analysis, and efficient use of advanced computing systems in order to maximize their impact on scientific applications. Research in this area typically involves modelling and characterizing computer architectures, identifying performance bottlenecks, developing software capable of exploiting hardware specificities, and giving insights to guide the design of future systems.

Modern supercomputers are the product of decades of continuous innovation in research, technology, and engineering. From the 1970s through the mid-1990s, HPC systems were typically designed as large, monolithic machines. In contrast, today's architectures are built from collections of commodity processors, tightly interconnected to deliver high performance while allowing their total computational power to be flexibly divided into partitions as needed. The fundamental unit in such systems is the node, usually implemented as a blade or rack-mounted server. These nodes are linked via high-speed interconnects, forming clusters whose scale can vary from a handful of servers in a single rack to massive datacenter-level infrastructures with tens or even hundreds of thousands of nodes, requiring dedicated facilities. This architectural paradigm enables scalability, cost-effectiveness, and resilience to hardware failures. Building on this clustered design, the internal structure of each node is composed of hardware comparable to that found in large-scale web service datacenters. Typically, these nodes rely on high-performance general-purpose CPUs with many cores, substantial banks of fast memory, and hierarchical storage solutions that balance speed with capacity. During the early 2000s, a wide range of processor architectures coexisted, often coupled with their own dedicated operating systems. In recent years, however, the market has consolidated, with x86 processors from Intel and AMD becoming the dominant choice. Some systems continue to use IBM's POWER architecture, while Arm-based processors from vendors such as Fu-

jitsu, Marvell, and Ampere have also emerged as viable alternatives. Regardless of the processor type, the vast majority of supercomputers today run on Linux [91]. Over the past decade, incremental improvements in component performance have been significant, yet the fundamental approach to integrating CPUs, memory, and storage within nodes has remained largely unchanged. Within each node, cost and energy efficiency considerations have historically guided design choices. The most widespread configuration is the dual-socket layout, where two processors are mounted on the same motherboard and share system resources. Single-socket nodes are deployed when the selected CPU does not support multi-socket operation, while some high-end designs employ up to four sockets per board. More commonly, however, vendors achieve higher density by integrating multiple motherboards into a single blade rather than further increasing the number of sockets on one motherboard [92]. In nodes equipped with multiple sockets, memory access patterns are shaped by the physical arrangement of processors and memory banks. Each CPU is directly connected only to a portion of the system’s main memory. When data resides in memory attached to another socket, the processor must issue a request across the inter-socket link and wait for the response. This design leads to a non-uniform memory access (NUMA) architecture, where the latency of memory operations varies depending on whether the data is local or remote. Since all remote accesses compete for the same interconnect bandwidth, performance degrades as the number of concurrent cross-socket requests increases.

While NUMA effects influence memory performance in multi-socket designs, additional performance gains can be achieved by complementing CPUs with specialized hardware. For certain workloads, co-processors—commonly referred to as accelerators—are employed to offload or speed up computationally intensive tasks. The most prominent examples today include general-purpose graphics processing units (GPGPUs), field-programmable gate arrays (FPGAs), and advanced network interface cards (NICs) capable of in-network computation. In such configurations, accelerators are connected to the CPU, which serves as the host responsible for orchestrating their use. The integration of accelerators alongside CPUs has significantly expanded the computational capacity of supercomputers. The next major milestone in HPC was the exascale computing. Exascale refers to the ability of a single system to perform 10^{18} Floating-Point operations per second (1 EFLOP/s) [93]. By the mid-2010s, this target had become a central focus for the HPC community, and as of mid-2021 it remained just out of reach. The term “road to exascale” is often used in the literature to describe both the technical obstacles encountered as increasingly larger systems are deployed and the strategies proposed to overcome them. Achieving exascale performance poses challenges on multiple fronts: maximizing efficiency across every system component, while ensuring that scientific applications remain maintainable and portable. Striking this balance is critical—applications must be optimized enough to exploit current hardware, yet not so specialized that they become incompatible with future architectures or programming models, which may differ substantially across vendors.

HPC refers also to the use of parallel and distributed architectures to solve computationally intensive problems that exceed the capabilities of conventional computer systems. Traditionally, HPC has relied on vector processors and large-scale supercomputers; however, over the past three decades, the paradigm has shifted toward clusters of commodity processors programmed with distributed-memory frameworks such as the Message Passing

Interface (MPI) [94], as well as shared-memory paradigms such as Open Multi Processing (OpenMP) [95]. The increasing heterogeneity of modern hardware, with multi-core CPUs tightly coupled to accelerators such as Graphics Processing Units (GPUs), has further refined the design space of HPC systems.

The development of hybrid programming models, where MPI coordinates computation across nodes (inter-nodes) and OpenMP (or CUDA/OpenCL) exploits intra-node parallelism, has become a well-used strategy to achieve scalability on current computing systems. Performance models such as the roofline framework [96] have been introduced to quantify the interaction between memory bandwidth, arithmetic throughput, and peak floating-point performance. The model gives insights into the possible algorithmic and architectural optimizations. Recent advances have extended this methodology to hierarchical memory systems and energy-aware computing [97, 98].

In the context of medical imaging, and particularly MRI, the adoption of HPC has been driven by the need for rapid image reconstruction and motion correction. MRI reconstruction problems often require solving large-scale inverse problems with high-dimensional data, making them particularly well-suited to parallel architectures. Early implementations distributed iterative solvers across CPU clusters using MPI [99, 100], while more recent approaches use GPU accelerators through CUDA to achieve real-time or near real-time reconstruction performance [101]. Toolboxes such as Gadgetron [102] and BART [103] embody this evolution, integrating parallel algorithms with modular frameworks designed for modern heterogeneous clusters.

4.2 Laws of Computing

In the study of computational performance, several foundational principles, commonly referred to as “laws”, have been proposed and continue to guide both theoretical and practical considerations. First, Moore’s Law [104] describes the exponential increase in transistor density on integrated circuits, reflecting the steady advancement of silicon fabrication technology and its implications for computational power. Amdahl’s [105] Law formalizes the theoretical speedup achievable when optimizing a portion of a multi-step algorithm, illustrating the diminishing returns of parallelization when sequential components dominate execution time. Gustafson’s Law [106] reinterprets scalability by showing that larger problem sizes can proportionally increase parallel efficiency, offering a new dimension of the benefits of parallel computing for large-scale applications. Finally, Wirth’s Law [107] emphasizes the software dimension of overall program performance, observing that software is getting slower more rapidly than hardware becomes faster. Improvements in hardware efficiency can be masked by growing software complexity. Consequently, algorithmic efficiency is as important as hardware capabilities to design high-performance applications.

4.2.1 Moore’s Law

Moore’s Law [104] is an empirical observation proposed in 1965, stating that *the number of transistors on an integrated circuit doubles approximately every two years*. This trend,

sustained for more than four decades, has driven the exponential growth of computational performance. Early gains in the 1990s and early 2000s were achieved by increasing clock frequency and pipeline depth, which directly improved uni-processor performance. However, the associated rise in dynamic power dissipation eventually reached the thermal limits of commodity cooling systems, making further frequency scaling impractical. As a result, the semiconductor industry shifted towards exploiting Moore’s Law through on-chip parallelism, integrating multiple cores per die rather than only increasing clock rates. While single-core performance has stagnated, overall throughput continues to rise with core counts. This architectural shift has also highlighted what we call the memory wall: improvements in transistor density outpace memory bandwidth growth, as Direct Random Access Memory (DRAM) remains fabricated with distinct processes and packaged separately. Consequently, memory-system performance has emerged as a critical bottleneck in HPC scientific computing applications.

4.2.2 Amdahl’s Law

Amdahl’s Law [105] introduces the principle that overall performance improvement is constrained by *the fraction of execution time spent in the part being optimized*. For instance, if a sub-computation accounts for 50% of the runtime, even a $1000\times$ speedup in that part only reduces the total runtime by a factor of two. More generally, if p denotes the fraction of execution time subject to optimization, s the speedup applied to that portion, and r and r' the original and optimized runtimes of the application, respectively, Amdahl’s Law reads

$$r' = r \left((1 - p) + \frac{p}{s} \right) \quad (4.1)$$

As $s \rightarrow \infty$, the runtime approaches $r(1 - p)$, emphasizing the diminishing returns of optimizing only part of a workload. A well-known corollary underscores the need for careful performance analysis before investing in optimization: *Premature optimization is the root of all evil*. Since optimization often requires significant effort and experimentation, it should be targeted at specific performance bottlenecks.

4.2.3 Gustafson’s Law

Gustafson’s Law [106] provides another perspective on scalability by *considering problem sizes that grow with computational resources*. In fact, increasing input resolution—such as refining the discretization in fluid dynamics simulations—yields more accurate results at the cost of larger workloads. Gustafson observed that, under such scaling, the asymptotically dominant computations begin to control execution time, and since these operations are often highly parallelizable, they constitute the most effective targets for optimization techniques. Mathematically, Gustafson’s Law reads

$$s(N) = (1 - p) + p.N \quad (4.2)$$

where p is the fraction of parallelizable work and N the number of processors. Unlike Amdahl’s Law, this formulation predicts near-linear scaling when p is close to 1. In MRI reconstruction, although problem size is constrained by acquisition time in the scanner

and the size of the encoding operator, Gustafson’s insight remains applicable: performance optimization should be guided by benchmarks at realistic problem scales to ensure effort is concentrated on the true computational bottlenecks.

4.2.4 Wirth’s Law

Wirth’s Law, proposed by Niklaus Wirth in 1995, states that “*software is getting slower more rapidly than hardware becomes faster*” [107]. This law highlights that despite exponential growth in hardware performance, end-user computing efficiency often stagnates or may even decline due to the increasing complexity and inefficiency of modern software. The observation is particularly relevant in the context of HPC. While Moore’s Law historically guaranteed constant improvements in raw processing power, software abstraction overhead has simultaneously reduced the effective performance that applications can get from available hardware resources. Hence, the practical gains of new architectures may be limited by software layers and the utilization of inefficient parallel programming paradigms.

4.3 Modern Processor Architecture

Modern processor architectures are mainly characterized by two components: the instruction execution hardware and the memory system. Performance optimization strategies can therefore be broadly categorized according to whether they enhance instruction throughput or memory access efficiency. With the industry-wide shift toward on-chip parallelism, single-core execution speed has largely plateaued, and memory-system performance has increasingly emerged as the dominant target for optimization. Recent innovations in processor architecture that have fueled advances in HPC parallel programming are largely tied to the mechanisms through which parallelism is exposed to the programmer. Historically, this trend gained importance when Graphics Processing Units (GPUs) began incorporating programmable cores. During the 1990s, GPUs relied primarily on fixed-function hardware dedicated to projective geometry and texture interpolation for 3D rendering. As graphics algorithms diversified, vendors introduced programmable cores into these pipelines, enabling Turing-complete instruction sets with arbitrary memory access. Despite this, a significant portion of GPU die area remains dedicated to fixed-function circuitry such as texture interpolation. In contrast, modern CPUs allocate substantial die area to out-of-order execution units designed to accelerate sequential workloads, particularly those represented by the long-standing SPEC [108] benchmarks.

4.3.1 The Memory System of Modern Processors

The memory systems of modern processors are inherently hierarchical, with Dynamic Random Access Memory (DRAM) representing the highest level typically considered in high-performance programming. To achieve efficient execution, it is generally undesirable to rely on magnetic storage (hard disks), whether explicitly through file I/O or implicitly through page-faulting, since both introduce prohibitive latency. Table 4.1 reports typical

access latencies across different memory levels. While DRAM provides access times on the order of ~ 100 ns, modern non-volatile memory such as NVMe-based Solid State Drives (SSDs) exhibit latencies around $\sim 100 \mu\text{s}$, already three orders of magnitude slower. Hard disk drives (HDDs) remain even further behind, with latencies in the millisecond range, i.e., five orders of magnitude slower than DRAM.

Table 4.1: Typical memory access latencies across technologies.

Memory type	Typical latency	Ratio vs DRAM
DRAM	100 ns	1
SSD NVMe	100 μs	$\sim 10^3$
HDD	10 ms	$\sim 10^5$

As a result, DRAM access latency is high—on the order of several hundred processor cycles. Bandwidth differs significantly between architectures: CPU systems achieve tens of gigabytes per second, whereas GPUs can reach several hundreds of gigabytes per second, largely due to more aggressive memory-controller designs and higher DRAM clock rates. These higher frequencies, however, come at the cost of reduced capacity: GPU DRAM typically ranges from 1–6 GB, compared to 4–128 GB in CPUs. Beneath DRAM, both CPUs and GPUs incorporate multiple levels of caches. In current Intel architectures, L3, L2, and L1 caches are used, with L3 shared across all cores of a socket and L2/L1 private to each core. By contrast, NVIDIA GPUs employ a smaller shared L2 cache along with per-core L1 caches. Additionally, GPUs are distinguished by exceptionally large register files, comparable in size to an L1 cache, which the CUDA programming model exploits by allowing much of a kernel’s working set to reside in registers while larger persistent data structures remain in DRAM. A key architectural distinction lies in cache coherence: CPU caches are coherent, ensuring that writes from one core are automatically visible to others via the cache hierarchy. GPU caches, however, are incoherent, requiring the programmer to explicitly manage inter-thread communication and data consistency.

A recent architectural innovation concerns the NVIDIA Blackwell generation of accelerators, which introduces two different deployment models. In the first configuration, standalone Blackwell GPUs (B100, B200) are delivered as traditional accelerators. These devices can be interconnected in dual-GPU modules through high-bandwidth links such as NV-IB1 or NVLink, which enables direct data exchange between the two dies within the same enclosure. This design differs from a conventional monolithic GPU organization and instead resembles a multi-die processor model. Moreover, NVIDIA implements a proprietary cache-coherence protocol that mimics certain aspects of a NUMA configuration, ensuring transparent data sharing and reduced overhead in memory transactions between dies. The second configuration, denoted GB100 (Grace + Blackwell integration), goes further by combining CPU and GPU resources in a System-on-Chip (SoC) multi-die package. Specifically, the GB100 consists of two primary components: (i) a “S-die” integrating an ARM-based CPU cluster designed by MediaTek alongside memory controllers and I/O subsystems, and (ii) a “G-die” hosting the Blackwell GPU cores. Both dies are

manufactured in TSMC’s 3 nm process and interconnected via a high-bandwidth NVLink C2C interface delivering up to 600 GB/s of bidirectional bandwidth. A distinctive feature of this design is the implementation of a shared Last-Level Cache (LLC), which is uniformly accessible by both the CPU and GPU. This departs from the traditional NUMA paradigm where cache and memory hierarchies are asymmetrically exposed across processing units. Instead, the GB100 architecture enables fine-grained cooperation between CPU and GPU, reducing data transfer latency, improving cache utilization, and offering a more unified memory model.

4.3.2 The Computing System of Modern Processors

The execution pipelines of CPUs and GPUs employ similar micro-architectural mechanisms to expose parallelism, though with some design-level trade-offs. Both architectures can integrate multiple sockets with preferentially accessible DRAM nodes in a Non-Uniform Memory Access (NUMA) configuration, and each socket contains multiple CPU cores / GPU Streaming Multiprocessors. Within each core, several threads can share execution resources through hardware multithreading. In Intel processors, this is implemented as Hyper-Threading, while in NVIDIA’s CUDA programming model, the equivalent abstraction is expressed in terms of thread blocks. At a lower level, each core contains numerous execution units that operate under Single-Instruction Multiple-Data (SIMD) control: the instruction fetch and decode logic issues a single instruction that is applied in parallel to multiple data elements. SIMD offers high efficiency by reducing the cost of complex instruction handling across many units. However, CPU SIMD extensions such as SSE or AVX are relatively inflexible and often difficult to exploit for general-purpose codes. Therefore, GPUs expose SIMD hardware through a more flexible multi-threading abstraction. In CUDA, up to 1024 threads may form a block, with each thread able to execute different instructions and access arbitrary memory locations. In practice, however, the hardware is optimized for groups of 32 threads executing the same instruction on contiguous data. This execution model, known as Single-Instruction Multiple-Thread (SIMT), is a defining feature of GPU architectures and enables efficient large-scale parallelism. Figure 4.1, (adapted from [109]) shows the classical layout of CPU and GPU.

4.4 CPU MRI

The recent advances in MRI algorithms and programs have led to numerous implementation challenges in order to reduce the reconstruction time [110]. To overcome the huge volume of acquisition data, and the computational complexity of such algorithms, CPU and GPU nodes with specific characteristics are required. Many researchers have previously developed efficient solutions with the aim of tackling several MRI reconstruction problems, including non-Cartesian parallel imaging [111, 112], compressed sensing [35], and non-linear reconstruction [110]. Moreover, GPU-based systems generally provide larger raw computational power for the same cost and consumption as CPU-only systems, but fully exploiting this potential can be challenging. This is the case for motion-

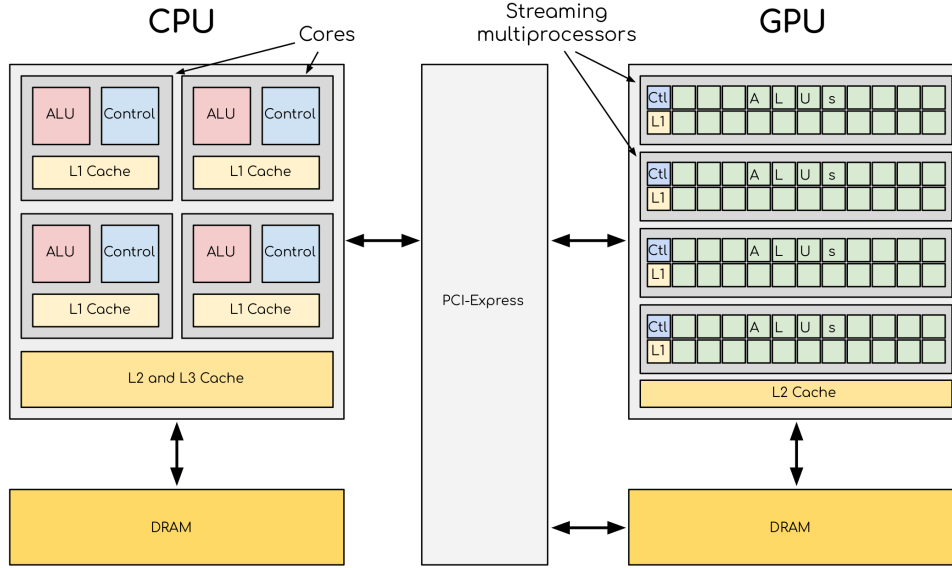


Figure 4.1: Comparison between CPU and GPU architectures. A CPU contains a small number of powerful cores, each containing arithmetic logic units (ALUs), control units, and private L1 caches, complemented by larger shared L2/L3 caches and system DRAM. A GPU integrates many streaming multiprocessors (SMs), each with lightweight control units, ALUs, and L1 caches, organized for massive parallel execution. The GPU relies on high-throughput memory hierarchies (L2 cache and device DRAM). Data exchange between CPU and GPU occurs through the PCI-Express interconnect.

compensated reconstruction, which has higher computational complexity [113, 75, 114], regardless of whether the correction deals with non-rigid motion, as with GRICS [75, 114], or rigid motion, as with DISORDER [69]. Some researchers have worked on accelerating motion-compensated reconstruction; however, only the motion estimation part of the algorithm was modified with a deep learning approach, whereas a classic optimization was still used for the image reconstruction part [115, 116, 117]. This limitation may be due to the challenge of generalizing deep-learning-based approaches over different types of MRI datasets. Huttinga et al. proposed a model-based framework to estimate motion, leveraging a reference image to inform the motion model. The authors suggested an efficient GPU implementation to estimate non-rigid 3D motion in real time (with low latency) from minimal k -space data [118]. However, this method was specifically designed for interventional MRI applications rather than for high-resolution imaging; its main purpose was to estimate 3D motion using a small number of parameters, which makes the optimization problem less demanding than that of a full motion-compensated reconstruction.

Efficient parallelization inevitably leads to large memory overheads; therefore, the total amount of available memory is still a limitation with GPU-based systems for 3D motion-compensated reconstruction in MRI. CPU-only architectures, using multiple cores, also have several advantages in addition to their superior memory space. They are easier to program and are highly portable across architectures and manufacturers (using C or C++ language), unlike GPU-based architectures (e.g., a wide community of developers

use CUDA, a vendor-specific language, rather than the OpenCL standard). P. Chen et al. [119] revisited the role of CPUs in image reconstruction problems, a choice that was motivated by power, low-cost, and space requirement considerations. The authors demonstrated that an ARM CPU outperformed high-end GPUs in the Computed Tomography domain. Roujol et al. conducted a comparative analysis of a real-time reconstruction of an adaptive TSENSE across CPU and GPU hardware [120], highlighting the significant acceleration achievable with GPUs. However, they did not explore the potential benefits of integrating and combining multiple parallelization paradigms on CPUs. Inam et al. introduced a GPU-accelerated Cartesian GRAPPA reconstruction using CUDA [121], which was 17 times faster than their CPU multicore implementation. It is worth noting that the CPU code utilized only eight threads, whereas the CUDA implementation used a considerably larger number of threads. This comparison may not be entirely fair since it compromises the possible benefits of massively parallelizing the application over a large set of CPU cores. Wang et al. conducted a survey of GPU-based acceleration techniques in MRI reconstruction [122], emphasizing one of the distinguishing features of GPUs: their superior memory bandwidth. Nevertheless, it is essential to recognize that recent CPUs are also designed with high memory bandwidth and a considerable number of cores, each capable of executing one or more threads. Indeed, CPU-only solutions may involve different parallelization strategies, including nested loop parallelization, e.g., using MPI and OpenMP, for which many optimizations may be performed without changing the algorithm code. This can be achieved through the investigation of the interaction between the code and micro-architecture, as well as the optimization of specific parameters during both the compilation and execution stages.

4.5 GPU MRI

GPUs are suited for iterative, high-throughput operations, critical to MRI workflows, such as Fourier transforms, image interpolation, and gridding [122, 123, 124]. Unlike CPUs, GPUs are designed for data-level parallelism, enabling the processing of thousands of threads concurrently. Moreover, GPUs facilitate memory-intensive processes such as multi-frame parallel reconstruction, leveraging their superior memory bandwidth and multi-core architecture [125]. In parallel with fast GPU-driven reconstruction, a new trend is emerging toward multi-sensor pipelines that fuse imaging with real-time safety monitoring. Smart electronic device platforms now enable continuous streaming of tissue temperature data and local Specific Absorption Rate (SAR) measurements, which can be co-processed on the same GPU hardware to guide adaptive control of power deposition and reconstruction parameters [126]. Integrating such wearable sensor feedback underscores the broader move toward unified, GPU-centric biomedical systems where imaging, physiology, and patient safety are handled in a single low-latency loop. Compute Unified Device Architecture (CUDA) [127] has gained particular attention in the GPU transition in MRI for its user-friendly programming model, its accessibility, and its high-level libraries. Domain-specific frameworks have recently emerged to accelerate the adoption of GPU computing in the medical imaging field. One promising example is the *Medical Open Network for AI* (MONAI) [128], an open-source library initiated by NVIDIA

and now widely adopted by the community. Built on top of PyTorch, MONAI provides optimized components for data preprocessing, augmentation, and deep learning-based reconstruction workflows.

Traditional reconstructions based on direct Fourier inversion are relatively inexpensive, but modern approaches such as compressed sensing, model-based reconstruction, and dynamic imaging—require redundant evaluations of forward and adjoint operators, iterative solvers, and regularization steps. These operations map naturally to the massively parallel architecture of GPUs, where thousands of cores can process concurrently voxel- or k -space-level computations. For example, conjugate gradient and split-Bregman algorithms have been reformulated for GPU execution, reducing runtimes from hours on CPUs to minutes or even seconds on commodity hardware, thereby making advanced reconstructions clinically feasible [129, 122, 110]. Multi-GPU implementations have further extended these capabilities to real-time MRI, where nonlinear inverse (NLINV) reconstructions now achieve frame rates of 20–30 fps, enabling applications such as cardiac imaging and interventional guidance [100, 110]. Beyond speed, GPUs also offer superior energy efficiency and cost-effectiveness, often allowing a single GPU-equipped workstation to rival or surpass the performance of small CPU clusters. The principal GPU limitation in this context is the restricted memory capacity, which can result in suboptimal host-to-device data transfers.

4.6 Performance Profiling

Performance profiling tools play a major role in understanding the behavior of modern CPU and GPU architectures, yet they vary widely in scope, granularity, and ease of use. Traditional low-level profilers rely on hardware performance counters, accessed through vendor-specific interfaces, to measure events such as cache misses, memory bandwidth utilization, or pipeline stalls. While such metrics are invaluable for diagnosing bottlenecks, they are often limited by counter availability, sampling overhead, and lack of portability across architectures [130]. To overcome these limitations, a new generation of standardized tools has emerged for both CPUs and GPUs.

On the CPU side, profilers include Intel VTune Profiler [131], which provides fine-grained microarchitectural exploration, Linux perf [132] for lightweight kernel-level event tracing, AMD uProf [133] for combined CPU/GPU performance analysis, and Arm Streamline [134] for heterogeneous SoCs. These tools offer insights into instruction throughput, pipeline utilization, cache efficiency, and thread-level scalability. On the GPU side, the ecosystem is centered around NVIDIA’s Nsight Systems [135] for application-level tracing, Nsight Compute for CUDA kernel profiling [136], the CUDA Profiling Tools Interface (CUPTI) [137] for hardware event collection, and the Data Center GPU Manager (DCGM) [138] for large-scale monitoring. These tools provide integrated environments for correlating low-level hardware events with application-level behavior. In the following subsections, we detail the specific tools and methodologies used for GPU profiling and CPU profiling.

4.6.1 GPU Profiling

On the GPU side, NVIDIA Nsight Compute [136] is the state of the art in CUDA kernel profiling. It provides fine-grained measurements at the kernel, warp, and instruction level, enabling developers to analyze issues such as warp divergence, memory coalescing, register pressure, and achieved occupancy. Nsight Compute leverages the CUDA Profiling Tools Interface (CUPTI) to access hardware counters, and integrates with Nsight Systems for application-level tracing. Together, these tools form a comprehensive suite for analyzing GPU performance, but they are limited to NVIDIA hardware and require careful interpretation to translate hardware events into algorithmic insights.

4.6.2 CPU Profiling

For CPUs, we mainly focus on Intel and AMD architectures. Although this work emphasizes these two vendors, it is important to note that ARM-based processors have gained increasing relevance in both mobile SoCs and high-performance computing environments. In ARMv8-A and ARMv9 designs, performance profiling relies on the Performance Monitoring Unit (PMU), which provides low-level event counters such as cycles, executed instructions, branch mispredictions, and last-level cache misses. Moreover, ARM introduces the Statistical Profiling Extension (SPE), a sampling-based mechanism that records statistical data at the instruction level, which enables attribution of performance bottlenecks to code regions and memory access patterns. Instruction tracing is further supported by CoreSight ETM/ETE modules, which allow fine-grained reconstruction of instruction flows and execution paths. At the operating system level, Linux perf and commercial tools such as Arm Streamline or NetSight provide system-wide profiling capabilities, covering both CPU and GPU workloads, context switches, and SoC-level metrics on heterogeneous ARM64 platforms.

For AMD architectures, the tool is AMD uProf [133], which supports both CPUs and GPUs in heterogeneous environments. uProf provides event-based sampling and time-based profiling, offering insights about pipeline utilization, memory hierarchy behavior, and thread-level performance. Its integration with ROCm allows developers to analyze HIP-based CPU/GPU applications. Intel VTune Profiler [131] remains one of the most comprehensive performance analysis environments. VTune supports fine-grained microarchitectural exploration, including hotspots, vectorization efficiency, branch mispredictions, and memory bandwidth utilization, across multi-threaded and heterogeneous workloads. It can be complemented by Intel Software Development Emulator (SDE) [139], which provides instruction-level simulation and enables evaluation of instructions not yet available in hardware, thus allowing forward-looking performance analysis. These tools are mainly used to derive the complete roofline model of an application on a specific hardware platform. This topic will be discussed in detail in Chapters 6 and 7.

4.7 Synopsis

In this chapter, we introduced some pillar concepts of HPC. Algorithms such as GRICS require repeated forward and backward evaluations of the encoding operator, coupled with

large-scale linear algebra operations. Without acceleration, these reconstructions would take hours, which makes them impractical for clinical use. By examining computing laws and computer architecture specificities, it becomes clear that raw computational power alone is not sufficient; optimal parallelization and memory optimization are required to make such frameworks feasible. The next chapter introduces both the mathematical formulation of GRICS and the theory behind the roofline performance model.

Chapter 5

Theory and Mathematical Formalism

5.1 Joint Optimization of image and motion

Let's consider the following optimization problem, which seeks to solve for the motion-corrected image ρ and the motion field $\mathbf{u}$:

$$\hat{\rho}, \hat{\mathbf{u}} = \arg \min_{\rho, \mathbf{u}} \|\mathbf{E}(\mathbf{u})\rho - m\|_2^2 + \lambda \|\rho\|_2^2 + \mu \|\mathbf{S}\mathbf{u}\|_2^2 \quad (5.1)$$

In this formulation, a second regularization term is added to impose a constraint on the displacement field $\mathbf{u}$. This term includes the operator $\mathbf{S}$, which will be defined later depending on the assumptions on motion. In order to reduce the computational complexity of this problem, an approach of motion parameterization is used. If a specific model is chosen for motion, $\mathbf{u}$ can be defined by a number of parameters α . Hence, only this set of parameters should be optimized. For this purpose, two motion models will be considered. The first model will be a spatially uniform displacement field (rigid). The second one assumes spatially varying displacements (non-rigid), under an additional temporal constraint, stating that the temporal variations of the displacement field are expressed as a linear combination of N_k known motion modes. Under these conditions, the joint reconstruction is expressed as a joint optimization problem of the static image $\mathbf{u}$ and the parameters of the motion model α :

$$\hat{\rho}, \hat{\mathbf{u}} = \arg \min_{\rho, \mathbf{u}} \|\mathbf{E}(\mathbf{u}_\alpha)\rho - m\|_2^2 + \lambda \|\rho\|_2^2 + \mu \|\mathbf{S}\alpha\|_2^2 \quad (5.2)$$

For the translational motion model, the regularization controlled by the parameter μ can be neglected, since the number of added parameters is small. An alternative is to set $\mathbf{S}$ to the identity operator. For a temporally constrained non-rigid model, $\mathbf{S}$ can be defined as the spatial gradient operator to enforce spatial smoothness of α and $\mathbf{u}$.

To solve Eq 5.2, it is possible to proceed with an alternating optimization scheme. The concern is that the cost function is non-linear in α . Eq 5.2 presents a non-linear least squares problem. The Gauss-Newton strategy is used to overcome this non-linearity problem. This approach consists of linearizing the cost function around the current estimate of motion. Hence, the non-linear problem turns to a sequence of linear least squares

problems. The linearization process is achieved mathematically with the so-called Jacobian matrix [114]. Considering an initial guess of motion parameters α , the problem can be solved iteratively by

1. α are fixed, we solve the optimization with respect to ρ , using the results of Chapter 2.

$$\hat{\rho} = (\mathbf{E}^H \mathbf{E} + \lambda \mathbf{I})^{-1} \mathbf{E}^H m. \quad (5.3)$$

2. Compute the residual reconstruction error

$$f(\rho, \alpha) = \mathbf{E}(\mathbf{u}_\alpha) \rho - m \quad (5.4)$$

3. Fix ρ , and solve the optimization problem with respect to the motion model parameters.

The process is then repeated until a stopping criterion is reached. The Gauss-Newton linearization is used in step 3. Since the error function f is a non-linear function of the parameters α , the idea is to search for an optimal increment $\delta\alpha$ that minimizes $f(\rho, \alpha + \delta\alpha)$ instead of directly minimizing $f(\rho, \alpha)$. The optimization step for step 3 reads:

$$\delta\hat{\alpha} = \arg \min_{\delta\alpha} \|f(\rho, \alpha + \delta\alpha)\|_2^2 + \mu \|\mathbf{S}(\alpha + \delta\alpha)\|_2^2 \quad (5.5)$$

and using the Jacobian matrix J_α

$$\delta\hat{\alpha} = \arg \min_{\delta\alpha} \|f(\rho, \alpha) + \mathbf{J}_\alpha(\rho, \alpha) \delta\alpha\|_2^2 + \mu \|\mathbf{S}(\alpha + \delta\alpha)\|_2^2. \quad (5.6)$$

The least squares solution reads:

$$\delta\hat{\alpha} \approx (J_\alpha(\rho, \alpha)^H J_\alpha(\rho, \alpha) + \mu \mathbf{S}^H \mathbf{S})^{-1} (J_\alpha(\rho, \alpha)^H (m - \mathbf{E}(\mathbf{u}_\alpha) \rho) - \mu \mathbf{S}^H \mathbf{S} \alpha). \quad (5.7)$$

This approximation is valid for small motion errors [114]. Generally, in order to handle also large displacements, a multiresolution approach is used. Assuming that motion is null, the low resolution k -space data is used to reconstruct a low resolution image, then a low resolution estimate of the parameters α . After each resolution level, the motion model is interpolated to the following resolution level, which gives an appropriate initial guess for the higher resolution estimate.

5.1.1 Rigid motion modeling

5.1.1.1 Case of a translational motion

The motion parameters are defined as the translations occurring at each shot. For the sake of simplicity, let's assume a 2D case (x and y dimensions), the motion parameters α read:

$$\alpha = \left[\tau_1^{(x)}, \tau_2^{(x)}, \dots, \tau_{N_m}^{(x)}, \tau_1^{(y)}, \tau_2^{(y)}, \dots, \tau_{N_m}^{(y)} \right]^T, \quad (5.8)$$

where N_m is the number of shots or motion states. The displacement field $\mathbf{u}_\alpha$ is then expressed using α .

$$\mathbf{u}_\alpha = \left[u_1^{(x)}, u_2^{(x)}, \dots, u_{N_m}^{(x)} \dots u_1^{(y)}, u_2^{(y)}, \dots, u_{N_m}^{(y)} \right]^T = \mathbf{P}\alpha. \quad (5.9)$$

$\mathbf{P}$ is a sparse matrix of size $N_x N_m N_{dims} \times N_m N_{dims}$ that duplicates the scalar transition value at each shot onto each pixel of the corresponding displacement field. $N_{dims} \in \{2, 3\}$.

5.1.1.2 Case of a rotational motion

For the rotational modeling, an approach similar to [140] was used. First, a given rotation by angle θ can be written as three shears:

$$\begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} = \begin{bmatrix} 1 & -\tan(\frac{\theta}{2}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \sin \theta & 1 \end{bmatrix} \begin{bmatrix} 1 & -\tan(\frac{\theta}{2}) \\ 0 & 1 \end{bmatrix}. \quad (5.10)$$

Let $\mathbf{F}_\ell$ denote the 1D DFT along dimension $\ell \in \{1, 2, 3\}$ and $\mathbf{F}_\ell^H$ its Hermitian. Define diagonal modulation matrices $\mathbf{V}_{\ell s}^{\tan}$ and $\mathbf{V}_{\ell s}^{\sin}$. The rotation affecting shot $s \leq N_m$ is

$$\begin{aligned} \mathbf{T}_s^{\text{rot}} &= \mathbf{F}_2^H \mathbf{V}_{1s}^{\tan} \mathbf{F}_2 \mathbf{F}_3^H \mathbf{V}_{1s}^{\sin} \mathbf{F}_3 \mathbf{F}_2^H \mathbf{V}_{1s}^{\tan} \mathbf{F}_2 \\ &\quad \times \mathbf{F}_3^H \mathbf{V}_{2s}^{\tan} \mathbf{F}_3 \mathbf{F}_1^H \mathbf{V}_{2s}^{\sin} \mathbf{F}_1 \mathbf{F}_3^H \mathbf{V}_{2s}^{\tan} \mathbf{F}_3 \\ &\quad \times \mathbf{F}_1^H \mathbf{V}_{3s}^{\tan} \mathbf{F}_1 \mathbf{F}_2^H \mathbf{V}_{3s}^{\sin} \mathbf{F}_2 \mathbf{F}_1^H \mathbf{V}_{3s}^{\tan} \mathbf{F}_1. \end{aligned} \quad (5.11)$$

Let k_ℓ and r_ℓ be the k -space and spatial coordinate vectors along dimension ℓ , and $\circ$ the Hadamard product. For rotation angles $\theta_{\ell s}$ about axes $\ell \in \{1, 2, 3\}$:

$$\begin{aligned} \text{diag}(\mathbf{V}_{1s}^{\tan}) &= \exp\{j \tan(\theta_{1s}/2) k_2 \circ r_3\}, \quad \text{diag}(\mathbf{V}_{1s}^{\sin}) = \exp\{-j \sin(\theta_{1s}) k_3 \circ r_2\}, \\ \text{diag}(\mathbf{V}_{2s}^{\tan}) &= \exp\{j \tan(\theta_{2s}/2) k_3 \circ r_1\}, \quad \text{diag}(\mathbf{V}_{2s}^{\sin}) = \exp\{-j \sin(\theta_{2s}) k_1 \circ r_3\}, \\ \text{diag}(\mathbf{V}_{3s}^{\tan}) &= \exp\{j \tan(\theta_{3s}/2) k_1 \circ r_2\}, \quad \text{diag}(\mathbf{V}_{3s}^{\sin}) = \exp\{-j \sin(\theta_{3s}) k_2 \circ r_1\}. \end{aligned} \quad (5.12)$$

Here, the 3D rotation is not represented as a dense transformation matrix, but as a sequence of three shears. This decomposition, illustrated in Eq. (5.10), is particularly convenient in the Fourier domain because each shear can be implemented using a combination of one-dimensional FFTs and simple diagonal modulation matrices. In practice, this reduces the computational complexity of applying a 3D rotation: instead of an explicit matrix multiplication with cost $\mathcal{O}(N^3)$, the operation can be carried out with FFTs at cost $\mathcal{O}(N \log N)$. The diagonal operators $\mathbf{V}_{\ell s}^{\tan}$, $\mathbf{V}_{\ell s}^{\sin}$ introduced in Eqs. (5.11)–(5.12) encode these shears as phase modulations acting along different spatial axes. Intuitively, they correspond to direction-dependent frequency shifts, which, when applied in sequence across the three dimensions, reproduce the effect of the original rotation matrix.

5.1.2 Non-Rigid motion modeling

In this case, it has been suggested to use a separable motion model of the following form:

$$\mathbf{u}(r, t) = \sum_{k=1}^{N_k} v_k(r) w_k(t), \quad (5.13)$$

where w_k are known motion signals extracted from external sensors or navigator data, and v_k are spatial coefficient maps controlling the motion direction and amplitude [141]. α is then defined as the concatenation of all v_k coefficients (in 2D, for the sake of simplicity)

$$\alpha = \left[v_1^{(x)}, v_2^{(x)}, \dots, v_{N_k}^{(x)} \dots v_1^{(y)}, v_2^{(y)}, \dots, v_{N_k}^{(y)} \right], \quad (5.14)$$

α is a vector of size $N_x N_k N_{dims}$. The displacement field $\mathbf{u}_\alpha$ can be expressed in a compact matrix formalism:

$$\mathbf{u}_\alpha = \mathbf{Q}\alpha, \quad (5.15)$$

where $\mathbf{Q}$ is a sparse matrix of size $N_x N_m N_{dims} \times N_x N_k N_{dims}$, formed with diagonal block matrices, where each diagonal has a constant value given by the motion signals w_k at the actual motion state M_s .

5.2 The Generalized Reconstruction by Inversion of Coupled Systems Method

Now, we introduce the framework that was used in this thesis. Motion-compensated reconstruction was accomplished using GRICS [75, 114]. The algorithm solves for both a motion-corrected image ρ and the parameters of a motion model α . The GRICS problem is mathematically formulated as follows:

$$\begin{cases} \rho, \alpha = \arg \min_{\rho, \alpha} \|E(u_\alpha)\rho - m\|^2 + \lambda\|\rho\|^2 + \mu R(\alpha), \\ u_\alpha(x, y, z, t) = \begin{cases} [x, y, z, 1]\alpha(t) \text{ (rigid model), or} \\ \alpha(x, y, z)s(t) \text{ (non-rigid model)} \end{cases} \end{cases} \quad (5.16)$$

m stands for the acquired MRI k -space data; m is a vector of size N_m . ρ is the motion-corrected image, a vector of N_ρ elements. u is the displacement field at each k -space sample time; it is a vector of $N_m N_\rho N_d$ elements, where N_d is the number of dimensions in the image ($N_d \in \{2, 3\}$). In the case of a rigid motion model (e.g., for cerebral imaging), $\alpha(t)$ is an affine transformation matrix using homogeneous coordinates. In the case of a non-rigid motion model (e.g., for chest or abdominal imaging), a separable motion model is used, with a spatial component $\alpha(x, y, z)$, and the temporal component $s(t)$ is a motion signal, acquired from a motion sensor (here, a pneumatic respiratory belt). A regularization term is added in the non-rigid case as $R(\alpha) = \|\nabla\alpha\|^2$. E is an encoding operator; it is expressed using the following operators:

$$E = \begin{bmatrix} S_{t_1} F C_1 T(u_{t_1}) \\ \vdots \\ S_{t_N} F C_1 T(u_{t_N}) \\ \vdots \\ S_{t_1} F C_{N_c} T(u_{t_1}) \\ \vdots \\ S_{t_N} F C_{N_c} T(u_{t_N}) \end{bmatrix}$$

The acquisition is subdivided into N_t motion states (shots) and N_c receiver coils. For a given $i \leq N_t$, each motion state t_i is described by a dense displacement vector field u_{t_i} and a sparse spatial transformation matrix $T(u_{t_i})$. The size of $T(u_{t_i})$ is $N_\rho \times N_\rho$. The matrices S_{t_i} describe the sampling pattern of the k -space data during the i^{th} motion state. In the Cartesian sampling case, S_{t_i} represents sparse matrices, having 0 and 1 as elements. The matrix S_{t_i} has N_ρ columns, and its number of rows is the number of k -space samples acquired at each shot t_i . F stands for the Fourier transform operator. $C_1, C_2, \dots, C_{N_c}$ denote the diagonal coil sensitivity operators associated with the receiver coils. For each $j \leq N_c$, the size of the matrices C_j and operator F is $N_\rho \times N_\rho$. In GRICS, the Fast Fourier Transform of a vector is computed using the FFTW implementation.

The two least squares problems in Algorithm 4 are solved using the conjugate gradient solver. The Gauss–Newton scheme is introduced to linearize the cost function, line 8 around the current estimate of α ; each iteration suggests a refinement $\delta\alpha$ of α . The tolerances are $\epsilon_r = 10^{-3}$ for the reconstruction solver and $\epsilon_m = 10^{-2}$ for the motion problem. In the last resolution level, only the reconstruction step is performed.

Algorithm 4 Joint reconstruction with GRICS.

- 1: **Input:** m the k -space data, and s the motion sensor signals
 - 2: **Output:** The image ρ and the motion model α
 - 3: Initialization of the motion model: $\alpha = 0$
 - 4: **for** resolution level = coarse to fine **do**
 - 5: **for** Gauss–Newton iteration = 1, 2, $\dots$, N **do**
 - 6: $\min_{\rho} ||E(u_\alpha)\rho - m||^2 + \lambda||\rho||^2$
 - 7: $\epsilon = m - E(u_\alpha)\rho$ $\triangleright \epsilon$ is the residual term
 - 8: $\min_{\delta\alpha} ||E(u_{\alpha+\delta\alpha})\rho - m||^2 + \mu R(\alpha + \delta\alpha)$
 - 9: $= \min_{\delta\alpha} ||J(\alpha, s, \rho)\delta\alpha - \epsilon||^2 + \mu R(\alpha + \delta\alpha)$ $\triangleright J$ is the Jacobian matrix
 - 10: $\alpha = \alpha + \delta\alpha$
 - 11: **end for**
 - 12: **end for**
-

The most computationally intensive part of Algorithm 4 lies in the conjugate gradient linear solvers. In particular, the application of the operator $E^H E$, where E^H is the Hermitian transpose of the operator E -, comprises an outer loop on motion states and

an inner loop on coil receivers. The operator $E^H E$ reads

$$E^H E = \sum_{t=1}^{N_t} T_t^H \left(\sum_{c=1}^{N_c} C_c^H F^H S_t^H S_t F C_c \right) T_t \quad (5.17)$$

The reconstruction kernel will be discussed in detail in Chapter 6. For the motion model problem, we have a similar intense computational part with the application of the operator $J^H J$, where J is the Jacobian matrix.

5.3 The Roofline Model

Performance modeling in HPC seeks to characterize the interaction between algorithmic demands and hardware capabilities. Among the most widely adopted profiling tools is the *Roofline model*, introduced by Williams et al. in 2009 [96]. The Roofline model provides an intuitive and rigorous framework to quantify the attainable performance of computational kernels on modern CPU and GPU architectures. It does so by simultaneously considering the peak floating-point throughput and the sustainable memory bandwidth of the target computing system. By coupling these architectural ceilings with the *operational intensity* of an algorithm, the Roofline model offers a visual tool to identify bottlenecks, classify workloads as either memory-bound or compute-bound, and to guide optimization alternatives in a specific direction.

5.3.1 Mathematical Formulation

The roofline model relies on three main concepts:

- The peak performance P_m of the CPU/GPU, which can be defined as the maximum speed at which all the components of a machine can operate [?]. In a considerable number of application codes, this limit is never reached. In this work, P_m is expressed in terms of GFLOP/s.
- The main memory bandwidth B_s , which is the rate at which data can be transferred from memory to CPU/GPU [142]. It is expressed in GB/s. In several scientific programs and modeling applications, DRAM bandwidth is considered the main bottleneck. Memory specifications refer to B_s that a certain memory chip can achieve. However, this theoretical bandwidth requires perfect circumstances and is not attainable in reality [143].
- The operational intensity I of a program; this metric characterizes the code itself, while the machine is characterized by P_m and B_s . I is defined as the ratio between the number of operations performed by a given kernel and the number of bytes of memory transferred from a memory level to the computing unit. In our case, the operations refer to Floating Point additions and multiplications. Hence, the operational intensity is expressed in FLOP/byte.

The graph of the roofline model is on a log-log scale, the X-axis is FLOP/byte, and the Y-axis is GFLOP/s. The peak Floating Point performance P_m of the machine is expressed as a horizontal line or ceiling. The peak memory performance is considered as the maximum FLOP performance that the memory can provide for a specific value of the operational intensity I . The peak memory performance is represented by a line at a 45-degree angle. The two ceilings intersect at the point of peak FLOP performance and peak DRAM bandwidth [96]. They also provide what we call the performance limits by setting upper bounds to the attainable performance of a kernel; see Figure 5.1. Hence, the performance P verifies the following inequality.

$$P \leq \min(P_m, I \times B_s). \quad (5.18)$$

The architecture efficiency e_i of an application a running on an architecture i is provided by

$$e_i(a) = \frac{P}{\min(P_m, I \times B_s)}. \quad (5.19)$$

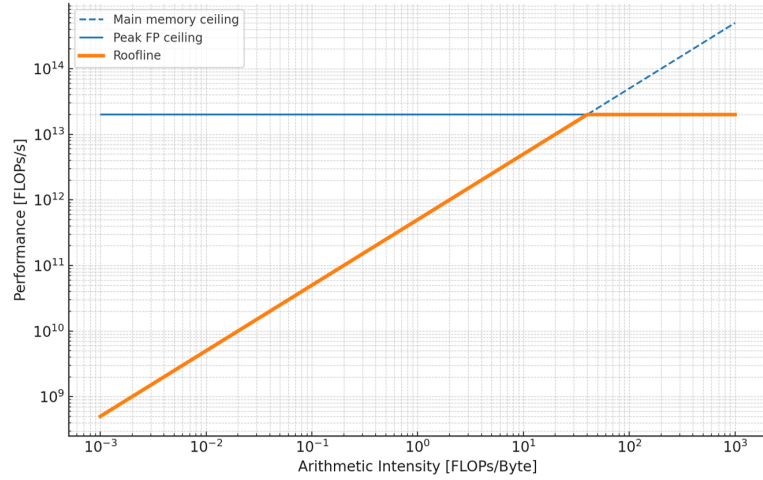


Figure 5.1: The basic roofline model

5.3.2 The basic roofline model Extensions

While the canonical Roofline model provides valuable insight, several extensions have been proposed to address more complicated performance issues, such as cache behavior, cache reuse, and vectorization. First, the **communication-aware Roofline model** incorporates inter-node communication costs in distributed-memory environments, where network bandwidth and latency impose additional ceilings on performance [144]. Second, the **energy-aware Roofline model** extends the framework by coupling performance with power and energy metrics, thereby enabling trade-offs between computational efficiency and energy consumption [97]. Third, the **hierarchical Roofline model** has been developed for heterogeneous platforms such as GPUs and FPGAs, in which multiple memory layers (registers, caches, off-chip DRAM) must be considered [98].

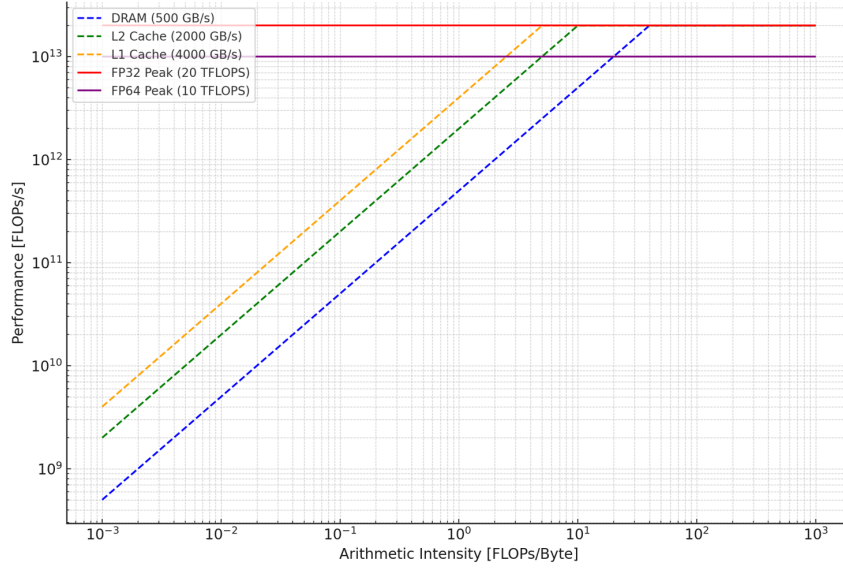


Figure 5.2: Example of the hierarchical roofline model.

5.4 Synopsis

In this chapter, the GRICS method has been presented in detail, with both rigid and non-rigid motion models integration. This formalism provided the mathematical framework needed to describe how physiological signals and deformation fields can be integrated into the reconstruction process. The roofline model was introduced as a theoretical performance tool to quantify computational limits and bottlenecks, guiding where optimizations should focus. This interaction between motion modeling and computational investigation forms the pillars of the thesis.

Chapter 6

CPU/ Matlab-GPU Parallel Implementations of GRICS

6.1 Parallel Programming

The design of a parallel program for a given algorithm is strongly correlated to the parallel computing system, with all its hardware and software components. The software performance is mainly controlled by the complex interaction between the compiled code and the micro-architecture of the computing platform, while the hardware architectural aspects are mostly related to the computing units (CPU, GPU, FPGA), the memory hierarchy, and the system interconnect. Highly efficient implementations can typically be achieved when the micro-architecture of the computing platform is taken into account during the development process, i.e., through careful consideration of software–hardware interactions. Many problems in scientific computing, such as MRI reconstruction, involve the processing of large quantities of data stored on a compute node. In the case where manipulation can be performed in parallel by multiple processors working on different parts of the data, programmers can investigate a parallel implementation of their codes. A parallel program runs on a parallel environment in which each processor executes one or more independent control flows. Depending on how these flows are coordinated, they are called processes or threads. A thread is a lightweight generalization of a process: multiple threads within the same process share a single address space, whereas each process has its own private address space. Choosing between processes and threads depends largely on the system’s memory architecture—processes map naturally to distributed-memory systems, while threads are the preferred abstraction on shared-memory machines, including recent multicore processors. Processes or threads in a parallel program can be created in a static manner, at the start-up of the program, or dynamically, whenever the workload requires additional parallelism. The execution and synchronization mechanisms of a given programming model determine whether these concurrent entities form a parent-child hierarchy or remain as independent peers. Some models permit (or even encourage) a fixed affinity mapping that pins each process or thread to a specific core (thread affinity); when such a mapping is enforced, migration to another core or processor is disabled during the execution of the program. In this work, three parallel programming models were

investigated: MPI and OpenMP on CPU systems, and CUDA on GPU.

6.1.1 Open Multi Processing parallelization

Shared memory offers the possibility to have immediate access to all data from all processors without explicit communication. Open Multi Processing is a de facto standard solution to perform parallelization on shared memory systems. OpenMP is a set of compiler directives that a non-OpenMP-capable compiler would just consider as comments and ignore. The central entity in an OpenMP program is not a process but a thread. Spawning a thread is much less costly than forking a new process, because a thread's instruction pointer, stack pointer, and register state are not shared among a group of threads. In a given OpenMP program, the master thread runs immediately after startup; this is the sequential part of the execution. Parallel execution happens inside parallel regions, which are delimited by OpenMP pragmas. Between two parallel regions, no thread except the master thread executes any code. This is called the “fork-join model”. Inside the parallel region, a team of threads executes instruction streams concurrently, see Figure 6.1

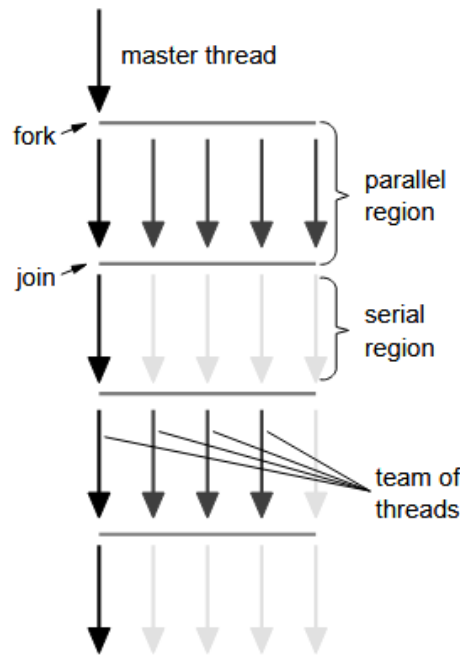


Figure 6.1: OpenMP follows a fork–join execution model. At the start of a parallel region, the master thread forks a team of worker threads; each thread executes a subset of the region’s code while sharing the same address space. When the region ends, an implicit barrier forces all threads to synchronize, after which they join back into a single thread context. The worker threads are then either terminated or parked by the runtime until the next parallel region begins. Because OpenMP allows the application, environment variables, or the runtime to pick the team size dynamically, the number of active threads can change from one parallel region to the next, letting the program adapt to workload and hardware availability.

Next, we present a simple example of adding how an addition kernel of two arrays a and b of the same length N can be parallelized with OpenMP by adding the pragma `#omp parallel` before the loop.

```

1 #pragma omp parallel for
2 for (int i = 0; i < N; ++i)
3     c[i] = a[i] + b[i];

```

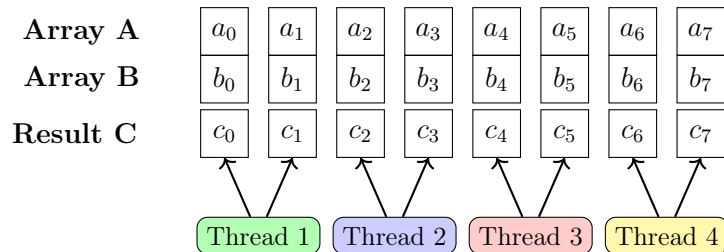


Figure 6.2: OpenMP parallelization: multiple threads compute portions of the output array by directly accessing shared memory.

When the compiler encounters `#pragma omp parallel for`, it creates a team of threads at runtime, each of which takes a portion of the loop’s iteration amplitude and executes it independently. OpenMP’s default chunking algorithm divides the iteration range into roughly equal-sized blocks so that each thread gets about the same workload; hence, the cores remain busy and idle time is minimized. a , b and c are shared among all threads, and each thread writes into a distinct slice of $c[]$ while reading the corresponding elements from $a[]$ and $b[]$. No synchronization is required as long as two threads never touch the same index. The loop counter i , on the other hand, is automatically made private, so each thread has its own copy and race conditions are avoided. After every thread finishes its assigned iterations, the implicit barrier at the end of the parallel region ensures all additions are complete before execution proceeds.

6.1.2 Message Passing Interface parallelization

Message Passing is required if a parallel computer is of the distributed-memory type, which means that one processor can’t directly access the address space of another process. It can also be considered as a programming model and employed on shared-memory or hybrid systems as well. Message Passing Interface, MPI, which is the de facto standard for the Message Passing approach, operates with a specific set of rules: In the MPI programming model, every process executes the same program—the Single-Program Multiple-Data (SPMD) paradigm—although modern MPI launchers can also start different executables on selected ranks, thereby accommodating the more general Multiple-Program Multiple-Data (MPMD) style when required. Each running instance is identified by a unique rank, which the program can query to decide which portion of the workload to perform. The source code itself is typically written in a sequential language such as Fortran, C, or C++; explicit library calls handle all communication, such as the sending and receiving of messages. Moreover, memory is private to each process: variables reside only within

the address space of the rank that created them, and there is no implicit shared-memory region. Cooperation, therefore, relies entirely on explicit message exchanges managed with the MPI library, see Figure 6.3

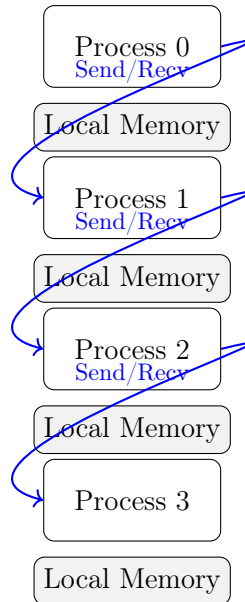


Figure 6.3: Illustration of MPI parallelization: each process operates independently with its own local memory. Communication is explicit and takes place via message passing (Send/Recv) between processes.

Next, a synthetic example of how the addition of two 1D arrays can be performed with MPI is presented.

Listing 6.1: MPI addition of two 1-D arrays

```

1 MPI_Init(&argc,&argv);
2
3
4 int rank, size;
5 MPI_Comm_rank(MPI_COMM_WORLD, &rank);
6 MPI_Comm_size(MPI_COMM_WORLD, &size);
7
8 /* each rank handles a contiguous slice */
9 int start = rank * (N/size);
10 int end   = (rank+1 == size) ? N : start + N/size;
11
12 for (int i = start; i < end; ++i)
13     c[i] = a[i] + b[i];
14
15 MPI_Finalize();

```

After all processes enter the program with ‘MPI_Init’, each one has access to two constants: size, the total number of MPI ranks, and its own rank within that team.

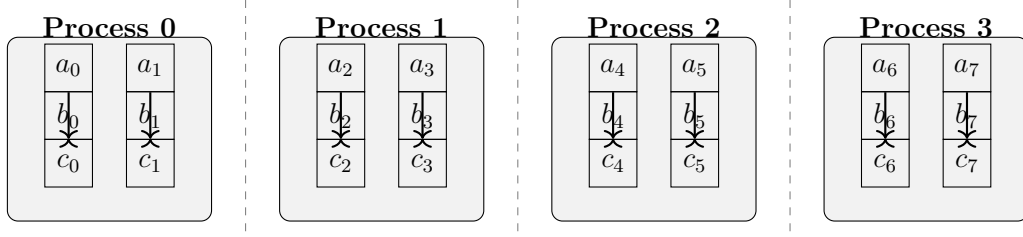


Figure 6.4: MPI parallelization: each process works independently on its local data, with results optionally gathered after computation. No shared memory exists between processes.

With this information, every process computes a unique, contiguous slice of the input arrays: the starting index is $\text{rank} \times (N/\text{size})$, while the ending index is either that start plus N/size or, for the last rank, the full length N (so any remainder elements are covered). Inside the for-loop, each process independently performs the element-wise addition $a[i] = b[i] + c[i]$ for the indices in its slice. Because every slice is disjoint, no two ranks touch the same array positions, eliminating the need for synchronization or data exchange during the computation. Once the local loop finishes, the work for that rank is done, and the program calls `MPI_Finalize`, leaving the partial results distributed across processes; an optional gather could be added later if a single copy of the resulting array is required.

6.2 Overview of the Reconstruction Kernel - RK

The most computationally-intensive part of GRICS lies in the Conjugate Gradient (CG) linear solvers. In particular, the application of the operator $E^H E$, where E^H is the Hermitian transpose of the operator E , comprises an outer loop on motion states N_t and an inner loop on coil receivers N_c . Algorithm 5 introduces the structure of this kernel. The operator $E^H E$ reads

$$E^H E = \sum_{t=1}^{N_t} T_t^H \left(\sum_{c=1}^{N_c} C_c^H F^H S_t^H S_t F C_c \right) T_t \quad (6.1)$$

6.3 Parallel Implementations of the RK

6.3.1 Pure OpenMP Implementation

In the pure OpenMP version of GRICS, the N_c coils are distributed over OpenMP threads by adding pragmas before the loop over coils, line 6 in Algorithm 5. The code is linked to threaded and serial FFTW-3.3.3 package [145], and to the Blitz-0.9 library [146]. There is no nested OpenMP parallelism in GRICS so far. The version of OpenMP is 4.5.2.

Algorithm 5 Main part of the reconstruction kernel (function $x \mapsto E^H E x$).

```

1:  $y = 0$ 
2: for  $t = 1, 2, \dots, N_t$  do
3:    $x_t = T(u_t)x$ 
4:    $y_t = 0$ 
5:   for coil  $c = 1, 2, \dots, N_c$  do
6:      $y_t = y_t + C_c^H (F^H S_t^H S_t F) C_c x_t$ 
7:   end for
8:    $y = y + T(u_t)^H y_t$ 
9: end for

```

6.3.2 Hybrid MPI & OpenMP Implementation

In the hybrid MPI and OpenMP version of the GRICS, the N_c coils are distributed over OpenMP threads, and the N_t motion states are distributed using the MPI, line 3 in Algorithm 5. Each MPI process represents an independent computational entity, operating on its allocated subset of motion states N_t . The MPI guarantees inter-process communication and synchronizes computations among distributed processes or ranks. This distributed approach enables efficient workload allocation across the compute node. The OpenMP distribution is created explicitly by adding specific pragmas before the loop over coils. The code is linked to threaded and serial FFTW-3.3.3 package [145], and to the Blitz-0.9 library [146]. The version of the MPI is 4.0.3.

6.3.3 MATLAB–GPU Implementation

In order to draw a comparison with CPU performance, we suggested a preliminary MATLAB–GPU implementation of the reconstruction kernel—the application of the operator $E^H E$, step 6 of Algorithm , with iterative calls to Algorithm 5. For this purpose, the MATLAB *gpuArray* object is used; it enables the storage of arrays on GPU memory, which allows the programmer to run a kernel on GPU with minimal software modifications. In particular, operators C, S, F of Algorithm 5 were declared as *gpuArray*. The spatial transformation operator T_t was the only operator that needed to be transferred from CPU to GPU memory at the beginning of each outer loop since their storage for all motion states $t = 1, 2, \dots, N_t$ on the GPU device memory resulted in an out-of-memory error on Quadro RTX 5000, which was not the case on A100. In MATLAB, the *gpuArray* object supports CUDA, allowing MATLAB to efficiently harness the parallel processing capabilities of GPUs. By using CUDA, MATLAB can offload computationally intensive tasks to the GPU, resulting in speedups for certain types of calculations, particularly those involving large datasets. Additionally, within the framework of the MATLAB Parallel Computing Toolbox, the MAGMA library [147] is used to enhance the performance and accuracy of linear algebra functions designed for *gpuArray* objects. Furthermore, MATLAB’s execution model incorporates Just-In-Time (JIT) compilation, where code is compiled dynamically during runtime, optimizing performance by converting high-level MATLAB code into machine code. Although this approach is not as efficient as a full

native GPU implementation, this JIT compilation process allows MATLAB to apply native GPU libraries to data stored in GPU memory, eliminating unnecessary data transfers between CPU memory and GPU memory. Additionally, we provided as supplementary material a benchmarking of 3D FFT, which is at the core of all iterative MRI reconstruction programs. This provides an estimate of the upper bound of performance that could be achieved if the whole problem could fit in GPU memory.

6.4 Methods

6.4.1 Data Acquisition

The MRI raw data were acquired on a 3T MR scanner (MAGNETOM Prisma, Siemens Healthineers, Erlangen, Germany). Four healthy volunteers were scanned, resulting in four in vivo datasets. Three datasets (D_1, D_2, D_3) with non-rigid motion were acquired with a T1-weighted 3D GRE (fast low angle shot gradient echo) sequence, covering the breast region. The breast MRI acquisition was performed during free breathing, with the subject lying in the supine position. Further details about the imaging protocol can be found in [148]. A dataset D_4 with subtle rigid motion was acquired using a T1-weighted 3D MPRAGE (Magnetization-Prepared Rapid Gradient Echo) sequence covering the head. The study was conducted in accordance with the ethical principles outlined in the Declaration of Helsinki (1975, revised 2013) and was approved by the ethics committee (approval number: [CPP EST-III, 08.10.01]) under the protocol “METHODODO” (ClinicalTrials.gov Identifier: NCT02887053). All participants provided written informed consent prior to their inclusion in the study.

The main parameters of the dataset of GRICS are the number of samples in frequency encoding direction, N_x , in phase-encoding direction in the image plane, N_y , and in the direction of slices, N_z . The size of the matrix is $N_x \times N_y \times N_z$. The number of coils in the coil array is N_c (“Body 18” and “Spine” coil arrays from the manufacturer; “Head 20” for brain MRI data); the number of shots or motion states is N_t . In this work, the acquisition type is 3D.

The datasets were carefully chosen by selecting different cases with varying coil numbers and dimensions. The study aims to comprehensively evaluate and understand the impact of these variables on the performance. Specifically, the datasets include different combinations of N_x , N_y , N_z , N_c , and N_t , reflecting real-world scenarios with distinct spatial resolutions and complexities; see Table 6.1

6.4.2 Performance Analysis

6.4.2.1 The test Machines

GRICS was executed on two CPU architectures, Intel Xeon Gold 5220 and AMD EPYC 75F3 processors. The CPUs have a sensibly similar architecture layout: two NUMA nodes and two sockets; the cores of each socket have private L1 and L2 level caches and share the L3 level cache; see Figure 6.5 and Table 6.2.

Table 6.1: The four test datasets.

Dataset	D_1	D_2	D_3	D_4
N_x	320	320	320	256
N_y	470	550	470	254
N_z	112	112	112	80
N_c	20	26	30	20
N_t	12	12	12	16

Table 6.2: The characteristics of the CPUs.

CPU Name	Intel Xeon Gold	AMD EPYC
Total Threads	72	128
Total Cores	36	64
Total Sockets	2	2
NUMA nodes	2	2
Frequency in GHz	2.2	2.95
Number of memory channels	6	8
DRAM type	DDR4	DDR4

Moreover, two GPU architectures are considered, NVIDIA Quadro RTX 5000 and NVIDIA A100. Sketches and characteristics of the two GPUs are presented in Figures 6.6 and 6.7 and Table 6.3.

6.4.2.2 The Empirical Roofline Toolkit

In order to generate the roofline model, theoretical values of peak performance and DRAM/HBM bandwidth could be used, but they will not provide realistic modeling of the architecture-code interaction. In this work, we used the Empirical Roofline Toolkit ERT-1.1.0; the tool generates the required characterization of a machine empirically. It does this by running specific kernel codes on the test machine so that the results are attainable in reality by some codes. The ERT generates the bandwidth of the DRAM/HBM and cache levels and estimates the empirical peak performance of the CPU/GPU device. The input of the ERT is a configuration file where the user mentions machine specification data, and the final output is the roofline chart in Postscript format and the roofline parameters in JSON format. The ERT is employed to gather the empirical values of memory bandwidth and peak performance. Then, we used the NERSC-Roofline repository [149] to generate the final plot with GRICS information in terms of operational intensities and achieved performance.

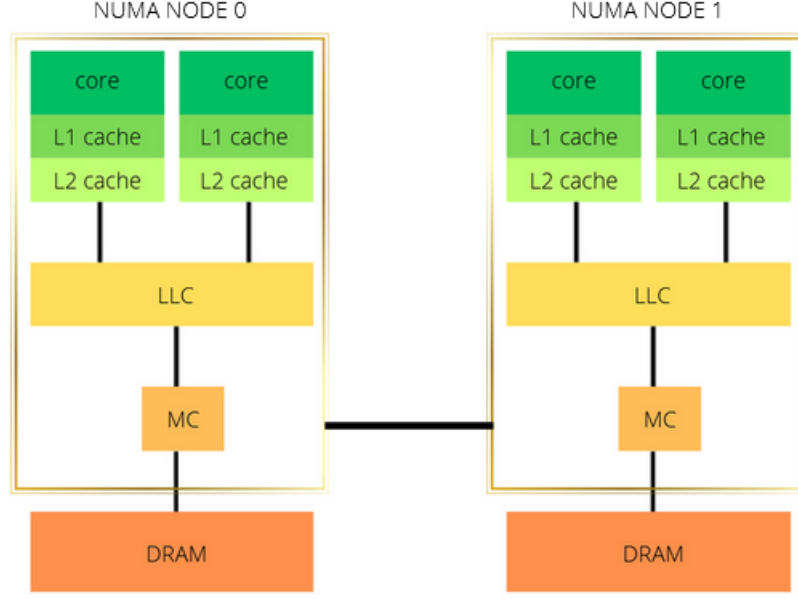


Figure 6.5: Modeling of a two-socket, two-NUMA-node architecture. LLC stands for the last-level cache, which is shared between all cores of the same socket/NUMA node. MC is the memory controller, and DRAM is the main memory.

6.4.2.3 Application Characterization

On HPC systems, the programmer must deal with the complexity of the hardware, the different parallelization paradigms, network topology, and software builds. Hence, it is important to quantify the application's performance, using performance profiling tools, to see whether it is possible or not to obtain a better performance on the same system. These tools generate different performance metrics, identify hotspots for code, and help in the determination of adequate optimization techniques. We present the tools used to profile the code on the considered architectures:

- Intel Vtune and Intel Software Development Emulator

Intel Vtune-19.0.4 [131] and Intel SDE-9.14 [139] are used to profile the code on Intel architecture. In particular, the SDE tool can be employed to collect the number of FLOPs and the L1 data movement. Vtune is used to capture data traffic in DRAM and cache levels [150]; the total bytes are read and written in each memory level. Then, the achieved performance on the architecture i is $P_i = \text{SDE FLOPs}/T$, where T is the application's elapsed time. The operational intensity of the code regarding a specific memory level m_l is $I_{m_l} = \text{SDE FLOPs}/\text{Vtune_Bytes_}m_l$, where $\text{Vtune_Bytes_}m_l$ is the number of Vtune memory accesses (read and write) for the memory level m_l .

- AMD μ Prof

AMD μ Prof-4.1 [133] is used to characterize the performance bottlenecks in the source code and to identify ways to optimize it on AMD architecture. The tool integrates

Table 6.3: The characteristics of the GPUs. SM stands for Streaming Multiprocessor. SMs are akin to CPU cores.

GPU Name	Quadro RTX 5000	A100
SMs	48	108
Tensor Cores	384	432
Unified L1/SMEM per SM	64 kB	192 kB
L2 Cache Size	4 MB	40 MB
Memory Size	16 GB	40 GB
Memory type	GDDR6	HBM2e
Driver Version	550.78	535.113.01
CUDA Version	12.4	12.2

a roofline support in order to generate the roofline data and the roofline plot. The achieved performance on the architecture P_i and the operational intensity of the DRAM are reported in the roofline support, while for a memory level m_l we have $I_{m_l} = \mu\text{Prof FLOPs}/\mu\text{Prof_Bytes_}m_l$, where $\mu\text{Prof_Bytes_}m_l$ is the number of μProf memory accesses (read and write) for the memory level m_l .

- NVIDIA Nsight Compute

Nsight Compute [136] is employed to collect performance data on Nvidia GPUs. It provides a set of metrics to evaluate in order to compute the number of FLOPs and data movement. We summarize the metrics for roofline data collection in Table 6.4.

Table 6.4: NVIDIA Nsight compute metrics. FP stands for Floating Point. In the following, M_{prec} will stand for `sm__sass_thread_inst_executed_op_fp64_pred_on.sum`, where $prec \in \{64, 32, 16\}$, M_{TC} stands for `sm__inst_executed_pipe_tensor.sum`, and M_{m_l} for the m_l memory level metric, where $m_l \in \{L1, L2, HBM\}$.

Data	Metric
FP 64	<code>sm__sass_thread_inst_executed_op_fp64_pred_on.sum</code>
FP 32	<code>sm__sass_thread_inst_executed_op_fp32_pred_on.sum</code>
FP 16	<code>sm__sass_thread_inst_executed_op_fp16_pred_on.sum</code>
Tensor Core	<code>sm__inst_executed_pipe_tensor.sum</code>
L1 cache	<code>l1tex__t_bytes.sum</code>
L2 cache	<code>lts__t_bytes.sum</code>
HBM	<code>dram__bytes.sum</code>

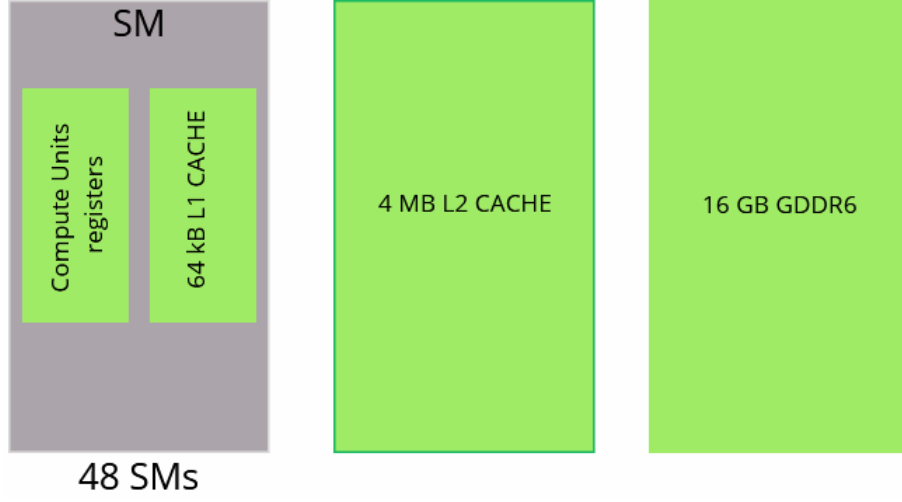


Figure 6.6: Modeling of NVIDIA Quadro RTX 5000 GPU.

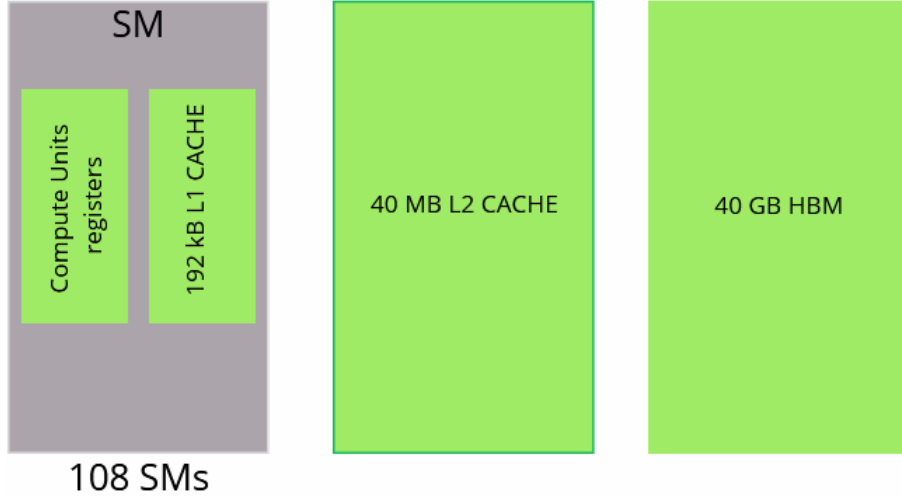


Figure 6.7: Modeling of NVIDIA A100 GPU.

The total number of FLOPs = $M_{64} + M_{32} + M_{16} + 512 \times M_{TC}$, the achieved performance P_i on GPU architecture i is $P_i = \text{FLOPs}/T$, where T is the kernel runtime, and the operational intensity I_{m_l} of memory level $m_l \in \{L1, L2, HBM\}$ is $I_{m_l} = \text{FLOPs}/M_{m_l}$. The operational intensities of each memory level are computed using the output of these metrics, in particular:

$$OI_{L_1} = \frac{\text{FLOPs}}{L_1 \text{ cache}}, \quad OI_{L_2} = \frac{\text{FLOPs}}{L_2 \text{ cache}}, \quad OI_{m_{HBM}} = \frac{\text{FLOPs}}{\text{HBM}}.$$

Giving that T is the elapsed time of the CUDA-based implementation, its achievable performance P is computed as

$$P = \frac{\text{FLOPs}}{T}$$

6.4.2.4 Memory Footprint

Using Table 6.5, we define M_{RK} , the total memory footprint of the reconstruction kernel.

Table 6.5: Memory footprint of Algorithm 5. $M = \max(\text{NNZ}(S_{t_i}))$ stands for the maximum number of non-zero elements per S_{t_i} row. The elements of operator T are pre-computed. N_{ps} stands for the number of physiological signals used to derive the spatial transformation matrix during acquisition. s_f refers to the size of the elements. In GRICS, we work with single-precision float, so $s_f = 4$ bytes (32 bits). FFT is conducted using FFTW implementation, so we do not need explicit storage of the FFT operator F .

Operator/Variable	Size	Memory Footprint
x	N_ρ	$N_\rho \times s_f$
x_t	N_ρ	$N_\rho \times s_f$
y_t	N_ρ	$N_\rho \times s_f$
y	N_ρ	$N_\rho \times s_f$
$T(u_{t_i})$	$N_\rho \times N_\rho$ sparse	$2 \times N_t \times N_\rho \times N_{ps} \times s_f$
S_{t_i}	$N_k \times N_\rho$ sparse	$N_t \times N_\rho \times M \times s_f$
C_c	$N_\rho \times N_\rho$ diagonal	$N_c \times N_\rho \times s_f$

$$M_{RK} = N_\rho(4 + N_c + N_t(2N_{ps} + M))s_f. \quad (6.2)$$

In terms of the dimensions N_x , N_y , and N_z , M_{RK} reads as follows:

$$M_{RK} = N_x N_y N_z (4 + N_c + N_t(2N_{ps} + M))s_f. \quad (6.3)$$

Using Equation 6.3, none of the 4 datasets of Table 6.1 can entirely fit in the cache levels of the studied CPUs/GPUs.

6.4.2.5 Computational and Communication Complexities

In this section, we present an analysis of the computational complexity of the reconstruction kernel (Algorithm 5) to identify the reconstruction parameters that contribute most significantly to the computational costs. Let us break the application of the operator $E^H E$ into simple multiplication steps:

- Applying $T(u_{t_i})$: Multiplication of a vector by a sparse matrix of size $N_\rho \times N_\rho$ has a complexity of $\mathcal{O}(\gamma_1 N_\rho)$, where γ_1 is the size of the interpolation kernel (in our case, we use a linear interpolation kernel $\gamma_1 = 8$).
- Applying $T(u_{t_i})^H$: Same as the application of $T(u_{t_i})$.
- C_j and its C_j^H are diagonal matrices, each with a complexity of $\mathcal{O}(N_\rho)$.

- FFT and its inverse are performed using FFTW implementation, each with a complexity of $\mathcal{O}(N_\rho \log N_\rho)$.
- Assuming that the maximal number of non-zero elements per row in the matrices S_{t_i} and $S_{t_i}^H$ is k , where $k \ll N_\rho$, the complexity of applying S_{t_i} and $S_{t_i}^H$ is $\mathcal{O}(kN_\rho)$.

The dominant term for each inner loop of Algorithm 5 is $\mathcal{O}(N_\rho \log N_\rho)$. Hence, each outer loop iteration of Algorithm 5 requires $\mathcal{O}(N_C N_\rho \log N_\rho)$. Finally, the overall complexity is provided by

$$C = \mathcal{O}(N_t N_C N_\rho \log N_\rho). \quad (6.4)$$

C can be expressed using the number of k -space samples in frequency and phase-encoding directions, N_x , N_y , and N_z (see Table 6.1) since $N_\rho = N_x N_y N_z$.

$$C = \mathcal{O}(N_t N_C N_x N_y N_z \log(N_x N_y N_z)). \quad (6.5)$$

In our case, the FFT is performed only on the phase-encoding directions; hence, C reads as follows:

$$C = \mathcal{O}(N_t N_C N_x N_y N_z \log(N_y N_z)). \quad (6.6)$$

Using similar reasoning, we can compute the memory complexity (the amount of memory a program requires to execute relative to the size of the input data) C_m too; C_m reads as follows:

$$C_m = \mathcal{O}(N_t N_C N_x N_y N_z). \quad (6.7)$$

On the other hand, and to understand the performance of the MPI, we suggest a process communication analysis. First, each process receives its share of the data for different motion states. The size of data communicated here depends on the number of motion states N_t and the size of the data N_ρ . The initial distribution of x (line 4, Algorithm 5) to all processes requires sending $N_\rho \times P$ elements. Hence, the communication complexity of the data distribution is $\mathcal{O}(N_\rho P)$ per process. Then, after computing partial results, each process will send its local y (line 9 in Algorithm 5) back to the root process. The communication complexity per process for gathering the vector y is $\mathcal{O}(N_\rho P)$. Summing these contributions, we derive the overall communication complexity C_c .

$$C_c = \mathcal{O}(N_\rho P). \quad (6.8)$$

In terms of the dimensions N_x , N_y , and N_z , C_c reads as follows:

$$C_c = \mathcal{O}(N_x N_y N_z P). \quad (6.9)$$

6.4.3 Optimization techniques on CPUs

6.4.3.1 Compilation-level optimization

The choice of the compilation flags is a crucial step: it helps to exploit the hardware specificities and capabilities of a system running a given program or application [151]. GRICS is a C/C++ program; it is compiled with GNU C++ 9.4.0 compiler. The flag **-Ofast** is applied to provide the highest level of compiler optimizations and to enable automatic

vectorization. The flag **-fopenmp** enables generating parallel code using OpenMP. The **-march=native** flags instruct the compiler to build binaries optimized for our specific AMD EPYC micro-architecture. We also used the flag **-mavx2** to generate codes using AVX2 instructions. Finally, we used the flag **-mfma** to enable the use of the FMA instructions. For the Intel Xeon processor, we enabled architecture optimization with **-march=cascadelake** flag, and we used **-Ofast** too.

6.4.3.2 Threads Binding and affinity

The operating system is able to move a thread around from its initial position while executing the code. This may cause performance degradation by losing both locality and data in caches. It is possible to prevent the system from completing such an operation by using the strategy of thread binding; it consists of binding threads to a specific core processor, a hardware thread, or a socket. OpenMP offers two environment variables to bind threads:

OMP_PLACES: The variable specifies on which CPUs the thread should be placed. This operation can be conducted either by using an abstract name, such as *threads*, *cores*, *sockets*, or by an explicit list of the places. With *threads*, each place is a single hardware thread, and, with *cores*, each place is a single-core processor, which may correspond to one or more hardware threads. With *sockets*, the places correspond to a single socket in a multi-socket system.

OMP_PROC_BIND: This variable indicates if a thread may be moved between CPUs. It can be set to *false*, *true*, *primary*, *master*, *close*, and *spread*. If set to *true*, the OpenMP thread should not be moved from its initial position. With *primary* and *master*, the worker threads are in the same partition as the master thread. With *close*, they are kept close to the partition of the master thread, in contiguous place partitions, which may be beneficial for data sharing and reuse. With *spread*, a sparse distribution within the place partitions is used, which may offer higher bandwidth since we use cores from both sockets.

For GRICS, controlling the number of OpenMP threads is conducted at the compile stage, with dedicated compilation flags. When it comes to thread binding, two lines were added before the execution of the code:

- export OMP_PLACES = threads, cores, or sockets.
- export OMP_PROC_BIND = spread or close.

Hence, these optimizations do not require a source code modification.

6.5 Results

6.5.1 The roofline analysis of the RK on CPUs

The CPU roofline models are presented below. We used dataset D_1 . The main difference between the two CPUs is the DRAM bandwidth. For the Intel Xeon processor, the memory bandwidth is equal to 70.3 GB/s, and for the AMD EPYC processor, it is equal to

221.1 GB/s. Regarding the Floating Point performance, Intel Xeon has the lead with a $P_m=1544$ GFLOP/s. However, since the GRICS reconstruction relies on repeated FFT, the code will likely be impacted by the memory bandwidth, and efforts should be undertaken in this direction to optimize the code. Intel SDE and Vtune report that the GRICS's operational intensity $I_{DRAM,Xeon}$ is equal to 0.8 FLOP/byte. AMD μ Prof reports that $I_{DRAM,EPYC} = 0.69$ FLOP/byte. Using Figures 6.8 and 6.9, we can see that, in both architectures, the GRICS is memory bound. In Intel Xeon, the code is mainly limited by the low value of the DRAM bandwidth; its actual performance P_{Xeon} is equal to 16 GFLOP/s. P_{Xeon} is far from the theoretical bound for this value of operational intensity; a code that has a similar value of $I_{DRAM,Xeon}$ and that uses the micro-architecture of the processor in a better way can operate at a performance P_{mXeon} equal to 56 GFLOP/s. Using Equation 4, the code's architectural efficiency on Xeon is 28%. On the AMD EPYC CPU, it performs better, with a performance P_{AMD} equal to 80 GFLOP/s. In this architecture, the code benefits from the high value of the memory bandwidth, $B_s = 221.1$ GB/s. Its architectural efficiency on EPYC is 52%. Moreover, the figures show that $I_{L2,Xeon} \approx I_{L3,Xeon} \approx I_{DRAM,Xeon}$, which means that there is poor data reuse; once the data are loaded from the main memory, they are not reused while being in caches and instead evicted. This is not the case regarding the AMD EPYC CPU, where $I_{DRAM,EPYC}$ is greater than $I_{L3,Xeon}$ and $I_{L2,Xeon}$, which means that the data are being reused once they are loaded from DRAM.

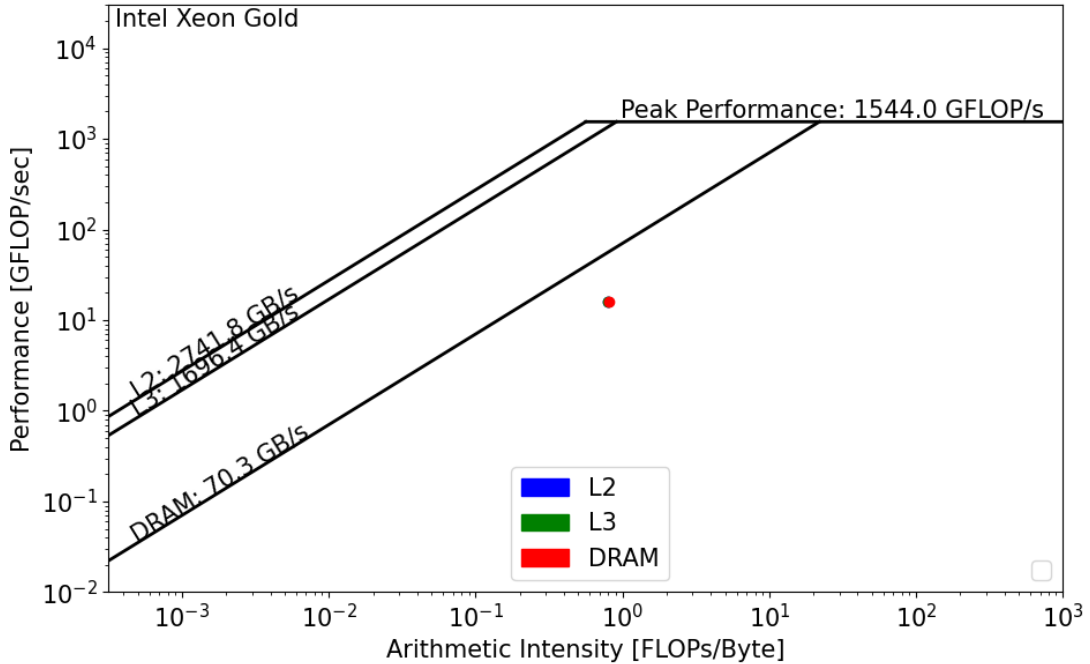


Figure 6.8: Roofline analysis of GRICS on Intel Xeon Gold.

Figure 6.10 presents the computational and memory complexity of Algorithm 5 as a function of the elapsed time per iteration on the AMD EPYC CPU. The complexities are computed using Equations 7 and 8. Datasets D_3 and D_4 align closely with the predicted trend, both for memory and computation. D_2 lies above the linear fitting lines, suggesting

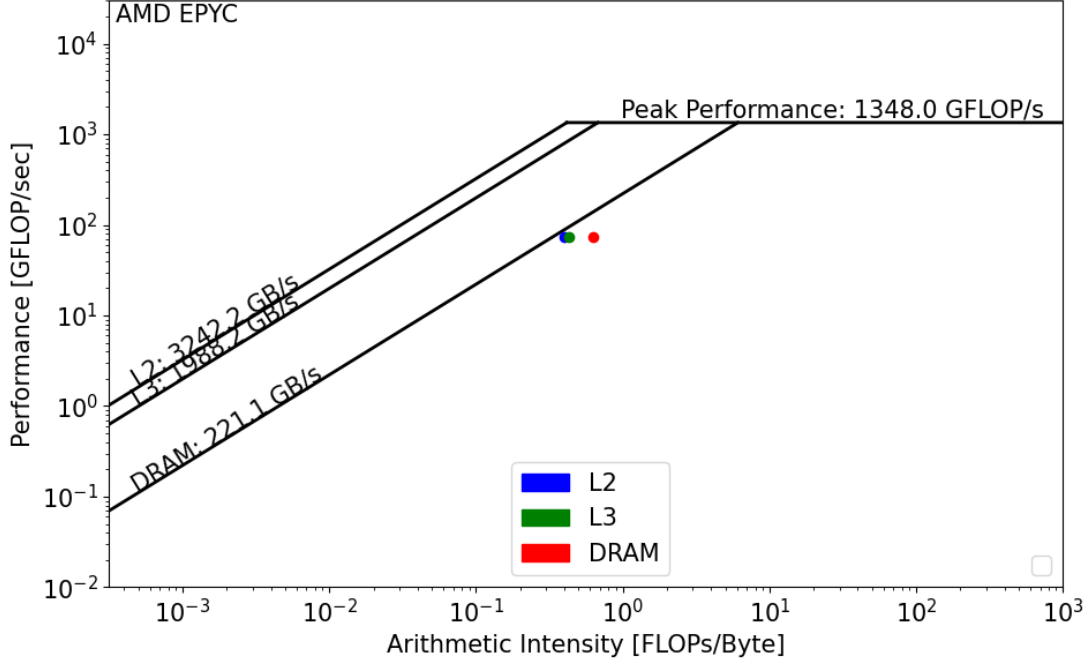


Figure 6.9: Roofline analysis of GRICS on AMD EPYC.

lower computational and memory efficiency than expected. D_1 shows both good memory and computational efficiency, positioned below the fitting lines.

6.5.2 The roofline analysis of the RK on GPUs

Figures 6.11 and 6.12 present the GPU roofline models. In both architectures, the reconstruction kernel is compute bound. The kernel's performance on RTX 5000 P_{RTX} is equal to 5 GFLOP/s, while its performance on A100, P_{A100} , is equal to 16 GFLOP/s. In both cases, the actual performance is far from the roofline boundaries, which means that the actual implementation does not properly use the GPU architecture and that source-code modifications should be considered in order to achieve better performance. Moreover, we notice the absence of data reuse on Quadro RTX, $I_{DDR,RTX} \approx I_{L2,RTX}$, which means that there are repetitive calls to the main memory in order to satisfy the load and store operations. However, the data are being reused on A100, which reduces the latency and improves the overall performance of the kernel.

NVIDIA Nsight compute offers features to evaluate the achieved percentage of utilization of each hardware component with respect to the achieved maximum. In Figure 6.13, we compare these throughputs on the two devices.

The A100 GPU has 40 MB of L2 cache, which is 10 times larger than Quadro's L2 cache. The increase in the L2 cache size improves the performance of the workloads since larger portions of datasets can be cached and repeatedly accessed at a much higher speed than reading from and writing to HBM memory, especially for workloads that require a considerable memory bandwidth and memory traffic, which is the case for this kernel. Moreover, the main memory bandwidth is intensively used by the kernel on

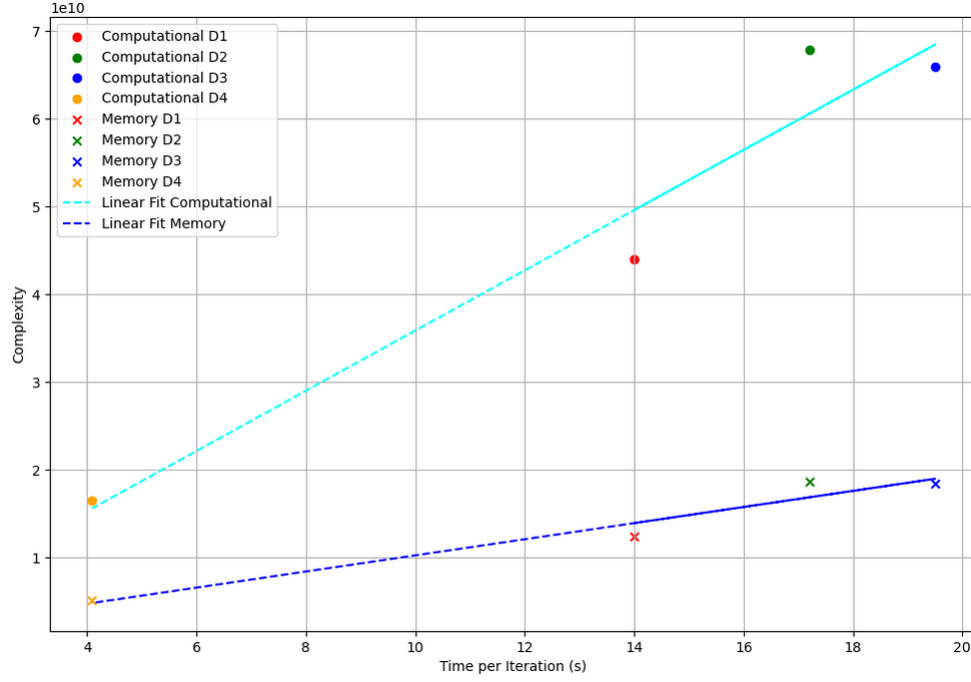


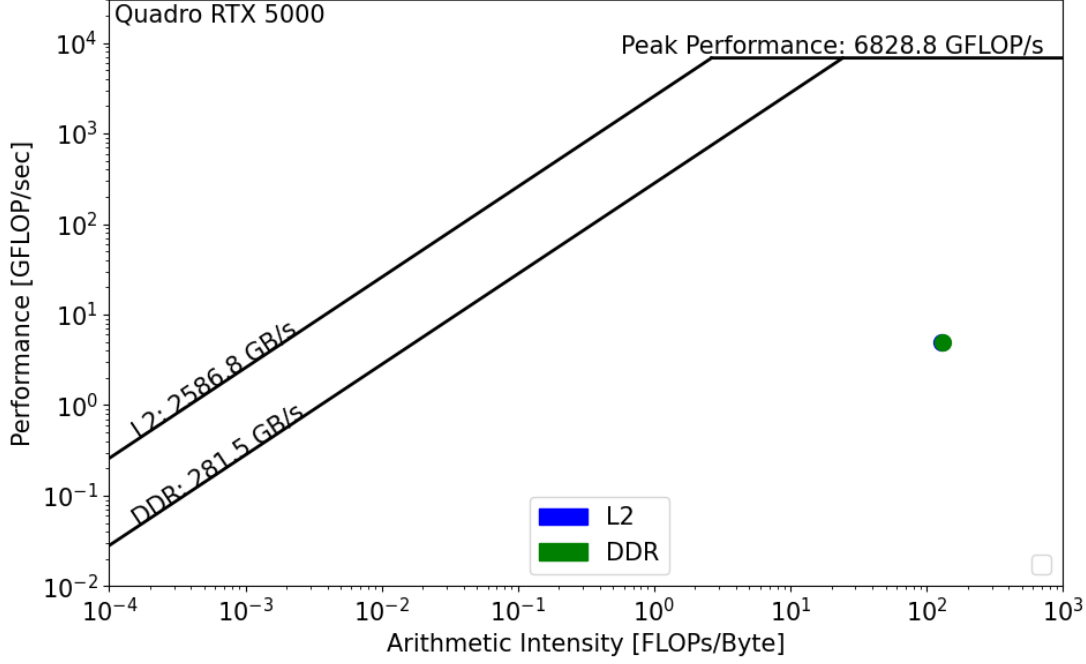
Figure 6.10: Computational and memory complexity of the reconstruction kernel as a function of elapsed time per iteration (on AMD EPYC CPU). The cyan and blue dashed lines in the plot represent the linear regression fits for computational and memory complexities, respectively. Time per iteration refers to the elapsed time of the reconstruction kernel per conjugate gradient iteration.

both architectures, while the SM throughput is low, which helps us in determining the kernel's bottleneck on GPUs. The compute units are not optimally used; architecture optimization would also greatly improve the GPUs' compute throughput regarding the Tensor and CUDA cores.

6.5.3 Optimization Results on CPUs

In order to understand the impact of thread binding and affinity, we present the different elapsed times of GRICS with different configurations using only OpenMP. The elapsed time of the code is insensitive to the variable `OMP_PLACES`. Hence, we choose to bind the threads to hardware threads. However, the elapsed time is remarkably sensitive to the binding strategy.

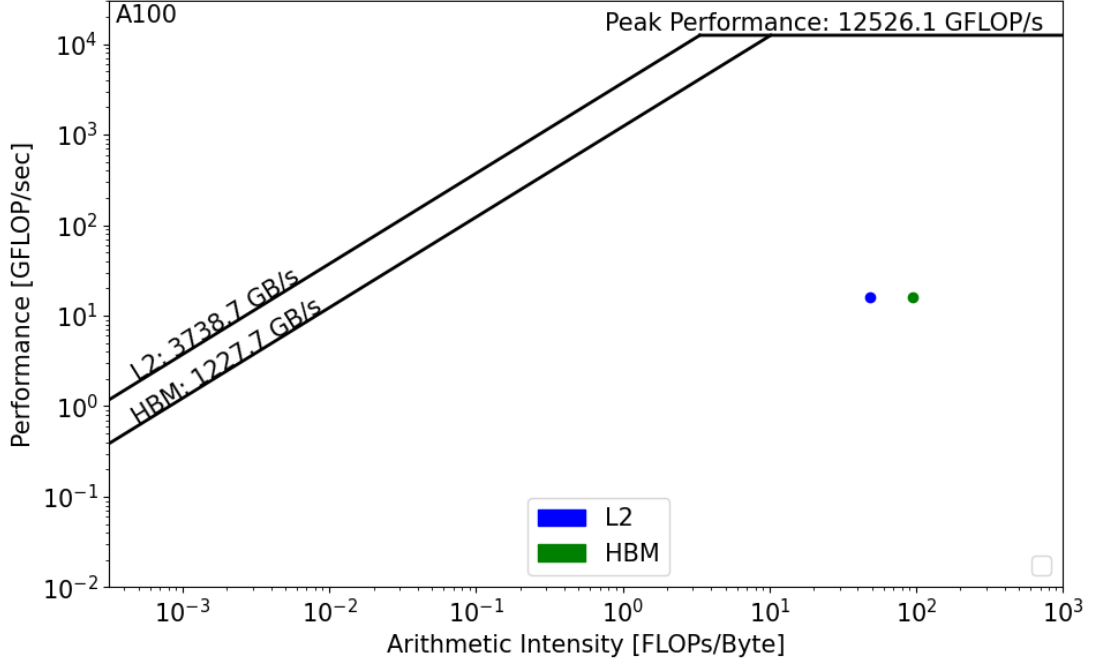
- OS OMP: Only OpenMP parallelization over the receiver coils is considered, with basic optimization flags and without thread binding. The GNU C++ compiler is considered.
- Close OMP: The thread affinity is enabled with the *close* option.
- Spread OMP: The thread affinity is enabled with the *spread* option. In the three cases, the compilation flags discussed in section III.D are enabled.

Figure 6.11: Roofline analysis of $E^H E$ on NVIDIA Quadro RTX 5000.

Each OpenMP configuration for each dataset is executed five times; the reported time is the mean of the five experiments. Since we are the only user at the workstation while conducting the testing, the differences in the execution times between multiple runs (for a given binding strategy) are not significant: the standard deviations range from 0.30 s to 1.001 s. The thread binding and compilation flags do not impact the numerical accuracy and stability of the program since the number of conjugate gradient iterations remains the same with spread/close bindings: 4, 5, 4, and 11 iterations, respectively, for D_1 , D_2 , D_3 , and D_4 . Moreover, we compute the Normalized Root Mean Square Error to prove that there is no image quality degradation with the proposed optimizations: I_1 , I_2 , and I_3 stand, respectively, for the reconstructed images with the spread, close, and no binding options; we calculate $\max(\text{NRMSE}(I_1, I_2), \text{NRMSE}(I_1, I_3), \text{NRMSE}(I_2, I_3))$. These values are, respectively, equal to 4.67×10^{-7} , 5.95×10^{-7} , 4.89×10^{-7} , and 2.40×10^{-3} for D_1 , D_2 , D_3 , and D_4 .

Thread binding and affinity decisions (spread or close) are architecture-dependent for the same program. Using Table 6.6, we can see that, on the AMD EPYC processor, the *spread* option enables a gain of 9–17% on the AMD EPYC CPU, while the *close* option degrades the performance compared to the OS allocation. On the Intel CPU, the code is much slower, and the *close* option yields better results compared to the *spread* binding. However, the OS thread allocation is more suitable to the GRICS on this architecture.

The roofline analysis and the performance of the GRICS’s OpenMP version prove that the EPYC CPU is more suitable for the code, notably in terms of its demanding data traffic. The high value of EPYC’s DRAM bandwidth accelerates the rate at which the data are transferred to the compute units and enables rapid execution. For these reasons, we decided to investigate the performance of the hybrid version only on the AMD EPYC

Figure 6.12: Roofline analysis of $E^H E$ on NVIDIA A100.

CPU; the different execution times are presented in Figure 6.14.

The GRICS's hybrid implementation improves the overall performance in terms of elapsed time. The best performances are achieved when the number of MPI processes n divides the number of motion states N_t ; for datasets D_1, D_2 , and D_3 , N_t is equal to 12 and the minimal times are reported for $n \in \{3, 4, 6\}$. The best performance is achieved with $n = 6$ ranks. For dataset D_4 with $N_t = 16$, the minimal elapsed time corresponds to the $n = 4$ and $n = 8$ processes. When the number of processes n divides the number of motion states N_t evenly, it leads to better load balancing. Each process receives an equal share of the workload, minimizing idle times. The empirical data show that the elapsed time initially decreases with increasing processes (up to a certain point) and then starts increasing. More processes may increase the number of communications required, leading to higher network traffic and contention. For the OpenMP parallelization of the loop over the N_c coils, choosing a number of threads m such that m divides N_c does not greatly improve the overall performance; the set of threads m when $m \in \{10, 12, 13, 15, 18, 20\}$ provides approximately the same result. A small improvement is noticed when $m = N_c/2$. The best MPI OpenMP settings are $n = N_t/2$ and $m = N_c/2$ for all the datasets.

6.6 Discussion

Using the proposed optimizations renders GRICS reconstruction available at the MRI console within an acceptable delay (80-102 s depending on the dataset), even in the case of very high-resolution 3D images. This delay is smaller than the typical acquisition time for one sequence of the examination. This is important for clinical use as the radiologist

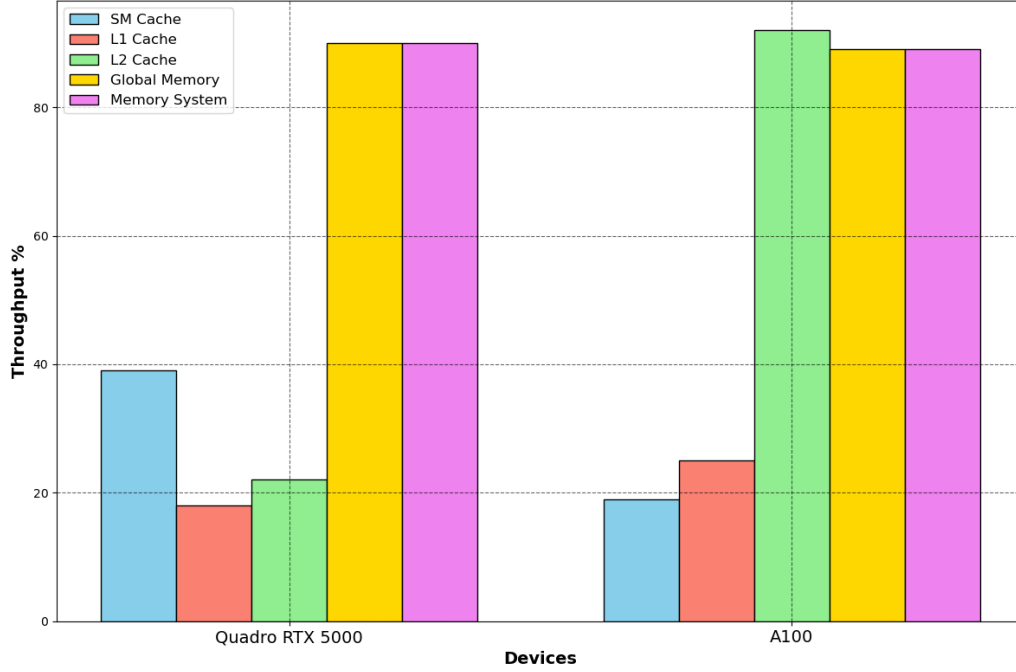


Figure 6.13: Performance throughputs on RTX 5000 and A100. On GPUs, the main memory system is used optimally, and the achieved throughputs are 90% and 89% for RTX 5000 and A100, respectively. The first difference concerns the L2 cache throughput, 22% for RTX 5000 and 92% for A100. However, the main hardware limitation is the SM utilization; the throughputs are low on both architectures.

needs to visualize the images shortly after scanning, evaluate whether the image quality is sufficient for diagnostic use, and decide whether the patient can be released from the scanner. The optimal processor architecture is algorithm-dependent, and the roofline model is a visual model that helps in evaluating the software–hardware interaction. It is difficult to suggest a parallelization strategy that would be beneficial to every architecture or to confirm that an optimization pattern would improve the performance on multiple architectures. However, in this work, we presented steps to follow in order to evaluate the actual performance of a program, and ways to improve it with no source-code modification, by establishing the roofline model of the considered workstation using performance profiling tools to generate the operational intensity of the code and performing a roofline analysis to capture the hardware limitations and bottlenecks. In this work, we determined the main hardware limitations of the iterative MRI reconstruction algorithms, which are not straightforward to estimate, depending on the workstation (CPUs or GPUs).

For the CPU optimizations, we suggested identifying the appropriate compilation flags for the architecture, testing several binding strategies for the OpenMP parallelization, and selecting the optimal number of MPI ranks to minimize the communication overhead and ensure effective parallelization.

We have shown that, without any code modification, using thread binding and affinity enables gains of 11%, 12%, 9%, and 17% for D_1 , D_2 , D_3 , and D_4 , respectively, with only OpenMP on AMD EPYC. The GRICS’s architectural efficiency on this CPU is equal to

Table 6.6: Elapsed time of GRICS with different OpenMP configurations on CPUs.

OpenMP Option	Dataset	GRICS EPYC (s)	GRICS Xeon (s)
OS OMP	D_1	214	485
	D_2	294	587
	D_3	293	614
	D_4	225	500
Close OMP	D_1	272	526
	D_2	376	630
	D_3	416	657
	D_4	256	520
Spread OMP	D_1	190	550
	D_2	259	658
	D_3	267	683
	D_4	186	545

52%. However, the OS OpenMP affinity provides better results when it comes to Intel Xeon compared to the *spread* and *close* options. On this CPU, the code’s architectural efficiency is 28%.

The optimal MPI and OpenMP combination yields gains of 62%, 65%, 68%, and 62% for D_1 , D_2 , D_3 , and D_4 , respectively, on AMD EPYC. These gains are computed relative to the OS OpenMP elapsed time for each dataset. The reconstruction kernel’s elapsed time on the A100 GPU was the lowest, even if the roofline analysis pointed out that the GPU architecture is not optimally used. A low execution time does not guarantee optimal GPU utilization. Figure 6.13 shows that the GPU memory bandwidth is used efficiently. However, the program is not fully utilizing the GPU’s available cores, leading to suboptimal performance. For instance, using only a fraction of the available cores will leave the rest of the GPU’s resources idle (low SM throughput on both devices) and lead to an imbalanced workload among the threads. Optimal GPU utilization requires a program to effectively leverage the parallelism, memory bandwidth, load balancing, and efficient memory access patterns. Therefore, even with a low execution time, the program may still be far from making the best use of the GPU’s capabilities. We acknowledge that the GPU part of the study is preliminary and aimed at providing initial insights into the performance benefits of GPUs for our specific application. The future work will include a CUDA implementation of the reconstruction kernel (Algorithm 5), taking into account the results of the roofline model on GPUs.

In Ref . [152], Driscoll introduced several HPC techniques for computational image reconstruction with specific applications in the MRI field. He presented an architectural efficiency (peak percentage) comparison of the non-Cartesian SENSE normal operator

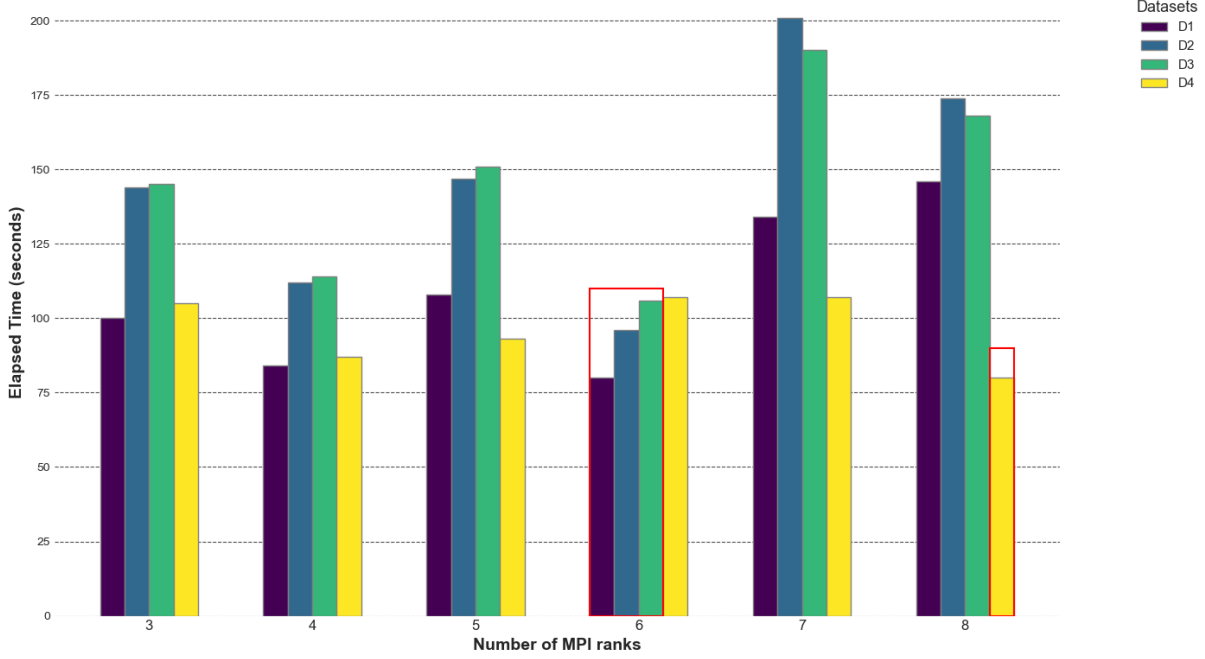


Figure 6.14: Elapsed time of GRICS for different MPI processes on AMD EPYC; the number of OpenMP threads was fixed to 20. The reported times are the mean of 5 executions. The maximum standard deviations were 0.96, 0.14, 0.39, and 0.13 for D_1 , D_2 , D_3 , and D_4 , respectively. We highlighted the best times per dataset in red.

$A^H A$ across different computing platforms. The operator achieved peak percentages of 39% and 10%, respectively, on the Intel Xeon E5-2698 CPU and Intel Xeon Phi 7250 KNL processor. The GRICS method achieved a superior architectural efficiency on the AMD EPYC CPU and a better peak percentage on the Intel Xeon CPU compared to the Intel Xeon E5-2698 CPU. We recall that, since the GRICS is memory bound on a CPU, the peak percentage is computed relative to the memory performance for the GRICS’s operational intensity value rather than the peak floating point performance.

The proposed performance study and optimizations are not limited to GRICS. It is completely possible to follow the same steps in order to evaluate other computationally intensive MRI motion-compensated reconstruction codes that share the same bottleneck code as that described in Algorithm 5 and are applied by many authors [114].

The performance study does not confirm that the AMD EPYC CPU will outperform every Intel CPU in resolving motion-compensated reconstruction problems. We have chosen a specific Intel processor with a specific memory bandwidth and compute performance. Our study showed that, on CPUs, GRICS will perform better on an architecture that offers high memory and cache bandwidths. For example, our work can lead us to propose a suitable Intel architecture for GRICS, which is the *Intel Xeon Phi 7235 Processor*; the GRICS would benefit from Xeon’s Phi Multi-Channel DRAM (MCDRAM); the multi-channel aspect is the key to more bandwidth. Affording more channels regarding the DRAM leads to more simultaneous accesses, thus improving the bandwidth. Intel Xeon Phi processors include an on-package MCDRAM to increase the bandwidth

available to a program at the same level in the memory hierarchy as the DDR memory but with higher bandwidth (High-Bandwidth Memory). The MCDRAM can serve as a high-bandwidth cache for DDR memory. It has a 450 GB/s bandwidth, and, assuming GRICS bottlenecks and hotspots, we would receive a performance improvement on this architecture. In [153], the authors proved that MCDRAM integration offers considerable speedup of the matrix–vector operations (the same type of operations in the GRICS) in Multilayer Perceptron and Recurrent Neural Networks compared to Tensor Product Units (TPUs) and an NVIDIA P4 GPU. A more recent choice would be the *Intel Xeon Platinum 8380* processor, suitable for data-intensive workloads. With support for eight channels of DDR4-3200, it offers a maximum memory bandwidth of 204.8 GB/s, making it appropriate for memory-intensive applications. Moreover, Intel DL Boost and AVX-512 provide significant performance improvements for HPC applications. It is crucial to deeply understand the hardware limitations and bottlenecks of the application in order to suggest a suitable architecture.

6.7 Conclusion

For clinical use, the optimization of reconstruction in MRI is an important subject. We evaluated the performance of motion-compensated reconstruction algorithms using the GRICS code as an example, and we determined their main bottlenecks and hotspots. Using the roofline model, we have shown that the memory bandwidth is the main hotspot for this class of advanced MRI reconstruction on CPUs. The thread binding strategy resulted in up to a 17% performance gain when considering only OpenMP parallelization. The architectural efficiency of the program was 28% on Intel Xeon Gold and 52% on AMD EPYC. On the GPU devices, the reconstruction kernel was limited by the computational performance and compute throughput, indicating that the actual implementation does not fully exploit the GPU computational capabilities. Efforts should be directed towards maximizing the workload distribution among threads. However, the roofline analysis of the reconstruction kernel on the A100 GPU, where host–device transfer was not an issue, highlighted that a native C++ GPU implementation of this program, using, for example, CUDA, could leverage the hardware capabilities more efficiently and achieve better performance. As a result of the findings, the next step involves implementing a native C++ version of the reconstruction kernel using CUDA, starting with the micro-kernels of Algorithm 5, particularly the sparse matrix vector product, the 1D and 2D FFT, and the element-wise matrix product.

Chapter 7

CUDA Implementation of GRICS

7.1 Compute Unified Device Architecture parallelization

Compute Unified Device Architecture parallelization - CUDA - was developed by NVIDIA and integrated into their GPUs. From a programming standpoint, CUDA is straightforward and user-friendly; it closely resembles traditional C/C++, with a few additional keywords and syntax specific to GPU platforms. The core of a GPU code is the *CUDA Kernel*.

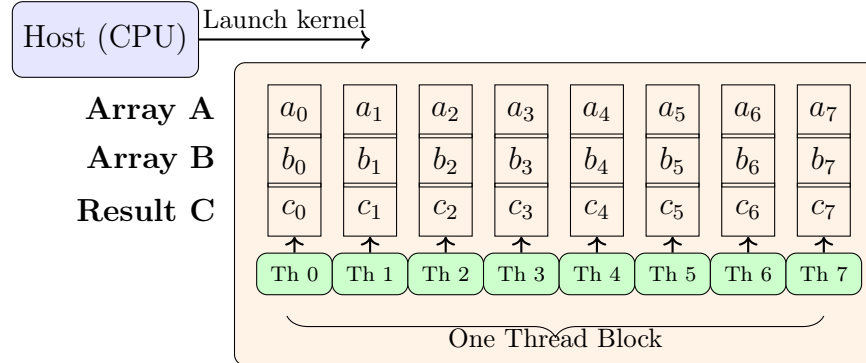


Figure 7.1: CUDA parallelization: the host CPU launches a GPU kernel. Each thread computes the sum of one (or more) elements independently in GPU memory. Threads are organized in blocks and access global memory directly.

7.1.1 CUDA Kernel

A kernel is a fundamental component of the CUDA programming model. It refers to a function designed explicitly for data-parallel computations on a GPU, which acts as a co-processor alongside the CPU (host). When launched, the kernel executes simultaneously across multiple GPU threads, enabling efficient parallel processing.

Listing 7.1: Example of CUDA Kernel to add two 1D arrays

```
1  __global__ void add_arrays(const float *a, const float *b, float *c, int n)
2  {
3      int idx = blockIdx.x * blockDim.x + threadIdx.x;
4      if (idx < n) {
5          c[idx] = a[idx] + b[idx];
6      }
7  }
```

7.1.2 Threads and Blocks Management

In CUDA programming, the GPU is treated as a computational device capable of executing a large number of threads simultaneously. A CUDA function, known as a kernel, is executed concurrently by multiple threads arranged into a structured grid of thread blocks. Each thread block contains threads that can concurrently work by exchanging data using fast, shared memory and synchronizing their execution to manage coordinated memory access. Threads are grouped into independent blocks, which collectively form a grid, as depicted in Figure 7.2 Each thread can be uniquely identified using built-in CUDA indexing variables, such as *threadIdx*, *blockIdx*, *blockDim*, and *gridDim*. To simplify complex thread addressing, CUDA allows blocks to be defined as multidimensional arrays (two- or three-dimensional) of customizable size, enabling threads to be referenced through two- or three-component indices. For example, in a two-dimensional block of size $D_x \times D_y$, the thread with coordinates (x, y) can be linearly indexed as $y \times D_x + x$.

7.1.3 Single Instruction Multiple Thread SIMT

A CUDA-capable GPU consists of a scalable array of multithreaded Streaming Multiprocessors (SMs). Each SM contains multiple scalar processors known as Streaming Processors (SPs), with the precise number varying according to the GPU architecture. When the host CPU launches a CUDA kernel, its blocks are evenly distributed across SMs based on their available execution resources. Threads within a thread block execute simultaneously as a single unit on a specific SM, and an SM can concurrently handle multiple thread blocks. Once executing blocks complete their tasks, the freed SM resources are promptly reassigned to inactive blocks. To efficiently handle the extensive number of threads, CUDA employs a specialized architecture known as Single-Instruction, Multiple-Thread (SIMT). In this model, each multiprocessor creates, manages, schedules, and executes threads in parallel groups of 32 threads, referred to as *warps*. Threads within a warp initially execute the same instruction at the same program address. However, each thread maintains its own instruction address counter and register state, allowing them to diverge and execute independently if conditional branching occurs. When a multiprocessor receives one or more thread blocks to execute, it divides them into warps, each consisting of 32 consecutive threads based on their thread IDs (with the first warp starting at thread 0). These warps are then independently scheduled for execution by a dedicated *warp scheduler*. A warp executes a single common instruction across all its

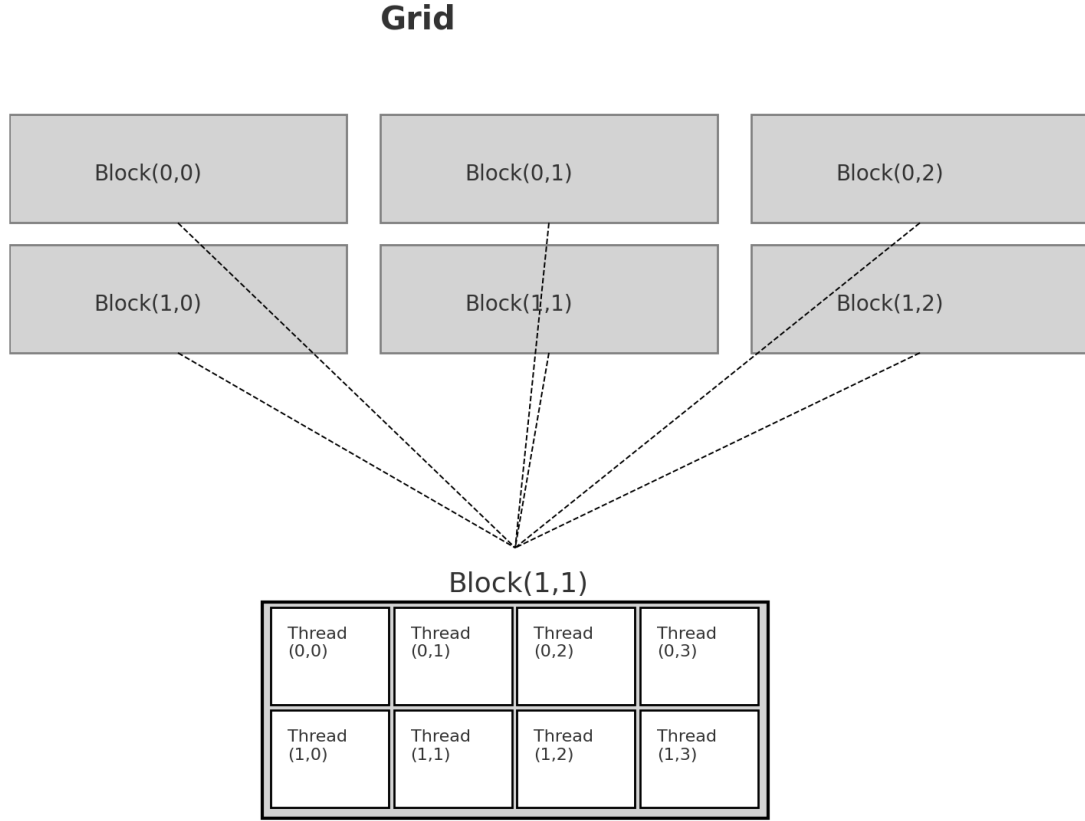


Figure 7.2: Threads and Block in CUDA

threads simultaneously, achieving peak efficiency when all 32 threads follow an identical execution path. If threads within a warp diverge due to conditional branching dependent on data, the warp scheduler sequentially executes each divergent path, temporarily disabling threads not following the current path. Once all branch paths are completed, the threads converge back to continue executing along the same path. This *branch divergence* affects only threads within the same warp; distinct warps operate independently, even if executing entirely different sections of code.

From a hardware perspective, each Streaming Multiprocessor (SM) integrates a heterogeneous set of execution units. While a large number of *Single Precision* (FP32) cores are available for add and multiply operations, the number of *Double Precision* (FP64) units is typically much smaller (e.g., only one FP64 unit for every 32 or 64 FP32 units, depending on the GPU architecture). This asymmetry has direct implications for MRI reconstruction, where Double Precision arithmetic may be required to maintain numerical stability. Each thread also relies on a set of registers, with registers within a warp mapped to a common register file, and communication between threads often relying on shared memory, which is explicitly managed by the programmer. In addition to scalar cores, recent GPUs also include specialized units such as *Tensor Cores*, designed for mixed-precision matrix operations, and integer ALUs.

7.1.4 Useful Libraries

In addition to the C-based programming language, CUDA offers a wide range of powerful libraries that enable GPU-accelerated computing.

7.1.4.1 cuBLAS

The CUDA Basic Linear Algebra Subroutines (cuBLAS) library is a GPU-accelerated implementation of the full standard BLAS library, delivering performance gains of $6\times$ to $17\times$ compared to the latest MKL BLAS. cuBLAS fully supports all 152 standard BLAS routines across single, double, complex, and double complex data types. Leveraging the high-performance BLAS operations in cuBLAS, heterogeneous LAPACK libraries like CULA Tools and MAGMA have also been developed to further extend GPU-accelerated linear algebra capabilities.

7.1.4.2 cuFFT

The Fast Fourier Transform (FFT) is an efficient algorithm for calculating the discrete Fourier transform of complex or real-valued data sets. NVIDIA's CUDA Fast Fourier Transform library (cuFFT) offers a straightforward interface for performing FFTs on the GPU, achieving speedups of up to $10\times$ compared to traditional CPU implementations. cuFFT features an API similar to the advanced interface of FFTW and supports 1D, 2D, and 3D transforms for both complex and real data types, making it easy to integrate high-performance spectral analysis into CUDA applications.

7.2 Methods

7.2.1 CUDA Implementation of the RK

In order to implement Algorithm 5 in a GPU with CUDA, several kernels and mathematical operations were designed. First, the application of the Sensitivity Maps operator C_c and its Hermitian transpose C_c^H required the implementation of an *element-wise* complex multiplication, where each thread processes one data point, performing concurrently the real and imaginary parts of the complex multiplication for all the elements. Another *element-wise* product kernel was designed to apply the sampling mask operator $S_t^H S_t$. The kernels for the displacement operator T_t and its Hermitian transpose T_t^H assign threads to individual voxels and compute displacement fields based on rigid or non-rigid motion parameters, warping the image using linear interpolation. In the C++ implementation, the non-zero elements of T_t were precomputed before the CG solver was run and stored in CPU memory. In the Matlab-GPU implementation, due to the GPU memory limitation, the precomputed elements had to be transferred from the CPU to the GPU memory at the beginning of the outer loop (before line 3 of Algorithm 5). To avoid this repeated and costly transfer, in the native CUDA implementation, the non-zero elements of T_t were computed dynamically within the kernel from the motion model definition. This resulted in a higher computational complexity; however, all the required problem variables were

stored in GPU memory before running the CG solver. The Fast Fourier Transform *FFT* algorithm was implemented using the GPU-optimized library cuFFT [154]. Moreover, in order to avoid race conditions, *atomicAdd* was explicitly employed at multiple locations, mainly in the reduction process (line 8 in Algorithm 5), to safely accumulate partial results into the shared memory location. Finally, the CG solver was implemented using the cuBLAS library [155].

7.2.2 Data Acquisition

In this study, we investigated data acquired through two separate MRI examinations conducted on different imaging systems. For the first acquisition, raw MRI data were obtained using a 3T MRI scanner (MAGNETOM Prisma, Siemens Healthineers, Erlangen, Germany). Three volunteers were enrolled, and three datasets (D_1, D_2, D_3) with non-rigid motion were acquired using a T1-weighted 3D fast low-angle shot GRAdient Echo (GRE) sequence covering the breast region in a 90 s free-breathing scan. Additional details regarding this acquisition and its ethical approval are available in [156, 148]. For the second acquisition, MRI data were acquired using a 7T MRI scanner (Magnetom 7T, Siemens Healthineers) equipped with an in-house built head RF coil having 8 transmit and 32 receive channels [157]. Here, 3 datasets (D_4, D_5, D_6) containing subtle rigid motion from 3 volunteers were obtained with a 3D Magnetization Prepared Rapid Gradient Echo (MPRAGE) [18] sequence covering the head at a resolution of $1 \times 1 \times 1$ mm³. To enhance motion sensitivity, the MPRAGE sequence was modified to randomize the k -space sampling order. This randomization evenly distributes motion-induced artifacts, effectively transforming them into less noticeable, noise-like distortions and making them more suitable to advanced reconstruction techniques [69]. Additionally, it improves temporal resolution by enabling more flexible sampling strategies that adapt to motion dynamics and capture more relevant temporal information. A navigator (the central k -space line, a line scan oriented along the inferior-to-superior IS direction) was acquired at the beginning of each RAGE shot and was processed in order to split the k -space data into N_t motion states. All participants in these studies provided written informed consent. Volunteers were not instructed to move. A summary of the dataset parameters is depicted in Table 7.1.

7.2.3 The test machines

The CPU-only implementation of Algorithm 5 was tested on an AMD EPYC 75F3 processor (AMD, Santa Clara, CA, USA). The C++ code benefits from a hybrid parallelization using MPI to distribute the N_t motion states over n MPI ranks and OpenMP to distribute the set of the N_c coils over m software threads. The GPU implementations were executed on NVIDIA A100 GPU (NVIDIA Corporation, Santa Clara, CA, USA). More details about the two devices can be found in Table 7.2.

Table 7.1: N_x, N_y, N_z are the number of samples in frequency encoding, phase-encoding, and slice directions, respectively. N_c is the number of coils in the coil array. N_t is the number of motion states.

Dataset	D_1	D_2	D_3	D_4	D_5	D_6
N_x	320	320	320	256	256	256
N_y	470	470	550	208	208	208
N_z	112	112	112	160	160	160
N_c	20	30	26	32	32	32
N_t	12	12	12	12	12	12
Magnetic Field (T)	3	3	3	7	7	7

Table 7.2: The main characteristics of the test machines. SM stands for streaming multi-processor (akin to CPU cores). FP stands for floating point. The bandwidth of the memory levels and the peak FP performance were estimated empirically with the Empirical Roofline Toolkit ERT, which provides more realistic benchmarking of a workstation.

Device	AMD EPYC	NVIDIA A100
Main memory bandwidth (GB/s)	221.1	1227.7
L2 bandwidth (GB/s)	3242.2	3738.7
Main memory size (GB)	4000	40
Main memory type	DDR4	HBM2e
Nb cores/SMs	36	108
Peak FP32 (GFLOP/s)	1348	12526.1
Release year	2021	2020

7.2.4 Performance Profiling

In order to evaluate the performance of our CUDA-based implementation, we employed the NVIDIA Nsight Compute profiling tool [136]. Nsight Compute is a performance analysis utility specifically designed for NVIDIA GPUs, providing kernel-level profiling with minimal overhead, with the aim of identifying and optimizing performance bottlenecks. In this context, Nsight Compute was used to collect performance metrics such as the FLOPs and memory traffic for both the main memory and the cache hierarchy. The set of the collected metrics is presented in Table 7.3.

The operational intensities of each memory level are computed using the output of these metrics, in particular:

$$OI_{L_1} = \frac{\text{FLOPs}}{L_1\text{cache}}, \quad OI_{L_2} = \frac{\text{FLOPs}}{L_2\text{cache}}, \quad OI_{m_{HBM}} = \frac{\text{FLOPs}}{\text{HBM}}.$$

Table 7.3: NVIDIA Nsight Compute metrics. FP stands for floating point.

Data	Metric
FP 64	sm__sass_thread_inst_executed_op_fp64_pred_on.sum
FP 32	sm__sass_thread_inst_executed_op_fp32_pred_on.sum
FP 16	sm__sass_thread_inst_executed_op_fp16_pred_on.sum
Tensor Core	sm__inst_executed_pipe_tensor.sum
L1 cache	l1tex__t_bytes.sum
L2 cache	l2s__t_bytes.sum
HBM	dram__bytes.sum

Given that T is the elapsed time of the CUDA-based implementation, its achievable performance P is computed as

$$P = \frac{\text{FLOPs}}{T}$$

7.3 Results

To compare the performance of the three implementations, we measured the elapsed time per conjugate gradient (CG) iteration across the six datasets listed in Table 7.1. Each reported time represents the average of five runs. The results are shown in Figure 7.3, where the primary y-axis (left) indicates the elapsed time (in seconds) per CG iteration, and the secondary y-axis (right) denotes the speedup relative to the CPU reference implementation. The CUDA-based implementation significantly reduced the elapsed time, achieving speedup factors of between 9.7 and 13.9 (green dashed line) compared to the CPU-only version. The Matlab-GPU implementation (blue dashed line) also improved speedup but was less pronounced than the CUDA solution. The NRMSE of the CPU and the CUDA images remained within the range of 10^{-6} to 10^{-4} for all datasets. Since these values are lower than the reconstruction kernel’s tolerance of $\epsilon_r = 10^{-3}$, no detectable image quality degradation was observed. Figure 7.4 presents a roofline analysis of the $E^H E$ operator, highlighting both the achievable performance and operational intensity for each operator in Algorithm 5. We relied on the Empirical Roofline Toolkit to measure realistic bandwidth and peak FP32/FP64 performance values rather than theoretical specifications. By computing the motion operator elements on the fly, the performance of operator T_t was significantly enhanced, reaching 105 GFLOP/s. The Fast Fourier Transform (FFT) steps, executed via the cuFFT library, achieved 310 GFLOP/s. Overall, the $E^H E$ function is bandwidth-bound, exhibiting a performance of approximately 250 GFLOP/s at a high-bandwidth memory (HBM) operational intensity of 2.94 FLOP/Byte. For comparison, [156] reported a performance of 16 GFLOP/s for the Matlab GPU implementation of Algorithm 5. In contrast, our native GPU implementation delivered a $15.6\times$ performance gain, narrowing the gap to the roofline limits that define the upper bounds of achievable performance. Figures 7.5 and 7.6 illustrate the reconstructed images for each implemen-

tation. In both the brain MRI dataset (D_4), subject to rigid-body motion, and the breast MRI dataset (D_2), subject to non-rigid respiratory and cardiac motion, the three methods effectively suppressed motion-induced artifacts.

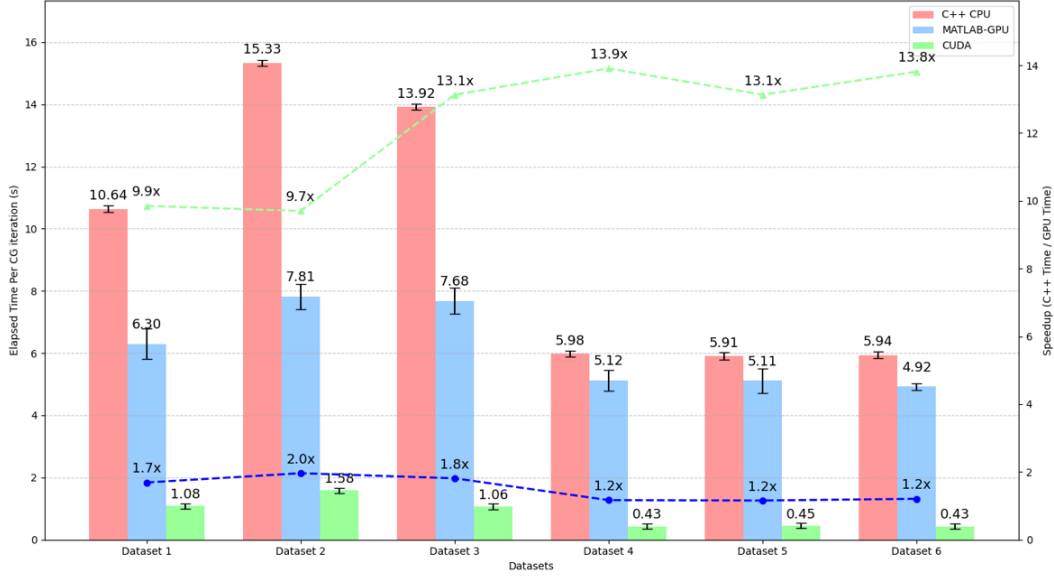


Figure 7.3: Comparison of the elapsed time per CG iteration for the three implementations.

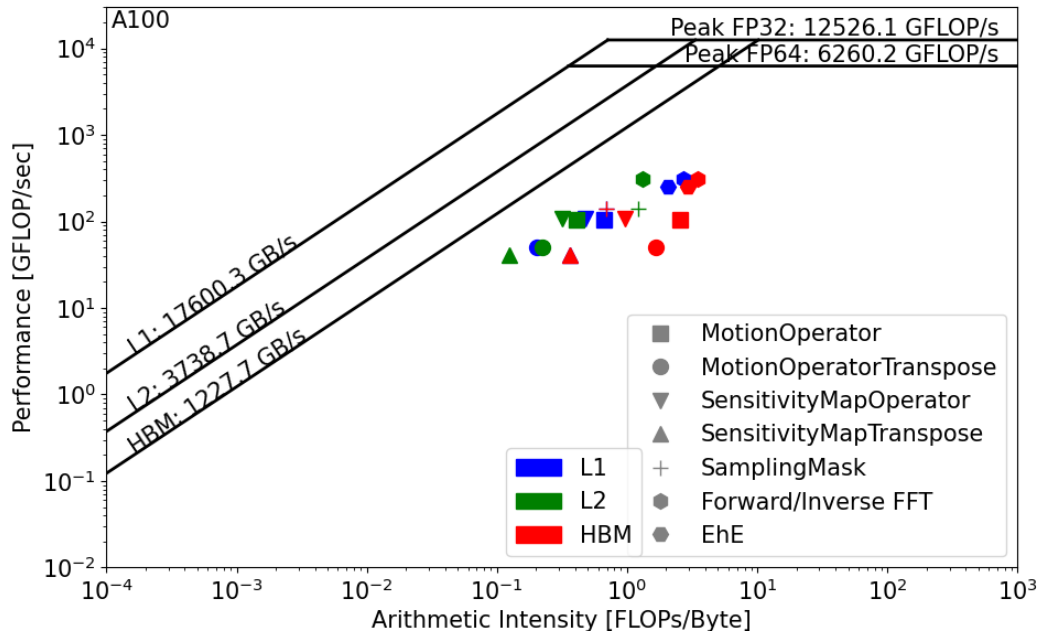


Figure 7.4: Roofline analysis of the implemented CUDA kernels on NVIDIA A100. Dataset D_1 .

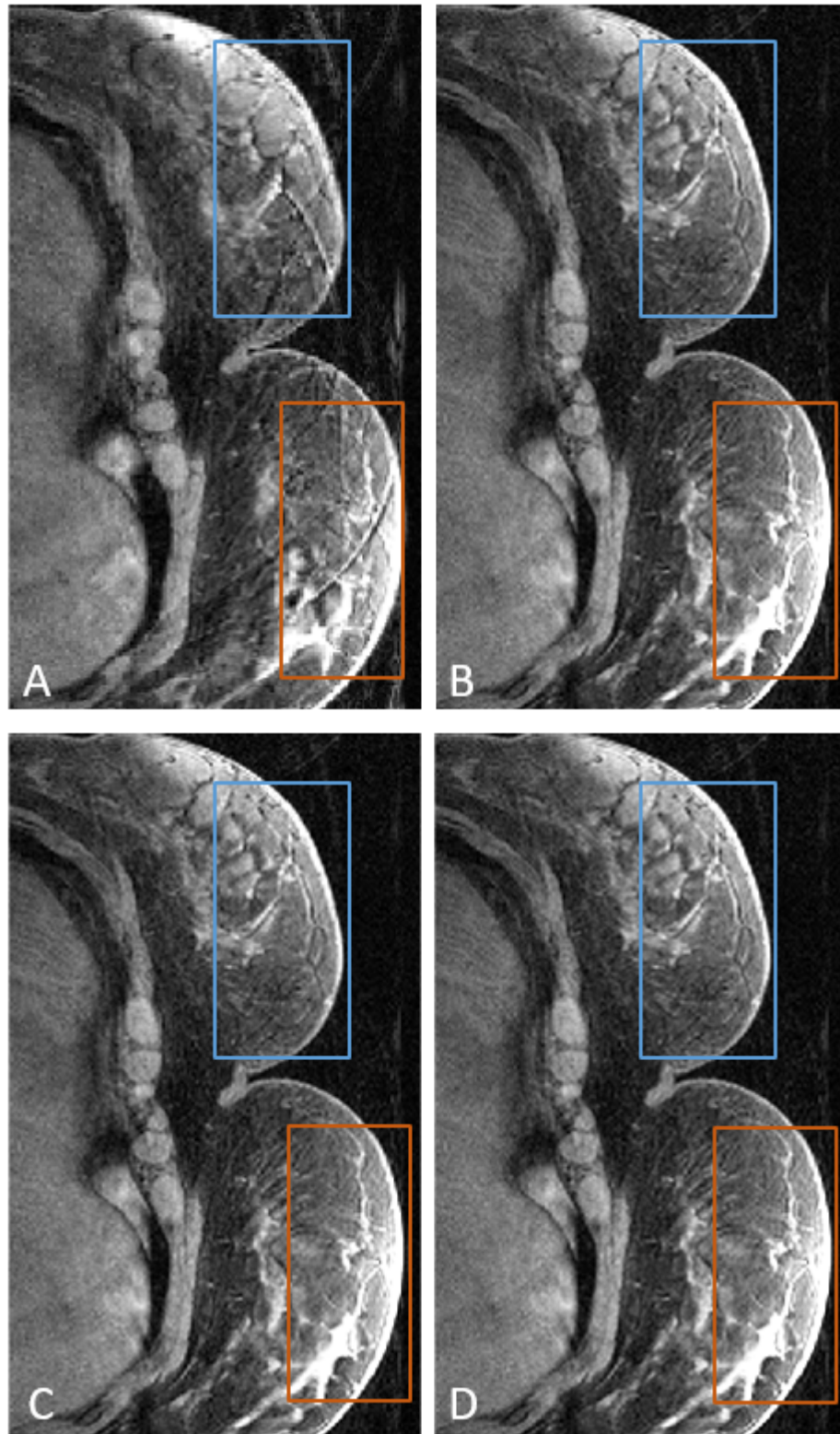


Figure 7.5: Comparison of GRICS reconstructions with the three implementations for dataset D_2 . (A) Uncorrected reconstruction. (B): The CPU-only solution. (C): The Matlab-GPU solution. (D): The CUDA solution. The three solutions markedly reduced the motion artifacts visible in the blue and orange squares.

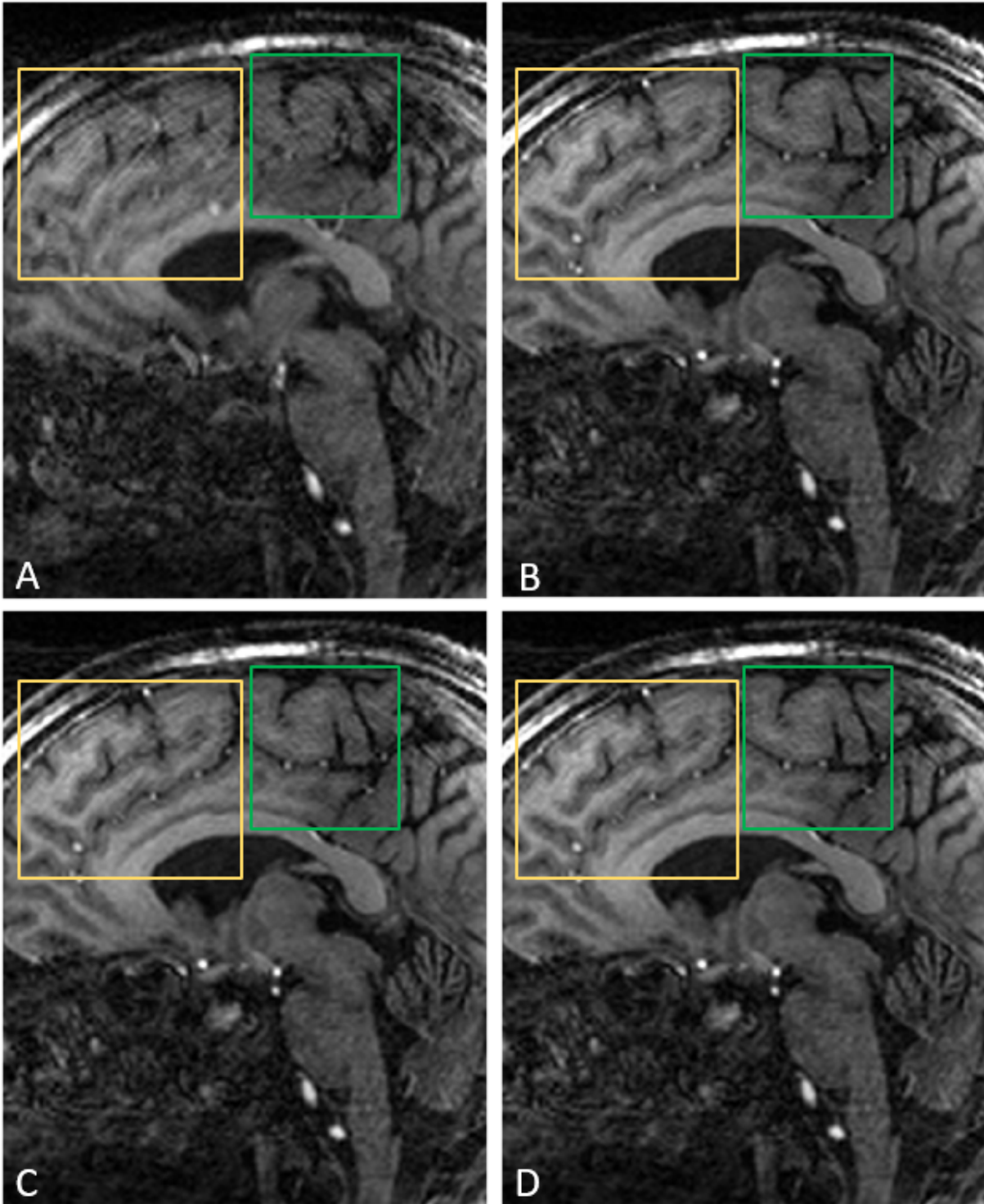


Figure 7.6: Comparison of GRICS reconstructions with the three implementations for dataset D_4 . (A) Uncorrected reconstruction. (B): The CPU-only solution. (C): The Matlab-GPU solution. (D): The CUDA solution. Improvements of the reconstruction quality were noticed on the frontal (yellow square) and the parietal (green square) lobes.

7.4 Discussion

Compared to the CPU-only implementation, the GPU-based solutions significantly decreased computation time and improved the performance of Algorithm 5. In particular, the CUDA solution was approximately 9–13 times faster than the hybrid MPI/OpenMP version, exploiting the GPU micro-architecture without compromising the quality of the reconstructed images. For all datasets, the normalized root-mean-square error remained within the specified tolerance of the reconstruction kernel. For many years, one of the primary bottlenecks in GPU-based computing was the limited amount of device memory, which often forced data to be shuttled back and forth between the host and the device to fit larger workloads. This frequent transfer not only added latency but also introduced complexity in managing data pipelines for HPC applications. Recently, however, a new generation of GPUs (such as A100 GPU) has emerged with significantly expanded memory capacities, enabling developers to hold entire datasets directly in GPU memory. By offering more robust storage space and reducing reliance on host-to-device transfers, these GPUs improve the overall performance of such applications [158, 159]. The roofline analysis showed substantial performance improvements with the CUDA implementation. However, the actual performance remains noticeably below the theoretical upper bound for the corresponding operational intensity. Potential future optimizations include the following:

- Data storage in half precision: Reducing the memory footprint by using FP16 could significantly enhance memory bandwidth utilization. An initial test was conducted with this constraint, and the reconstructed image quality was not compromised.
- Improvement of cache usage: Achieved by optimizing memory access patterns to benefit from the large L2 cache on the A100 GPU.
- The Utilization of tensor cores could be also investigated to accelerate matrix-based kernels.

The CUDA back-end is markedly cheaper to run. In our code, the A100 GPU performed each reconstruction in 5.3–18 s, whereas the 36-core EPYC 75F3 needed 53–180 s, a roughly 10-fold speedup. Even though a data-center A100 draws more instantaneous power than a single-socket EPYC (400 W versus 280 W, per vendor TDPs), the order-of-magnitude shorter runtime means the energy per study falls by approximately 80%. In a department that performs 2000 such motion-compensated scans a year, that difference translates to roughly 1 MWh saved, enough to pay the GPU’s electricity bill for the entire 5-year depreciation window.

7.5 Conclusion

We demonstrated the feasibility of a native GPU implementation of the GRICS method. By evaluating the CUDA-based implementation of the reconstruction kernel, which accounts for 70% to 80% of the total GRICS elapsed time, we achieved a speedup ranging

from $9.7\times$ to $13.7\times$ with an elapsed time ranging between 5.3 s and 18 s across six different datasets, having different problem sizes and MRI acquisition types. The GPU acceleration did not compromise the reconstruction quality. The roofline analysis highlights an improvement in the overall performance with the CUDA implementation, with an achievable performance of 250 GFLOP/s. The results show the potential applicability of the method in clinical settings.

Chapter 8

MRI Sequence Design: MPRAGE & MP2RAGE

8.1 k -space Sampling Randomization Strategies

Incoherent (random or pseudo-random) reordering of k -space measurements is a way to control how patient motion appears in MR data: instead of producing structured ghosts along phase encode directions, motion induced phase and displacement errors are dispersed across k -space so that residual artifacts are much more similar to noise. In MRI compressed-sensing, randomized undersampling improves incoherence and preserves recoverability in sparse models [34]. Within Cartesian 3D fast spin-echo, *T2 Shuffling* use the same idea by randomizing the k_y - k_z view order across the echo train and re-acquiring phase encodes throughout, then reconstructing the resulting 4D dataset with low-rank subspace and sparsity constraints to suppress coherent blur/ghosts while yielding multiple T_2 -weighted contrasts from a single scan [160]. More generally, DISORDER (*Distributed and Incoherent Sample Orders for Reconstruction Deblurring*) designs sample orders that are explicitly distributed and incoherent to perform joint retrospective motion estimation and reconstruction, leveraging coil-encoding redundancy; it requires no navigators or external sensors and has been demonstrated on volumetric brain exams with substantial motion [69]. In the non-Cartesian realm, golden-angle radial sampling provides a pseudo-random angular ordering that repeatedly traverses k -space center, enabling self-navigation and retrospective sorting into motion states; the XD-GRASP framework exploits this property to perform motion-resolved reconstructions under sparsity constraints, and the golden-angle family has since been reviewed and generalized extensively [161, 162]. While randomization improves incoherence and motion sensitivity, practical sequence engineering must respect gradient-moment continuity, RF/SAR, and steady-state constraints; for short-TE 3D UTE, the reordered 2D golden-angle (r2DGA) scheme was introduced to smooth encoding-gradient moment changes while preserving motion robustness—illustrating that view-order randomness often needs carefully constrained structure to avoid eddy-current-related artifacts [163]. In summary, the advantage of randomization is not simply due to the addition of diffuse noise, which would trivially support the denoising process. The key mechanism is its ability to improve the

conditioning of the motion estimation stage: by spreading the acquired k -space information more uniformly across the different motion states. Our acquisition strategy follows this line by designing randomness subject to hardware/state constraints and coupling it with motion-aware reconstruction.

8.2 The pseudo-randomization methods implemented : ReorderInfo Class

The choice of k -space sampling strategy can significantly affect the success of motion correction. Concretely, randomizing the acquisition order of k -space samples can enhance image quality in several ways [69]. First, distributing samples more evenly across different motion states helps to break the spatial coherence of motion-induced artifacts. Instead of producing structured ghosting patterns, randomization transforms these effects into pseudo-random noise. Moreover, acquiring k -space lines in a random or pseudo-random order can decorrelate data collection from physiological motion and, therefore, suppress ghosting or blurring without extending the scan. Early work by [164] showed that completely random Cartesian phase-encode re-ordering reduced translation-induced artifacts in phantoms. Extending the concept to free-breathing 3D Cartesian acquisitions, [165] used Quasi-Random numbers to reorder the lines and partitions, in order to desynchronize sampling from the respiratory cycle, yielding fewer residual ghosts. Most recently, [166] introduced a semi-randomized frequency-selective space-filling curve that significantly reduced physiological-induced artifacts in vivo while preserving steady-state contrast.

In the IDEA environment, k -space acquisition ordering is managed by the ReorderInfo class. In this work, we implemented ReorderInfoDisorder, a subclass that inherits from ReorderInfo. Three pseudo-random methods were implemented:

- Shuffle Lines (SL): Randomization was introduced only on the lines dimension, while the partition sampling was kept sequential.
- Shuffle All (SA): Randomization was performed on both Phase Encoding dimensions. This scheme can also be implemented in elliptical mode, where the peripheral corners of k -space are omitted during acquisition. This mode is called Elliptical Shuffling (ES).

Randomization was performed using the Fisher-Yates algorithm [167]. For SA (and ES), the samples closest to the center of k -space were acquired at the center of the train.

Next we present some useful functions that were implemented in this class.

```
1
2 enum DISORDER_MODE {
3     DEFAULT_ORDER = 0,
4     SHUFFLE_LINES = 1,
5     ELLIPTICAL_SHUFFLING = 2,
6     SHUFFLE_ALL=3
7 };
```

Listing 8.1: Reordering Modes

```

1 class ReorderInfoDisorder : public ReorderInfo
2 {
3 public:
4     ReorderInfoDisorder(long lMaxNoOfReorderIndices = -1,
5                         long lCalcBufferSize = -1);
6
7     virtual BOOL reorder3dE(MrProt* pMrProt, SeqLim* pSeqLim);
8     virtual ~ReorderInfoDisorder() {};
9
10    void setDisorderMode(DISORDER_MODE mode);
11    long getDisorderMode() const;
12
13    void setReorderMode(long orderLINPAR,
14                      REORDERING_MODE outerLoop,
15                      REORDERING_MODE innerLoop);
16
17 protected:
18     DISORDER_MODE m_eDisorderMode ;
19     REORDERING_MODE m_eReorderingMode ;
20     NAVIGATOR_NUMBER m_NbOfNavig ;
21 };

```

Listing 8.2: ReorderInfoDisorder Class.

```

1 OrderLINPAR = pMrProt->wipMemBlock().alFree[WIP_LINPARSelection];
2 Reorder.setReorderMode(OrderLINPAR, LINEAR_DESCENDING, LINEAR_ASCENDING);
3
4 if ( pMrProt->wipMemBlock().alFree[WIP_Disorder] == WIP_Disorder_Off )
5 {
6     Reorder.setDisorderMode(DEFAULT_ORDER);
7 }
8 else if ( pMrProt->wipMemBlock().alFree[WIP_Disorder] == WIP_Disorder_Lines )
9 {
10    Reorder.setDisorderMode(SHUFFLE_LINES);
11 }
12 else if ( pMrProt->wipMemBlock().alFree[WIP_Disorder] ==
13           WIP_Disorder_Elliptical_Shuffling )
14 {
15    Reorder.setDisorderMode(ELLIPTICAL_SHUFFLING);
16 }
17 else if ( pMrProt->wipMemBlock().alFree[WIP_Disorder] == WIP_Disorder_Shuffle_All )
18 {
19    Reorder.setDisorderMode(SHUFFLE_ALL);
20 }
21 else
22 {
23    Reorder.setDisorderMode(DEFAULT_ORDER);
24 }

```

Listing 8.3: Reordering Selection.

```

1 if (m_bPATAActive)
2 {

```

```
3 // gives the correct result even if iPAT is disabled
4 lNbLines      = getNoOfPATLines()      + getNoOfPATRefLinesInGap();
5 lNbPartitions = getNoOfPATPartitions() + getNoOfPATRefPartitionsInGap();
6 }
7 else
8 {
9     lNbLines      = getLinesToMeasure();
10    lNbPartitions = getPartitionsToMeasure();
11 }
```

Listing 8.4: Computation of lines and partitions with or without iPAT

```
1 void ReorderInfoDisorder::array_shuffle(long arr[], long n, long SeedPoint)
2 {
3     srand(SeedPoint);
4     for (long i = n - 1; i >= 1; --i)
5     {
6         long j = rand() % (i + 1);
7         long temp = arr[j];
8         arr[j] = arr[i];
9         arr[i] = temp;
10    }
11 }
```

Listing 8.5: Fisher-Yates shuffling of a 1D array.

```
1 /* Randomly permute all the indices of the Reorder array */
2 for (lJ = 0; lJ < lNoOfReorderIndices; lJ++)
3 {
4     al_indiceOrder[lJ] = lJ;
5 }
6
7 array_shuffle(al_indiceOrder, lNoOfReorderIndices, 2);
8
9 for (lJ = 0; lJ < lNoOfReorderIndices; lJ++)
10 {
11     al_Lines[lJ]      = getLinNo(al_indiceOrder[lJ]);
12     al_Partitions[lJ] = getParNo(al_indiceOrder[lJ]);
13 }
14
15 for (lJ = 0; lJ < lNoOfReorderIndices; lJ++)
16 {
17     setLinNo(lJ,      al_Lines[lJ]);
18     setParNo(lJ,      al_Partitions[lJ]);
19 }
```

Listing 8.6: Case study : SA

```
1 /* Add navigators at the beginning of each shot */
2 setNoOfReorderIndices(lNoOfReorderIndices + lNoNavigToAdd);
3
4 cpt = 0;
5 if (m_NbOfNavig > 0)
6 {
```

```

7   for (lJ = 0; lJ < lNoOfReorderIndices; lJ++)
8   {
9       if (lJ % lNbInInnerCycle == 0)
10      {
11          for (long k = 0; k < m_NbOfNavig; k++)
12          {
13              setLinNo(lJ + cpt, getKSCenterLin());
14              setParNo(lJ + cpt, getKSCenterPar());
15              setFreeBit1(lJ + cpt, true);
16              cpt++;
17          }
18      }
19
20      setParNo(lJ + cpt, al_Partitions[lJ]);
21      setLinNo(lJ + cpt, al_Lines[lJ]);
22      setFreeBit1(lJ + cpt, false);
23  }
24  }
    
```

Listing 8.7: Navigators Insertion at the beginning of each RAGE shot in the SA.

8.3 *k*-space Center Line Navigator and Motion Eigenmodes extraction

The center *k*-space line navigator ($k_y = k_z = 0$ phase-encodes) was added at the beginning of each shot of the two sequences. It contains information about motion in the whole FOV, since it provides a projection of the image content onto the k_x (frequency-encoding) dimension. In additional scans (datasets D_7 - D_{10}), 2 and 3 navigators were collected in 2 and 3 orthogonal directions successively, by applying a rotation matrix.

In an acquisition with an array of $N_c = 32$ coils, where each coil c has a specific sensitivity, each c records a different image projection. The processing of navigator data is achieved using a method similar to that in [168], consisting of the following steps:

1. Choosing a reference central *k*-space line profile, which is the median of all profiles over time.
2. Non-rigid 1D registration of the navigator using an optical flow-based algorithm [?] for each coil c .
3. Principal component analysis (PCA) of the resulting set of 1D + T (one spatial dimension (x), and time) motion fields $u_c(x, t)$.
4. Extract the motion eigenmodes, as they are the right singular vectors of the matrix P see (1).

$$P = [U_1, \dots, U_{N_c}]^T, \quad U_c = \begin{bmatrix} u_c(x_1, t_1) & \dots & u_c(x_1, t_{N_k}) \\ \vdots & & \vdots \\ u_c(x_{N_x}, t_1) & \dots & u_c(x_{N_x}, t_{N_k}) \end{bmatrix} \quad (8.1)$$

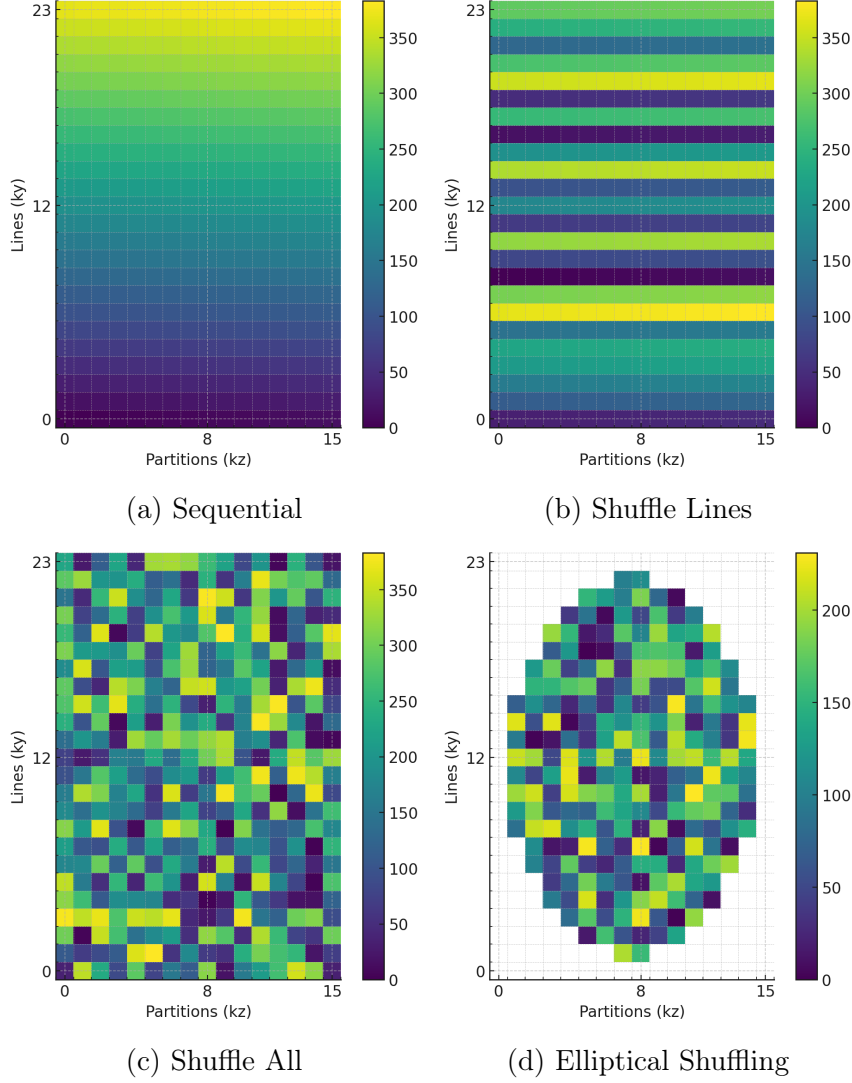


Figure 8.1: k -space acquisition reordering. The color gradient stands for the acquisition order.

where N_k is the number of navigator samples. Under the assumption that the 1D + T motion fields are highly correlated, the PCA is applied to the matrix P , and the motion eigenmodes are the right singular vectors of P , or the eigenvectors of the symmetric matrix $P^T P$.

8.4 Motion Clustering and Motion States Determination

To reduce the high computational cost and complexity of the GRICS problem, the number of motion states N_t is reduced by grouping k -space data that are acquired in similar motion states. First, the motion signals are quantized onto $L = 32$ levels, resulting in a number

$N'_t \leq 2^L$. The total k -space energy contained in each of the N'_t motion states is calculated. A reduced number $N''_t = 12$ is selected as those motion states that contain the largest k -space energy. Data from remaining states are grouped with the closest to the selected ones, see Figure 8.2.

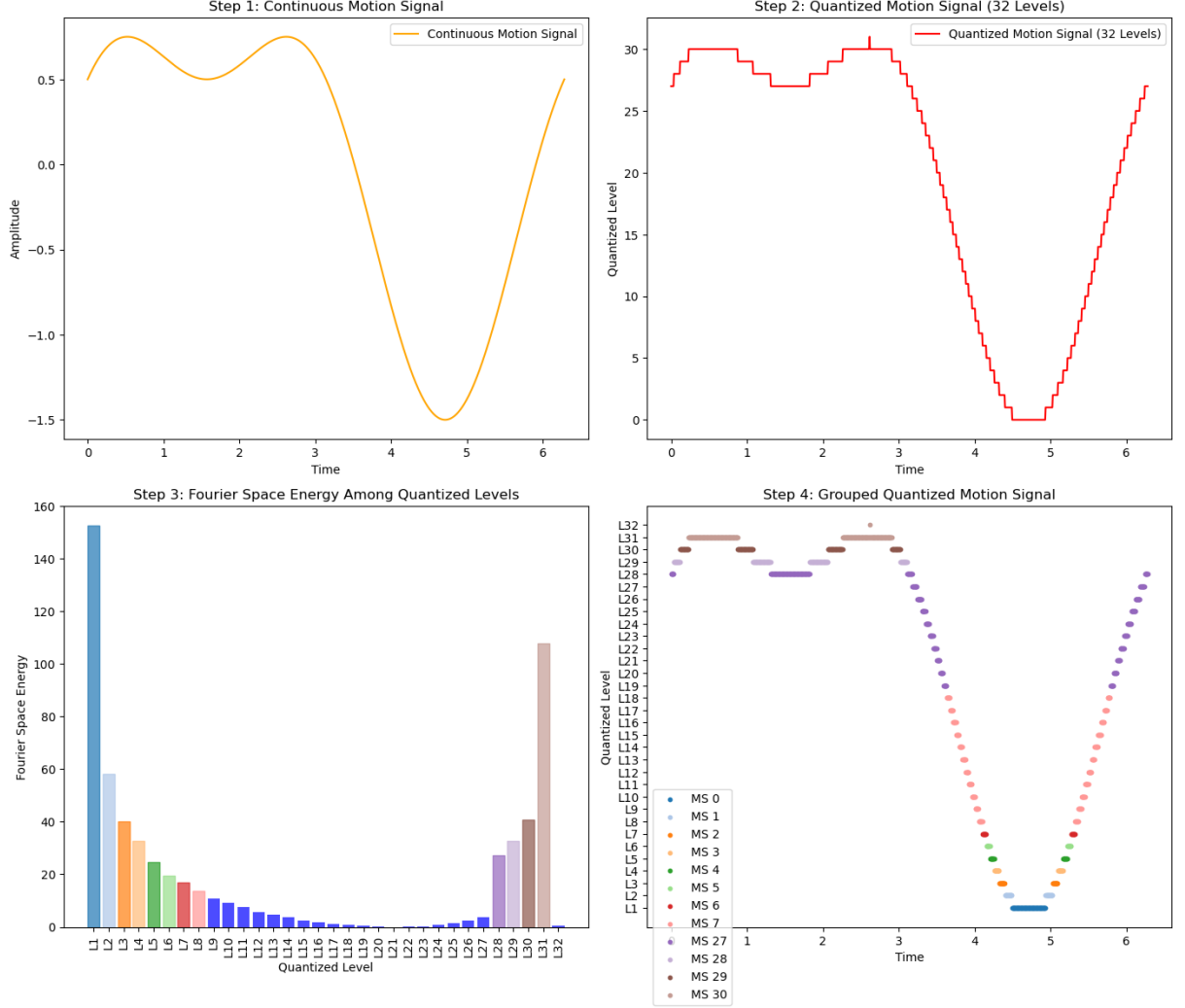


Figure 8.2: Pipeline for motion states determination. (Step 1) The continuous motion signal illustrating physiological motion during MR acquisition. (Step 2) Quantization of the signal into 32 discrete amplitude levels. (Step 3) Computation of Fourier-space energy per quantized level, and highlighting the highest N''_t , here 12, levels. (Step 4) Describing the motion signal with only N''_t motion states.

Chapter 9

Motion Compensated Reconstruction for brain MRI at 7T

9.1 Data Acquisition

The MRI raw data were acquired on a 7T MR scanner (Magnetom 7T, Siemens Healthineers, Erlangen, Germany) equipped with an in-house-built head RF coil featuring 8 transmit and 32 receive channels [157]. The subjects in this study were healthy volunteers; they provided written informed consent. The study protocol was approved by the *Comité de protection des personnes Sud-Méditerranée IV*. 10 volunteers were scanned with variable FOV and resolutions. Brain scans were performed using a 3D MPRAGE or MP2RAGE sequences, see Table 9.1. To distribute k -space data more evenly through motion states, which is favorable for motion estimation, the sequences code was modified by introducing the randomization of the sampling order, as explained in the previous Chapter. To preserve the T_1 contrast within each shot, the samples closest to the center of k -space were acquired first. Randomization was performed using the Fisher-Yates algorithm [167]. To evaluate the effect of changing the sampling scheme on image quality, owing to T_1 contrast but also eddy currents caused by phase encoding jumps, a phantom scan was performed with the different schemes and compared to the standard sampling order using a 3D MPRAGE acquisition at 0.9 mm isotropic resolution with iPAT factor 3. Moreover, additional phantom experiments were performed using an MP2RAGE sequence. Two MP2RAGE acquisitions were done: a 0.9 mm isotropic and a 0.6 mm isotropic slab-selective, both accelerated (iPAT=3), see Table 9.2.

Table 9.1: The datasets of the study. SS stands for Slab-Selective. A_F is the GRAPPA acceleration factor. N_x , N_y and N_z stand respectively for the number of samples in frequency encoding direction, phase-encoding direction in the image plane, and in the direction of slices.

Dataset	Sequence	Resolution (mm^3)	Shuffle Mode	N_x	N_y	N_z	A_F
D_1	MPRAGE	$1 \times 1 \times 1$	ES	256	208	160	1
D_2	MPRAGE	$1 \times 1 \times 1$	SL	256	208	160	1
D_3	MPRAGE	$1 \times 1 \times 1$	ES	256	208	160	2
D_4	MPRAGE	$1 \times 1 \times 1$	ES	256	208	160	2
D_5	SS MPRAGE	$0.6 \times 0.6 \times 0.6$	SL	448	363	208	3
D_6	MPRAGE	$0.9 \times 0.9 \times 0.9$	SL	256	256	192	2
D_7	MPRAGE	$0.9 \times 0.9 \times 0.9$	SL	256	226	176	3
D_8	MP2RAGE	$0.9 \times 0.9 \times 0.9$	SL	256	256	176	3
D_9	MP2RAGE	$0.8 \times 0.8 \times 0.8$	SL	288	234	192	3
D_{10}	MP2RAGE	$0.9 \times 0.9 \times 0.9$	SL	256	256	176	3

Table 9.2: Phantom acquisitions used to evaluate the effect of k -space sampling order on image quality.

Phantom	Sequence	Resolution (mm^3)	AF	Shuffle modes
A	MPRAGE	$0.9 \times 0.9 \times 0.9$	3	Sequential, SL, SA
B	MP2RAGE	$0.9 \times 0.9 \times 0.9$	3	Sequential, SL, SA
C	SS MP2RAGE	$0.6 \times 0.6 \times 0.6$	3	Sequential, SL, SA

9.2 Comparison with a Deep-Learning based solution: MC-NET

For comparison purposes, a deep-learning based method, MC-NET introduced by Zhang et al. in [169], was tested. The authors described the Motion Correction-Net (MC-Net), a lightweight (approximately 8.6 M parameters) UNet-derived auto-encoder designed for retrospective slice-based correction of rigid in-plane head motion in brain MRI. MC-Net was trained using brain MRI scans acquired with an MPRAGE sequence at 3T. The training consisted of two sequential stages: an initial 200-epoch pass minimized an ℓ_1 loss to suppress gross artifacts, after which the network was fine-tuned with a hybrid loss that augments the ℓ_1 term with a TV regularizer to restore edge sharpness. Training employed 13 700 synthetically corrupted axial MPRAGE slices generated from 35 motion-free brains using 25 realistic motion trajectories, magnified eight-fold. MC-Net casts

retrospective slice restoration as supervised image-to-image regression. Let $\mathcal{S} = \{(I_c, I_0)\}$ be paired motion-corrupted (I_c) and motion-free reference (I_0) slices and $F_{\Theta}(\cdot)$ the UNet-style mapping with learnable parameters Θ . Training seeks for the motion parameter set Θ^* minimizing

$$\Theta^* = \arg \min_{\Theta} \frac{1}{|\mathcal{S}|} \sum_{(I_c, I_0) \in \mathcal{S}} \mathcal{L}(F_{\Theta}(I_c), I_0),$$

with the hybrid loss

$$\mathcal{L}(I, I_0) = \underbrace{\alpha \sum_{i,j} |I_{ij} - I_{0,ij}|}_{\ell_1} + \underbrace{\beta \sum_{i,j} [(I_{i+1,j} - I_{ij})^2 + (I_{i,j+1} - I_{ij})^2]^{1.25}}_{\text{TV}^{1.25}}.$$

Stage 1 uses $(\alpha, \beta) = (1, 0)$ to suppress gross artifacts via pure ℓ_1 minimization; Stage 2 fine-tunes the network with $(\alpha, \beta) = (1, 1)$, activating the fractional-order TV term to recover edge sharpness and remove residual streaks [169]. In this work, we directly applied the pretrained model from the authors to our uncorrected MPRAGE/MP2RAGE images at 7T, which is available at https://github.com/MRIMoCo/DL_Motion_Correction.

9.3 Quantitative Image Quality Metrics : The Sharpness Index - SI

The common indicators for image quality evaluation include image noise, artifacts, and sharpness. This last criterion refers to the clarity of the borders of various parts of an image. Sharpness evaluation can be performed in the spatial or the spectral domain. In this work, we used the Sharpness Index introduced in [170]. In this work, Moisan and Blanchet suggested a Sharpness Index (SI) closely related to the Global Phase Coherence (GPC) but with much lower computational cost, using Gaussian random fields instead of random phase images. This was done by estimating the probability that a random image u has a given Total Variation $TV(u)$, and then by computing $\mathbb{E}(TV(u))$ and $\mathbb{V}ar(TV(u))$. SI is mathematically expressed as follows:

$$SI(u) = -\log_{10} \left(\Phi \left(\frac{\mu - TV(u)}{\sigma} \right) \right)$$

where:

$$\mu = \mathbb{E}[TV(u)], \quad \sigma^2 = \mathbb{V}ar(TV(u)),$$

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^x e^{-t^2/2} dt$$

In [148], SI showed a great correlation with the radiological scores compared to other well-known metrics, such as Average Strength Enhancement, Average Edge Strength, and Sharpness Enhancement. Here, SI was used to quantitatively evaluate the efficiency of

GRICS reconstructions under different motion models (rigid/non-rigid), with one, two, or three navigators. MC-NET reconstructions were assessed with the same metric. For each 3D reconstruction, the SI was first computed on a slice-by-slice basis. To obtain a single value per volume and reconstruction method, we then calculated the arithmetic mean of the slice-wise SI values.

Moreover, a statistical evaluation of the SI across the ten datasets was elaborated. For each dataset D_i , four reconstructions were performed: without correction (W), with rigid MoCo (R), with non-rigid MoCo (NR), and using the DL approach MC-NET (MC). To assess statistical significance, we employed the Wilcoxon signed-rank test. Given paired SI values ($SI_{i,j}, SI_{i,k}$), the differences

$$d_i^{(j,k)} = SI_{i,j} - SI_{i,k}$$

were ranked according to $|d_i^{(j,k)}|$, and the Wilcoxon statistic was computed as

$$W = \min(W^+, W^-),$$

where W^+ and W^- denote the sum of positive and negative signed ranks, respectively.

9.4 Results

9.4.1 Phantom Evaluation : Impact of the sampling scheme

In Figure 9.1, phantom experiments show that while the SL strategy preserved uniformity and structural fidelity comparable to conventional sequential ordering, the SA and ES approaches introduced some artifacts. Specifically, images exhibited structured intensity modulations and ringing, indicative of PSF distortion and signal non-uniformity. Figure 9.2 shows phantom MP2RAGE reconstructions. With SL, the reconstructed phantom interior and boundary definition remained sharp for both acquisitions. However, the SA/ES strategies introduced wave-like modulations and ringing, particularly visible at the phantom center and edges.

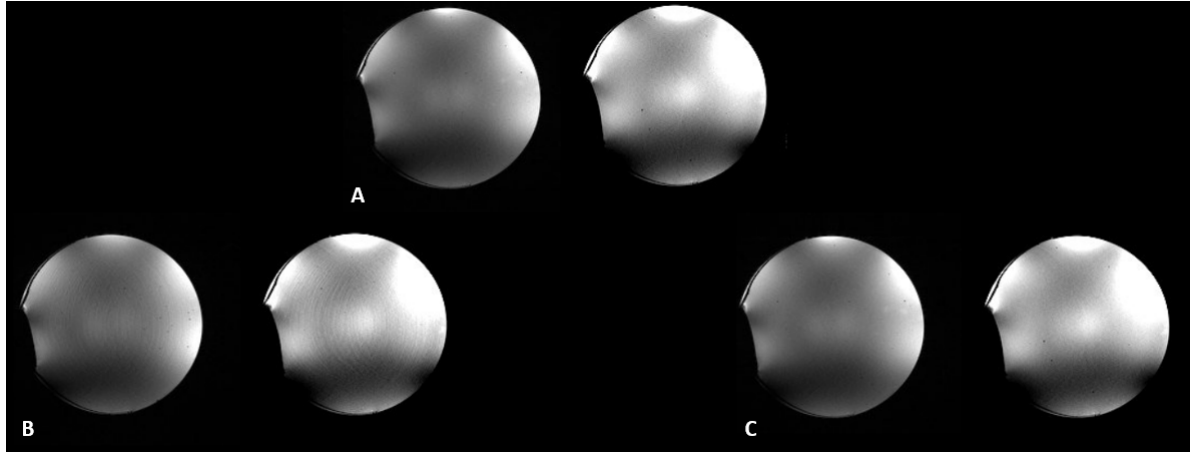


Figure 9.1: Phantom evaluation of the three sampling strategies. For each sampling scheme, the right image is the same as the left image but with magnified window level, to highlight subtle artifacts. A: Without shuffling. B: Shuffle All (SA). C: Shuffle Lines (SL).

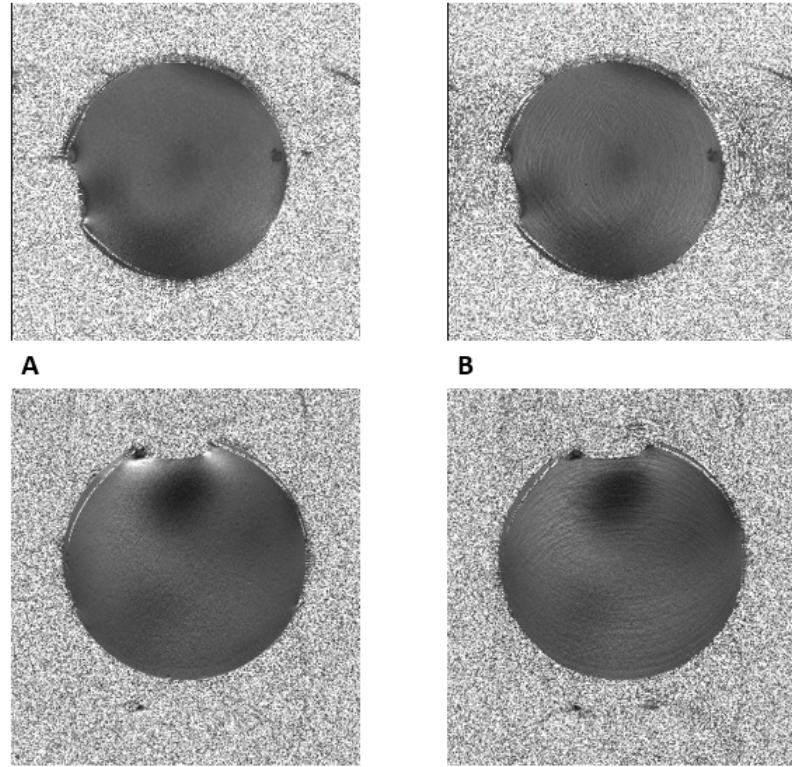


Figure 9.2: Phantom MP2RAGE (iPAT=3) reconstructed with different k -space randomization strategies, from both 0.9 mm isotropic acquisition (top row) and 0.6 mm isotropic slab-selective acquisition (bottom row). (A): Shuffle Lines (SL). (B) Shuffle All (SA).

9.4.2 Impact of motion model

Figure 9.3 illustrates sagittal MRI scans of the dataset D_1 using three different reconstruction strategies to correct for head motion. In (A), the uncorrected reconstruction exhibits pronounced motion artifacts, mainly blurring and ghosting across the frontal lobe (yellow square). These artifacts compromise fine anatomical details and reduce the overall sharpness of the image. (B) shows the motion-corrected reconstruction with a rigid motion model, which substantially mitigates motion-related blurring. The cortical contours, particularly around the gyri and sulci, are notably sharper, and the signal homogeneity across the brain is improved. This improvement highlights the effectiveness of the rigid model in this case. (C) presents the motion-corrected reconstruction with a non-rigid motion model, which, in this specific case, does not yield additional improvements beyond the rigid motion model. Residual motion artifacts remain visible, indicating that the non-rigid approach did not successfully compensate for the type of motion encountered in this dataset. The subject experienced a peak-to-peak shift of 4.85 mm in translation and 7.03 degrees in rotation.

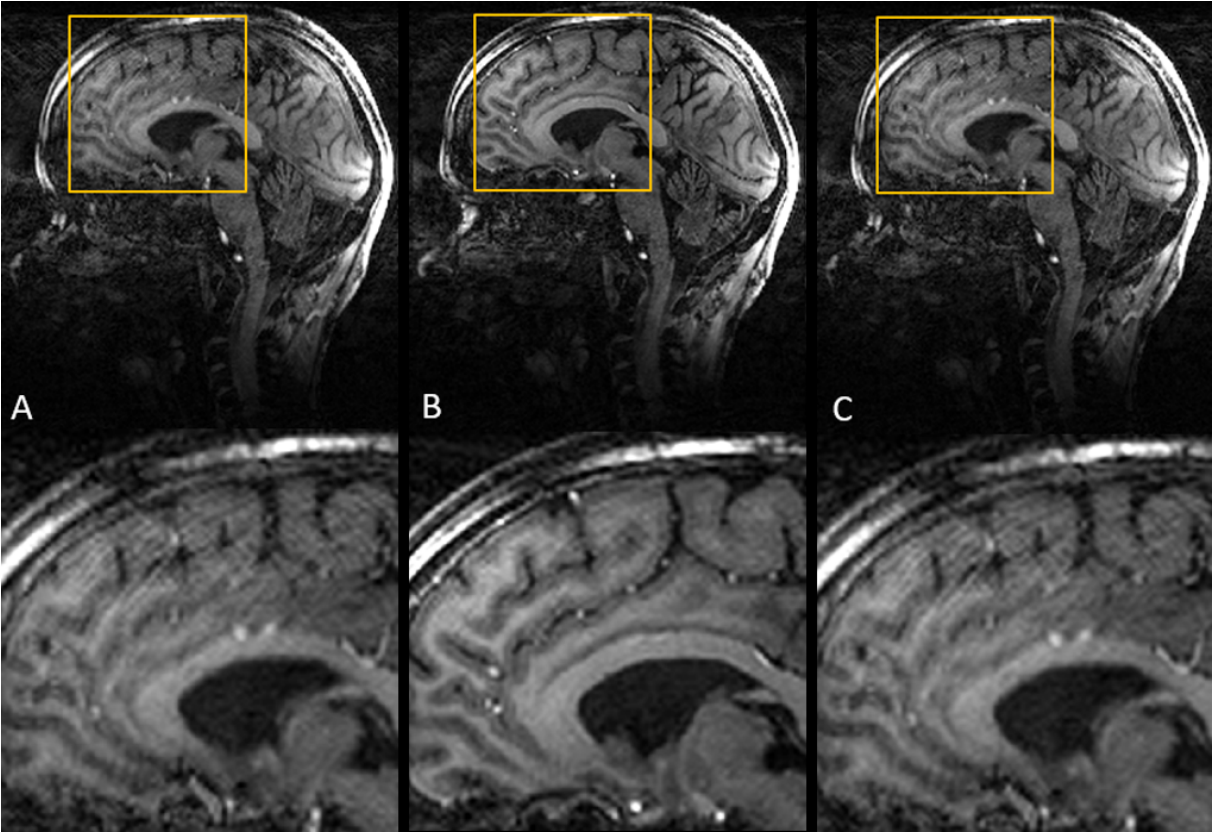


Figure 9.3: GRICS Reconstructions for dataset D_1 . A) Uncorrected reconstruction. B) Motion Corrected reconstruction with rigid model. C) Motion Corrected reconstruction with non-rigid model.

Figure 9.4 presents sagittal reconstructions from dataset D_8 , with rigid correction (A) and non-rigid correction (B). While rigid correction reduces some of the gross motion

artifacts, residual ringing is still apparent, particularly along the parietal lobe (the blue rectangle in the images of the second row). In contrast, the non-rigid approach improves image sharpness; it reduced considerably the artifacts left uncorrected in the rigid-based reconstruction. This finding suggests that the proposed non-rigid motion model can help improve the image sharpness for high-resolution brain imaging. Depending on the motion during acquisition, the non-rigid MoCo may provide a better reconstruction than the rigid MoCo.

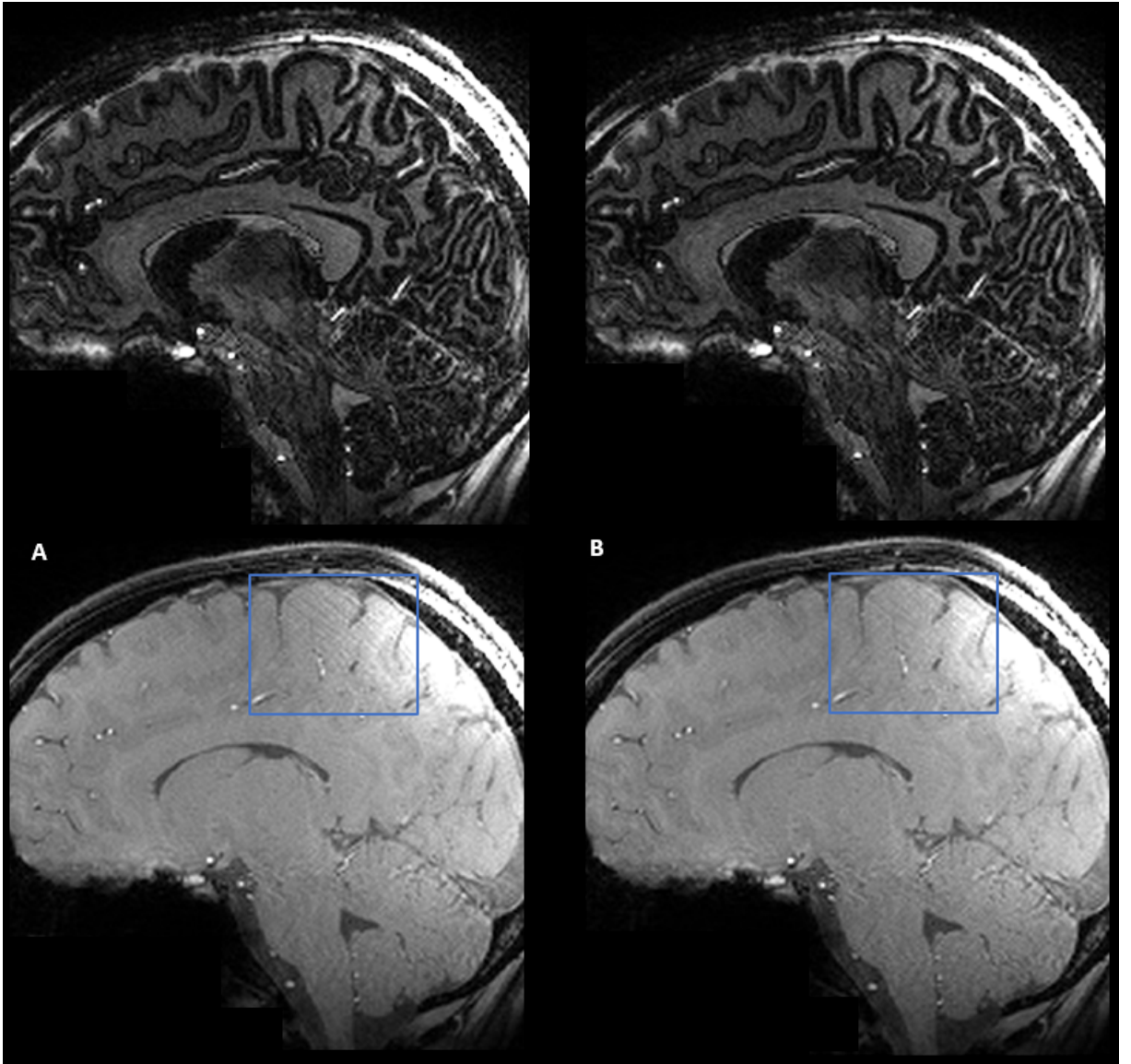


Figure 9.4: GRICS reconstructions for dataset D_8 . A: Rigid correction. B: Non rigid correction. Top row: MP2RAGE images at TI_1 . Bottom row: MP2RAGE images at TI_2 .

9.4.3 Impact of Navigators

Figure 9.5 shows reconstructions using either one navigator k_x (B and C), two navigators k_x and k_y (D and E), and three navigators in the three spatial directions (F and G). A represents the uncorrected reconstruction. B, D, and F use a rigid motion model, whereas C, E, and G use a non-rigid model. In A, motion artifacts are most prominent in the frontal (orange square) and parietal (green square) lobes. When using a single k_x navigator or the combination of k_x and k_y , both rigid and non-rigid motion corrections substantially reduce the artifacts and enhance the sharpness of the corrupted anatomical structures in A. However, with three navigators, although the ringing artifact in the parietal lobe is diminished, the frontal lobe artifacts remain in F and G, indicating that including a k_z navigator may introduce a suboptimal motion estimation and considerably reduce reconstruction quality. To confirm this, Figure 9.6 illustrates reconstructions using a single navigator in each of the three possible directions (k_z , k_y , or k_x). Qualitatively, reconstructions using the k_x or k_y navigator reduce significantly ringing artifacts, whereas the k_z navigator alone does not remove the ringing artifact in the parietal lobe.

9.4.4 Quantitative Assessment and Comparison With MC-NET

Quantitative evaluation using the sharpness index is shown in Table 9.4. A_D is the peak-to-peak maximal displacement field value in mm, obtained from the navigator processing described in section 2.2. Motion amplitude was low in a considerable number of the datasets, where $A_D \leq 2$ mm in 6 datasets. Hence, motion artifacts were not clearly noticeable in these acquisitions. The 4 remaining datasets presented moderate artifacts, where $2 \text{ mm} \leq A_D \leq 5 \text{ mm}$. The most severe artifacts were noticed on the D_1 acquisition, Figure 9.3. The SI values indicate that the GRICS reconstruction, either under rigid or non-rigid motion assumptions, improves quantitatively the image quality, even in the cases where motion was minimal. In some acquisitions, the non-rigid motion model was more beneficial and captured more relevant motion information, resulting in a higher SI. MC-NET consistently improved image sharpness in all six datasets for which a valid motion-corrected reconstruction was obtained. In the remaining four datasets, however, the network failed to provide a plausible solution: large intensity fluctuations appeared in the output, severely degrading the reconstructed image. An example of failed MC-Net reconstruction is shown in Figure 9.7. These cases are therefore reported as NA in Table 9.4.

The statistical evaluation of the SI across the datasets of the study is reported in Table 9.3.

Both rigid and non-rigid motion correction provided statistically significant improvements over the uncorrected reconstruction ($p = 0.0019$ and $p = 0.0098$, respectively). In contrast, no significant difference was observed between rigid and non-rigid corrections ($p = 1.0$), indicating comparable performance. However, in datasets (D_3 , D_5 , D_6), the non-rigid model provided additional SI gains of 2–4%. In some cases, non-rigid GRICS improved image sharpness compared to rigid GRICS. Nevertheless, non-rigid GRICS requires tuning of the smoothness penalty $\mu R(\alpha)$ to avoid overfitting noise. Due to the complexity of the model, the non-rigid based GRICS requires longer runtime; GPU ac-

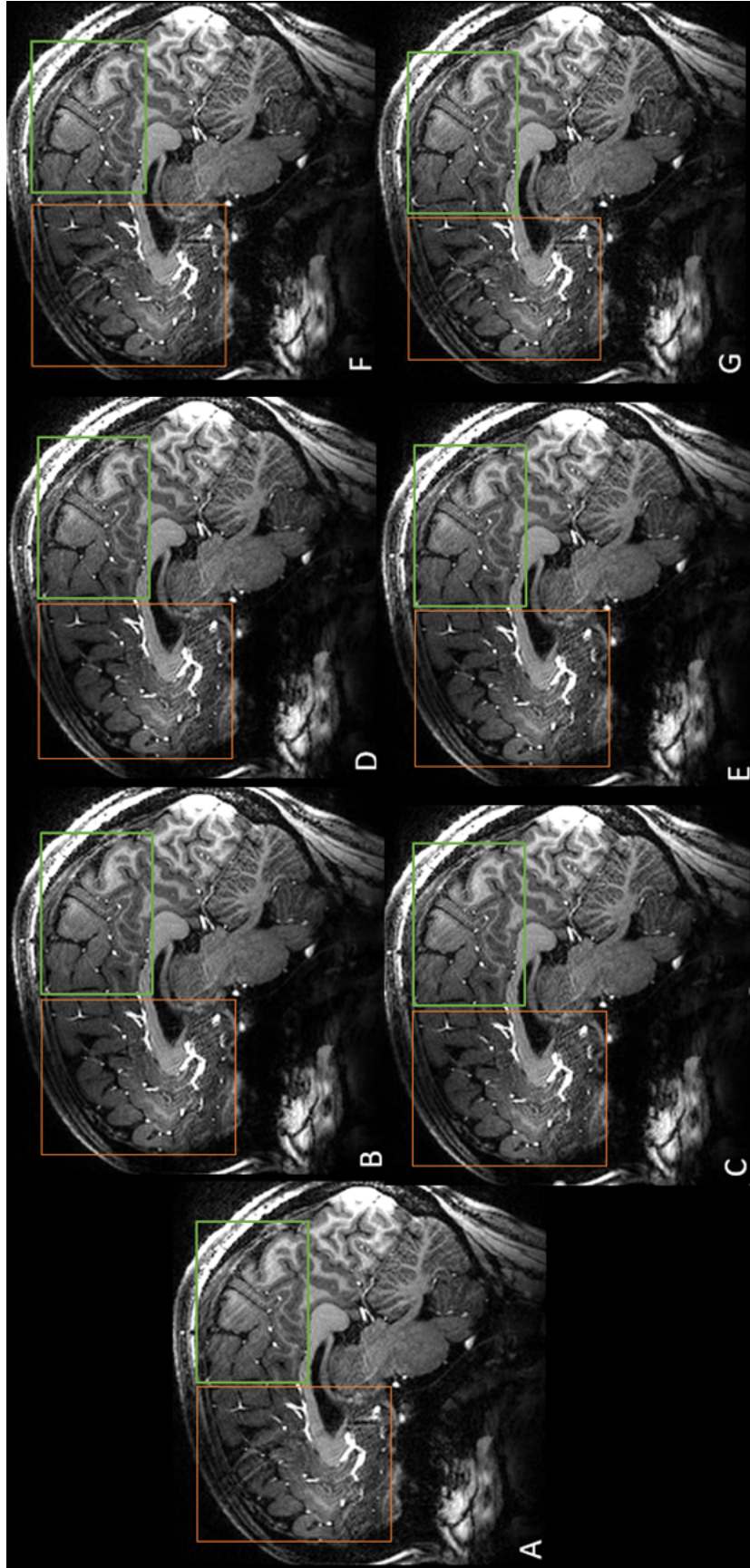


Figure 9.5: GRICS reconstructions for dataset D_7 . A : Uncorrected reconstruction. B : Rigid correction with 1 navigator. C : Non-Rigid correction with 1 navigator. D : Rigid correction with 2 navigators. E : Non-Rigid correction with 2 navigators. F : Rigid correction with 3 navigators. G : Non-Rigid correction with 3 navigators.

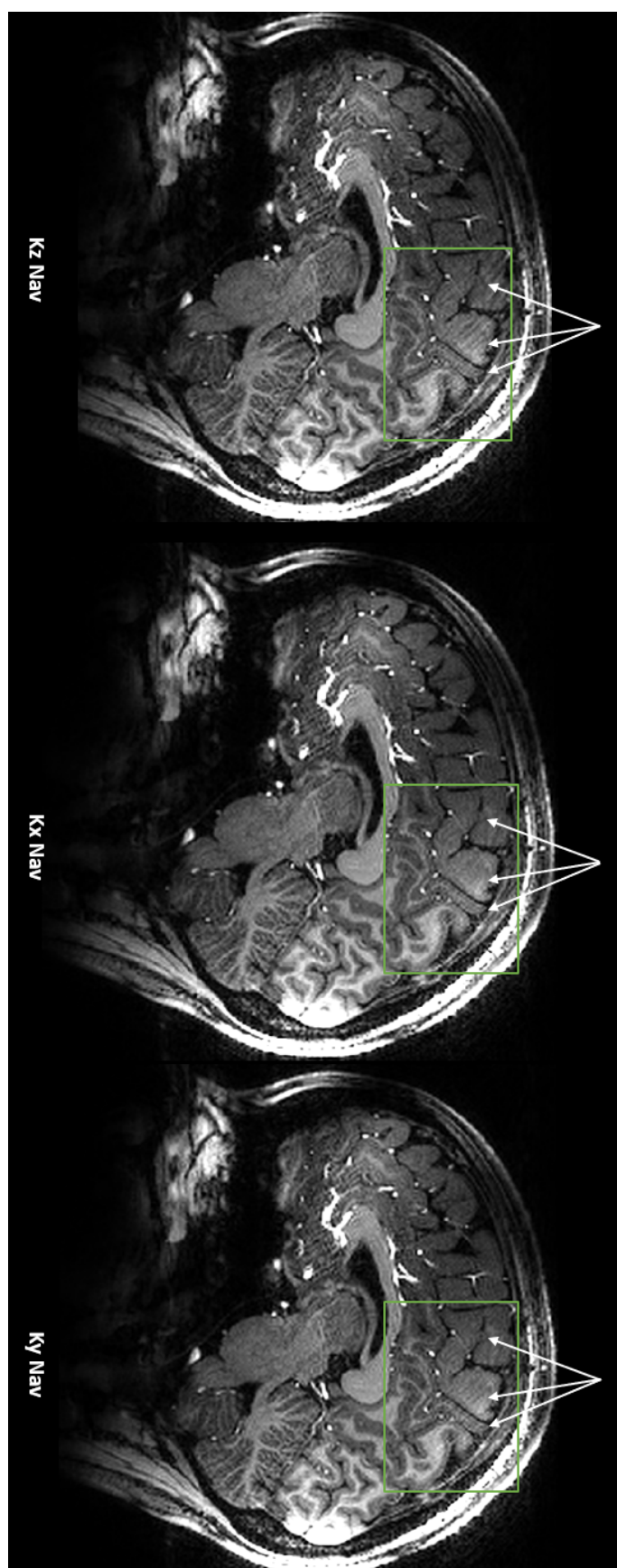


Figure 9.6: The impact of the navigator orientation on the reconstructed image.

Table 9.3: Wilcoxon signed-rank test results comparing Sharpness Index (SI) across reconstruction methods. N indicates the number of datasets included in the test. Significant differences ($p < 0.05$) are highlighted in bold. Since MC-NET provided interpretable reconstruction in only 6 datasets, we did not perform a statistical analysis involving this method (NA).

Comparison	N	Wilcoxon Statistic	p -value	Significance
Rigid vs Without	10	0.0	0.0019	Yes
Non-Rigid vs Without	10	3.0	0.0098	Yes
Non-Rigid vs Rigid	10	27.0	1.0000	No

celeration may accelerate the reconstruction process [171]. For MC-NET, results were available in only six datasets due to reconstruction failures in the four remaining cases. Hence, we did not perform a statistical analysis of this method.

Qualitatively, Figure 9.8 compares the motion-corrected reconstruction quality on datasets D_1 (first row) and D_2 (second row), which both presented visual ringing and blurring artifacts. GRICS eliminated the bright cortical rims and background streaks that MC-NET occasionally left intact; it preserved sulcal sharpness and deep-grey boundaries more faithfully and deleted the motion artifacts presented in Figure 9.8 (A). For both datasets, the anatomical structures are much sharper in the GRICS solution, and the blurring is considerably reduced compared to the MC-NET solution. Although MC-NET is faster and requires no external hardware, our qualitative evaluation confirms that explicit modeling delivers a more efficient reconstruction when accurate sensor traces are available.

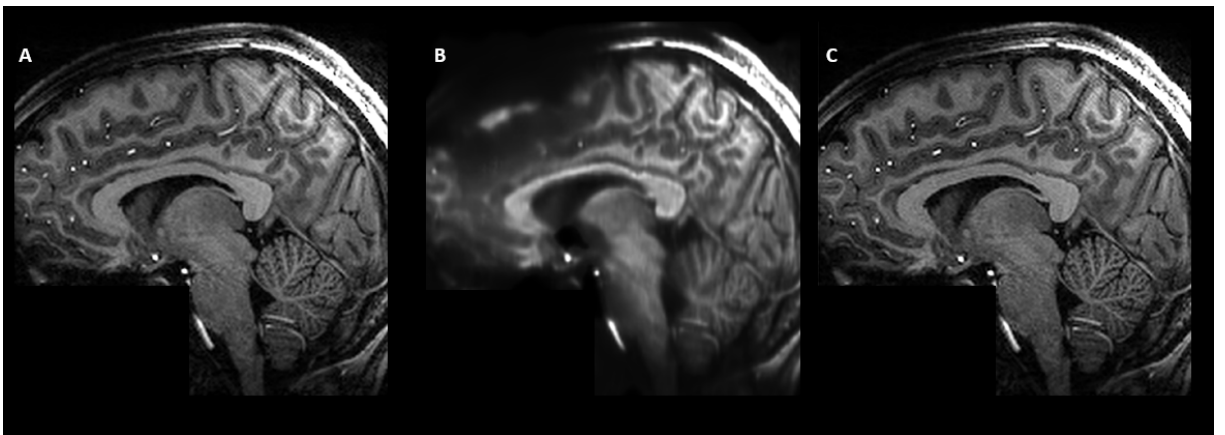


Figure 9.7: An example of an MC-NET failed reconstruction(B). (A) shows the uncorrected reconstruction. (C) is the Rigid GRICS reconstruction.

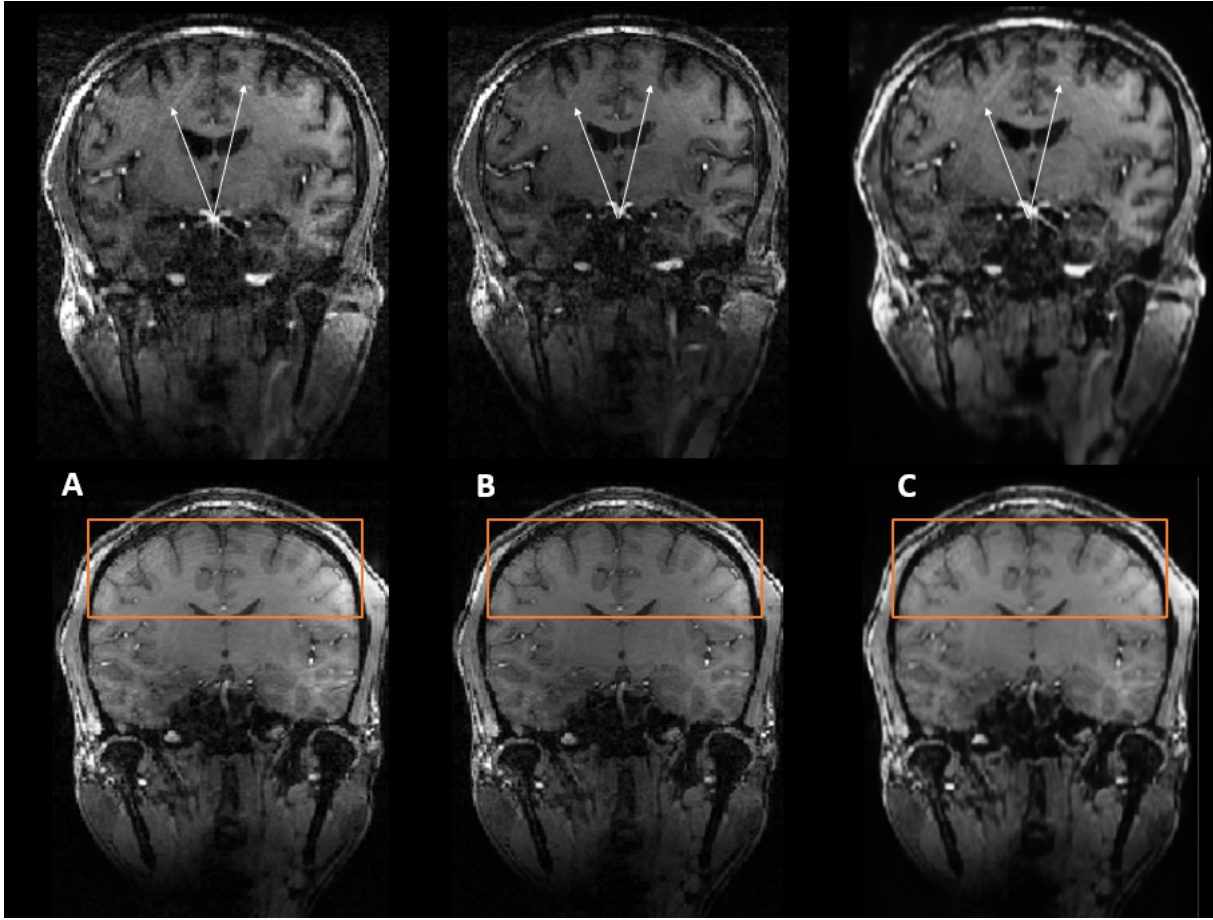


Figure 9.8: GRICS vs MC-NET. A: GRICS without MoCo. B: GRICS with MoCo. C: MC-NET. Datasets D_1 and D_2 .

Table 9.4: Sharpness Index (Higher is better) & Motion Model. A_D is the peak-to-peak maximal displacement field in mm. NA*: Not Available: excluded from analysis due to failed reconstruction

Dataset	MoCo Type	SI	$A_D(mm)$
D_1	Without	2230	4.37
	Rigid	2630	
	Non-Rigid	2310	
	MC-NET	2420	
D_2	Without	3260	3.5
	Rigid	3610	
	Non-Rigid	3420	
	MC-NET	3585	
D_3	Without	5104	2.2
	Rigid	5649	
	Non-Rigid	5843	
	MC-NET	5670	
D_4	Without	2906	1.20
	Rigid	2988	
	Non-Rigid	2925	
	MC-NET	2940	
D_5	Without	16400	1.62
	Rigid	18734	
	Non-Rigid	19371	
	MC-NET	NA*	
D_6	Without	5684	1.93
	Rigid	7496	
	Non-Rigid	7673	
	MC-NET	7224	
D_7	Without	4718	1.97
	Rigid	4992	
	Non-Rigid	4924	
	MC-NET	4973	
D_8	Without	6141	1.98
	Rigid	6519	
	Non-Rigid	6586	
	MC-NET	NA*	
D_9	Without	4742	3.28
	Rigid	5128	
	Non-Rigid	5214	
	MC-NET	NA*	
D_{10}	Without	5881	1.11
	Rigid	6295	
	Non-Rigid	6030	
	MC-NET	NA*	

9.5 Discussion

Our phantom experiments highlight a trade-off between motion-robustness and sequence-induced artifacts. Randomizing both PE dimensions with SA or ES schemes introduced ringing artifacts that were absent in the default sampling strategy. These artifacts most likely originate from (i) large, irregular k_z gradient jumps that drive eddy currents and (ii) intra-shot T_1 -weighting inconsistencies when the inversion-recovery read-out is no longer sequential. Since the SA scheme can introduce additional in vivo artifacts on top of motion-induced ones, we opted for the more stable SL scheme for the later subjects ($D_5 - D_{10}$). Because the SL strategy preserves a sequential k_z ordering while still randomizing in-plane phase encodes, it maintains image uniformity and provides sufficient sampling incoherence for GRICS to resolve motion. SL produces a more stable point-spread function while maintaining sufficient randomness to decorrelate motion from acquisition. Hence, our data suggest that SL should be the option to test for high-field structural imaging, particularly when parallel imaging acceleration amplifies even small signal non-uniformity.

Reconstructions with three navigators (k_x, k_y, k_z) occasionally underperformed those with only k_x/k_y , notably leaving residual frontal lobe ghosting. Isolating each navigator showed that k_z alone failed to capture the dominant in-plane head motion in our acquisitions, whereas k_x or k_y did. This underlines the importance of adapting navigator placement to the principal motion axes for better motion estimation.

The non-rigid motion model provides additional improvements in image quality, particularly in very high-resolution or slab-selective acquisitions, where even subtle motion can significantly compromise reconstruction. Hence, considering non-rigid motion correction in brain MRI is essential to fully exploit the resolution gains offered by high-field MRI.

MC-NET operates in image space, learning a regression from synthetically corrupted 3T data. This mismatch (different field strength, SNR and motion types) explains its instability at 7T: in 40% of the datasets. When reconstructions were valid, MC-NET improved SI but remained 5–10% below GRICS, especially in cases with through-plane or non-rigid motion. The DL method was limited to the data type on which it was trained and is not adapted to protocol changes (e.g., different resolution, slab selection, or MP2RAGE contrast). A promising direction is to hybridize DL with physics-informed operators — e.g. using GRICS for data consistency while replacing the regularization with a learned prior, or using networks to accelerate navigator parameters estimation.

In some of the reconstructions presented in this chapter, we observed that residual aliasing artifacts remained visible when the acceleration factor was important ($AF = 3$). This limitation can be attributed to the way coil sensitivity maps were estimated, namely from the central lines of k -space. While this estimation strategy is widely used in iterative reconstruction frameworks, it inherently provides a low-resolution approximation of the coil profiles and may be sensitive to both the size of the auto-calibration region, contrast variation, and motion during the acquisition. These inaccuracies can degrade the conditioning of the inverse problem and amplify residual undersampling artifacts when AF is high. To explore an alternative approach, we performed a complementary evaluation using an $0.9 \times 0.9 \times 0.9 \text{ mm}^3$ MP2RAGE acquisition (D_{11}) in SL mode reconstructed with two different sensitivity operators: one estimated from the conventional central lines of k -

space, and the other derived from explicit B_1 field maps. The latter were obtained using a separate 3D scan, acquired at the beginning of the examination, designed for quantitative mapping of both the transmit (B_1^+) and receive (B_1^-) fields. In this case, the B_1 -based operator directly encodes the receive sensitivities of the coils. Results are shown in Figure 9.9, where the top row presents the MP2RAGE images at the first inversion time, and the bottom row shows the images at the second inversion time. In (A), the sensitivity maps were derived from B_1^- field maps, while in (B) they were estimated from the central lines of k -space. The visual comparison demonstrates the benefit of the B_1 -based estimation: aliasing artifacts that remain visible in (B) are considerably reduced, as highlighted by the colored squares. Overall image quality is improved, with sharper anatomical delineation and fewer undersampling residuals. These results confirm the potential benefit of using B_1 -based sensitivity maps estimation in the reconstruction process.

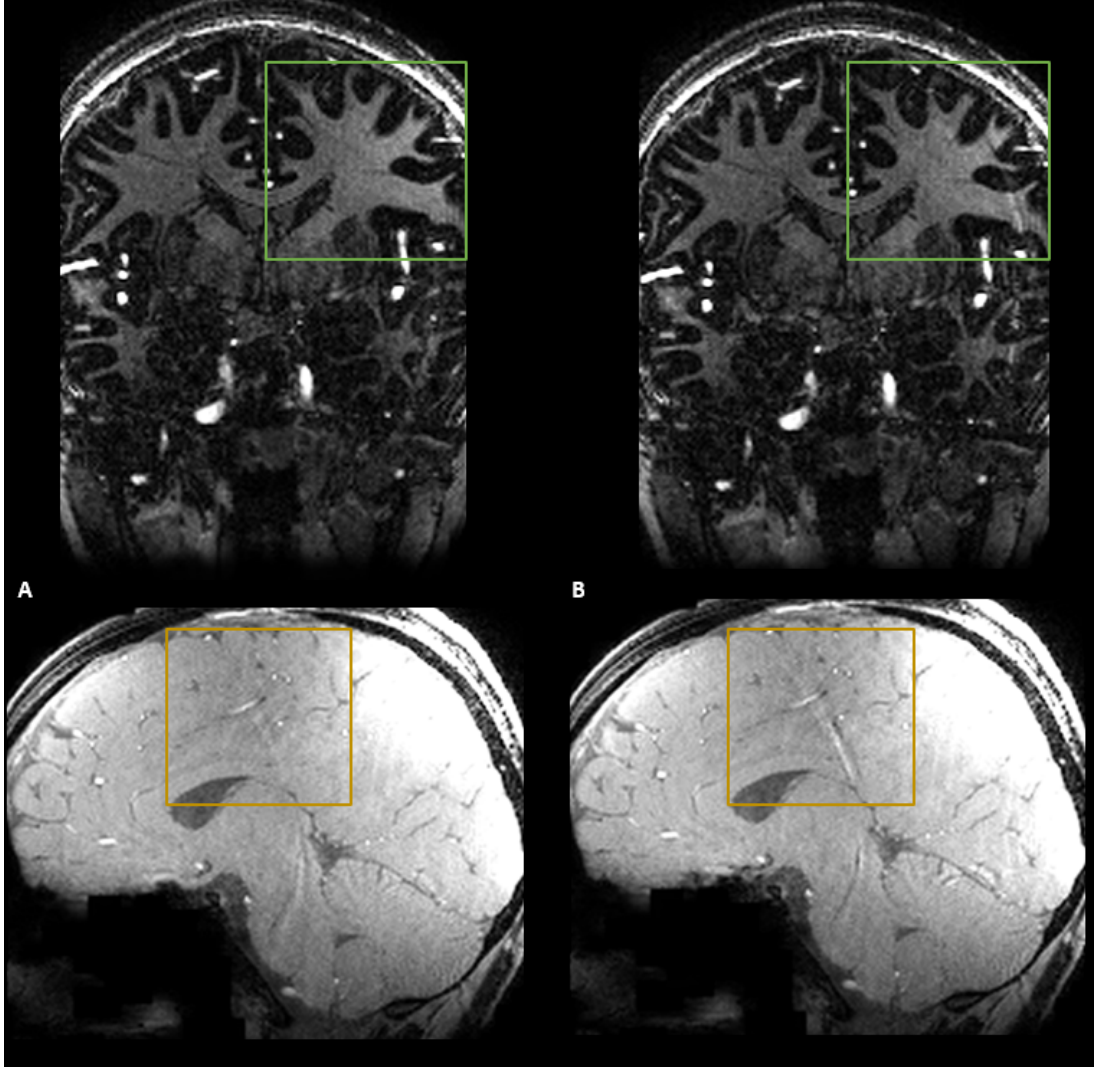


Figure 9.9: GRICS reconstructions for dataset D_{11} . (A) Reconstruction performed using sensitivity maps derived from the B_1^- field maps. (B) Reconstruction performed using sensitivity maps estimated from the central lines of k -space.

We emphasize that retrospective motion correction at 7T accommodates both rigid and non-rigid motion, whereas current prospective schemes remain mainly focused on rigid motion alone—highlighting an improvement possibility. To deliver a solution that can be adopted in routine clinical practice, we limit sequence modifications to a simple addition of central k -space navigator lines combined with randomized sampling, and we aim to keep the reconstruction time under 10 minutes. Within this framework, we systematically optimize the reconstruction parameters, sampling pattern, navigator strategy, and choice between rigid and non-rigid models, and benchmark the proposed pipeline against a fast pre-trained end-to-end DL reconstruction solution.

9.6 Conclusion

In this work, we have validated the GRICS framework for performing motion compensated brain MRI at 7T. At Ultra High-Field, even subtle head and physiological motion significantly degrades image quality, making motion correction essential. Our results show that SL sampling provides the best trade-off between artifact suppression and intensity uniformity. Rigid GRICS correction already yields strong improvements, but non-rigid modeling becomes important when residual physiological deformations are present, and may further improve the reconstruction quality. Our results demonstrate that non-rigid motion must be considered in ultra-high-field brain MRI, as rigid modeling alone may be insufficient in some cases. Regarding navigators, single in-plane lines (k_x and/or k_y) are most effective, while k_z introduces instability in the motion correction process. Compared with a MC-NET, GRICS provided more robust and consistent reconstructions, highlighting the importance of explicit physics-based modeling in the MRI motion-compensated reconstruction.

Chapter 10

Conclusion and Future Works

This thesis addressed one of the main challenges in Ultra-High-Field MRI: the degradation of image quality caused by patient motion during long and high-resolution brain acquisitions. By combining iterative motion-compensated reconstruction frameworks (with the GRICS method as an example), high-performance computing implementations, and sequence design strategies, this work demonstrated the feasibility of clinically relevant, motion-robust neuro-imaging at submillimetric resolution. Beyond the specific case of GRICS, the HPC results obtained in this work can be extended to other reconstruction pipelines. For classical iterative methods such as generalized SENSE or Compressed Sensing, the underlying formalism and computational kernels are very similar to those of GRICS, and therefore one can expect comparable performance and scalability conclusions. In the DL context, a distinction should be made between physics-based approaches, such as PINNs, and purely end-to-end methods. In the former case, since the same forward and backward operators are involved in the loss function, the observations reported here remain valid. However, for end-to-end networks where the physical model is no longer explicitly integrated, our conclusions cannot be directly transferred. The generalization of such methods to our specific reconstruction problems remains to be demonstrated, as already illustrated by the limitations observed with MC-NET.

10.1 Contributions

1. **Performance Characterization and Optimization of GRICS via the Roofline Model**
 - The performance of GRICS reconstruction framework was investigated using the roofline performance model, we elaborated a quantification of compute-versus memory-boundedness across CPU and GPU architectures.
 - This approach guided the design of optimized implementations (the CUDA implementation, the optimal OpenMP MPI configuration) and provided a reproducible methodology to benchmark iterative MRI reconstruction algorithms, since most of them share the same bottleneck as GRICS (the application of the operator $E^H E$).

2. High-Performance Implementations of GRICS on CPU and GPU Architectures

- Parallel implementations were developed on modern multi-core CPUs using OpenMP and MPI, achieving reconstruction times on the order of minutes and tested on brain and breast MRI at 3T, on different types of motion.
- The CUDA version provided up to 9–14 $\times$ acceleration on NVIDIA A100 GPU compared to CPU versions. The implementation reduced considerably the reconstruction time, and approached the framework to near-clinical feasibility.

3. Integration of Randomized Acquisition Strategies into MRI Sequences

- MPRAGE and MP2RAGE sequence codes were modified to incorporate randomized k -space sampling strategies (Shuffle Lines, Shuffle All, Elliptical Shuffling).
- The rationale behind the randomization is to distribute evenly the effects of motion across all motion states. In order to get relevant motion information, navigators (the central line of k -space) were acquired in one or all the spatial directions.

4. Quantitative and Qualitative Evaluation of GRICS reconstruction for brain MRI at 7T

- Phantom and in vivo experiments at 7T were conducted with different motion models and different randomization strategies.
- GRICS was compared against a state-of-the-art deep learning approach (MC-NET). Results demonstrated that GRICS produced superior sharpness index, GRICS succeeded also in reducing the severity of motion artifacts in all of the in-vivo acquisition.
- Considering non-rigid modeling for brain MRI lead to an improvement of the reconstruction quality and the sharpness of the brain anatomical structures.

10.2 Future Work

Several perspectives can be derived from our study in order to extend the presented results:

1. **Optimization of Imaging Protocols** Although the current work demonstrated the feasibility of motion-compensated reconstruction for brain MRI at 7T using the GRICS framework, motion remained limited in a large proportion of the datasets. A future direction would be to design and test a dedicated imaging protocol with explicit instructions provided to volunteers during the MR examination. The aim would be to introduce controlled motion events, and to accurately identify them in the plot of the DOFs temporal variations after the reconstruction.

2. **Sensitivity Map Estimation** A natural perspective of this work will be the systematic validation of B_1 -based sensitivity operators for motion-compensated reconstructions. The preliminary MP2RAGE results presented in the Discussion of Chapter 9 already demonstrate that using B_1 maps to define coil sensitivities can considerably reduce undersampling artifacts compared to conventional estimation from central k -space data. Extending this evaluation to larger datasets and varying acceleration factors would provide a more comprehensive assessment of its robustness and reproducibility. Such validation would also clarify the conditions under which B_1 -based calibration is most beneficial, particularly at UHF where RF inhomogeneities are prominent and accurate sensitivity modeling becomes essential.
3. **Use of the Scattering Matrix** The present implementation in MPRAGE and MP2RAGE sequences relied on the acquisition of central k -space lines along the three spatial directions as navigators before each RAGE shot for motion estimation. Future work should investigate the use of the scattering matrix (S -matrix) as a motion surrogate. The S -matrix is a robust alternative to conventional navigator echoes. Unlike central k -space lines, which may provide limited information on bulk phase shifts, the S -matrix encodes the full transmit–receive interactions between coil elements and is directly modulated by subject motion through changes in loading, coupling, and B_1 field distributions. Because even small and subtle displacements alter the S -parameters, the S -matrix offers a highly sensitive and spatially rich motion information.

Bibliography

- [1] I. I. Rabi, “Space quantization in a gyrating magnetic field,” *Phys. Rev.*, vol. 51, pp. 652–654, Apr 1937.
- [2] I. I. Rabi, J. R. Zacharias, S. Millman, and P. Kusch, “A new method of measuring nuclear magnetic moment,” *Phys. Rev.*, vol. 53, pp. 318–318, Feb 1938.
- [3] F. Bloch, “Nuclear induction,” *Phys. Rev.*, vol. 70, pp. 460–474, Oct 1946.
- [4] E. M. Purcell, H. C. Torrey, and R. V. Pound, “Resonance absorption by nuclear magnetic moments in a solid,” *Phys. Rev.*, vol. 69, pp. 37–38, Jan 1946.
- [5] H. Y. Carr, *Free Precession Techniques in Nuclear Magnetic Resonance*. PhD thesis, Harvard University, Massachusetts, Jan. 1953.
- [6] P. C. Lauterbur, “Image Formation by Induced Local Interactions: Examples Employing Nuclear Magnetic Resonance,” , vol. 242, pp. 190–191, Mar. 1973.
- [7] P. Mansfield and P. K. Grannell, “NMR ‘diffraction’ in solids?,” *Journal of Physics C Solid State Physics*, vol. 6, pp. L422–L426, Nov. 1973.
- [8] L. HANSON, “Is quantum mechanics necessary for understanding magnetic,” *Concepts Magn. Reson. Part A*, vol. 32, 09 2008.
- [9] R. R. Ernst and W. A. Anderson, “Application of fourier transform spectroscopy to magnetic resonance,” *Review of Scientific Instruments*, vol. 37, no. 1, pp. 93–102, 1966.
- [10] R. R. Ernst and W. A. Anderson, “Equivalence of fourier spectroscopy and slow passage in nuclear magnetic resonance,” *Pure and Applied Chemistry*, vol. 37, no. 1, pp. 47–60, 1974.
- [11] R. Damadian, M. Goldsmith, and L. Minkoff, “Nmr in cancer: Xvi. fonar image of the live human body,” *Physiological Chemistry and Physics*, vol. 9, no. 1, pp. 97–100, 108, 1977.
- [12] R. Pohmann, O. Speck, and K. Scheffler, “Signal-to-noise ratio and mr tissue parameters in human brain imaging at 3, 7, and 9.4 tesla using current receive coil arrays,” *Magnetic Resonance in Medicine*, vol. 75, no. 2, pp. 801–809, 2016.

- [13] X. He, M. A. Ertürk, A. Grant, X. Wu, R. L. Lagore, L. DelaBarre, Y. Eryaman, G. Adriany, E. J. Auerbach, P.-F. V. de Moortele, K. Ugurbil, and G. J. Metzger, “First in-vivo human imaging at 10.5t: Imaging the body at 447 mhz,” *Magnetic Resonance in Medicine*, vol. 84, no. 1, pp. 289–303, 2020.
- [14] N. Boulant, F. Mauconduit, V. Gras, A. Amadon, C. Le Ster, M. Luong, A. Massire, C. Pallier, L. Sabatier, M. Bottlaender, A. Vignaud, and D. Le Bihan, “In vivo imaging of the human brain with the iseult 11.7-t mri scanner,” *Nature Methods*, vol. 21, no. 11, pp. 2013–2016, 2024.
- [15] D. I. Hoult and P. C. Lauterbur, “The sensitivity of the zeugmatographic experiment involving human samples,” *Journal of Magnetic Resonance*, vol. 34, no. 2, pp. 425–433, 1979.
- [16] C. M. Collins and M. B. Smith, “Signal-to-noise ratio and absorbed power as functions of main magnetic field strength, and definition of “90 degrees” rf pulse for the head in the birdcage coil,” *Magnetic Resonance in Medicine*, vol. 45, no. 4, pp. 684–691, 2001.
- [17] E. L. Hahn, “Spin echoes,” *Physical Review*, vol. 80, no. 4, pp. 580–594, 1950.
- [18] J. P. I. Mugler and J. R. Brookeman, “Three-dimensional magnetization-prepared rapid gradient-echo imaging (3d mp rage),” *Magnetic Resonance in Medicine*, vol. 15, no. 1, pp. 152–157, 1990.
- [19] J. P. Marques, T. Kober, G. Krueger, W. van der Zwaag, P.-F. Van de Moortele, and R. Gruetter, “Mp2rage, a self bias-field corrected sequence for improved segmentation and t1-mapping at high field,” *NeuroImage*, vol. 49, no. 2, pp. 1271–1281, 2010.
- [20] G. Okubo, T. Okada, A. Yamamoto, M. Kanagaki, Y. Fushimi, T. Okada, K. Murata, and K. Togashi, “MP2RAGE for deep gray matter measurement of the brain: A comparative study with MPRAGE,” *Journal of Magnetic Resonance Imaging*, vol. 43, no. 1, pp. 55–62, 2016.
- [21] R. Metere, T. Kober, H. E. Möller, and A. Schäfer, “Simultaneous quantitative MRI mapping of T1, T2* and magnetic susceptibility with multi-echo MP2RAGE,” *PLOS ONE*, vol. 12, no. 1, p. e0169265, 2017.
- [22] J. Fessler and B. Sutton, “Nonuniform fast fourier transforms using min-max interpolation,” *IEEE Transactions on Signal Processing*, vol. 51, no. 2, pp. 560–574, 2003.
- [23] K. P. Pruessmann, M. Weiger, M. B. Scheidegger, and P. Boesiger, “Sense: Sensitivity encoding for fast MRI,” *Magnetic Resonance in Medicine*, vol. 42, pp. 952–962, Nov. 1999.

-
- [24] M. A. Griswold, P. M. Jakob, R. M. Heidemann, M. Nittka, V. Jellus, J. Wang, B. Kiefer, and A. Haase, “Generalized autocalibrating partially parallel acquisitions (GRAPPA),” *Magnetic Resonance in Medicine*, vol. 47, pp. 1202–1210, June 2002.
 - [25] M. Lustig and J. M. Pauly, “SPIRiT: Iterative self-consistent parallel imaging reconstruction from arbitrary k-space,” *Magnetic Resonance in Medicine*, vol. 64, pp. 457–471, Aug. 2010.
 - [26] M. Uecker, P. Lai, M. J. Murphy, P. Virtue, M. Elad, J. M. Pauly, S. S. Vasanawala, and M. Lustig, “ESPIRiT—an eigenvalue approach to autocalibrating parallel MRI: where SENSE meets GRAPPA,” *Magnetic Resonance in Medicine*, vol. 71, pp. 990–1001, Mar. 2014.
 - [27] F. Wiesinger, P. Boesiger, and K. P. Pruessmann, “Electrodynamics and ultimate snr in parallel mr imaging,” *Magnetic Resonance in Medicine*, vol. 52, pp. 376–390, Aug. 2004.
 - [28] D. O. Walsh, A. F. Gmitro, and M. W. Marcellin, “Adaptive reconstruction of phased array MR imagery,” *Magnetic Resonance in Medicine*, vol. 43, pp. 682–690, May 2000.
 - [29] K. P. Pruessmann, M. Weiger, P. Börnert, and P. Boesiger, “Advances in sensitivity encoding with arbitrary k-space trajectories,” *Magnetic Resonance in Medicine*, vol. 46, pp. 638–651, Oct. 2001.
 - [30] J. W. Murakami, C. E. Hayes, and E. Weinberger, “Intensity correction of phased-array surface coil images,” *Magnetic Resonance in Medicine*, vol. 35, pp. 585–590, Apr. 1996.
 - [31] A. Deshmane, V. Gulani, M. A. Griswold, and N. Seiberlich, “Parallel MR imaging,” *Journal of Magnetic Resonance Imaging*, vol. 36, pp. 55–72, July 2012.
 - [32] M. Blaimer, F. Breuer, M. Mueller, R. M. Heidemann, M. A. Griswold, and P. M. Jakob, “SMASH, SENSE, PILS, GRAPPA: how to choose the optimal method,” *Topics in Magnetic Resonance Imaging*, vol. 15, pp. 223–236, Aug. 2004.
 - [33] A. N. Tikhonov, “On the stability of inverse problems,” *Doklady Akademii Nauk SSSR*, vol. 39, no. 5, pp. 195–198, 1943.
 - [34] M. Lustig, D. Donoho, and J. M. Pauly, “Sparse MRI: The application of compressed sensing for rapid MR imaging,” *Magnetic Resonance in Medicine*, vol. 58, pp. 1182–1195, Dec. 2007.
 - [35] M. Lustig, D. L. Donoho, J. M. Santos, and J. M. Pauly, “Compressed sensing mri,” *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 72–82, 2008.
 - [36] B. Sutton, D. Noll, and J. Fessler, “Fast, iterative image reconstruction for mri in the presence of field inhomogeneities,” *IEEE Transactions on Medical Imaging*, vol. 22, no. 2, pp. 178–188, 2003.

- [37] A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-Posed Problems*. Washington, DC; New York: Winston; John Wiley & Sons, 1977. English translation from the Russian edition.
- [38] A. Ribes and F. Schmitt, “Linear inverse problems in imaging,” *IEEE Signal Processing Magazine*, vol. 25, no. 4, pp. 84–99, 2008.
- [39] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, vol. 87 of *Applied Optimization*. New York, NY: Springer, Apr. 2014. Softcover reprint of the 2004 edition.
- [40] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, UK: Cambridge University Press, 2004.
- [41] K. T. Block, M. Uecker, and J. Frahm, “Undersampled radial MRI with multiple coils. iterative image reconstruction using a total variation constraint,” *Magnetic Resonance in Medicine*, vol. 57, pp. 1086–1098, June 2007.
- [42] L. Rudin, S. Osher, and E. Fatemi, “Nonlinear total variation based noise removal algorithms,” *Physica D: Nonlinear Phenomena*, vol. 60, pp. 259–268, 11 1992.
- [43] S. Lefkimmiatis, A. Bourquard, and M. Unser, “Hessian-based norm regularization for image restoration with biomedical applications,” *IEEE Transactions on Image Processing*, vol. 21, no. 3, pp. 983–995, 2012.
- [44] S. Lefkimmiatis, J. P. Ward, and M. Unser, “Hessian Schatten-norm regularization for linear inverse problems,” *IEEE Transactions on Image Processing*, vol. 22, p. 1873–1888, May 2013.
- [45] M. Figueiredo and R. Nowak, “An em algorithm for wavelet-based image restoration,” *IEEE Transactions on Image Processing*, vol. 12, no. 8, pp. 906–916, 2003.
- [46] A. Beck and M. Teboulle, “A fast iterative shrinkage-thresholding algorithm with application to wavelet-based image deblurring,” in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 693–696, 2009.
- [47] J. Eckstein and D. P. Bertsekas, “On the Douglas–Rachford splitting method and the proximal point algorithm for maximal monotone operators,” *Mathematical Programming*, vol. 55, no. 1-3, pp. 293–318, 1992.
- [48] M. V. Afonso, J. M. Bioucas-Dias, and M. A. T. Figueiredo, “Fast image recovery using variable splitting and constrained optimization,” *IEEE Transactions on Image Processing*, vol. 19, no. 9, pp. 2345–2356, 2010.
- [49] N. Parikh and S. P. Boyd, “Proximal algorithms,” *Foundations and Trends in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.
- [50] A. Beck, *First-Order Methods in Optimization*, vol. 25 of *MOS-SIAM Series on Optimization*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2017.

-
- [51] J. Nocedal and S. Wright, *Numerical optimization*. Springer series in operations research and financial engineering, New York, NY: Springer, 2. ed. ed., 2006.
- [52] J. B. Andre, B. W. Bresnahan, M. Mossa-Basha, M. N. Hoff, C. P. Smith, Y. Anzai, and W. A. Cohen, "Toward quantifying the prevalence, severity, and cost associated with patient motion during clinical mr examinations," *Journal of the American College of Radiology*, vol. 12, pp. 689–695, Jul 2015.
- [53] Y.-C. Fung, *Biomechanics: Mechanical Properties of Living Tissues*. New York: Springer, 1st ed., 1981.
- [54] M. Zaitsev, J. Maclaren, and M. Herbst, "Motion artifacts in mri: A complex problem with many partial solutions," *Journal of Magnetic Resonance Imaging*, vol. 42, no. 4, pp. 887–901, 2015.
- [55] S. Bates, R. M. Taub, S. E. Black, D. D. McGowan, M. A. Gilbert, and R. M. Henkelman, "The role of swallowing in head and neck mri artifacts," *Magnetic Resonance in Medicine*, vol. 36, no. 2, pp. 191–196, 1996.
- [56] J. Schulz, M. Siegert, J. B. Pierre-Louis, T. Maclaren, T. H. Michael, M. Zaitsev, and R. Turner, "Prospective motion correction in functional mri," *NeuroImage*, vol. 59, no. 2, pp. 1322–1331, 2012.
- [57] C. Dresel, R. Castrop, C. Haslinger, B. Wohlschlaeger, A. Hennenlotter, and M. Ceballos-Baumann, "Physiological brain pulsations and cardiac cycle-related motion: Implications for fmri," *NeuroImage*, vol. 28, no. 1, pp. 116–123, 2005.
- [58] P. Andersen, A. Smith, L. Stæger, K. G. Jespersen, T. E. Lund, and L. K. Hansen, "Micro-movements in head mri: Contribution of cervical muscle activity," *Human Brain Mapping*, vol. 31, no. 3, pp. 311–323, 2010.
- [59] E. M. Haacke, R. W. Brown, M. R. Thompson, and R. Venkatesan, *Magnetic Resonance Imaging: Physical Principles and Sequence Design*. New York: Wiley, 1999.
- [60] J. Maclaren, M. Herbst, O. Speck, and M. Zaitsev, "Prospective motion correction in brain imaging: A review," *Magnetic Resonance in Medicine*, vol. 69, no. 3, pp. 621–636, 2013.
- [61] D. Stucht, K. A. Danishad, J. Schulz, F. Godenschweger, M. Zaitsev, and O. Speck, "Highest resolution *in vivo* human brain mri using prospective motion correction," *PLOS ONE*, vol. 10, no. 7, p. e0133921, 2015.
- [62] M. Aksoy, C. Forman, M. Straka, S. Skare, S. Holdsworth, J. Hornegger, and R. Bammer, "Real-time optical motion correction for diffusion tensor imaging," *Magnetic Resonance in Medicine*, vol. 66, no. 2, pp. 366–378, 2011.
- [63] M. D. Tisdall, A. T. Hess, M. Reuter, E. M. Meintjes, B. Fischl, and A. J. W. van der Kouwe, "Volumetric navigators for prospective motion correction and selective reacquisition in neuroanatomical mri," *Magnetic Resonance in Medicine*, vol. 68, no. 2, pp. 389–399, 2012.

- [64] P. M. Johnson, J. Liu, T. Wade, M. A. Tavallaei, and M. Drangova, “Retrospective 3d motion correction using spherical navigator echoes,” *Magnetic Resonance Imaging*, vol. 34, no. 9, pp. 1274–1282, 2016.
- [65] D. Gallichan, P. Marques, José and R. Gruetter, “Retrospective correction of involuntary microscopic head movement using highly accelerated fat image navigators (3d fatnavs) at 7 t,” *Magnetic Resonance in Medicine*, vol. 75, no. 3, pp. 1030–1039, 2016.
- [66] D. Atkinson, D. L. G. Hill, P. N. R. Stoye, P. E. Summers, and S. F. Keevil, “Automatic correction of motion artifacts in magnetic resonance images using an entropy focus criterion,” *IEEE Transactions on Medical Imaging*, vol. 16, no. 6, pp. 903–910, 1997.
- [67] E. J. Candès, J. Romberg, and T. Tao, “Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information,” *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, 2006.
- [68] D. L. Donoho, “Compressed sensing,” *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.
- [69] L. Cordero-Grande, G. Ferrazzi, R. P. A. G. Teixeira, J. O’Muircheartaigh, A. N. Price, and J. V. Hajnal, “Motion-corrected mri with disorder: Distributed and incoherent sample orders for reconstruction deblurring using encoding redundancy,” *Magnetic Resonance in Medicine*, vol. 84, no. 2, pp. 713–726, 2020.
- [70] C. M. Pirkel, M. Cencini, J. W. Kurzański, D. Waldmannstetter, H. Li, A. Sekuboyina, S. Endt, L. Peretti, G. Donatelli, R. Pasquariello, M. Costagli, G. Buonincontri, M. Tosetti, M. I. Menzel, and B. H. Menze, “Learning residual motion correction for fast and robust 3d multiparametric mri,” *Medical Image Analysis*, vol. 77, p. 102387, 2022.
- [71] J. Xu, D. Moyer, B. Gagoski, J. E. Iglesias, P. E. Grant, P. Golland, and E. Adalsteinsson, “Nesvor: Implicit neural representation for slice-to-volume reconstruction in mri,” *IEEE Transactions on Medical Imaging*, vol. 42, no. 6, pp. 1707–1719, 2023.
- [72] J. Hossbach, D. N. Splitthoff, S. Cauley, B. Clifford, D. Polak, W.-C. Lo, H. Meyer, and A. Maier, “Deep learning-based motion quantification from k-space for fast model-based magnetic resonance imaging motion correction,” *Medical Physics*, vol. 50, no. 4, pp. 2148–2161, 2023.
- [73] H. Eichhorn, V. Spieker, K. Hammernik, E. Saks, K. Weiss, C. Preibisch, and J. A. Schnabel, “Physics-informed deep learning for motion-corrected reconstruction of quantitative brain mri,” in *Proceedings of MICCAI 2024*, vol. 15007 of *Lecture Notes in Computer Science*, pp. 562–571, Springer, 2024.
- [74] D. Atkinson and D. L. G. Hill, “Reconstruction after rotational motion,” *Magnetic Resonance in Medicine*, vol. 49, no. 1, pp. 183–187, 2003.

-
- [75] F. Odille, P.-A. Vuissoz, P.-Y. Marie, and J. Felblinger, “Generalized reconstruction by inversion of coupled systems (grics) applied to free-breathing mri,” *Magnetic Resonance in Medicine*, vol. 60, no. 1, pp. 146–157, 2008.
- [76] L. Feng, R. Grimm, K. T. Block, H. Chandarana, S. Kim, J. Xu, L. Axel, D. K. Sodickson, and R. Otazo, “Golden-angle radial sparse parallel MRI: combination of compressed sensing, parallel imaging, and golden-angle radial sampling for fast and flexible dynamic volumetric MRI,” *Magnetic Resonance in Medicine*, vol. 72, pp. 707–717, Sept. 2014. Epub 2013 Oct 18.
- [77] A. Loktyushin, H. Nickisch, R. Pohmann, and B. Schölkopf, “Blind retrospective motion correction of MR images,” *Magnetic Resonance in Medicine*, vol. 70, pp. 1608–1618, Dec. 2013. Epub 2013 Feb 11.
- [78] V. Spieker, H. Eichhorn, K. Hammernik, D. Rueckert, C. Preibisch, D. C. Karampinos, and J. A. Schnabel, “Deep learning for retrospective motion correction in mri: A comprehensive review,” *arXiv preprint arXiv:2305.06739*, 2023.
- [79] K. Sommer, A. Saalbach, T. Brosch, C. Hall, N. M. Cross, and J. B. Andre, “Correction of motion artifacts using a multiscale fully convolutional neural network,” *American Journal of Neuroradiology*, vol. 41, no. 3, pp. 416–423, 2020.
- [80] S. Chatterjee, A. Sciarra, M. Dünnwald, S. Oeltze-Jafra, A. Nürnberger, and O. Speck, “Retrospective motion correction of mr images using prior-assisted deep learning,” in *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS)*, 2020.
- [81] M. Abdi, K. C. Bilchick, and F. H. Epstein, “Compensation for respiratory motion-induced signal loss and phase corruption in free-breathing self-navigated cine dense using deep learning,” *Magnetic Resonance in Medicine*, vol. 89, no. 5, pp. 1975–1989, 2023.
- [82] J. Hossbach, J. Untenberger, C. Forman, A. M. Nagel, V. Heuveline, M. Zaitsev, and B. Miller, “Deep learning-based motion quantification from k-space for fast model-based mri motion correction,” *Medical Physics*, vol. 49, no. 12, pp. 7844–7859, 2022.
- [83] M. W. Haskell, A. K. Cauley, B. Bilgic, C. S. Hu, J. T. Ngo, R. S. Shaikh, S. M. Lee, L. L. Wald, and E. Adalsteinsson, “Network accelerated motion estimation and reduction (namer): Convolutional neural network guided retrospective motion correction using a separable motion model,” *Magnetic Resonance in Medicine*, vol. 82, no. 4, pp. 1452–1461, 2019.
- [84] C. Wang, Y. Liang, Y. Wu, S. Zhao, and Y. P. Du, “Correction of out-of-fov motion artifacts using convolutional neural network,” *Magnetic Resonance Imaging*, vol. 71, pp. 93–102, 2020.
- [85] E. Kuzmina, A. Razumov, O. Y. Rogov, E. Adalsteinsson, J. White, and D. V. Dylov, “Autofocusing+: Noise-resilient motion correction in magnetic resonance

- imaging,” in *Medical Image Computing and Computer-Assisted Intervention – MIC-CAI 2022* (L. Wang, Q. Dou, P.-T. Yap, and D. Shen, eds.), vol. 13436 of *Lecture Notes in Computer Science*, pp. 365–375, Springer Nature Switzerland, 2022.
- [86] L. Cui, Y. Zhou, M. Gao, T. Yang, C. Li, Y. Wang, and J. Liu, “Motion artifact reduction for magnetic resonance imaging with deep learning and k-space analysis,” *PLOS ONE*, vol. 18, no. 1, p. e0278668, 2023.
- [87] H. Qi, Y. Dong, J. Yang, X. Bi, H. Xue, Y. Zheng, T. B. Nguyen, H. Luo, L. Jin, Y. Xue, and D. Li, “Non-rigid respiratory motion estimation of whole-heart coronary mr images using unsupervised deep learning,” *IEEE Transactions on Medical Imaging*, vol. 40, no. 1, pp. 444–454, 2021.
- [88] I. Oksuz, C. Clough, B. Ruijsink, G. P. Toth, A. P. King, J. A. Schnabel, A. F. Ferreira, and A. P. D. Rueckert, “Deep learning-based detection and correction of cardiac mr motion artefacts during reconstruction for high-quality segmentation,” *IEEE Transactions on Medical Imaging*, vol. 39, no. 12, pp. 4001–4010, 2020.
- [89] C. Qin, J. Schlemper, J. Caballero, A. N. Price, J. V. Hajnal, and D. Rueckert, “Convolutional recurrent neural networks for dynamic mr image reconstruction,” *IEEE Transactions on Medical Imaging*, vol. 38, no. 1, pp. 280–290, 2019.
- [90] T. Küstner, K. Armanious, J. Yang, B. Yang, F. Schick, and S. Gatidis, “Retrospective correction of motion-affected mr images using deep learning frameworks,” *Magnetic Resonance in Medicine*, vol. 82, no. 4, pp. 1527–1540, 2019.
- [91] TOP500, “Top500 historical charts: Development over time,” 2021. Accessed: 2021-04-14.
- [92] I. Cutress, “Managing 8 rome cpus in 1u: Cray’s shasta direct liquid cooling.” HPC Wire, 2021. Accessed: 2024-02-09.
- [93] P. Messina, “The exascale computing project,” *Computing in Science & Engineering*, vol. 19, no. 3, pp. 63–67, 2017.
- [94] Message Passing Interface Forum, “Mpi: A message-passing interface standard,” *International Journal of Supercomputer Applications and High Performance Computing*, vol. 8, no. 3–4, 1994.
- [95] L. Dagum and R. Menon, “Openmp: An industry-standard api for shared-memory programming,” *IEEE Computational Science and Engineering*, vol. 5, no. 1, pp. 46–55, 1998.
- [96] S. Williams, A. Waterman, and D. Patterson, “Roofline: an insightful visual performance model for multicore architectures,” *Communications of the ACM*, vol. 52, no. 4, pp. 65–76, 2009.

-
- [97] Y. Wang, T. Zhao, L. Li, Z. Hou, and J. Gu, “Roofline model based performance-aware energy management for scientific computing,” in *2018 9th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*, pp. 74–80, 2018.
- [98] C. Yang, Y. Wang, T. Kurth, S. Farrell, and S. Williams, “Hierarchical roofline performance analysis for deep learning applications,” in *Lecture Notes in Computer Science*, vol. 284, pp. 473–491, June 2021.
- [99] E. Borisch, R. Grimm, P. Rossman, C. Haider, and S. Riederer, “Real-time high-throughput scalable mri reconstruction via cluster computing,” in *Proceedings of the 16th Annual Meeting of ISMRM*, (Toronto, Canada), p. 1492, 2008. Uses MPI for distributed reconstruction.
- [100] M. Uecker, S. Zhang, D. Voit, A. Karaus, K.-D. Merboldt, and J. Frahm, “Real-time mri at a resolution of 20 ms,” *NMR in Biomedicine*, vol. 23, no. 8, pp. 986–994, 2010.
- [101] S. Schätz and M. Uecker, “A multi-gpu programming library for real-time applications,” in *Algorithms and Architectures for Parallel Processing (ICA3PP 2012)*, vol. 7439 of *Lecture Notes in Computer Science*, (Berlin, Heidelberg), pp. 114–128, Springer, 2012.
- [102] M. S. Hansen and T. S. Sørensen, “Gadgetron: An open source framework for medical image reconstruction,” *Magnetic Resonance in Medicine*, vol. 69, no. 6, pp. 1768–1776, 2013.
- [103] M. Uecker, F. Ong, J. I. Tamir, D. Bahri, P. Virtue, J. Y. Cheng, T. Zhang, and M. Lustig, “Berkeley advanced reconstruction toolbox,” in *Proceedings of the 23rd Annual Meeting of ISMRM*, (Toronto, Canada), p. 2486, 2015. Abstract.
- [104] G. E. Moore, “Cramming more components onto integrated circuits,” *Electronics*, vol. 38, pp. 114–117, Apr. 1965. April 19, 1965.
- [105] G. M. Amdahl, “Validity of the single processor approach to achieving large scale computing capabilities,” in *AFIPS Conference Proceedings, Spring Joint Computer Conference*, vol. 30, (Atlantic City, NJ), pp. 483–485, AFIPS Press, 1967.
- [106] J. L. Gustafson, “Reevaluating amdahl’s law,” *Communications of the ACM*, vol. 31, pp. 532–533, May 1988.
- [107] N. Wirth, “A plea for lean software,” *Computer*, vol. 28, no. 2, pp. 64–68, 1995.
- [108] J. Bucek, K. Lange, and J. v. Kistowski, “SPEC CPU2017 – next-generation compute benchmark,” in *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering (ICPE ’18 Companion)*, (New York, NY, USA), pp. 41–42, ACM, 2018.

- [109] ENCCS, “The gpu hardware and software ecosystem — gpu programming: why, when and how?.” <https://enccs.github.io/gpu-programming/2-gpu-ecosystem/>, 2023. Accessed: 2025-09-17.
- [110] S. Schaetz, D. Voit, J. Frahm, and M. Uecker, “Accelerated computing in magnetic resonance imaging: Real-time imaging using nonlinear inverse reconstruction,” *Computational and Mathematical Methods in Medicine*, vol. 2017, p. e3527269, 2017.
- [111] M. Hansen, D. Atkinson, and T. Sorensen, “Cartesian sense and k-t sense reconstruction using commodity graphics hardware,” *Magnetic Resonance in Medicine*, vol. 59, no. 3, pp. 463–468, 2008.
- [112] T. Sorensen, T. Schaeffter, K. Noe, and M. Hansen, “Accelerating the nonequispaced fast fourier transform on commodity graphics hardware,” *IEEE Transactions on Medical Imaging*, vol. 27, no. 4, pp. 538–547, 2008.
- [113] P. Batchelor, D. Atkinson, P. Irarrazaval, D. Hill, J. Hajnal, and D. Larkman, “Matrix description of general motion correction applied to multishot images,” *Magnetic Resonance in Medicine*, vol. 54, no. 5, pp. 1273–1280, 2005.
- [114] F. Odille, “Chapter 13 - motion-corrected reconstruction,” in *Advances in Magnetic Resonance Technology and Applications* (M. Akçakaya, M. Doneva, and C. Prieto, eds.), vol. 7 of *Magnetic Resonance Image Reconstruction*, pp. 355–389, Academic Press, 2022.
- [115] T. Küstner, J. Pan, H. Qi, G. Cruz, C. Gilliam, T. Blu, and C. Prieto, “Lapnet: non-rigid registration derived in k-space for magnetic resonance imaging,” *IEEE Transactions on Medical Imaging*, vol. 40, no. 12, pp. 3686–3697, 2021.
- [116] G. Balakrishnan, A. Zhao, M. R. Sabuncu, J. Guttag, and A. V. Dalca, “Voxel-morph: a learning framework for deformable medical image registration,” *IEEE Transactions on Medical Imaging*, vol. 38, no. 8, pp. 1788–1800, 2019.
- [117] J. Pan, D. Rueckert, T. Küstner, and K. Hammernik, “Learning-based and unrolled motion-compensated reconstruction for cardiac mr cine imaging,” in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 686–696, Springer Nature Switzerland, 2022.
- [118] N. R. Huttinga, C. A. Van den Berg, P. R. Luijten, and A. Sbrizzi, “Mr-motus: model-based non-rigid motion estimation for mr-guided radiotherapy using a reference image and minimal k-space data,” *Physics in Medicine Biology*, vol. 65, no. 1, p. 015004, 2020.
- [119] P. Chen, M. Wahib, X. Wang, S. Takizawa, T. Hirofuchi, H. Ogawa, and S. Matsuoka, “Performance portable back-projection algorithms on cpus: Agnostic data locality and vectorization optimizations,” in *Proceedings of the ACM International Conference on Supercomputing*, pp. 316–328, June 2021.

-
- [120] S. Roujol, B. de Senneville, E. Vahala, T. Sørensen, C. Moonen, and M. Ries, “Online real-time reconstruction of adaptive tsense with commodity cpu/gpu hardware,” *Magnetic Resonance in Medicine*, vol. 62, pp. 1658–1664, 2009.
 - [121] O. Inam, M. Qureshi, Z. Laraib, H. Akram, and H. Omer, “Gpu accelerated cartesian GRAPPA reconstruction using CUDA,” *Journal of Magnetic Resonance*, vol. 337, p. 107175, 2022.
 - [122] H. Wang, H. Peng, Y. Chang, and D. Liang, “A survey of gpu-based acceleration techniques in mri reconstructions,” *Quantitative Imaging in Medicine and Surgery*, vol. 8, no. 2, pp. 19608–19208, 2018.
 - [123] P. Després and X. Jia, “A review of gpu-based medical image reconstruction,” *Physica Medica*, vol. 42, pp. 76–92, 2017.
 - [124] S. S. Stone, J. P. Haldar, S. C. Tsao, W. m. W. Hwu, B. P. Sutton, and Z.-P. Liang, “Accelerating advanced mri reconstructions on gpus,” *Journal of Parallel and Distributed Computing*, vol. 68, no. 10, pp. 1307–1318, 2008.
 - [125] I. Buck and P. Hanrahan, “Data parallel computation on graphics hardware,” tech. rep., Stanford University, 2003.
 - [126] F. Laganà, L. Bibbò, S. Calcagno, D. D. Carlo, S. A. Pullano, D. Praticcò, and G. Angiulli, “Smart electronic device-based monitoring of sar and temperature variations in indoor human tissue interaction,” *Applied Sciences*, vol. 15, no. 5, p. 2439, 2025.
 - [127] S. Cook, *CUDA Programming: A Developer’s Guide to Parallel Computing with GPUs*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1st ed., 2012.
 - [128] M. J. Cardoso *et al.*, “Monai: An open-source framework for deep learning in health-care,” *arXiv preprint arXiv:2211.02701*, 2022.
 - [129] M. Sabbagh, M. Uecker, A. J. Powell, M. Leeser, and M. H. Moghari, “Cardiac MRI compressed sensing image reconstruction with a graphics processing unit,” in *2016 10th International Symposium on Medical Information and Communication Technology (ISMICT)*, pp. 1–5, 2016.
 - [130] B. Welton, “Performance analysis of modern computing systems.” Technical report, 2020. Available online.
 - [131] Intel, “Intel vtune.” <https://www.intel.com/content/www/us/en/developer/tools/oneapi/vtune-profiler.html#gs.yebp2q>.
 - [132] B. Gregg, “Linux performance tools,” *ACM Queue*, vol. 17, no. 2, pp. 70–102, 2019.
 - [133] “Amd uprof.” <https://www.amd.com/en/developer/uprof.html>.
 - [134] Arm Ltd., *Arm Streamline Performance Analyzer*, 2024. Accessed: 2025-09-12.

- [135] NVIDIA Corporation, *NVIDIA Nsight Systems User Guide*, 2024. Accessed: 2025-09-12.
- [136] “Nsight compute.” <https://docs.nvidia.com/nsight-compute/NsightCompute/>.
- [137] NVIDIA Corporation, *CUDA Profiling Tools Interface (CUPTI) Documentation*, 2024. Accessed: 2025-09-12.
- [138] NVIDIA Corporation, *NVIDIA Data Center GPU Manager (DCGM) Documentation*, 2024. Accessed: 2025-09-12.
- [139] Intel, “Intel® software development emulator.” <https://www.intel.com/content/www/us/en/developer/articles/tool/software-development-emulator.html>.
- [140] L. Cordero-Grande, R. P. A. G. Teixeira, E. J. Hughes, J. Hutter, A. N. Price, and J. V. Hajnal, “Sensitivity encoding for aligned multishot magnetic resonance reconstruction,” *IEEE Transactions on Computational Imaging*, vol. 2, no. 3, pp. 266–280, 2016.
- [141] F. Odille, N. Cîndea, D. Mandry, C. Pasquier, P.-A. Vuissoz, and J. Felblinger, “Generalized MRI reconstruction including elastic physiological motion and coil sensitivity encoding,” *Magnetic Resonance in Medicine*, vol. 59, pp. 1401–1411, June 2008.
- [142] G. Hager and G. Wellein, *Introduction to High Performance Computing for Scientists and Engineers*. CRC Press, 2010.
- [143] S. Eyerman, W. Heirman, and I. Hur, “Dram bandwidth and latency stacks: Visualizing dram bottlenecks,” in *2022 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 322–331, May 2022.
- [144] D. Cardwell and F. Song, “An extended roofline model with communication-awareness for distributed-memory HPC systems,” in *Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia '19)*, (Guangzhou, China), pp. 26–35, ACM, 2019.
- [145] M. Frigo and S. Johnson, “The fastest fourier transform in the west,” 2002.
- [146] blitzpp, “blitz.” <https://github.com/blitzpp/blitz>.
- [147] E. Agullo, J. Demmel, J. Dongarra, B. Hadri, J. Kurzak, J. Langou, H. Ltaief, P. Luszczek, and S. Tomov, “Numerical linear algebra on emerging architectures: The plasma and magma projects,” *Journal of Physics: Conference Series*, vol. 180, no. 1, p. 012037, 2009.

-
- [148] K. Isaieva, C. Meullenet, P.-A. Vuissoz, M. Fauvel, L. Nohava, E. Laistler, M. A. Zeroual, P. Henrot, J. Felblinger, and F. Odille, “Feasibility of online non-rigid motion correction for high-resolution supine breast mri,” *Magnetic Resonance in Medicine*, vol. 90, no. 5, pp. 2130–2143, 2023.
- [149] “Example scripts for plotting roofline.” <https://github.com/cyanguwa/nersc-roofline>.
- [150] C. Yang, “Hierarchical roofline analysis: How to collect data using performance tools on intel cpus and nvidia gpus,” *arXiv*, 2020.
- [151] K. Halbiniak, R. Wyrzykowski, L. Szustak, A. Kulawik, N. Meyer, and P. Gerner, “Performance exploration of various c/c++ compilers for amd epyc processors in numerical modeling of solidification,” *Advanced Engineering Software*, vol. 166, p. 103078, 2022.
- [152] M. Driscoll, *Domain-Specific Techniques for High-Performance Computational Image Reconstruction*. PhD thesis, University of California, Berkeley, 2018.
- [153] H. Shin, D. Kim, E. Park, S. Park, Y. Park, and S. Yoo, “Mcdram: Low latency and energy-efficient matrix computations in dram,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2613–2622, 2018.
- [154] N. Corporation, *cuFFT Library User Guide*. Santa Clara, CA, USA, 2023. [Online].
- [155] N. Corporation, *cuBLAS Library User Guide*. Santa Clara, CA, USA, 2023. [Online].
- [156] M. A. Zeroual, K. Isaieva, P.-A. Vuissoz, and F. Odille, “Performance study of an mri motion-compensated reconstruction program on intel cpus,” *Applied Sciences*, vol. 14, no. 21, p. 9663, 2024.
- [157] M. Luong *et al.*, “A compact 16tx-32rx geometrically decoupled phased array for 11.7 t mri,” in *Proc. 31st Annu. Meeting of the ISMRM*, vol. 707, (London, UK), 2023.
- [158] J. Hong *et al.*, “Bandwidth-effective dram cache for gpus with storage-class memory,” in *Proc. IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, (Edinburgh, United Kingdom), pp. 139–155, 2024.
- [159] C. Kenyon, G. Volkema, and G. Khanna, “Overcoming limitations of gpgpu-computing in scientific applications,” in *Proc. IEEE High Performance Extreme Computing Conference (HPEC)*, (Waltham, MA, USA), pp. 1–9, 2019.
- [160] J. I. Tamir, M. Uecker, W. Chen, P. Lai, M. T. Alley, S. S. Vasanawala, and M. Lustig, “T2 shuffling: Sharp, multicontrast, volumetric fast spin-echo imaging,” *Magnetic Resonance in Medicine*, vol. 77, pp. 180–195, Jan. 2017. Epub 2016-01-20.

- [161] S. Winkelmann, T. Schaeffter, T. Koehler, H. Eggers, and O. Dössel, “An optimal radial profile order based on the golden ratio for time-resolved mri,” *IEEE Transactions on Medical Imaging*, vol. 26, pp. 68–76, Jan. 2007.
- [162] L. Feng, L. Axel, H. Chandarana, K. T. Block, D. K. Sodickson, and R. Otazo, “Xd-grasp: Golden-angle radial mri with reconstruction of extra motion-state dimensions using compressed sensing,” *Magnetic Resonance in Medicine*, vol. 75, pp. 775–788, Feb. 2016. Epub 2015-03-25.
- [163] X. Zhu, F. Tan, K. M. Johnson, and P. E. Z. Larson, “Optimizing trajectory ordering for fast radial ultra-short te (ute) acquisitions,” *Journal of Magnetic Resonance*, vol. 327, p. 106977, June 2021.
- [164] C. K. Macgowan and M. L. Wood, “Phase-encode reordering to minimize errors caused by motion,” *Magnetic Resonance in Medicine*, vol. 35, pp. 391–398, Mar. 1996.
- [165] S. Weick, M. Völker, K. Hemberger, C. Meyer, P. Ehse, B. Polat, F. A. Breuer, M. Blaimer, C. Fink, L. R. Schad, O. A. Sauer, M. Flentje, and P. M. Jakob, “Desynchronization of cartesian k-space sampling and periodic motion for improved retrospectively self-gated 3-d lung mri using quasi-random numbers,” *Magnetic Resonance in Medicine*, vol. 77, pp. 787–793, Feb. 2017.
- [166] A. Seginer and R. Schmidt, “Messing up to clean up: Semi-randomized frequency-selective space-filling curves to suppress physiological signal fluctuations in mri,” *Magnetic Resonance in Medicine*, vol. 90, pp. 2275–2289, Dec. 2023.
- [167] R. Fisher and F. Yates, *Statistical Tables for Biological, Agricultural and Medical Research*. Edinburgh: Oliver and Boyd, 6th ed., 1963.
- [168] F. Odille, S. Uribe, P. Batchelor, C. Prieto, T. Schaeffter, and D. Atkinson, “Model-based reconstruction for cardiac cine mri without ecg or breath holding,” *Magnetic Resonance in Medicine*, vol. 63, no. 5, pp. 1247–1257, 2010.
- [169] L. Zhang, X. Wang, M. Rawson, R. Balan, E. H. Herskovits, E. R. Melhem, L. Chang, Z. Wang, and T. Ernst, “Motion correction for brain mri using deep learning and a novel hybrid loss function,” *Algorithms*, vol. 17, p. 215, 2024.
- [170] G. Blanchet and L. Moisan, “An explicit sharpness index related to global phase coherence,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1065–1068, 2012.
- [171] M. A. Zeroual, N. Dudysheva, V. Gras, F. Mauconduit, K. Isaieva, P.-A. Vuissoz, and F. Odille, “Feasibility of implementing motion-compensated magnetic resonance imaging reconstruction on graphics processing units using compute unified device architecture,” *Applied Sciences*, vol. 15, no. 11, p. 5840, 2025.

Abstract

Ultra-high-field Magnetic Resonance Imaging (MRI) provides submillimetric resolution and enhanced tissue contrast, which is particularly valuable for neuro-imaging, mainly for detecting subtle epileptogenic lesions in pediatric patients with drug-resistant epilepsy. However, long acquisition times and high sensitivity to motion can compromise image quality. To address this challenge, this thesis develops and optimizes motion-compensated reconstruction methods based on the GRICS (Generalized Reconstruction by Inversion of Coupled Systems) framework. A rigorous mathematical formalism was established for rigid and non-rigid motion models, and the computational performance of the parallel version of GRICS was analyzed using the roofline model. A CUDA version of the reconstruction kernel was implemented; it achieved up to $14\times$ acceleration on NVIDIA A100 GPUs compared to CPU implementations, approaching clinical feasibility. Moreover, MRI sequences (MPRAGE, MP2RAGE) were modified to integrate randomized acquisition orders (Shuffle Lines, Shuffle All, Elliptical Shuffling), dispersing evenly motion effects across motion states. Evaluations on phantom and in vivo datasets demonstrated that Shuffle Lines sampling combined with rigid/non-rigid motion modeling yielded the most robust reconstructions. Overall, this thesis demonstrates that combining motion modeling, high-performance computing optimization, and randomized sequence design enables effective motion correction for 7T brain MRI.

Keywords: 7T MRI, GRICS, iterative reconstruction, CUDA, MPRAGE/MP2RAGE

Résumé

L'imagerie par résonance magnétique (IRM) à très haut champ magnétique 7T offre une résolution submillimétrique, elle ouvre des perspectives majeures en neuro-imagerie, par exemple dans le cadre de la détection précoce de lésions épileptogènes. Toutefois, les acquisitions IRM sont typiquement longues et fortement sensibles aux mouvements, pouvant dégrader considérablement la qualité des images. Pour relever ce défi, cette thèse développe et optimise des méthodes de reconstruction compensées en mouvement basées sur la méthode GRICS (Generalized Reconstruction by Inversion of Coupled Systems). Un formalisme mathématique a été établi pour les modèles rigides et non rigides, et les performances computationnelles de GRICS ont été évaluées par le modèle de Roofline sur différentes architectures CPU et GPU. Une implémentation CUDA du noyau de reconstruction de GRICS a permis d'atteindre jusqu'à $14\times$ d'accélération sur GPU NVIDIA A100 par rapport aux versions CPU, rapprochant la méthode d'une potentielle application clinique. D'autre part, les séquences IRM (MPRAGE, MP2RAGE) ont été modifiées pour intégrer des stratégies de randomisation de l'ordre d'acquisition des échantillons de l'espace k (Shuffle Lines, Shuffle All, Elliptical Shuffling), l'objectif étant de distribuer uniformément l'impact du mouvement sur l'ensemble des états de mouvement. Des évaluations sur données fantômes et in vivo ont montré l'efficacité de l'approche Shuffle Lines

combinée avec une modélisation de mouvement rigide/non rigide. Cette thèse démontre, en combinant la modélisation de mouvement rigide/non rigide, les optimisations de calcul hautes performances et le design de séquences avec schémas de randomisation, la faisabilité d’une application clinique de la méthode GRICS.

Mots-clés: IRM 7T, GRICS, reconstruction itérative, CUDA, MPRAGE/MP2RAGE