Universidad
Rey Juan Carlos

TESIS DOCTORAL

Accelerated Computational Methods in Image Processing using GPU Computing: Variational Saliency Detection and MRI-CT Synthesis

Autor:
Eduardo ALCAÍN BALLESTEROS

Directores:
Dr. Emanuele SCHIAVI
Dr. Antonio SANZ MONTEMAYOR

Programa de Doctorado en
TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES
Escuela Internacional de Doctorado

2021

Declaration of Authorship

I, Eduardo ALCAÍN BALLESTEROS, declare that this Thesis titled, "Accelerated Computational Methods in Image Processing using GPU Computing: Variational Saliency Detection and MRI-CT Synthesis" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the Thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"IT IS A SAD thing to be unrequitedly in love, I can tell you. The truth is that I love mathematics and mathematics is completely indifferent to me."

Isaac Asimov

"Damit das Mögliche entsteht, muss immer wieder das Unmögliche versucht werden."

Hermann Hesse

"Ten confianza en ti mismo. En la inteligencia que te permitirán ser mejor de lo que ya eres y en el instinto de tu amor, que te abrirá a merecer la buena compañía."

Fernando Savater

Abstract

The parallel computing paradigm has made a significant impact on many scientific areas whose algorithms have experienced a substantial improvement. In Computer Vision, there has been an explosion of real-time applications ranging from medical science to telecommunications, defense, aerospace, etc. However, designing and implementing algorithms that make use of this paradigm introduce some challenges.

The main goal of this Thesis is to provide efficient algorithms for two relevant computer vision problems capable of real-time resolution based on multicore on CPU and manycore on GPU. The first problem is Saliency detection and the second one is Computed Tomography synthesis.

Nowadays, there is a large amount of visual content (videos and images). As humans, our visual attention can discriminate the relevant information of each scene in a fast way that allows our limited resources to interact with the environment naturally. Computer Vision has recently started proposing algorithms to mimic the human visual attention system. In particular, saliency detection belongs to a binary segmentation problem. These algorithms extract the relevant part of the scene automatically. Saliency detection models are of much interest to the research community. These saliency models aim to understand how visual attention works in humans and to be able to discriminate important information automatically and process it adequately.

The first contribution of this Thesis is the development of a variational model for the resolution of the saliency detection problem in natural images. Variational methods have a long history in Mathematics and Engineering. Its application to low-level image processing such as optical flow, denoising, inpainting, deblurring, etc. has provided state-of-the-art results for unsupervised methods in the computer vision community. One common drawback of these methods, based on local differential operators, is their inability to handle textures. Non-local variational methods, based on non-local differential operators, have been introduced successfully in image processing to overcome this problem. They model not only proximity but also similarity features.

Few algorithms solve the saliency detection problem using the variational setting in the literature. In a non-local framework, given by a 5-dimensional feature space, we propose a new general model based on the non-local Total Variation (NLTV) operator. Different scenarios are explored in the modelling exercise introducing a new explicit term for saliency detection. The resulting models are validated and the related optimization problems are solved on CPU and GPU platforms for comparison. A primal-dual algorithm dictated by the mathematical formulation of the problem is implemented.

A comparison with previous variational approaches is presented. The Quantitative results, under typical metrics used in saliency in complex public datasets, indicate that our method obtains almost the best score in all metrics. Furthermore, the implementation of the algorithm either on CPU or GPU achieves up to 33 fps and 62 fps respectively for 300 $\times$ 400 image resolution, making the method eligible for real-time applications such as surveillance video, temporal video classification, background subtraction, object detection, and general unmanned aerial vehicle (UAV) scene image recognition.

On the other hand, computer vision and image processing techniques have had a tremendous impact on Medical Imaging during the last few decades. For example, these applications have made it possible to diagnose diseases at an early stage. The research community has focused on multimodal imaging using two or more acquisition techniques to facilitate the diagnostic. It has several advantages because every modality aids the diagnosis process by providing specific information about the anatomy of the body. The combination of data from different modalities can be used for accurate construction of patient-specific tissue models for dosimetry applications in electromagnetics or the use of tissue information for attenuation correction in Positron Emission Tomography and Magnetic Resonance Imaging (PET/MR) among other applications.

Methods that allow generating a new modality image from a given data set through image processing are interesting because they reduce acquisition time and sometimes some harmful modality.

The second contribution of this Thesis is a novel patch-based approach to generate Computed Tomography volumes (CT) from Magnetic Resonance (MR) data. The proposed method can be useful for several applications such as electromagnetic simulations, cranial morphometry, and attenuation correction in PET/MR.

The proposed algorithm implemented using GPU computing techniques improves $\times$15.9 against a multicore CPU solution and up to about $\times$75 against a single core CPU solution. The proposed solution produces high-quality pseudo-CT images when a neighbourhood of $9 \times 9 \times 9$ and 10 atlases are used because the patient-specific CT and pseudo-CT images are similar according to the metric normalized cross-correlation (NCC = 0.93). Furthermore, the algorithm has been revisited with new hardware (NVIDIA DGX Station with NVIDIA V100 GPUs) in the last part of this Thesis. The new results have achieved almost $\times$6 speedup in comparison with the best previous configuration. These results have confirmed the scalability of the method in new architectures.

Resumen

Antecedentes Debido a las dificultades para aumentar las frecuencias de reloj de los procesadores, para acceder a la memoria principal disminuyendo la latencia y aumentando el ancho de banda, y para aumentar el paralelismo a nivel de instrucción (ILP), todo ello conocido como la pared de ladrillo (*Brick-Wall*) de la computación, las arquitecturas de cómputo han necesitado una estrategia de procesamiento paralelo para evolucionar y poder proporcionar mejoras en el rendimiento. Dentro de las arquitecturas de cómputo, se pueden distinguir dos importantes enfoques a nivel de diseño, el enfoque multicore más adecuado para las arquitecturas CPU, y las arquitecturas masivamente paralelas (*manycore*) para co-procesadores o procesadores especializados en cómputo paralelo a nivel de datos (*data level parallelism*, DLP).

A su vez estos caminos adoptados por la industria del hardware han influido en la manera en que se deben diseñar e implementar algoritmos (computación y programación paralela). La computación en paralelo ha influenciado considerablemente muchas áreas en la ciencia donde los algoritmos han experimentado un progreso sustancial. En el campo de la visión artificial, en el que la estructura de datos principal, la imagen, es una gran colección de múltiples elementos constitutivos (*picture elements*, pix-els), ha favorecido una explosión de aplicaciones en tiempo real desde medicina hasta las telecomunicaciones, defensa, aeronáutica, etc. Sin embargo, el diseño e implementación de algoritmos que usan este paradigma introduce nuevos retos.

Los problemas de visión artificial pueden ser tratados mediante multitud de técnicas: modelos estocásticos (*stochastic modelling*), ondículas (*wavelets*) y procesamiento en el espacio de la frecuencia, métodos variacionales (*variational methods*), ecuaciones diferenciales parciales (EDP) (*partial differential equations*), aprendizaje automático (*machine learning*), y enfoques más modernos dentro del aprendizaje automático como aprendizaje profundo (*deep learning*). Esta última ha conseguido ocupar las primeras posiciones en competiciones en una gran variedad de problemas, como por ejemplo ImageNet, eclipsando a otros métodos.

Sin embargo, las técnicas variacionales siguen siendo una referencia en muchos campos de visión artificial. El uso de estas técnicas combinado con las redes convolucionales neurales proporciona robustez y estabilidad. Los métodos variacionales tienen una larga y consolidada historia en matemáticas y en ingeniería. Su aplicación al procesamiento de imagen como el flujo óptico (*optical flow*), eliminación de ruido (*denosing*), eliminación de desenfocado (*deblurring*), restauración de imagen (*inpainting*), etc. ha obtenido resultados del estado del arte, entre los métodos no supervisados. Además, el tiempo de cómputo de los algoritmos en algunas áreas de visión artificial para resolver problemas específicos es un factor crítico, como pueda ser la reconstrucción de imágenes en imagen médica.

Por eso existe una necesidad constante en encontrar nuevos métodos para mejorar los algoritmos, pero esto presenta algunas dificultades. El diseño e implementación de los algoritmos usando técnicas paralelas para una tarea específica no es fácil y requiere amplio conocimiento en áreas como arquitectura de ordenadores, algoritmos, software, etc. El paradigma de programación en paralelo involucra desafíos porque el sistema operativo ejecuta las tareas de una manera asíncrona pudiendo

pararlas o retrasarlas haciendo variar el resultado de los algoritmos. Además, la naturaleza de los algoritmos introduce otros problemas cuando se optimizan. Algunos consisten en operaciones independientes. Las arquitecturas modernas pueden ejecutar las operaciones en paralelo, mejorando el rendimiento de estos considerablemente. La mejora esperada es alta y se incrementa cuando se utiliza nuevo hardware porque la naturaleza del método es escalable.

Por otro lado, hay otro tipo de algoritmos cuyas operaciones no son independientes. Estos algoritmos no tienen una naturaleza paralela, pero la investigación para mejorarlos con arquitecturas paralelas es también importante. Históricamente, el diseño de los algoritmos era secuencial. Intentar optimizar estos algoritmos puede producir límites teóricos en el rendimiento e incluso direcciones para reemplazarlos con métodos paralelos.

Hipótesis y Objetivos La propuesta de algoritmos eficientes para su consumo en tiempo real basado en soluciones multi-núcleo en CPU y en GPU que solucionen problemas en visión artificial son esenciales.

Los métodos variacionales tienen unos fundamentos teóricos que los hacen atractivos para resolver una gran cantidad de problemas en visión artificial. Sin embargo, la resolución de estos métodos tiene una naturaleza iterativa que limita la eficiencia de la computación en paralelo por sus dependencias de los datos al aproximar sus derivadas. Otra dificultad radica en la imposición de un criterio de parada basada en la propiedades de convergencia del algoritmo. En cambio, existen otros tipos de técnicas que se adecuan perfectamente a las arquitecturas en paralelo. Dos problemas en visión artificial han sido seleccionados para ilustrar el uso de técnicas de computación paralela. El primer problema es la detección de Saliencia y el segundo es la síntesis de imágenes de tomografía axial computarizada (CT). Las hipótesis principales de esta Tesis Doctoral se centran de la siguiente manera:

- **Primera Hipótesis**: Los métodos variacionales son formidables herramientas de modelado en visión artificial que han sido aplicados con éxito en una gran variedad de problemas de bajo nivel en visión artificial entendiendo por ello el preprocesado de imágenes (filtrado, eliminación de ruido, segmentación, recuperación de información, registro, fusión y super-resolución entre otros) previo a la extracción de estadísticas, clasificación e interpretación semántica de las imágenes. Convenientemente modificados estas formulaciones variacionales clásicas pueden resolver el problema de la detección automática de la saliencia en imágenes naturales y producir resultados del estado del arte para métodos no supervisados.

- **Segunda Hipótesis**: Los métodos iterativos como los esquemas numéricos utilizados para resolver modelos variacionales introducen desafíos cuando se trata de optimizarlos computacionalmente. La combinación de técnicas de computación paralela en GPU o CPU y un esquema numérico que converge rápidamente a la solución pueden producir implementaciones en tiempo real.

- **Tercera Hipótesis**: Los métodos de parches en los enfoques que usan multi atlas han sido utilizados con éxito para segmentación de imágenes médicas. También pueden ser adaptados para realizar síntesis de otras modalidades como por ejemplo CT. De esta manera, la imagen sintética se podría utilizar para corregir imágenes de tomografía por emisión de positrones (PET) en los escáneres PET/MR.

- **Cuarta Hipótesis**: Los métodos de parches en los enfoques de multi atlas consisten en tareas independientes. Con la ayuda de las técnicas de computación paralela en CPU o GPU, el rendimiento computacional del algoritmo puede mejorar sustancialmente. Además, algunos parámetros se pueden relajar en el algoritmo para reducir el tiempo de cómputo sin perder exactitud en los resultados.

Podemos formular el objetivo de esta tesis de la siguiente manera "*El estudio de los métodos clásicos para el procesamiento de imagen aplicado a detección de saliencia y problemas en imagen médica acelerados con plataformas multi-núcleo y arquitecturas masivamente paralelas*". Siendo los objetivos operativos:

1. Analizar el estado del arte de los métodos para resolver el problema de la detección de saliencia en imágenes naturales y métodos para corregir imágenes de PET en la modalidad de PET/MR.

2. Estudio de las plataformas multi-núcleo y arquitecturas masivamente paralelas para mejorar el tiempo de cómputo de los algoritmos.

3. Estudio de la formulación variacional en el procesamiento de imagen: desde un análisis local a no local en grafos discretos.

4. Estudio de los métodos de parches usados en la segmentación en imagen médica y su adaptación para síntesis.

Metodología Para el problema de la detección de saliencia en imágenes naturales se ha propuesto un nuevo modelo variacional de saliencia, de tipo no local, adecuado a la estructura de los datos dada por un espacio de características de dimensión 5 que se modela como un grafo. El cálculo variacional no local, tan antiguo como el local (Liouville, 1832), sólo recientemente ha sido desarrollado e introducido en visión artificial y procesamiento de imagen. Inicialmente motivado para solucionar el problema de la detección y reconstrucción de texturas puede ser empleado para detectar similitud y proximidad en espacios de características.

En este marco se proponen dos nuevos modelos basados en el operador de variación total (NLTV) no local. Diferentes escenarios son investigados en el ejercicio de modelado introduciendo un nuevo término explícito para la detección de saliencia. La formulación propuesta, en términos de la norma dual del operador NLTV, dicta la implementación de un algoritmo primal-dual que se adapta al marco no local.

Para el problema de la generación de modalidad CT a partir de imagen MRI (síntesis) se propone un algoritmo basado en los fundamentos del filtro de media no local (*non-local means*) para reducir el ruido. Este filtro se puede adaptar para conseguir otras finalidades, por ejemplo, generar una imagen de otra modalidad desde una modalidad diferente. La generación del pseudo-CT se consigue mediante técnicas multi atlas. Para ello relacionamos la imagen de entrada (MRI) con las imágenes de referencia en el atlas (pares MRI-CT) y propagamos la correspondencia en su par CT dentro del atlas. La relación entre las imágenes se realiza a nivel de parche dentro de un vecindario dando como resultado un peso que se combina con las etiquetas de CT para obtener la imagen de la nueva modalidad. Las ventajas de este algoritmo son su independencia en las operaciones y no tener ningún esquema iterativo que se traducen en una utilización casi óptima de los recursos de una arquitectura paralela.

Ambos problemas han sido optimizados en CPU y GPU dando como resultado soluciones para su consumo en tiempo real. El acceso eficientemente a la memoria es una de las prioridades para reducir tiempo en los algoritmos. Para lograr este objetivo hemos organizado mejor los datos (*spatial locality*) o su acceso local (*temporal locality*). Además, hemos aplicado patrones paralelos (*reductions*) para algunas operaciones en nuestros algoritmos (mínimos, máximos, sumas, etc.).

Resultados Las herramientas clásicas como los métodos variacionales y EDP, y los parches en multi atlas han resultado ser bastante poderosas para resolver los problemas propuestos.

En el problema de la detección de saliencia, los resultados cuantitativos, en base a típicas métricas usadas en saliencia en complejos conjuntos de datos, comparados con otros modelos variacionales, muestran que nuestro método obtiene casi los mejores resultados en todas las métricas. Además, nuestra implementación tanto en CPU como en GPU consigue hasta 33 fps y 62 fps respectivamente para una imagen con dimensiones 300×400, haciendo posible que el método sea elegible para aplicaciones en tiempo real como video vigilancia, substracción de fondo, detección de objetos y vehículo aéreo no tripulado (VANT) para reconocimiento de escenas.

El algoritmo propuesto para la síntesis que ha sido implementado con técnicas de computación en GPU obtiene un rendimiento de 15.9 veces mejor que una solución multi-núcleo en CPU y hasta cerca de 75 veces en comparación con una solución de un solo núcleo en CPU. La solución propuesta produce imágenes de pseudo-CT de alta calidad cuando se utiliza un vecindario de $9 \times 9 \times 9$ y 10 atlas porque la imagen de CT de referencia y el pseudo-CT son similares mediante la métrica de correlación cruzada normalizada (NCC = 0.93). Además, en la última etapa del estudio de la presente Tesis Doctoral se ha revisado el algoritmo con un nuevo hardware (NVIDIA DGX servidor con NVIDIA V100 GPUs). La mejora conseguida es de casi 6 veces respecto a la mejor configuración anterior. Esto confirma la escalabilidad del método cuando se utilizan nuevas arquitecturas.

Conclusiones El énfasis de esta Tesis Doctoral se ha centrado en la utilización de técnicas de computación paralela en arquitecturas modernas para resolver problemas de relevancia en visión artificial. El uso de estas técnicas permite tener aplicaciones en tiempo real las cuales son de mucho interés actualmente en la comunidad de la visión artificial y el procesamiento de imágenes. Estas técnicas de computación paralela también se han convertido en una pieza fundamental para muchas áreas fuera del ámbito de visión artificial debido a que siempre existe un deseo de mejorar el rendimiento de los algoritmos y el hecho de que los avances producidos por la miniaturización se hayan ralentizado. Por lo tanto, las alternativas, para satisfacer las demandas de muchas áreas de la ciencia, deberían de provenir del diseño de arquitecturas paralelas, software paralelo o algoritmos con naturaleza paralela.

En nuestro trabajo, hemos propuesto un nuevo algoritmo y un modelo para resolver dos problemas de relevante importancia y de naturaleza diferente en visión artificial con el uso de técnicas de computación en paralelo. La naturaleza diferente de las soluciones propuestas nos ha proporcionado otra perspectiva a la hora de diseñar algoritmos o modelos cuando aparezcan nuevos retos: la escalabilidad.

El método para la síntesis de MRI-CT tiene una naturaleza paralela, en cambio, la detección de saliencia es un método iterativo. Basado en los resultados de esta Tesis Doctoral para ambos métodos, hemos observado que la escalabilidad en el problema de síntesis MRI-CT ha producido un impacto considerable en el tiempo de cómputo con nuevo hardware. La naturaleza paralela en los algoritmos tiene una importante ventaja competitiva cuando el tiempo de cómputo es un requisito esencial, véase por ejemplo aplicaciones de tiempo real. Esta ventaja refuerza la idea de pensar de una manera diferente algunas herramientas de matemáticas como los métodos iterativos para intentar adecuarlas lo máximo posible a arquitecturas paralelas. Sin duda, modelar, computar o pensar en paralelo es el presente y el futuro.

La computación también evoluciona para satisfacer las demandas que surgen en los nuevos problemas. La computación en GPU nació como un modelo puramente paralelo, pero ha ido integrando maneras de expresar muchos algoritmos que inicialmente no eran posible.

Acknowledgements

I confess myself a dreamer and my hobby, reading, has strengthened this rare disease considerably. Dreaming carried me to this untouchable fantasy world, waking my curiosity up, after all, literature is pure inspiration. Books have been devoured by me on the long road to this PhD except for almost the end when I was so absorbed in writing this manuscript. In those books, I have been commanded by Trajan in Parthia (I wish it had happened) or explored the secrets of quantum computing *programming the universe*...From all of them, I have, of course, good memories, but I would like to break the ice with something that I read ages ago, and if lady luck is with me, these lines may arise the curiosity, and who knows? Read until the end. I cross my heart that my Acknowledgements are short and intense. Let's give it a go....

When we dream we give free rein to our aptitude for madness. At the same time we suspect that all madness is a dream that has taken root.
Popular wisdom: *"The poor guy is crazy, a dreamer... "*.[1]

A dream has become a reality at the time of writing this **Acknowledgements**. A wish in my life was and IS to become a Doctor of Philosophy. I am a lucky guy because digital letters cannot be spoiled by tears that are pouring to my desktop at the moment of writing. They are tears of happiness, satisfaction, and relief, above all, relief. Looking back at the journey I made, no one would describe it as a piece of cake. Honestly, let's look forward just in case that by chance we go back in time and I must start from scratch.

After having finished my Masters' Degree, I made the unhealthy informed democratic decision to continue and pursue a PhD. Without a doubt, it has been one of the biggest challenges in my life. I have been fighting for almost seven years and went through many personal situations, difficult ones and pleasant ones. My determination has made me what I am today and whenever I needed to encourage myself I shouted *Vamos Rafa!!*.

The topics of this Thesis are fascinating to me. I hope that the readers agree with me. Mathematics forms part of the essential blocks for Science. On the other hand, Computation has become the vehicle for everything and yet more to come with the incredible ideas of *Quantum Computing*. Being able to command in some degree these pillars of knowledge is a remarkable achievement itself. I am very privileged.

PhD should be a synonym for madness. Keeping you up late at night with ideas and excitation is something that under no circumstances is extending your life expectancy. But I could not counteract Nature's law that everything shall beget its like and I have been most of the time upon something no matter what.

Distance has been proved to be a fierce and strong enemy, but not as strong as I am, and living in Germany while studying in Spain was only another challenge.

I would like to wink at history and pay tribute to my mentors

[1]Lines extracted from Chapter 80 in Hopscotch book by Julio Cortázar

From my grandfather *Diego*, I learnt to relish the beauty of manners and to restrain all anger. My grandfather used to say that he had a grandson who was a go-getter. Thanks so ever much for these priceless words wherever you are.

I have mastered a bit of mathematics for image processing with the great mathematician *Emanuele Schiavi* and as he always used to say, "and the only one". Learning variational is a real challenge, but his knowledge has guided me to understand the key concepts and the beauty of it and made me a little mathematician apprentice.

I owe *Antonio Sanz Montemayor* some fundamental values to be a better professional and researcher: control my rebellion, soften the tone, and be practical. Thanks for leaving everything in the court that Christmas to publish the saliency paper.

I learnt from *Ana I. Muñoz* that people are not always more intelligent than you. They typically came first to the problem and spent more hours on it. I owe you one of the greatest discoveries in my PhD, your friendship, my sweet Ana. Introducing *Minaya* to you triggered everything. As you said, *Antonio Gala* makes things easy.

You may remember me from "Holidays at the University" where I spent with *Iván* some fun times having coffees without coffee at the cafeteria on my days off from work. The collaboration with *Ángel* and *Norberto* led me to experience the practical side of Medical Imaging and be a co-author of a paper cited 59 times.

I cannot forget my parents: *Ana* and *Eduardo*. They have been on the front line of this PhD and their support was immeasurable. My father always had words to soothe my desperation and my mother's blind faith in her son carried me to the final destination: this PhD.

My cousin *Hugo*, "el Peter Lynch de Los Boliches", has shown me that reading and motivation are crucial to pave the path to success.

I discovered from my friend *Amador* that people accuse others of being intelligent because they read. Thanks for your support in the good and the bad times, and, of course, for believing in my talent.

I met *Pacos* in a boring German class sponsored by my company. He was brave to review the document (my respect). Now, we will make a killing with my new hobby.

My friend *Joaquín* taught me that "Scientists should provide answers to problems". I hope that I have NOT let you down. Please ring *El Professore* up! it is high time for the Symposium of Nebraska…

I feel deeply indebted to *Monica* for reviewing this document. We did NOT spend that much time at work (separated by the Atlantic ocean, unfortunately), but Washington D.C. with you and the lovely dinner with your parents are in my heart.

My *harni ochi* has scented my life with things that are not in books, **affection**. YOU have made me feel special since we met. Now, I am free, fancy a walk in the Boca Raton promenade?

The PhD is in the post, that is for sure. What is next? Let's see what happens.

Contents

List of Figures

List of Tables

List of Algorithms

pArA Mi fAmiLiA y Mi гарні очі

Chapter 1

Introduction

The technological progress in building computers and the innovations in computer design such as processor optimization, memory, etc. have driven the rapid evolution of computer performance. Miniaturization in semiconductor devices led to an outstanding improvement growth over the past 50 years.

Gordon Moore, Intel co-founder, observed a gradual miniaturization rate [184]. His prediction, known as *Moore's Law*, was that the number of transistors that could be integrated into each computer chip would double about every two years [185]. Robert Dennard, at IBM, also observed that in parallel with doubling transistors the power consumption remained constant for a given area of silicon [76].

In the past few decades, the improvements in the computer industry have been driven by Moore's Law and Dennard scaling meaning that more transistors can be packed in the same space and transistors could go faster with less power [115]. This evolution has made algorithms run faster with new hardware for almost 50 years.

However, Moore's law and Dennard scaling no longer hold because the physical limits (energy consumption and dissipation problems) have been reached and therefore, the rate has been reduced [115]. In 2005, the growth in computation performance was limited by the convergence of three factors [175]: power, Instruction-level parallelism (ILP), and memory. Therefore, the performance obtained transparently in each new computer generation has slowed down considerably [17].

These factors triggered a shift in the design of new hardware architectures. To continue the evolution of the performance of computers as before, the semiconductor industry has adopted two paths [123]. On the one hand, there exists a multicore strategy intending to improve serial programs when moving them into multicore architectures [145]. Multicore processors provide access to more than one core for computer computation to make it possible to execute many calculations in parallel.

On the other hand, the manycore strategy aims to keep a large population of threads for calculation in data parallelism problems. GPU Computing is an important representative of this strategy [145]. In each GPU generation, this population of threads (more active cores at the same time) is increasingly giving more power for computation.

The use of the accelerator-based architectures such as Graphics Processing Units (GPUs), Field Programmable Gate Arrays (FPGAs), and Application-specific Integrated Circuits (ASICs) in the high-performance computing (HPC) field and, especially, supercomputing area started in 2008. In June 2008, IBM Roadrunner was the first number one supercomputer of the Top500 ranking including Cell processors as a form of accelerators [270]. After it, accelerators consistently formed part of the top 10 fastest computers in the world. For example, in 2011, the GPUs from the vendor NVIDIA were included in the second supercomputer on the ranking list for the Top500 of the world. The first supercomputer including GPUs as accelerators was Titan in November 2012 equipped with NVIDIA K20x GPUs. More recently, in 2018-19 Summit occupied the first position equipped with NVIDIA V100 GPUs demonstrating that heterogeneous architectures with CPUs (Central Processing Units) and GPUs still appear on the high-performance computing ranking [182].

Over the years, GPU technology has simplified the programming languages when writing code for GPUs and improved the relationship between computation performance and cost. These considerations make this technology very attractive for everyone who wants to develop efficient solutions at a reasonable price [44].

Multicore and manycore architectures have been used in many areas to improve algorithms. Among them, Computer Vision has witnessed an explosion of real-time applications through this performance increase. Computer Vision is a complex scientific field in which many other areas are interconnected together (Mathematics, Computation, Optics, Physics, Psychology, etc.). With this shared knowledge, the aim is to try to emulate the behaviour of the human visual system. This emulation is achieved by processing the data from images or videos through algorithms that usually follow the principle of single instruction multiple data (SIMD) and these are adequate for multicore and manycore architectures. Furthermore, the amount of data to process in videos or images is large and continues increasing every year [61], making it advantageous to use parallel programming paradigms.

Addressing computer vision problems can be done using many approaches. We can trace the roots of image processing back with techniques of manipulating and processing 1D signals. After this, more powerful and complicated tools have emerged, such as stochastic modelling, wavelets, and Partial Differential Equation (PDE), to name some relevant ones [19]. Machine learning techniques such as clustering, support vector machines, neural networks, etc., have also provided the necessary tools to tackle many computer vision problems, for example, image recognition. Furthermore, *deep learning* approaches belong to this family and they have been breaking accuracy records in ImageNet competition since 2012 with Alexnet [148].

Continuous approaches allow the use of continuous tools that are abundant in Mathematics such as functional analysis [40, 51, 12], differential equations [19], optimization theory [35, 233], etc. to tackle computer vision problems (Figure 1.1). Furthermore, the continuous setting can easily model the rotational invariance property and this a fundamental property for the low-level tasks in Image Processing [68].

Calculus of variations is a branch of functional analysis and provides a transparent framework to solve low-level image processing problems as an alternative to heuristic classical approaches. Variational methods are based on the calculus of variations. They define the properties that the solution should have in the functionals and provide a result through optimization. Although some parameters are inherent in this approach, their meaning is also well understood and their effects. An energy functional is constructed and then minimized. Variational methods have achieved great success when applied in many low-level image processing problems [260, 54,

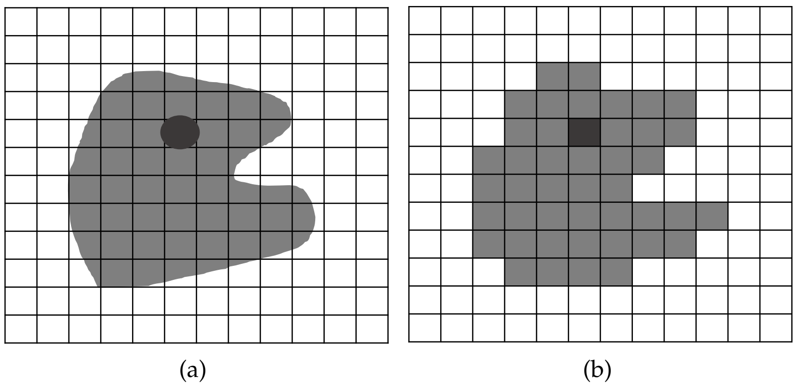


Figure 1.1: (a) Continuous representation of the image (b) Discretization version of (a).

87].

There are many real-world applications using computer vision or image processing algorithms such as machine inspection to check possible failures in industry parts [23], in the hardness testing industry to measure indents and determine the hardness of the materials [94], image manipulation [20], video summarization [164], image compression [108], enhance image quality in graphics [147, 161], MRI reconstruction [250], interactive segmentation [192], denoising in microscopy [107] or multi-atlas segmentation using label fusion for medical images [273, 45] to name a few. The production of new algorithms to satisfy the demands of new applications and new problems is an ongoing task, and new challenges appear every year. There are some areas in Computer Vision where the computational time of the algorithm is critical, for instance, Medical Imaging.

However, there are some difficulties when trying to improve algorithm computations. The design and implementation of them using the parallel paradigm for a specific task is not effortless and it requires expertise in different fields such as software, computer architecture, algorithms, etc. Parallel programming brings some challenges because the operating system executes tasks in the computer asynchronous and can halt them or delayed varying the effect in the algorithms [116]. Furthermore, the nature of the algorithms implies some difficulties when optimizing them. Some algorithms consist of independent operations. Modern architectures can execute them in parallel, improving the performance of the algorithms considerably. The speedup expectation for them is high and it increases with new hardware generation because the nature of the algorithm is scalable.

On the other hand, there are types of algorithms composed of operations that are not independent. These algorithms are not parallel in nature, but the research for improving them with parallel architectures is also important. Historically, the design of algorithms was sequentially [6]. Trying to optimize them can produce theoretical bound performance and even directions to replace them with parallel methods.

Thus, this Thesis is devoted to solving two relevant problems in Computer Vision and providing an efficient implementation in parallel architectures. These problems shall be briefly described in the next section.

1.1 Motivation

The multicore and manycore architectures introduced by the semiconductor industry have changed the way that we should design and implement algorithms. This shift motivates our research to bring parallel computing for two appealing computer vision problems in modern architectures. Our goals are twofold: proposing new algorithms or models to solve them and make an efficient solution that may be deployed in production. Linking both parts is a challenging task because it requires broad expertise in areas that are disjoint most of the time. This research focuses on two areas in Computer Vision: Saliency and Medical Imaging.

Saliency is a relatively new field in Computer Vision. Saliency detection algorithms aim to segment automatically the objects which capture human attention. Pre-processing steps in pipelines typically use saliency algorithms. Therefore, they must be quick methods. That is the reason why this problem is an appealing candidate to propose an efficient algorithm. Many algorithms have been proposed in literature since this new field was born.

However, there are a few using variational methods. Variational methods provide a way to model a problem in an optimization framework. They are still the state-of-the-art results among classical and unsupervised methods in many disciplines (3D image reconstruction, medical image reconstruction, optical flow, etc.). They can also accomplish more specific tasks, such as saliency detection, but the variational approach must introduce new dynamics to facilitate the almost binary separation of foreground and background. Nonetheless, variational algorithms have an iterative nature (numeric scheme), making it difficult to parallelize. They have data dependencies and depend on converge criterion to stop. To the best of our knowledge, algorithms using the variational setting for saliency have not been implemented yet by parallel computing paradigms. A new model using this framework with an efficient computation would be of much interest.

On the other hand, the use of several modalities to analyze data in Medical Imaging introduces new challenges. One of these challenges is to improve the quality of Positron Emission Tomography (PET) images when using Positron Emission Tomography and Magnetic Resonance Imaging (PET/MR) modality. The attenuation correction maps used for PET images are not directly from MR images as it is from CT images in Positron Emission Tomography and X-ray Computed Tomography (PET/CT) modality.

In Medical Imaging, there is also a high demand for fast solutions for clinical purposes (see for example medical image reconstruction algorithms for acquisition scanners). The correction of PET images problem exhibits a large amount of data (images are volumes). Since this correction is not the final step in the pipeline, PET images must be analysed by experts, a good computation performance is also necessary. These requirements (quality improvement and fast algorithm) motivate the proposal of an algorithm that makes the most out of parallel architectures: atlas-based methods, in particular, patch-based methods. These methods extrapolate information from a dictionary of pairs (MRI, CT) constructed offline to generate a pseudo-CT when an MRI input image is given. Then, the synthetic CT may be used to correct the PET images. Furthermore, these algorithms do not include either any iterative scheme or stop criterion. Lastly, their operations are independent and therefore high acceleration of the method is expected.

Although these problems will be solved using classical approaches, we are aware of the growth of new modern approaches like *deep learning* techniques to solve computer vision problems. However, our motivation not to use it is twofold: firstly, we have the intention to apply multicore and manycore architectures to speed up the algorithms. Machine learning approaches make typically use of frameworks (Caffe [137], PyTorch [216], TensorFlow [1], etc.) where the use of this multicore and manycore architectures is transparent and limited optimization is possible.

Secondly, the increase of available data and computation power has made machine learning approaches like deep learning solve increasingly complicated problems with better accuracy [103]. Nowadays, deep learning techniques produce outstanding performance in many areas in Computer Vision: localization, classification, recognition, etc (see, for example, [148]). Nevertheless, this technology remains a black box and some efforts are ongoing to make a stable and understood theory (for further details see [102]). For example, it has been proven that the inference result can be fooled by slightly altering the input data to produce high confidence results in classification when the input image is not that real object (see for example [190]). The lack of stability in the outputs makes for sceptical usage in critical applications such as Medical Imaging.

1.2 Hypotheses

Once we have presented our motivations and the selected problems, we shall define our goals and objectives based on the following considerations. We can formulate the principal hypotheses for the saliency detection problem:

- **Hypothesis One:** Variational methods can be successfully applied to the automatic saliency detection problem for natural images. This is based on the fact that variational methods have shown to be powerful modelling tools for many low-level image processing tasks providing, state-of-the-art results among classical, unsupervised methods. Based on the works of [179, 281] we then assume that, if conveniently modified, classical variational models can also address the saliency detection problem. Since the variational methods rely on unsupervised clues (contrary to the recent deep learning approaches based on data processing and training), in this work, we wish to explore the possibility to provide unsupervised, automatic segmentation (classification) state-of-the-art results.

- **Hypothesis Two:** Iterative methods such as the numerical schemes used to solve the proposed variational models introduce some challenges when optimizing them computationally. The combination of parallel techniques either on GPU or CPU and a numerical scheme that converges rapidly to the solution can be capable of real-time processing video.

and the hypotheses for the MRI-CT synthesis problem as follows:

- **Hypothesis Three:** Patch-based methods in multi-atlas have been applied successfully as a segmentation technique [236]. They can also be applied to generate other image modalities like CT. In this sense, PET/MR modality can use synthetic CT for the attenuation correction. This approach can reach accuracy results in comparison to the patient-specific CT when using an atlas composed of human head images. In this sense, clinical protocols could incorporate these methods.

- **Hypothesis Four:** Patch-based methods in multi-atlas consist of independent tasks. Using computing parallel techniques either on CPU or GPU can improve the performance substantially. Furthermore, some parameters in the patch-based algorithm can be relaxed to reduce time without losing accuracy.

In the following chapters, we shall try to provide the answers for these hypotheses and we will finalize with some conclusions at the end of this document.

1.3 Objectives

We tackle two computer vision problems from classical approaches. Therefore, we can formulate the goal of this Thesis in the following way *"The study of classical image processing techniques applied to saliency detection and medical image problems accelerated by multicore and manycore platforms"*. We can decompose the main goal into several operative objectives:

1. To study the state-of-the-art methods for solving the saliency detection problem in natural images which are general everyday scenes of contemporary life, animals, and landscapes.
2. To study the state-of-the-art attenuation correction approaches in PET/MR modality.
3. To study the multicore and manycore platforms for improving both algorithms.
4. To study the variational formulation in Image Processing: from local pixel-wise analysis to non-local discrete graphs.
5. To study patch-based approaches as segmentation methods in Computer Vision.

1.4 Thesis Organization

This Thesis has six chapters:

- **Introduction** (Chapter 1): This chapter is about giving a brief presentation about computation and its importance in Computer Vision. Furthermore, it summarizes ways to treat images and process them. It also presents the difficulties of optimizing algorithms due to their nature. Finally, we expose the motivation of this research and the choice of the problems, the hypotheses, and the objectives.
- **State of The Art** (Chapter 2): We illustrate the use of General Purpose Computation for solving problems in the literature. We also present the problems in our research and the proposals to solve them in the literature:
  - *General Purpose Computation Using Multicore and Manycore Architectures*: This section introduces the use of multicore and manycore architectures to accelerate algorithms either in Computer Vision or in other fields.

  - *Saliency Detection Problem*: We highlight the interest of the saliency algorithms in Computer Vision and present the different approaches to produce the saliency detection: unsupervised algorithms, supervised algorithms, and combined. At the end of the section, we describe some applications using saliency.
  - *Attenuation Correction in PET/MR: MRI-CT Synthesis*: We describe three medical image acquisition modalities in this section to better understand the difficulties and the challenges to correct PET images in PET/MR multimodality. We also present methods to correct attenuation for PET images in PET/MR multimodality: emission data, segmentation, and atlas-based.

- **Methodology** (Chapter 3): We present the techniques to build our proposals:
  - *Computation in Shared Memory*: alternatives that exist today for high computation in shared memory.
  - *Variational and PDE-based models in Computer Vision*: We describe how to formulate image problems in the variational setting. These problems are discretized and feasible numerical schemes are presented. Furthermore, we also present a non-local framework dictated by the graph structure of the data.
  - *Saliency Variational Model on Graphs*: We explain in detail our approach to solve the saliency detection problem with novel methods in the variational setting.
  - *Patch-Based Methods for MRI-CT Synthesis*: We explain in detail the method proposed for MRI-CT synthesis based on the patch approach.
- **Saliency Detection in Natural Images** (Chapter 4). We describe the implementation and present the results:
  - *Implementation*: We describe the implementations on CPU with vectorization as well as three different optimizations in the CUDA environment.
  - *Results*: We conduct an intensive evaluation of different benchmarks for quantitative purposes and time performance.
- **MRI-CT Synthesis** (Chapter 5). We conduct an intensive evaluation of different benchmarks for quantitative purposes and time performance.
  - *Implementation*: We describe the implementations on CPU using a single core and CPU multicore as well as three different optimizations in the CUDA environment.
  - *Results*: We conduct an intensive evaluation of 18 subjects with different configurations showing the accuracy of our method as well as the short time to produce an output.
- **Conclusions and Future Work** (Chapter 6): We highlight the main contributions of this Thesis, review the hypotheses, propose possible future work, and a summary of the publications produced during the Thesis (presentation, articles, etc.).

In addition to that, there are two appendixes A.1 and A.2. The first one is devoted to illustrating an optimization process for a real problem on multicore CPU and the second one describes how to derive the Euler-Lagrange equations for image processing problems modelled by differentiable energy functionals.

Chapter 2

State of The Art

In this manuscript, we provide a new model and an algorithm with an efficient implementation for two computer vision problems: the saliency detection in natural images and the synthesis of CT given an input MRI image using a dictionary of pairs (MRI, CT). The aim of this chapter is threefold. First, we emphasize the use of computation in many areas of science and industry, and how computation on multicore and manycore architectures is a natural way to improve algorithms and methods. Second, we present the saliency detection problem, highlight its importance in Computer Vision and we review current unsupervised and supervised methods to tackle the saliency detection problem. Finally, we describe briefly the image acquisition modalities we will work with, as well as the importance of producing a new image from another modality to correct the attenuation for PET images in PET/MR modality and the methods in the literature.

2.1 General Purpose Computation Using Multicore and Manycore Architectures

Multicore and manycore architectures are the main strategies that the semiconductor industry has adopted since the performance delivered by Dennard scaling and Moore's law has slowed down [123]. Multicore designs aim to maintain the pace for sequential algorithms or programs and produce optimized code with out-of-order, multi instructions, simultaneous multithreading, etc. in addition to the cores [145]. Manycores, on the other hand, have their focus on parallel algorithms because their designs have eliminated complex control flows or large caches to exploit parallelism with many core units for calculation. In this sense, they have a large number of core units that perform slower in comparison with cores in multicore processors. Manycore architectures are throughput oriented, for example, GPUs, while multicore architectures are latency oriented, e.g., maximizing the total execution task instead of reducing the time for a single task [145].

Algorithms use those architectures to improve performance. The nature of the algorithm is an important aspect when migrating from sequential implementation to a parallel one. We can roughly classify them into two types: algorithms in which the operations are independent, and they might process a large amount of data, for example, applying a global threshold to a large image. On the other hand, algorithms which operations usually depend on other steps in the algorithm creating data dependencies and might compute an insufficient amount of data, for example, iterative methods in a small linear system of equations.

Both types of algorithms are interesting from the point of view of how to accelerate them. They may be part of pipelines or pre-processing steps. However, the first kind of algorithm is going to experience a better speedup gain, and because the nature of the algorithm is parallel new hardware or software with better capabilities will improve the outcome further. The second type of algorithm is more dependent on the hardware characteristics and sometimes may have no significant acceleration when using better hardware or software. The following subsections try to describe research falling into these types of problems. Chapter 3 describes in detail the techniques used for our proposal implementations.

2.1.1 Multicore Architectures

The introduction in the computer industry of architectures with more than one core opened more opportunities to exploit parallelism for computation [124]. Before this milestone, some works accelerated algorithms via multiprocessors. A parallel chess program on a machine with a theoretical peak of 65.54 GFlops[1], which tied for third in the 1994 ACM International Computer Chess Championship, is proposed in [139]. The chess algorithm was implemented by Cilk 1.0.

In the multicore architectures, the use of a multithreading programming interface like OpenMP [72] allows rearranging the problems in a parallel way (problem decomposition) and achieves much better performance. Early works tried to speed up building blocks in scientific computation, such as Gaussian elimination [176]. The authors proposed a method in shared memory using OpenMP and a distributed memory architecture using MPI. Their conclusions highlighted the obstacle of transferring data with MPI and its better scalability for larger problems. In [188], a solver for the parallel finite-element platform for solid earth simulation, GeoFEM, is implemented with OpenMP, achieving a performance of 335.2 GFlops.

In Machine Learning, many algorithms need acceleration to apply them to, for example, image processing and pattern recognition problems. The following paper [136] proposed an architecture to combine GPU and multicore CPU to improve data coordination. The results of the experiments highlighted that the combined solution has $\sim \times 15$ speedup against an implementation on only CPU and only $\sim \times 4$ without OpenMP.

The authors in [26] implemented the Lattice Boltzmann Method for solving fluid dynamic problems with OpenMP. They achieved $\sim \times 6$ in the best configuration against a serial solution. In the field of materials science, a parallelization code to simulate molecular dynamics with OpenMP is proposed in [265]. On dual-processor Xeon systems, they achieved a speedup $\sim \times 1.7$. Another interesting work is an algorithm for DNA sequence alignment on multicore processor architectures [246]. The authors used OpenMP with tiling techniques and the experiments demonstrated a significant acceleration of the considered sequence alignment algorithms. In Ecology, the use of parallel architectures has also brought good performance. An approach for how to parallelize spatially-explicit structured ecological model is presented in [280]. The authors achieved $\sim \times 16$ speedup against a single core implementation, reducing the time from 11 hours for the single core execution to 39

[1]NCSA's 512 processor CM-5 `https://www.top500.org/system/167057/`

minutes in the parallel version.

In Computer Vision, an algorithm for image dehaze removal is proposed in [282]. The implementation is a parallel OpenMP code using four threads executing 50 images with input size 600×525 per second achieved $\times 1.6$ speedup against the sequential algorithm on Intel i7-4820K. The authors in [193] improved JPEG 2000 and MPEG-4 Visual Texture Coding (VTC) applying sequential as well as parallel strategies with shared memory programming paradigm OpenMP. A study on OpenMP and OpenCL platforms of the non-local means algorithm for denoising is conducted in [291]. The authors designed a parallel non-local means with OpenCL and OpenMP and reported results in different platforms for different image sizes. For a 8192×8192 image, the OpenMP solution is almost $\times 11$ faster than a single core solution and OpenCL has almost $\times 9$ speedup against the OpenMP solution.

A Canny Edge Detector is implemented using multicore processors in [183]. They proved the nature scalability of the algorithm for parallel architectures. Cilk is used as a parallel programming model for the codification of the code, and platforms with 4 and 8 CPUs are used. New frameworks have been published to make use of this parallel paradigm and encapsulate them for Image Processing. For example, Video++ is a novel framework targeting image and video applications running on multicore processors [95]. They offer an abstraction on how to represent and resolve problems in Computer Vision achieving up to $\times 32$ speedup using Cilk and OpenMP against naïve equivalent solutions.

In Medical Imaging, a performance study in 2D/3D registration is accomplished in [178]. The authors used shared memory multicore architectures and several parallel programming models such as OpenMP, Cilk++, Intel Threading Building Blocks (TBB), OpenCL, and others. A sequential and parallel data decomposition for medical image reconstruction is presented in [141]. The authors implemented the algorithm Particle Swarm based on Artificial Neural Network.

2.1.2 Manycore Architectures

The exploration of graphic card architectures for computation has been of much interest in the last two decades. Before a programming model for GPU was established, some works tried to exploit its good performance using the graphics contexts like OpenGL [284] or Direct3D [181] and shading techniques [173, 174]. For example, physically-based, visually-realistic cloud simulation, suitable for interactive applications such as flight simulation, is enhanced with GPU techniques in [113]. Furthermore, the author derived general techniques using GPUs to simulate fluid dynamics to chemical reaction-diffusion. Another early work using GPUs is the first implementation of Partial Differential Equations (PDEs) for applying diffusion filters in Computer Vision [239]. The authors implemented a modified version of the Perona-Malik model for filtering in Image Processing. They argued that GPUs are very suitable for memory complex schemes like filtering, since bandwidth became a major limiting factor in many scientific computations, and GPUs provide much better bandwidth output. In that direction, the technique of deformable isosurfaces, implemented with level-set methods used in segmentation, had an intensive computation. However, implementations via GPU have demonstrated a substantial gain in performance [151]. The authors implemented a GPU-based level-set solver with

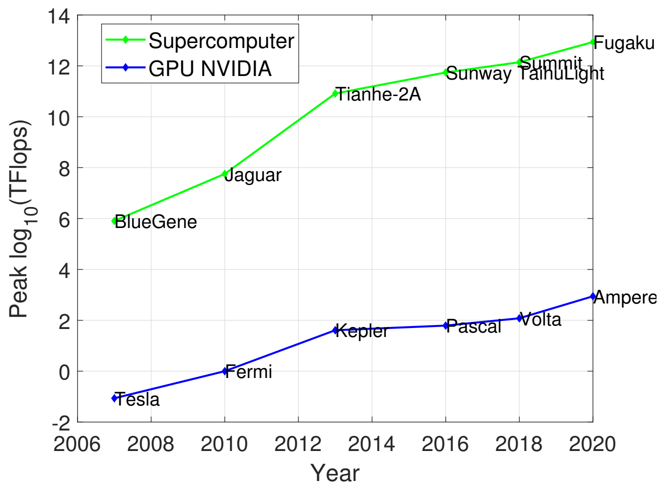


Figure 2.1: Performance comparison in Teraflops (TFlops) between the most relevant GPU architectures in NVIDIA vendor and supercomputers between 2007-2020. Data collected from NVIDIA and `https://www.top500.org/`

$\sim \times 15$ against a highly optimized CPU configuration.

In Medical Imaging, reconstruction algorithms are very computationally intensive tasks. For example, the use of accelerators like Application-specific Integrated Circuits (ASICs) to improve the algorithm time in CT scanners were common for manufactures because the CPU cannot compete with these configurations in three-dimensional reconstruction [86]. The evolution of GPUs allows more flexibility and good performance. The reconstruction of a CT image is based on backprojection operations with a complexity of $\mathcal{O}(N^3)$, N being the volume of the reconstruction. More backprojections imply more time for the algorithm. The authors in [86] achieved a $\times 12$ speedup against CPU implementation for a volume $128 \times 128 \times 128$ with similar accuracy. An efficient GPU implementation of the Fast Fourier Transform (FFT) use in MRI reconstruction algorithms is presented in [250]. They evaluated this implementation on two algorithms: look-up table based gridding algorithm and the filtered backprojection method. The results showed that the backprojection $\times 120$ and gridding algorithms had $\times 3.5$ speedups against CPU implementations respectively. The accuracy is also comparable to CPU outputs.

The creation of patterns for common problems like reduce [44], scan [253], sort [247], etc. was also of much interest. Furthermore, common programming languages as an abstraction to code algorithms with GPUs such as BrookGpu [44] made the codification on GPUs simpler.

After the adoption of the accelerators like GPUs for supercomputing in 2008 [270], the explosion of applications with GPUs has increased every year. Figure 2.1 shows the performance during 2007-2020 between supercomputers and GPUs from the NVIDIA vendor.

The effort continues in the theory direction to create efficient representations to exploit spatial locality, for example, the multiplication of vector-matrix with new ways to store the spare matrix [24], highlighting the importance of the low occupancy for specific problems using instruction-level parallelism (ILP) [278], creating more sophisticated libraries for parallel primitives such as reduction, sort, etc., which in the end facilitates the creation of new algorithms from a high level like Thrust [25]. There was also an effort to design parallel low-level primitives to improve the codification on the GPU side, being more efficient, more flexible, and more performance-portable [180].

In Computer Vision, variational methods for segmentation have used GPUs. For example, a Mumford-Shah variational formulation for computing piecewise smooth and piecewise constant approximations, given an input image at VGA resolution (640×480), is presented in [260]. The authors achieved ~ 20 frames per second and real-time video cartooning. Another interesting application is that of the interactive segmentation with scribbles and segmenting the image accordingly, the method proposed in [192] is a variational method considering the spatial variation of colour distributions in a local framework. A prima-dual scheme is solved on GPU at 0.43 seconds per image. In [220], a study of different variational formulations TV-L^1, TV-L^2, and Mumford-Shah is conducted. The authors obtained speedups of $\times 200$ against Matlab implementations. In [251] an algorithm to accelerate dynamic MR applications using infimal convolution of Total Generalized Variation functionals (TGV) is presented. The reconstruction spends only a few minutes triggering the applicability of the proposed method in clinical practice. A parallel implementation to remove multiplicative noise that appears in microscopy, ultrasound, and infrared imaging is proposed in [107]. The speedup achieved is $\times 10.65$ against a CPU-based implementation.

In a non-local framework, the authors in [166] implemented the nonlocal means denoising algorithm [43]. They used several graphics cards for reporting the results and highlighted that since the algorithm is bandwidth-limited a graphic card with better memory bandwidth will perform better. They achieved a $\times 718$ gain against a naïve implementation on CPU.

Nowadays, with the increase of machine learning approaches and the proliferation of frameworks, libraries, etc., it has become crucial to improve inferring and training processes for these models by GPU computation. For example, the convolution neural network *Alexnet* for the classification problem was implemented by GPUs [148]. There are several frameworks such as PyTorch [216], Caffe [137], Tensorflow [1] to name a few which can map the operations on different hardware transparently (for further details see [1]). It is out of the scope of this Thesis to cover all the works in any area using GPUs. However, we would like to highlight some important work in areas outside of Computer Vision, which looks very promising using GPUs.

Data storage has increased exponentially this decade [258] and the way to process data is changing as well. Databases were optimized for CPU [39], but the new demands of industry driven by analysing more data quicker have forced the exploration of new ideas. Since GPU is a mature technology, the database research community has started to accelerate databases via GPU [39].

Another relevant topic outside Computer Vision is to apply GPU to Finance. The daily trading of matching orders (buy/sell) in financial stock markets is a very challenging task. In [240], the authors accelerated the Automatic Order Matching method, achieving $\times 160 - 240$ speedup. Algorithms like STAC-A3 have also used GPUs. As a result, they can simulate more workloads and trading strategies per minute [205]. Using machine learning frameworks like Tensorflow, we can find a work trying to predict stock prices via financial news [154].

Many fields in Science use mathematical models. For example, epidemiological models have been accelerated in [89]. The authors achieved $\times 20$ speedup in the best configuration against a CPU version. A molecular dynamics simulation and molecular docking of the NS3 helicase of ZIKV is presented in [21] for the Zika outbreak. The simulations are accelerated by CUDA. Nowadays, we are immersed in the SARS-CoV-2 outbreak and the research community has also been using GPUs in HPC centres to accelerate the simulations[2]. For example, a supercomputer-driven pipeline for in-silico drug discovery using enhanced sampling molecular dynamics and ensemble docking is proposed in [4]. Since the authors use Autodock-GPU [245] on Summit, they can perform many simultaneous docking calculations that greatly decrease the time-to-solution for screening a large dataset of ligands.

2.2 Saliency Detection Problem

2.2.1 Introduction

There has been a growing interest in visual attention models and their resolution by computational methods in the last years. First, it is interesting to understand how the brain can concentrate on a few important things when considering complex scenes. Every second, around $10^8 - 10^9$ bits $\sim$ 125 MB of visual information reaches our human vision system [32]. The brain does not have enough resources to process all this information [153]. Instead, it applies some analysis optimization mechanisms to tackle this massive income data. Secondly, statistical reports for the use of network regarding visual information show an increase every year [61]. Processing this data efficiently has become an important task and has produced many applications equipped with visual attention models to tackle problems such as advertising, content aware, video summarization, compression, image matching to name a few.

Attention is a general term to group all possible factors which might influence the decision in a selection mechanism [32]. However, we are interested in a specific concept called *saliency*. Saliency can refer to different tasks: eye-fixation prediction [140], objectness estimation, and image-based salient object detection [34]. We shall concentrate on image-based salient object detection.

There is not an exact definition of what saliency is, but it could be formulated as follows: it is the process to identify the parts in an image (regions or objects) that stand out in comparison with their neighbours (Figure 2.2) [32]. Many visual attention models have their foundation in the *Feature Integration Theory* [275]. This theory aims to select which features are important for constructing visual attention models such as colour, orientation, location, etc. (Figure 2.3a). After this, Koch and Ullman [146] proposed a model to combine these features providing a final *saliency map* where the salient elements in the images were highlighted. Based on this model, Itti

[2]The COVID-19 High Performance Computing Consortium `https://covid19-hpc-consortium.org/projects`

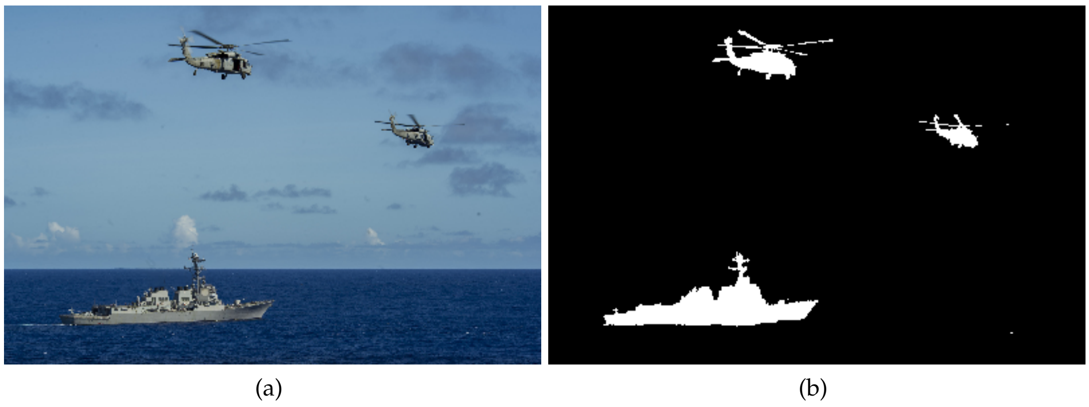


Figure 2.2: (a) Original image from HKU-IS dataset and (b) the saliency ground truth associated to the original image.

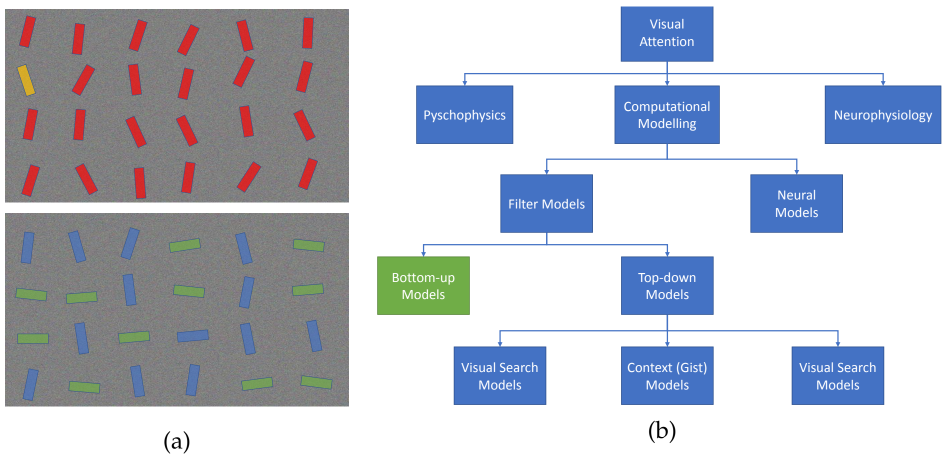


Figure 2.3: (a) Discrimination problem in visual attention systems: it is easy to find the unique item on the top image, but on the bottom image it is difficult because it requires the features of orientation and colour. Image inspired by *Attention-aware rendering, mobile graphics, and games at SIGGRAPH 2014* (b) Taxonomy of the visual attention models. Adapted from [32].

et al. [133] implemented a system and validated it for digital images, giving birth to a new research area in Computer Vision.

A taxonomy is shown in Figure 2.3b which describes possible models proposed in the literature to model Human Visual Attention. Among them, there are two main directions: *bottom-up* pure computationally (pre-attentive data-driven) models or *top-down* (task-dependent) approaches. In this manuscript, we shall focus on bottom-up approaches, which are task-free and do not rely on learning, training, or contextual information. They can also be considered as fundamental building blocks for advanced, robust, hybrid bottom-up, and top-down models.

2.2.2 Saliency Methods

A computer vision system for the *saliency detection* or *saliency segmentation* problem aims to accomplish two stages [34]. First, to localize the salient regions or objects and to label them as *foreground* and second to segment them from the unimportant part called *background*. Saliency algorithms are typically involved in pipelines as pre-processing steps. Then, algorithms for solving the saliency segmentation problem should have some requirements [34]:

- Minimize regions marked as salient when they are not and vice versa.
- The computational time to produce saliency maps should be short.
- Saliency maps should be as accurate as possible. Therefore, objects marked as salient should have a good delimitation.

There are several possible classifications for saliency models (see for further details [34]), but, in this manuscript, we are going to classify them according to the nature of the algorithm: classical ones (unsupervised) with or without optimization formulation and machine learning approaches (supervised). Unsupervised and supervised can also be combined [122, 159, 254, 96].

Some classical approaches build their models in heuristic assumptions *hand-crafted* features to produce the saliency maps. These heuristics are colour, orientation, texture, etc., and they extract these features from blocks or regions in the images [34]. Blocks are simple pixels or identical patches in size of an image, whereas regions are a group of pixels that have some degree of similarity according to some characteristics, for example, colour. Regions estimation in digital images can be achieved by watershed [277], meanshift [63], graph-based algorithms [88], and superpixels: Simple Linear Iterative Clustering (SLIC) [3], Superpixels with Contour Adherence using Linear Path (SCALP) [99], Superpixel Sampling Networks (SSN) [135] to name a few.

Superpixels represent the state-of-the-art decomposition of the image in the feature domain [3]. Superpixels have interesting properties for a partitioning algorithm: **1)** Accuracy in the sense it adheres to the boundaries of the object, **2)** Efficiency as part of a pipeline, it must be a quick method, **3)** Regularity producing high-quality partitions of the image which are compact and uniform, so making the construction of the graph more meaningful [3]. The saliency score s for each region r_i of a partition $P = \{r_i\}_{i=1}^{N}$ may be calculated

$$s(r_i) = \sum_{j=1}^{N} w_{i,j} D_r(r_i, r_j)$$

where $D_r(r_i, r_j)$ is a measure of similarity between two regions in terms of some features (salient prior) and $w_{i,j}$ is a parameter between regions r_i and r_j weighting spatial distance and region size [34].

Most of these classical approaches do not formulate the problem as an optimization one. There exist more structured alternatives to model the saliency detection problem as an optimization one. Bayesian and variational methods are examples of well-established techniques in Computer Vision [19, 37].

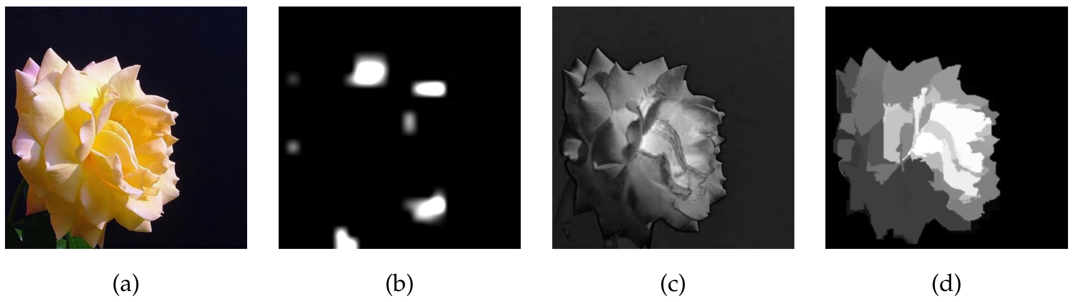


Figure 2.4: Saliency results by classical algorithms without optimization formulation: (a) Original image from MSRA10K [58] (b) Saliency proposed by [133] (c) Saliency proposed by [2] (d) Saliency proposed by [58].

As classical algorithms are based on handcrafted features, authors during the years have proposed several priors to improve the models: centre-bias, optical-flow, face localization to name a few (see for further details [32]).

Since Alexnet [148] occupied the first position with a large margin in ImageNet Large Scale Visual Recognition Competition (ILSVRC [241]) in 2012, the proliferation of computer vision algorithms in any area with machine learning has been enormous [103]. Many deep learning models have achieved successful results in dealing with the saliency detection problem [120, 290]. They typically use trained models for image segmentation, which can extract representative features for saliency detection via transfer learning. However, these approaches are computationally expensive and require extensive labelled datasets. For further details about the challenges, achievements, and future development in saliency with deep learning see [33].

Saliency algorithms typically produce a *saliency map* as output. However, these outputs are images in gray-scale encoding the probability to be salient, white, otherwise black. They must be binarized to obtain the final discrimination in the salient or not salient region. The binarization is usually obtained with a hard fixed threshold or with some adaptive algorithm for automatic discrimination [2, 58].

In the following sections, we shall briefly review the state-of-the-art saliency methods for each class of algorithms.

Classical Algorithms

The first algorithms for saliency detection can be traced back to the seminal work of Itti et al. [133]. A saliency map is determined by using centre-surround operations on colour, intensity, and orientation features using a Difference of Gaussians (DoG) in a multiscale framework (Figure 2.4b).

Later, Liu et al. formulated the saliency detection problem as a binary segmentation problem [160]. They described the likelihood of the salient objects in terms of multi-scale contrast, centre-surround histogram, and colour spatial distribution. They combined these features using a Conditional Random Field (CRF) to produce a bounding box where the salient object is located. A first database was also published to evaluate the model. The database contains salient object annotations with

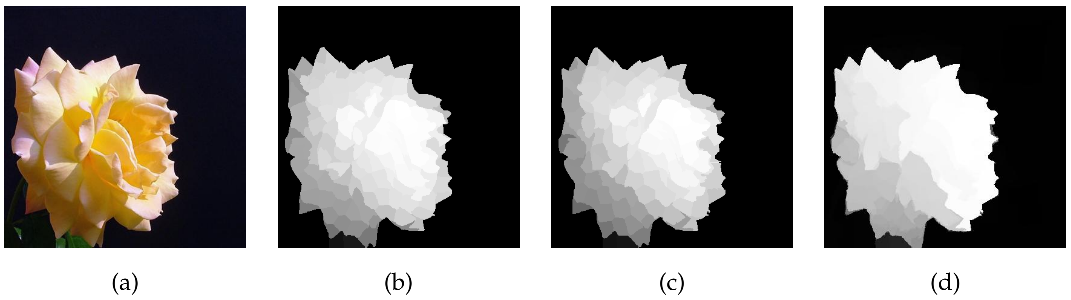


Figure 2.5: Saliency results by optimization algorithms: (a) Original image from MSRA10K [58] (b) Saliency proposed by [293] (c) Saliency proposed by [257] (d) Saliency proposed by [122]

a bounding box.

In the work [2], the authors presented an automatic saliency detection algorithm in the CIE $L*a*b$ colour space as an alternative to the classical RGB. Subtracting the image mean colour to each of the components, a saliency map is produced through a meanshift [63] segmentation and a dynamic threshold. They also provided the first benchmark (1000 images) with pixel-wise annotations for evaluation (Figure 2.4c).

The methods presented above perform the extraction of the saliency maps in the pixel domain. As a segmentation procedure, saliency detection methods can make use of graph-based segmentation techniques [138, 47, 81]. The starting point is an over-segmented image in which regions are progressively merged using different criteria or features, such as colour or contrast. The over-segmentation task can rely on fine-grained morphological operators such as the watershed segmentation algorithm [138, 81] or superpixels approaches [155]. In [58], authors proposed a histogram-based contrast (HC) procedure to measure the saliency of pixels as well as a region-based contrast (RC) metric for regions obtained after a graph partitioning algorithm [88]. Furthermore, the authors derived a way to produce a binary output from the saliency map adapting the GrabCut algorithm [235]. They provided a new benchmark pixel-wise with 10000 images (Figure 2.4d).

Classical Algorithms Using Optimization

Now, we shall focus our attention on those saliency models solved by optimization methods. A formulation to mimic the process in the brain to separate salient and not salient regions in an image was proposed in [286]. In this paper, the authors formulated the problem of saliency as a matrix decomposition of an image I in form

$$I = L + S$$

where the redundant information should be approximately low-rank (L) and the saliency information tends to be sparse (S). The authors transform images into patches and form a low-rank matrix using a sparse over-complete bases dictionary. The optimization problem is solved by a low-rank matrix recovery.

Based on a general low-rank matrix recovery framework, a unified approach to incorporate traditional low-level features (colour, texture, etc.) with higher-level guidance (centre prior, colour prior, and face prior) is proposed in [254]. It solves the optimization problem by low-rank matrix recovery.

In [293] is observed that spatial layout for background regions is different because *"object regions are much less connected to image boundaries than background ones"*. Following this idea, they proposed a measurement to quantify the boundary connectivity of a region (with area and length) (Figure 2.5b). First, the authors decompose the image in the CIE $L{*}a{*}b$ colour space into a partition $P = \{p_i\}_{i=1}^{N}$ of N superpixels p_i with SLIC [3] and calculate the probability of the background w_i^{bg} accordingly to this boundary measurement. The weights w_i^{fg} can be any meaningful saliency measure. The following optimization problem is solved with s_i being the saliency value of region i and $\mathbf{s} = \{s_i\}_{i=1}^{N}$:

$$\min_{\mathbf{s}} \underbrace{\sum_{i=1}^{N} w_i^{bg} s_i^2}_{\text{background}} + \underbrace{\sum_{i=1}^{N} w_i^{fg} (s_i - 1)^2}_{\text{foreground}} + \underbrace{\sum_{i,j} w_{ij} (s_i - s_j)^2}_{\text{smoothness}} \tag{2.1}$$

where w_{ij} are smoothness weights defined as

$$w_{ij} = \exp\left(-\frac{d^2(p_i, p_j)}{2\sigma^2}\right) + \mu$$

with $d(p_i, q_j)$ being the Euclidean distance between the average colours in the CIE $L{*}a{*}b$ colour space. The σ and μ parameters are experimentally fixed.

In the same direction in [257], it is argued that the boundary prior may fail when the objects are on the borders of the image. They proposed a new measurement for the boundary as follows: first, a rough estimation for the background in the image is computed by using objectness proposals (Figure 2.5c). For this, the Binarized Normed Gradients for Objectness Estimation (BING), which is a fast algorithm delivering many proposal objects (bounding box) at 300 fps [59], is adapted for this problem. The image is then partitioned into superpixels (SLIC [3]) and the optimization (2.1) is solved to determine foreground superpixels.

More recently object proposals have been combined with learning techniques. In [122], it is observed that object proposals should be used according to some criteria and the authors proposed a way to filter the objects generated by [295]. The criteria are removing the oversized and undersized proposals and eliminating proposals without enough salient seeds. They created bag instances for the multiple instance learning process (MIL) with the proposals and introduced an iterative algorithm for optimizing the model (Figure 2.5d).

In [159], an adaptive PDE LESD (Linear Elliptic System with Dirichlet boundary) is proposed. A bottom-up and top-down information is incorporated into a saliency diffusion model and the specific formulation, and boundary conditions of LESD are learnt from the given image.

Optimization problems are faced when we model the problem in a Bayesian or

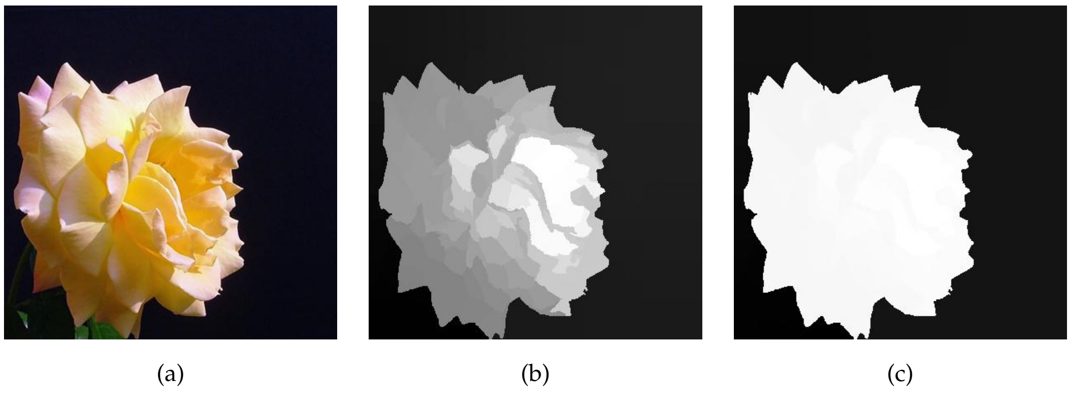


Figure 2.6: Saliency results by variational methods: (a) Original image from MSRA10K [58] (b) Saliency proposed by [281] (c) Saliency proposed by [10].

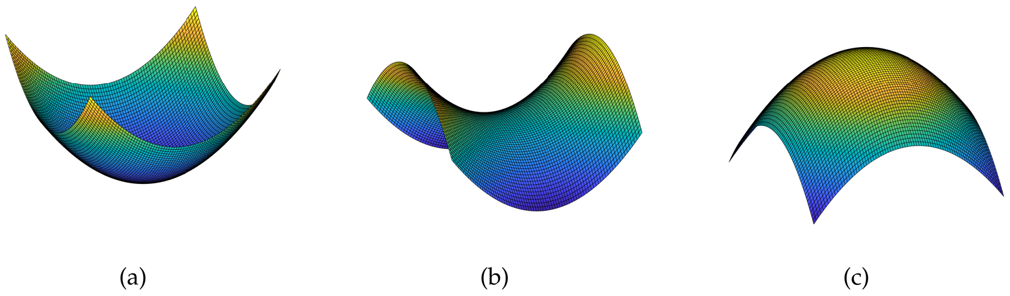


Figure 2.7: (a) A convex surface in $\mathbb{R}^3$. It is the graphic of the quadratic function $f(x,y) = x^2 + y^2$ which has a Positive Definite hessian matrix and a unique global minimum. (b) A convex-concave surface in $\mathbb{R}^3$. It is the graphic of the quadratic function $f(x,y) = x^2 - y^2$ which has an Indefinite hessian matrix and a saddle-point at the origin. (c) A concave surface in $\mathbb{R}^3$. It is the graphic of the quadratic function $f(x,y) = -x^2 - y^2$ which has a Negative Definite hessian matrix and a unique global maximum.

variational framework.

A general variational model for saliency segmentation can be formulated by the following energy minimization problem: Given a function f (the data), compute a solution u in the image domain such that it minimizes the energy

$$E(u) = J(u) + \lambda F(u) \tag{2.2}$$

where $J(u)$ is the regularizer term (*a-priori* information), $F(u)$ is the fidelity term. The λ parameter is a positive constant which allows balancing the relative importance of the terms in the energy functional (2.2).

Typical energy functionals in image processing are convex, just like the function $f(x,y) = x^2 + y^2$ in the two-dimensional case (Figure 2.7). The curvature of its surface is positive and a (global) minimum exists.

The alternating-direction method of multipliers (ADMM) can be applied to solve the general convex optimization problem. Variants of ADMM have been proposed

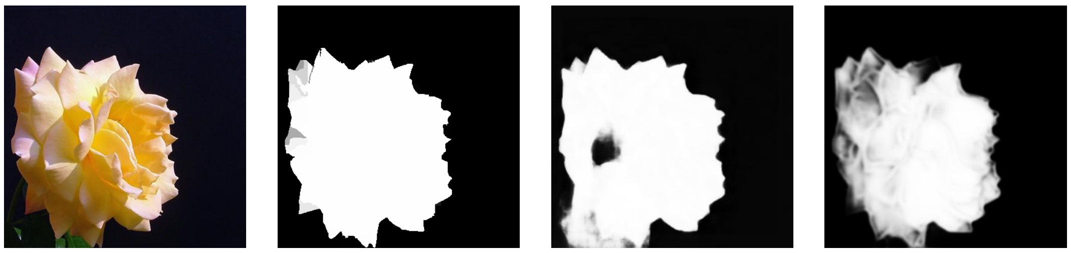


Figure 2.8: Saliency results by deep learning approaches: (a) Original image from MSRA10K [58] (b) Saliency proposed by [96] (c) Saliency proposed by [120] and (d) Saliency proposed by [290].

to solve a (possibly non-convex) optimization problem defined over a graph [281]. Non-convex energy functionals have many attractive properties, see [225, 207], but lack of a general optimization theory [233, 35].

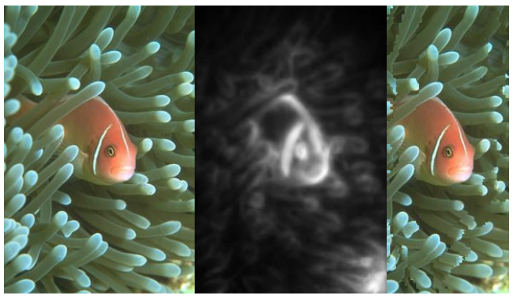
Some works have been published making use of the variational setting for the task of saliency segmentation. Wang et al. formulated a non-local non-convex model, considering the minimization of the L_0 semi-norm in a feature space [281]. The over-segmentation is achieved by superpixels (SLIC [3]) and a graph in the feature space (colour and distance) is constructed, providing a mechanism to link regions with the same similarity using a k-NN methodology. To solve the minimization problem and produce the saliency map (Figure 2.6b), the authors derived an Alternative Direction Method of Multipliers (ADMM).

Based on the framework [281], Alcaín et al. proposed a non-local convex model in the feature space using the Total Variation operator (NLTV) as a regularizer [10]. Since the nature of the model is convex, the authors provided an efficient numeric scheme through a primal-dual algorithm (Figure 2.6c).

There are other approaches for addressing saliency detection with variational formulation, for example, a non-local, pixel-wise convex model is proposed in [179] to segment natural images. In the same non-local setting, non-convexity is investigated in [225] to detect and segment glioblastomas (tumours) in MRI images.

Deep Learning Approaches

The application of modern machine learning methods like deep learning has achieved good results making full use of transfer learning techniques to derive saliency detection algorithms [33] (Figure 2.8b). However, being computationally intensive methods, they perform slowly (about 2 fps) even when using modern hardware to produce saliency maps [96]. They also suffer from images without clear objects, but the latest works have improved these deficiencies. For example, the network models in [120, 290] have achieved good results in different complex datasets (Figures 2.8c and 2.8d). We shall not deepen anymore into this class of neural approaches in this memory. We just observe that hybrid models will drive future, oncoming research.


(a)

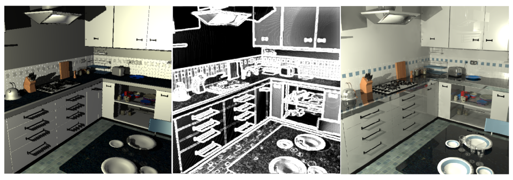

(b)

Figure 2.9: (a) Retargeting algorithm [101] original image (left), saliency map (middle), retargeting result using the saliency map (right). (b) Scenes represented by image preview (left), saliency map (middle), and selective rendered image (right). Image taken from [161].

2.2.3 Applications

In the introduction of this chapter has been mentioned that the saliency estimation is a powerful tool that can be used in many applications mimicking human perception. We list some compelling applications in different fields (Image Processing, Computer Vision and Graphics, Advertising, etc.) to illustrate the importance of the research.

In Computational Photography which is a branch of Image Processing, we can find several applications making use of the saliency information: **Content Aware Image Resizing** is also known as *Retargeting* (to resize the image keeping the aspect according to the important information in the image). This application could use the information coded in the saliency maps to guide the removal of unimportant information and resize the images appropriately to be displayed in different devices and resolutions. An improvement is proposed for retargeting in [101] with saliency maps using Seam Carving as retargeting technique [20] (Figure 2.9a). **Video summarization** uses the information of the saliency maps (where the relevant objects are) as input for machine learning approaches to summarize videos [164]. **Compression** techniques could take advantage of relevant and irrelevant information in the image to compress, for example, in video compression [108].

In Computer Vision and Graphics, saliency maps can serve to deliver higher accuracy resolution for salient areas than non-salient parts in the rendering process [147, 161] (Figure 2.9b). The utilization of the Internet has increased exponentially. Above all, the visualization of videos represents a large percentage of the share [61]. Some works have used the idea of saliency to insert ads in better positions to maximize the user experience [177, 109].

2.3 Attenuation Correction in PET/MR: MRI-CT Synthesis

The use of image processing and computer vision techniques for medical purposes has been one of the revolutions in the past century. Nowadays, the diagnosis of diseases, the effectiveness of cancer treatment [75], detecting tumours [75], predicting Alzheimer's [162] or even strokes [252] by algorithms analysing images is common practice.

Medical images are represented in 2D (slice), 3D (volumes), and even 4D (acquiring multiple 3D images over time). The data to treat is usually large for the algorithms and the use of the acceleration solutions by multicore and manycore is well established (see for example [250]). There are also many medical imaging modalities such as Radiography (X-rays), X-ray Computed Tomography (CT), Magnetic Resonance Imaging (MRI), functional Magnetic Resonance Imaging (fMRI), Magnetic Resonance-Diffusion Tensor Imaging (MR-DTI), Magnetic Resonance Arterial Spin Labeling (MR-ASL), Ultrasound Imaging (US), Nuclear Medicine such as Positron Emission Tomography (PET) to name some of them [261].

Multimodality imaging, also known as hybrid imaging, combines two or more complementary modalities. It brings together the advantages for diagnosis from these techniques, for instance, morphological and functional information, and in this way, it provides better analysis than from only one modality.

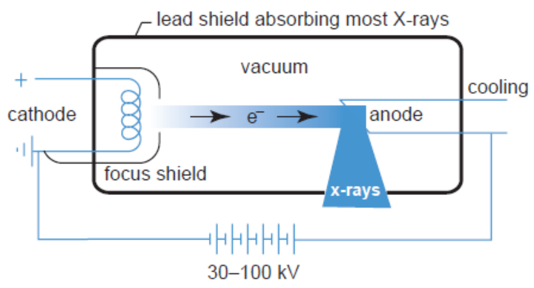
This approach can be achieved by software and hardware. Software methodology acquires images taken at different moments in time and fuses them through image processing while hardware scanners take images simultaneously and create the fusion automatically [168]. The hardware methodology works better because the alignment between both modalities is more accurate [274], but it introduces some engineering challenges for the construction of the scanners [218].

There is also an increasing interest to produce images for a modality from another modality with software. The synthesis from one modality to another modality has several advantages: firstly, the synthesis takes less time than with the normal acquisition process. Secondly, some acquisition modalities use high-energy radiation, which is harmful to the human body. Finally, the produced modality can help provide some relevant information to correct another modality like in Positron Emission Tomography and Magnetic Resonance Imaging (PET/MR) with the attenuation correction (AC).

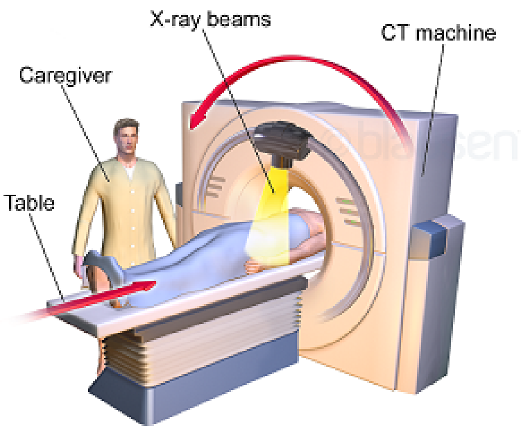
Now, we shall focus on three acquisition techniques: CT, MRI, and PET. They are involved in the multimodality PET/CT and PET/MR. At the end of this section, we present some methods proposed in the literature to address the AC in PET/MR multimodality.

2.3.1 X-ray Computed Tomography

The effects of X-rays were discovered by the German physicist Wilhelm Conrad Röntgen in 1895 after experimentation with cathode tubes. X-ray beams are formed inside a vacuum tube that has a *cathode* and *anode* (Figure 2.10a). When an electrical current circulates between a cathode and anode, the electrons hit the anode and release energy. Some part of this energy is in the form of short-wave electromagnetic



(a)



(b)

Figure 2.10: (a) X-rays formation scheme. Image taken from [261] (b) Illustration of a Computed Tomography (CT) scanner. PET and MRI scanners are similar, but with different detectors. Image taken from `http://publications.lib.chalmers.se/records/fulltext/255299/255299.pdf`

radiation called *X-rays* [261]. These waves have special properties when penetrating the objects that there are in their path. Depending on what kind of tissue or material the objects are made of, they will attenuate more or fewer electrons and this effect will be reflected in the photographic film giving an image where bright parts mean more electrons have been attenuated and black parts mean the electrons pass through [261].

CT image is composed of a stack of cross-sectional X-ray images introduced by Hounsfield in 1972 [121]. CT scanners accomplish two tasks: move the patient to take X-ray images from different sections of the body (*slices*) and move around the X-ray tube to cover 360° for image reconstruction (Figure 2.10b). The X-ray detectors are distributed in a ring and the signal information is transmitted to the computers where the reconstruction takes place. This reconstruction is calculated by complex mathematical algorithms which are based on the Radon Transform. With more data (more images), better accuracy in the images can be achieved. Since X-rays are a type of electromagnetic wave with high energy (*ionizing radiation*), this technique even at low doses can damage or destroy cells [38, 28]. The reconstruction algorithms try to produce better results (higher contrast-to-noise ratio and improved spatial and temporal resolution) with as few images as possible.

The use of CT technology has advantages over conventional X-rays. X-rays can detect hard tissue, but CT includes both hard and soft tissue giving much richer information about the structure of the part scanned. Furthermore, 3D images achieved by CT easily localize the cause of the problems, for example, fractures in the bones.

X-ray is a mature technology that many different fields use to solve problems. For example, as the miniaturization is going further and further and classical approaches for inspection have reached their limits, CT is used today in automobile, aerospace sectors, or electronics equipment (Industry 4.0) for inspection purposes [23]. For clinical use and further information about X-rays and CT see [261].

2.3.2 Magnetic Resonance Imaging

Magnetic Resonance Imaging (MRI) is based on the physics principle of Nuclear Magnetic Resonance (NMR) first described in the 1940s [223, 30]. There was a long journey to produce the recent scanners and obtain the images with NMR technology. In 1971, the pioneering work from the American physician Raymond V. Damadian [73] suggested that the NMR signal could differentiate tumours in the body from normal tissues. No producing an image but analysing the radio frequencies and concluding that the tumour was somewhere in your body. After this, American chemist Paul C. Lauterbur contributed to the basis of creating MRI images by demonstrating that different tissues provide different MRI signals [150] and British physicist Peter Mansfield formalized the reconstruction of the images [214, 163]. Both were laureated by Nobel prizes in Physiology and Medicine in 2003.

The explanation presented here about the concepts in MRI follows the reference book "MRI in Practice" [259]. MR image consists of a set of voxels or volume elements from our body tissue. The MR scanners measure the NMR signal from each of the small volumes through coils or antennas, localize them in 3D space, and plot them on a matrix, typically 256×256 or 512×512, to make a visible picture. The NMR signals are produced by the combination of the spinning motion of the nucleus in the atoms present in our tissues, magnetic field, and specific radio frequency. Atoms have a different type of motions: electrons and nucleus spinning around its own axis, and electrons orbiting the nucleus. A nucleus with a different number of protons and neutrons has angular momentum (net charge). These kinds of atoms are called *MR active nuclei*. For example, hydrogen belongs to this classification with only one proton and the human body has many of them because our composition is 60% water (H_2O) [263]. MR active nuclei, which are spinning and have a net charge, automatically acquire a magnetic moment. The sum of these magnetic moments of the patient in a determined volume is the net magnetization vector (NMV) in the longitudinal direction.

When no magnetic field is applied, the magnetic moments from the nuclei do not present any organisation in the orientations. However, if a magnetic field B_0 is applied, the magnetic moments of the hydrogen nuclei align with B_0. According to the energy of the nuclei, they align in the same direction to B_0 (less energy) or in the opposite direction to B_0 (high energy). The interaction of the NMV with B_0 is the basis for MRI. B_0 also provokes a spin of the nuclei around B_0 itself called *precession*. Every MR active nucleus has different precession values.

Using a radio frequency (RF) in the transverse direction to the nucleus with the same precession frequency initiates a resonance phenomenon. Firstly, there is an absorption of energy changing some energy states in the nuclei population from low energy to high energy. Secondly, the resonance causes the NMV to lie at an angle to B_0 but not being parallel. The angle to which the NMV moves away from B_0 is the flip angle and the flip angle magnitude depends on the magnitude and duration of the RF pulse. A flip angle of 90° transfers the NMV in the longitudinal direction to the transverse. Thirdly, magnetic moments start spinning in-phase and a voltage is induced in the antenna or coil when coherent magnetization crosses the section (*MR signal*). This is the signal to form the images.

Once the RF has finished, NMV goes to the initial position governed by B_0. During this process, the nuclei lose energy *relaxation*. Relaxation comprises:

- **T1 Recovery**: The nuclei release energy in the surrounding environment or lattice until longitudinal magnetization reaches the initial position.
- **T2 Decay**: Loss of the phase of the magnetic moments and therefore loss of the magnetization in the transverse plane.

T1 recovery and T2 decay are intrinsic contrast parameters in MRI. Furthermore, some extrinsic parameters control the RF pulse and are fundamentals in the generation of contrast in MRI:

- **Repetition Time (TR)**: TR is the time between an RF pulse emission and the next in milliseconds. This parameter controls T1 Recovery.
- **Echo Time (TE)**: TE is the time since emitting the RF pulse until the peak of the signal. This parameter controls T2 Decay.

One of the main advantages of MRI compared with other imaging modalities is the excellent soft tissue discrimination of the images. Unlike CT images, bone has a weak signal in MRI and as a result, it appears dark in the images. In addition to that, the electromagnetic waves used to generate the MRI images are not ionizing. For further information about MRI see [259].

2.3.3 Positron Emission Tomography

Nuclear Medicine Imaging makes use of radioactive substances (*radiotracers*) to help to produce functional images. In this classification, we can find Positron Emission Tomography (PET). There is not only one contributor to this area of science but many (physicists, chemists, biologists, physicians, etc.). PET dates to the 50's when the first image device prototype with two detectors was developed at Massachusetts General Hospital (MGH) by Brownell, G.L., W.H. Sweet, and the results for brain tumour localization were presented in [264]. Several refinements to this prototype were added producing a hybrid scanner in the mid 60's. This scanner improved the sensitivity and the possibility to obtain a three-dimensional image. PC-I with 2-dimensional arrays was the first tomographic imaging device [42, 46]. Sensitivity was increased with a circular or cylindrical array of detectors. This approach was proposed by James Robertson et al. in 1973 [231] and Z.H. Cho et al. in 1975 [60]. The ring system is implemented in the PET scanners.

The explanation presented here about the concepts in PET follows the reference book "Positron Emission Tomography" [104] and "Fundamentals of biomedical imaging" lectures [106]. PET imaging is based on measuring the emission of radioactivity caused by an unstable nucleus (number of protons more than neutrons) inside a subject. First, the molecule to produce the emission is prepared in the cyclotron, for example, in Oncology, the use of $^{18}F\text{-}FDG$ is very common. Injection into the bloodstream, inhalation, or swallow methods are typical used to introduce the radioactive substance to the subject (Figure 2.11).

The unstable nuclei emit positrons for a short period. Once a positron is out of the orbit of the atom and has almost zero kinetic energy, it annihilates with the nearest neighbour electron, releasing two γ-photons. The annihilation provokes that the

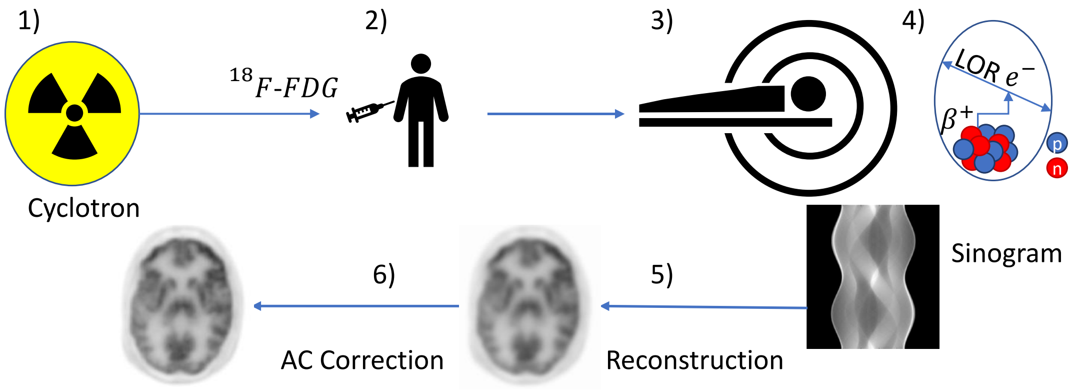


Figure 2.11: Steps in PET: 1) Produce the molecule, 2) Inject it into the subject 3) Emission positron 4) Annihilation process 5) Reconstruction 6) AC correction.

two photons with the same amount of energy (511 keV) move in opposite directions at almost 180° apart (close to collinearly). The distance from the nucleus to the annihilation point determines sensitivity in PET scanners. The length achieved by the positron until the annihilation point depends on its energy and density of the environment.

The annihilation coincidence detection registers two events at the same time. Two events are considered simultaneous if they occur inside a coincidence window, typically from 4.2 to 12 ns, in the detector blocks from the ring of the scanner. This is a *true coincidence*. Apart from the true coincidences (T_{ab}), there exist other types: *scattered* (S_{ab}) and *random* (R_{ab}) events. Scattered events mean that one of the photons does not follow the line of response (LOR) and they are registered in a wrong detector block. This effect is reduced with the introduction of the energy window centre in 511 keV and ignoring coincidence detection outside the area where the emission is not possible. On the other hand, random events mean that two annihilations occur at the same time (coincidence window), but a photon in each pair is lost, making it possible to take it as a true coincidence. Reducing the coincidence window reduces the random events. The energy measured between two detectors A and B can be summarized by the equation:

$$Y_{ab} = N_{ab}(AC_{ab}T_{ab} + S_{ab} + R_{ab}) \tag{2.3}$$

where N_{ab} is a normalization to compensate imperfections in the equipment and AC_{ab} is the attenuation correction. Attenuation is the probability of detecting the photon pair after annihilation. The detector blocks are responsible for the annihilation registration. They consist of crystal and photomultiplier. Once the photon hits the crystal, it produces a visible light *scintillation light* through the photomultiplier, in the detector block. After that, an electronic signal is created. The electronic signal is characterized by polar coordinates (θ and radius) creating a sinogram for every line of response (Figure 2.11). Using the sinograms and reconstruction techniques such as filtered-back projection or iterative method, we pass from raw data to an image.

PET provides different information about the body in comparison with CT and MRI. The principal difference is that PET foundation lies in biochemistry changes at

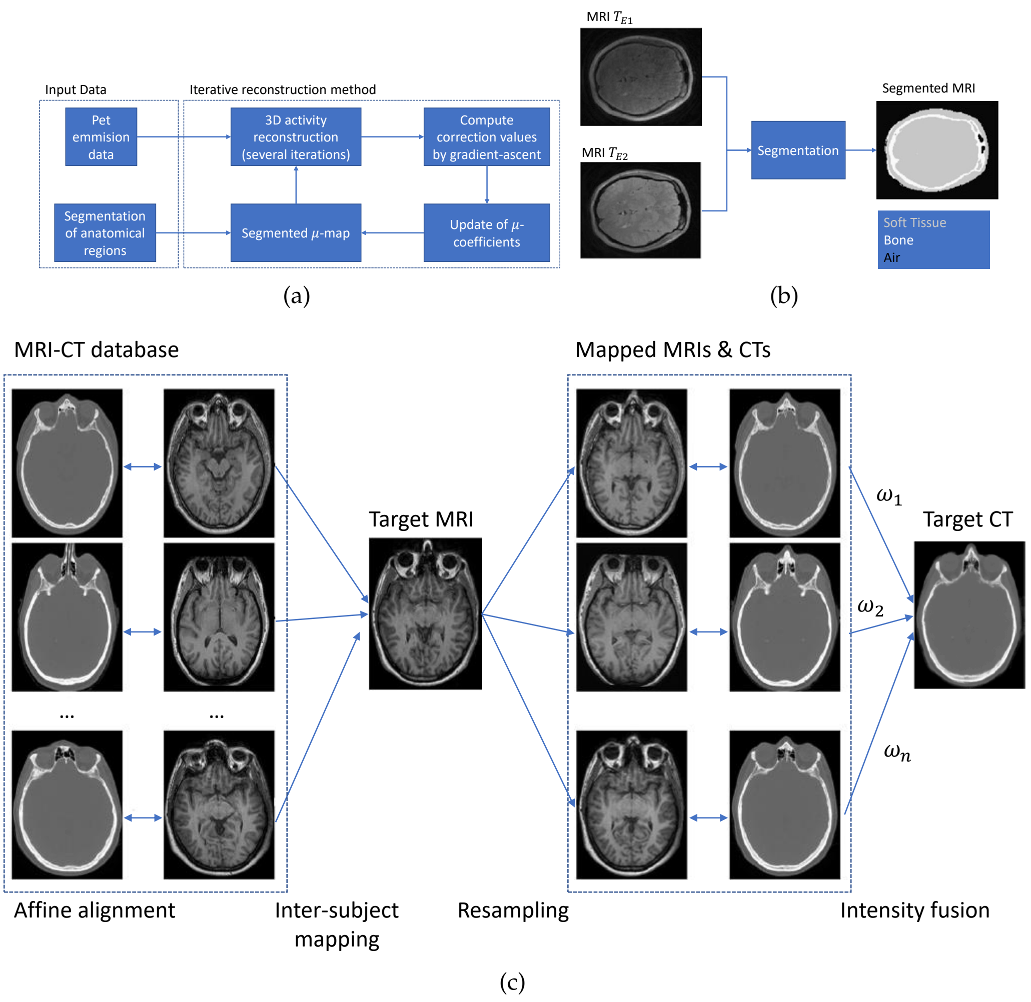


Figure 2.12: Some approaches for AC in PET/MR (a) emission data method from [244] (b) MRI segmentation method from [142] (c) atlas-based method from [45].

the molecular level giving quantitative information [75]. CT registers very well tissue density and MRI proton density and relaxation dynamics. Once anatomic alterations are present, then CT and MRI can diagnosis the diseases. However, chemical changes (in most diseased conditions) precede any anatomic alterations making PET imaging useful to notice them and providing unique information [75].

PET imaging is applied in Oncology: to detect tumours and the spread of cancer, and evaluation of the effectiveness of cancer or in Neurology: Alzheimer's disease and Parkinson's disease to name a few.

2.3.4 Multimodality PET/MR

PET has become a ubiquitous tool when a functional analysis is required, for example, in Oncology [75]. However, to overcome some limitations using this modality (not enough anatomical data and low spatial resolution), it has been combined with anatomical information: CT and MRI.

The first PET/CT prototype was clinically evaluated with success in [29]. CT provides the spatial resolution to identify more accurately the oncological diagnosis and a direct way to correct the attenuation in PET images.

The application success of PET/CT scanners opened the research to explore new modality combinations, for example, MRI and PET. The idea to use together PET/MR is an appealing choice since MRI is a richer modality than CT providing different kinds of images and a high tissue contrast [186]. Furthermore, this modality introduces a lower radiation dose than PET/CT being of importance for repeated studies aimed to evaluate disease progression and therapy response.

However, PET/MR has some disadvantages: the magnetic fields used in MRI interfere with scintillation detectors and multiplier tubes that amplify the PET signal making it difficult to integrate them into one scanner [218]. Apart from that, the AC (Eq. 2.3) for PET imaging is less direct with MRI than CT, as MR provides information on proton density while the attenuation is proportional to electron density [279].

2.3.5 Methods for Attenuation Correction in PET/MR

There has been a great interest in developing AC maps for PET to overcome these difficulties. Several authors classify the approaches to tackle this problem into three categories [45, 56, 134]: emission data, segmentation, and atlas-based. Other authors divide the approaches into two main classes [142]: template-based methods and segmentation-based. We follow the first classification for our discussions.

Emission Data Methods

The core of these approaches to estimate the attenuation information is based mainly on the PET emission data. Attenuation is the probability of detecting the photon pair after annihilation (Section 2.3.3). This estimation of this probability is achieved by using consistency conditions or simultaneous statistical reconstruction [244].

A proposal based on time-of-flight (TOF) PET data and a segmented MR image as an anatomical reference is presented in [244]. TOF technology in the scanners allows approximating where the annihilation occurred in the LOR. They proposed an iterative reconstruction scheme to estimate the local tracer concentration (activity) and the distribution of the attenuation at the same time (Figure 2.12a). Their approach uses a maximum-likelihood estimation for the activity and a gradient ascent based algorithm for the attenuation distribution.

Some authors used non-TOF PET data to construct their proposals. For example, a maximum likelihood reconstruction of attenuation and activity is presented in [194]. Recently, in [256] a convolutional neural network is proposed to estimate the direct attenuation correction of PET images from non-attenuated corrected PET without anatomical information from MR.

Segmentation Methods

Segmentation methods classify tissues in MRI images. In this way, these approaches can map tissue to attenuation values (μ). One of the challenges in this alternative is how to segment bone areas in MRI images. Bone and air space in scans produce weak MRI signals. Nevertheless, their attenuation effects are very different [171]. This discrepancy affects the AC when applied to PET images from the brain and pelvis. We can find segmentation-based methods which give for each tissue class a uniform linear attenuation after the segmentation, for instance, the authors in [171] proposed a correction for the whole body through segmenting (using MRI Dixon) the MRI images into four classes: background, lungs, fat, and soft tissue and assign the μ to all the voxels which belong to this type.

There are some approaches to try to mitigate the difficulties for bone signals using some extrinsic properties in the MRI acquisition. Ultrashort-echo-time (UTE) sequences were designed to visualize tissues with a short T2, such as tendons or ligaments [232], but the cortical bone has even a shorter relaxation time [142]. The authors in [142] solve this problematic realizing a study about the MRI properties of cortical bone and classify the MRI images for the brain taken by UTE sequences into bone, soft tissue, or air (Figure 2.12b). Each tissue class has an attenuation value, which the correction procedure uses.

Atlas-based Methods

Atlas-based methods extrapolate information from a dictionary of pairs (MRI, CT) to generate a pseudo-CT given an input MRI image. The synthesis CT is used to derive the attenuation values in the PET correction.

Two steps are usually necessary: first, there is a stage to align the pairs within the dictionary. The alignment is achieved by a matrix transformation *image registration*: linear transformation (affine registration) or nonlinear registration (deformable registration). The second stage uses a method that relates the input MRI image to the registered MRI/CT images in the dictionary to produce the pseudo-CT. Following [56], we can classify these methods according to how the pseudo-CT is produced into three subgroups: machine learning, voxel-based, and patch-based methods.

Machine learning approaches use the trained data to generate pseudo-CT. A convolutional neural network to map Dixon MRI images (water, fat, in-phase, and out of phase) to their CT image values is presented in [272]. The synthesis is fast allowing the use in the clinical procedures. The authors in [119] combined local pattern recognition with image registration for pseudo-CT estimation. A support vector machine approach to synthesise CT images from Dixon-volume and UTE images was proposed in [189]. The generation of μ-map algorithm was implemented on an Intel Xeon processor with 8 cores producing a result in approximately 3 minutes.

In voxel-based approaches, the work in [45] described a method that uses nonrigid registration to an atlas, followed by label fusion based on patch-similarity measurements using normalized correlation. The authors utilized a ranking scheme in combination with the voxel similarity metric to promote the voxels which have a better similarity. The pseudo-CT is generated based on the weighting among the CT images (Figure 2.12c). An approach to synthesise a pseudo-CT based on nonrigid registration to an atlas is proposed in [134]. They used the standard SPM8 software with a CT template.

In patch-based approaches, a work used T1 weighted MRI images dictionary and a non-linear symmetric diffeomorphic registration to generate the pseudo-CT output [57]. They made use of an air distribution probability calculated from CT atlases images to better discriminate air space in input T1 weighted MRI images. A sparse regression for selecting the most relevant weights for each patch $5 \times 5 \times 5$ in a "search window" ($7 \times 7 \times 7$) is calculated. The registration accuracy is not that important in this method because a neighbour is employed. Since a spare regression is used, the computational is very intense $\sim$ 10 hours.

The work in [15] also used T1 weighted MRI images for the pseudo-CT estimation and affine registration. Weights derived from patch similarity were used to produce the CT values. They applied the structural similarity measure (SSIM) for each patch in a neighbour to discard highly dissimilar patches with a maximum of 8 elements in the search volume. If all patches in the neighbour are discarded, the voxel in the pseudo-CT is marked as unknown. A post-processing procedure labels these unknown voxels.

In [237], the authors presented another patch-based method to generate attenuation values (μ) for the whole head from target MRI images using UTE sequences. They calculate the patch similarities between the reference and target images in the atlas. Then, there is a combination process via Bayesian estimation for the corresponding CT patches to produce the pseudo-CT without registration or segmentation.

Chapter 3

Methodology

In this chapter, we present the techniques which we use in our proposals. Since we intend to implement efficient solutions on multicore CPU and GPU, the first part of this chapter is devoted to describing the computation using shared memory model. We start by explaining parallel computing. After this, we dig into multicore CPU and GPU architectures. Finally, we discuss the challenges when optimizing code.

In the second part of this chapter, we present the computer vision techniques used in our proposals. To understand the techniques used for solving the saliency problem (Section 3.3), we described briefly the variational methods in Computer Vision from local framework to non-local framework. We also present how to solve the formulation with a rigorous derivation of the primal-dual algorithm. After this, we present the patch-based methods used in MRI-CT synthesis (Section 3.4). At the end of this chapter, we analyse the computational complexity of both proposals.

3.1 Computation in Shared Memory

This section aims to provide the foundation of the techniques used in the implementation of saliency and patch-based methods (Chapters 4 and 5).

3.1.1 Parallel Computing

The abstraction between software and hardware is the programming model [172]. Hardware architecture typically influences the design of parallel programming models to exploit the parallelism that they provide. Some authors classify the hardware architecture through memory organization [172, 115]: own address and shared address space. The first group relies on a network for communication among nodes since they have different memory address spaces (Figure 3.1a). There are parallel programming models to exploit this architecture for computation, for instance, Message Passing Interface (MPI) [62]. MPI includes primitives to split the data among nodes and collect the data so that the computation can be achieved in parallel. Clusters and warehouse-scale computers are typical examples of this classification.

In shared memory architectures, we have symmetric (shared memory) multiprocessors (SMPs) (Figure 3.1b) and distributed shared memory (DSM) depending on the number of processors because a large number of them must be organized in a distributed way instead of centralized [115]. Some examples to represent this model are consumer multicore processors, multiprocessors, and GPUs.

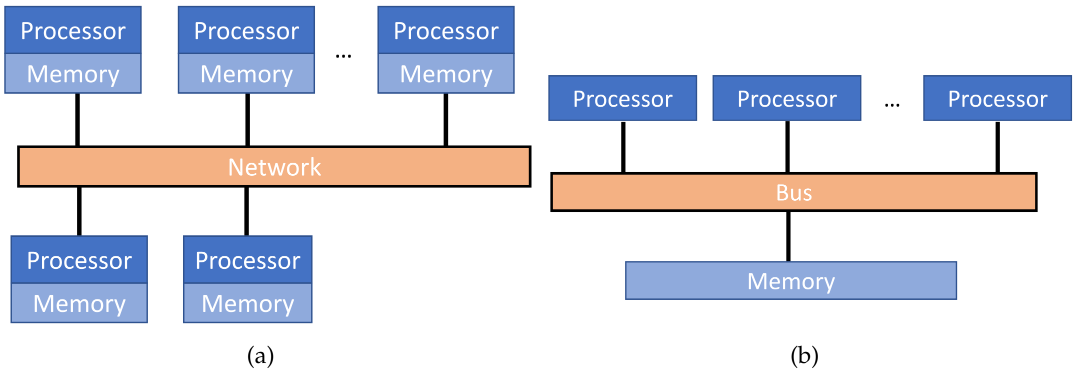


Figure 3.1: (a) Clusters of processors example (different address space). There is a network to connect each processor and with dedicated memory (b) Symmetric (shared memory) multiprocessor (SMPs) example (shared address space). The memory is shared by all the processors through a bus.

Some authors also proposed a combination of distributed and shared memory in hybrid architectures and exploit parallelism [11, 172] or combining accelerators and CPUs in the same chip [143].

In computation, *classical computing* accomplishes the calculations one after another whereas in *parallel computing* achieves many calculations simultaneously [156]. To exploit the capabilities of modern architectures (more than one processor) for computing, we should map the processes or threads to these processors for execution. Parallel programming models are used to express tasks for parallelization. Some authors classify the models according to the nature of the parallelism: implicit, partly explicit, and completely explicit [18]. Other authors, [77], decompose them into pure parallel models for shared and distributed memory architectures (OpenMP [72], OpenMPI [62]), heterogeneous parallel programming models (CUDA [195], OpenCL [187]), Partitioned Global Address Space (PGAS) model for DSM architectures (Unified Parallel C), hybrid (pure parallel models for shared-distributed), languages with parallel support like C# [132], and the distributed programming model (Simple Object Access Protocol (SOAP)). In this work, we shall focus on pure parallel programming models for shared memory and heterogeneous parallel programming models. Parallelism can be defined as

> *The use of concurrency to decompose an operation into finer grained constituent parts so that independent parts can run on separate processors on the target machine* [82]

There are several levels of parallelism according to the amount of the data and number of instructions (Flynn's Taxonomy [90]), (Figure 3.2). In the present report, our attention will focus on Data Level Parallelism (DLP). This level of parallelism applies the same instruction to a set of data. Within this group, our interest lies in data low-level parallelism. This level executes the same instruction for a small data set. Processors must expose these kinds of instructions. They are called Single Instruction, Multiple Data (SIMD). Modern computers include these instructions [126] and they are also the type of arithmetic instructions used by GPUs [157].

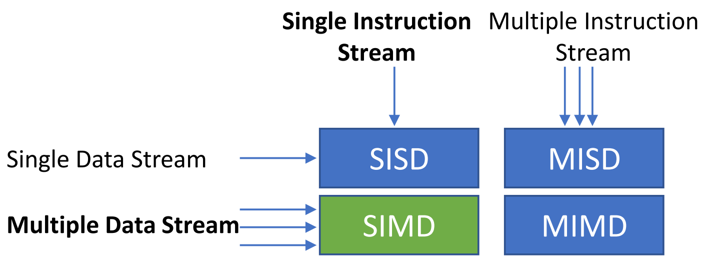


Figure 3.2: Computer architecture classification according to the instruction and data stream (Flynn's Taxonomy). Single Instruction, Single Data stream (SISD), Single Instruction, Multiple Data streams (SIMD), Multiple Instruction, Single Data stream (MISD), and Multiple Instruction, Multiple Data streams (MIMD).

It is also fundamental to assess whether parallel computing techniques will improve an algorithm. For this reason, the improvement when using these parallel computing techniques could be measured, at least theoretically, by Amdahl's law:

$$S = \frac{1}{s + \frac{p}{n}} \tag{3.1}$$

where S is the speedup of a given task, p is the percentage of invested time in the task that can be computed in parallel, n number of processors, and s is the percentage of invested time for the serial part and $s + p = 1$. Amdahl's Law indicates that the theoretical maximum speedup S when used n processors collaborating on a task, cannot be more than $1/s$ (Eq. 3.1) and in this way, the speedup is bounded to how fast the serial part is [13]. However, John L. Gustafson argued that there was a misuse of Amdahl's formula in the assumption p is independent of the problem size. In this sense, we should measure the speedup of a specific task by scaling the problem size to the number of processors, not fixing the problem [110].

It is also crucial to consider the nature of the operations in the algorithm so that deciding which approach to a specific problem is better. We can differentiate two problems: *compute-bound* and *memory-bound* [145]. The first one means that the bottleneck for a specific task lies in the calculations necessary for the algorithm. Expensive functions such as trigonometric, square root, complex custom ones, etc., are typical examples of compute-bound. The second one means that the calculation occupies a small portion in comparison with the reads and the writes of data. Furthermore, the pattern of memory access in the algorithm makes a substantial difference in the performance. Memory usage has become an influential topic in Computer Science since many applications do not improve performance when using faster processors [285].

All these considerations make parallel computing a challenging exercise. Now, we shall describe the typical approaches when using shared memory model for parallel computing.

3.1.2 Parallel Computing on CPU

Parallel computing techniques usually achieve performance improvements on multiprocessors with multi-threading, vector instructions, or both. In the first case, threads represent the way to carry the calculations in parallel computing in shared memory. As multicore architectures support shared memory [115], the idea is to divide the computation into threads. In this way, these threads will be executed in different processors cooperating among them via shared memory. The synchronization of computing threads to avoid non-deterministic outputs of the program (*race conditions*) is necessary [116]. Synchronization is a mechanism provided by the programming model. It assures that processors execute the events or steps in the order that they were designed for. There are many concurrent primitives such as mutex, locks, monitors, etc., for synchronization in shared memory [116].

However, implementing solutions for parallel problems with these primitives could lead most of the time to suboptimal results for real problems not fulfilling the three desired properties (scalability, code simplicity, modularity) [152]. Standards and libraries have emerged to help focus on improving algorithms giving building blocks that encapsulate the parallel knowledge. This encapsulation aims to transparently map the resources of the machine for a specific algorithm. In this way, the code produced could be improved when better hardware is available.

There are several parallel-programming models for shared memory computation [172, 77]. In this manuscript, we follow the classification in [7]: Threading models, tasking models, and directive-based models. Threading models use low-level library routines for parallelizing. Tasking models have their foundation of the concept of specifying tasks instead of threads like Thread Building Blocks (TBB) [267], or Cilk [31]. Directive-based models use high level compiler directives to parallelize the algorithms through compiler support like Open Multi Processing (OpenMP) [72].

TBB is a C++ library that works under ISO C++. The library provides a very highly sophisticated set of parallel primitives to parallelize the code [229]. Apart from the *data parallelism* applying the same transformation to all data, it offers *task parallelism* applying different operations to all data.

Cilk was a research project of the Massachusetts Institute of Technology (MIT) to provide a programming language (C alike) into the multicore landscape [31]. It adds a few keywords to control parallelism and synchronization and provides a compiler to create the executables. When used on only one core, the program in Cilk behaves like a program written serially. The first release of Cilk, called Cilk-1, was in 1994. Today, Cilk is deprecated in Intel compilers as well in other compilers[1]. However, Cilk has been recast in an open source platform supported by an organization *Cilk Hub* (`http://cilkhub.org`) to satisfy demands from software developers, researchers in multicore, and parallel-computing field, and university courses [248].

On the other hand, OpenMP is an industry standard API for shared memory programming, first published in 1997. It covers C, Fortran, and C++ languages, and many compilers support the OpenMP extensions and generate parallel code [72]. The standard became popular for the simplicity of how to parallelize existing code.

The OpenMP API consists of compiler directives, library routines, and environmental variables which are encapsulated in a specification [211] (the latest version is OpenMP 5.0 [210]). OpenMP uses a fork-join thread model for the creation and

[1]`https://gcc.gnu.org/gcc-7/changes.html`

termination of threads. It also uses basic and very powerful programming directives for spawning them and creating work-sharing constructs [72]. It mainly targets intensive data-parallel problems where algorithms perform computation over massive data. The launching of threads for computing a data-parallel problem in a multicore CPU using OpenMP is effortless at the code level. We can make use of OpenMP directives for parallelizing for-loops using a predefined number of threads. Small coding changes are applied to the source code to parallelize it using OpenMP. OpenMP allows configuring the number of threads statically or dynamically, and the operating system would try to spread these threads to the available number of computing cores.

Simultaneous multithreading (SMT) is a technology that allows theoretically to have two computing threads per physical processor [276]. Intel was the first vendor to launch this technology for the computer desktop market in 2002 [127] and modern processors nowadays are equipped with it. On the software side, this technology means the possibility to schedule threads to these logical processors and experiment performance gains up to 30% [167]. Using Intel processors with SMT support (Hyper-Threading Technology in Intel terminology), it can be reasonable to launch two computing threads per available core in the system. The impact of this new technology in the applications was analysed in [243].

In the second case, using vector instructions provided by the CPU manufacturer can be an alternative when the overhead for switching threads consumes more time than the computation. Vectorization is the combination of unrolling techniques and the use of SIMD instructions to speed up calculations [130]. The use of vectorization techniques is a very low-level task and difficult, but some efforts can be made to ease some transparent auto-vectorization. Compilers with optimization choice (-O2) or higher activate the mode to seek possible vectorization opportunities in the code and transform them into the final machine code. These transformations include unrolling loops, memory adjustments, and the use of SIMD instructions when no data dependency is detected, for example, for multiplication operations in an array (Figure 3.3).

Although compilers identify and optimize part of the code automatically [130], organizing the code by the programmer could produce important speedups as well as using different compilers. There is active research to improve the automatic vectorization proposed by the compiler [221] or using machine learning techniques to insert *pragmas* to speed up loops [112]. When using vectorization instructions either manually or transparently through the compiler, the knowledge of assembly code is fundamental to determine whether the result compilation is the expected one (see `https://godbolt.org/`).

3.1.3 Parallel Computing on GPU

A graphics processing unit (GPU) is another hardware device that could be integrated into the system via a bus. GPUs are specialised in computation (computer graphics and video games). The fundamental differences between CPU architecture and GPU architecture are that the design of the GPU has more Arithmetic Logic Unit (ALU) and, in this way to be able to compute in parallel more operations in exchange for losing flow control [145]. Furthermore, they possess small caches to improve the

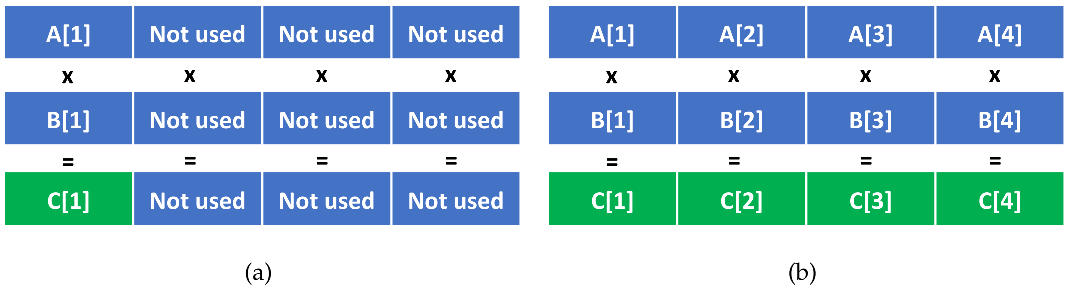


Figure 3.3: (a) No vectorizated operation for multiplication (b) Vectorizated operation for multiplication. Diagram inspired by [130].

bandwidth output instead of reducing latency [283].

GPU has passed through a continued evolution from its origin as a simple peripheral for displaying data on monitors (first generation). The second generation included its memory and specialised processors which allowed rendering effects in 3D configurable but not programmable [213]. It is also the time for the communication standard between the graphic processor and CPU processor as well as the first graphics libraries 3D OpenGL [284], C for graphics (Cg) [165], and DirectX [181]. The third generation had two specific graphic processors which allowed to program two stages: vertex shaders and fragment shaders through a rendering pipeline [213]. In this way, the programmer could extend some effects in the 3D rendering from the prior stage. The next generation moved on to unified device architecture, after merging these two processors into one [195]. Although graphic APIs had provided a wide range of functions, the programming, known as General-Purpose Computation on GPUs (GPGPU), was still difficult [113]. There was an effort in the software direction to create a model to make a better manipulation of the resources and let the programmers map more general problems (GPU Computing) [117].

As has happened in many fields of Computer Science, there has been a great effort to standardise the GPU computing model. This effort is still a work in progress many years after the initial release of OpenCL (Open Computing Language) (2008) [187]. OpenCL is currently being developed by the Khronos Group (known for OpenCL and other standards). OpenCL emerged from a collaboration among software vendors, computer system designers (including designers of mobile platforms), and microprocessors (embedded, accelerator, CPU, and GPU) manufacturers. The idea behind this standard is to promote multi-platform adoption rather than being bound to a single vendor. OpenACC (Open Accelerators) is a compiler specification to target multicore CPUs and accelerators like GPUs for high productivity. Similar to OpenMP, compiler directives are used to determine which part of the code should be parallelized transparently [209].

There are several GPU manufacturers such as AMD/ATI, Intel, and NVIDIA to name the most relevant ones. In this report, we shall discuss NVIDIA manufactured GPUs.

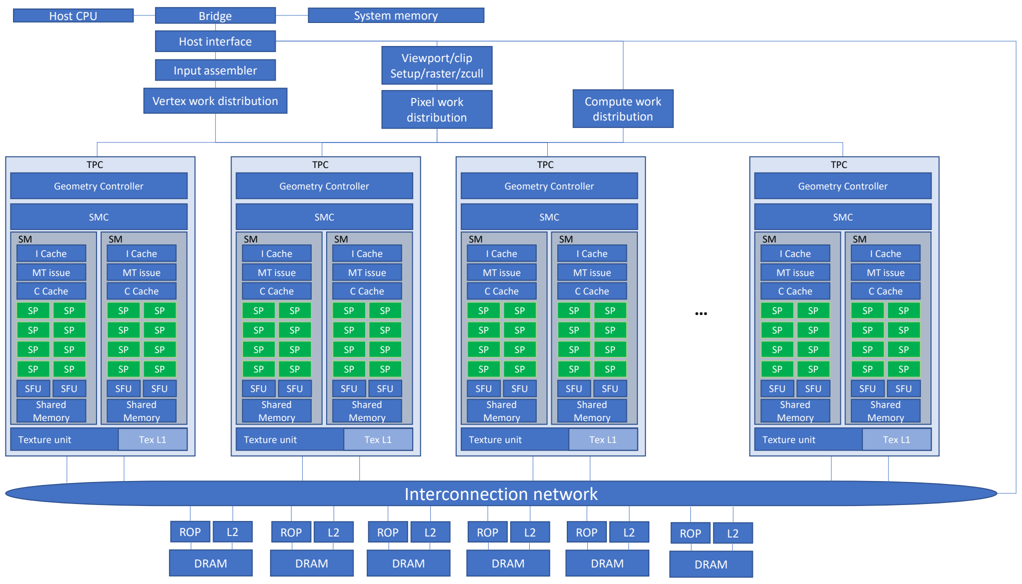


Figure 3.4: Tesla unified graphics and computing GPU architecture. TPC: texture/processor cluster; SM: streaming multiprocessor; SP: streaming processor; Tex: texture, ROP: raster operation processor. We only show 8 SMs to keep the diagram simple, but Tesla contains 16 SMs. Diagram inspired by [157].

3.1.4 NVIDIA CUDA

The lack of high precision operations and difficulties of mapping real problems using graphics APIs led to the exploration of alternatives that would allow developers to exploit the power of GPUs [113]. Several research projects looked at designing programming languages that would help simplify this task like Sh (Shader Algebra) [174, 173] and in 2007 NVIDIA introduced its CUDA architecture which provided tools to make data parallel computing more direct. This work was the continuation of the BrookGPU project at Stanford University [44]. Our attention will now focus on the architecture and programming model for the NVIDIA vendor.

3.1.5 CUDA Hardware Architecture

NVIDIA's GPU microarchitectures have evolved since CUDA was born in 2007 [195] and several microarchitectures have been proposed to improve the performance (Table 3.1). For simplicity, we take *Tesla* microarchitecture to illustrate the general ideas of how the GPU computes and its organization. The GPU architecture is built around a scalable processor array with Streaming Multiprocessors (SMs) and each SM has several streaming-processor (SP) or CUDA cores (Figure 3.4) [157].

Stream multiprocessors are responsible for creating, managing, and executing the threads. CUDA cores execute threads. The thread is the element unit for computation in CUDA [65].

The threads belong to a block and the block is scheduled by Compute Work Distribution (CWD) in any order [157]. CWD fills each SM with blocks up to the execution capacity (Figure 3.5). The threads of a thread block execute concurrently on one SM. When the thread blocks finish the computation, the CWD unit launches new blocks on the vacated multiprocessors [157]. This ability to handle the execution of

Table 3.1: NVIDIA micro-architecture evolution during the years (2007-2020). Architectures: Tesla [157], Fermi [200], Kepler [201], Maxwell [202], Pascal [203], Volta [204], Ampere [197] and new features [242]. CUDA c.c. stands for CUDA Compute Capability. * FP64 double precision available from this version onwards. Die in nm, Bandwidth in GB/s and Peak FP32 in TFflops. Name refers to relevant graphic card products in a specific architecture.

	Info			Features			
Year	**Arch**	**CUDA c.c.**	**Die**	**Cores**	**Bandwidth**	**Peak FP32**	**Name**
2007	Tesla	1-1.3	90	128	76.80	0.345	Tesla-C870
2010	Fermi	2.1	40	448	150.36	1.028	Tesla-C2075*
2013	Kepler	3-3-3.7	28	2880	288.46	5.046	Tesla-K40c
2013	Maxwell	5-5.3	28	2560	332	6.688	Tesla-M10
2016	Pascal	6-6.2	16	3584	732	9.526	Tesla-P100
2018	Turing	7.0-7.2	12	2560	320	8.141	Tesla-T4
2018	Volta	7.5	12	5120	897	14.3	Tesla-V100
2020	Ampere	8	7	6912	1555.80	19.5	Tesla-A100

the thread transparently produces program scalability (Figure 3.5). It is up to the scheduler to feed these multiprocessors with work. It also maximizes the maximum number of multiprocessors in use at any given time which, will, in turn, increases performance.

All threads cannot be executed at the same time. To balance this large population of threads efficiently, the GPU employs a Single Instruction Multiple Data (SIMD) architecture [191] in which the threads of a block are executed in groups of 32 (*warps*). A warp executes a single instruction at a time across all its threads. The threads of a warp are free to follow their own execution path and all such execution divergence is handled automatically in hardware [24]. Therefore, if no diverge occurs all threads follow the same execution path and the computation is efficient (see [74] for further details).

CUDA threads may access data from multiple memory spaces during their execution (Figure 3.6) [70]. Each thread has private local memory and registers for fast access. Each thread block also has shared memory (L1 with high bandwidth and low latency access) visible to all threads of the block and with the same lifetime as the block. There are also two additional read-only memory spaces accessible by all threads: the constant and texture memory spaces.

All threads have access to the same global memory (DRAM). Global memory is conceptually organised into a sequence of 32-, 64-, or 128-byte segments [71] and connected to the host through a high-speed IO bus slot, typically a PCI-Express and in current high-performance systems NVLink [71]. The data which is manipulated by the threads come from the host to the global memory through this bus. Read and write operations are executed through transactions. The memory requests are serviced as segments for a half-warp (16 threads) and operated by transactions [70]. *Coalesced memory access* refers to aggregate the memory accesses in a half-warp into a single transaction, in this way, it reduces the transactions and improves the performance [70]. The number of memory transactions performed for a half-warp will

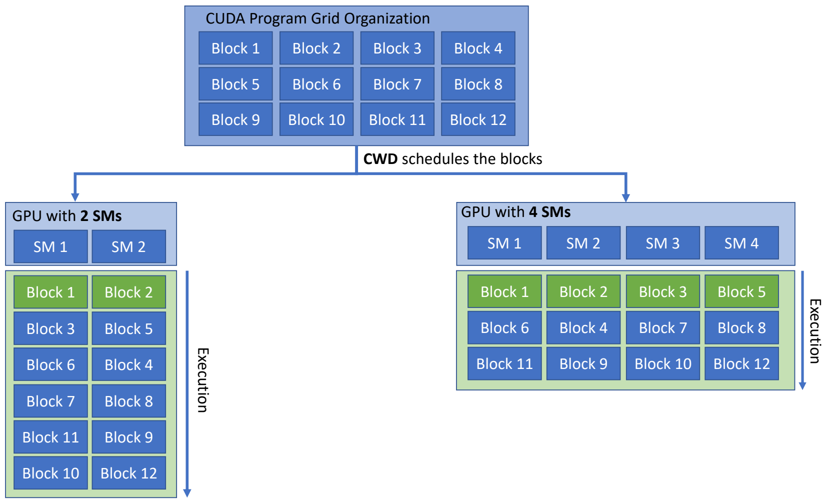


Figure 3.5: Tesla architecture execution. The program is split into several blocks of threads and the scalability is achieved by the type of graphic processor. Note that the execution of the blocks can be in any order. Diagram inspired by [71].

be the number of segments touched by the addresses used by that half-warp. Non-contiguous access to memory in thread blocks is detrimental to memory bandwidth efficiency and the overall time.

The features of the hardware, set of instructions supported by the graphic card as well as other specifications, such as the maximum number of threads per block and the number of registers per multiprocessor are specified in the *compute capability* [71].

3.1.6 CUDA Programming Model

In the CUDA programming model, the program, typically in C++, is divided into two parts: host code and device code *kernels* [71]. It is up to the host code to transfer and retrieve the data from the graphic card memory. It is also a host task to allocate and release memory from the graphic card memory, and, lastly, organizes the kernel calls synchronously or asynchronously.

On the host side, programmers describe the problem as a composition of grid and blocks, (Figure 3.7), to cover as much as possible the processors (SP)[2]. A thread block is a set of concurrent threads that can cooperate among themselves through barrier synchronization and shared access to a memory space private to the block [283]. A grid is a set of thread blocks that may be executed independently, i.e., in parallel. The data access inside the blocks is achieved by a unique identification which determines the element by a tuple (Figure 3.7).

Kernels represent the single instruction in the SIMD architecture and are executed on the graphic processor using the data transferred by the host code [157]. Kernels are typically written in C++, but the programmer has always the choice to write the code in the Parallel Thread Execution language (PTX) [215]. In both cases,

[2]This is a good practice to hide latency doing operations while waiting, but sometimes it is more beneficial to exploit Instruction-level parallelism (ILP) [278].

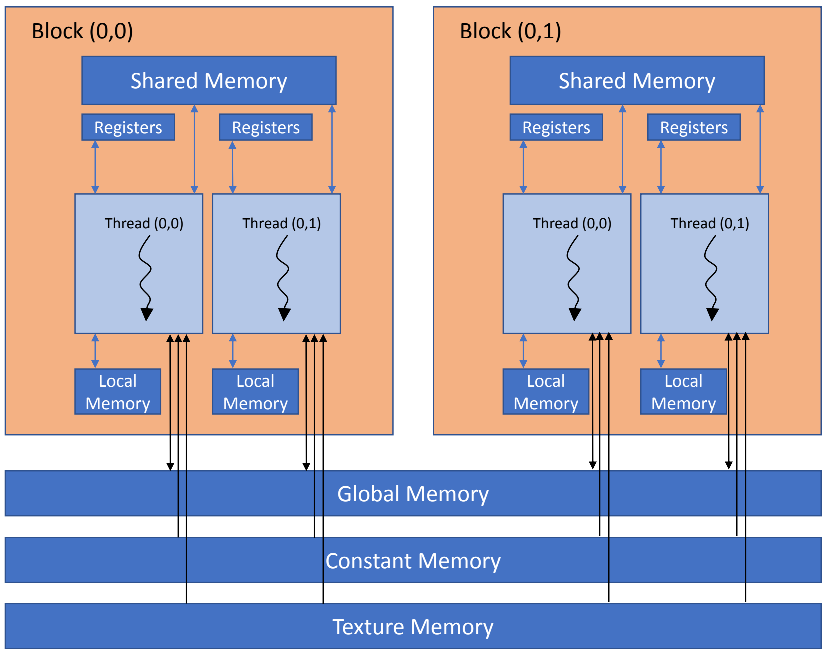


Figure 3.6: Threads can access during execution different types of memory. Coalescing memory patterns in a block of threads improve the final throughout. Diagram inspired by [70].

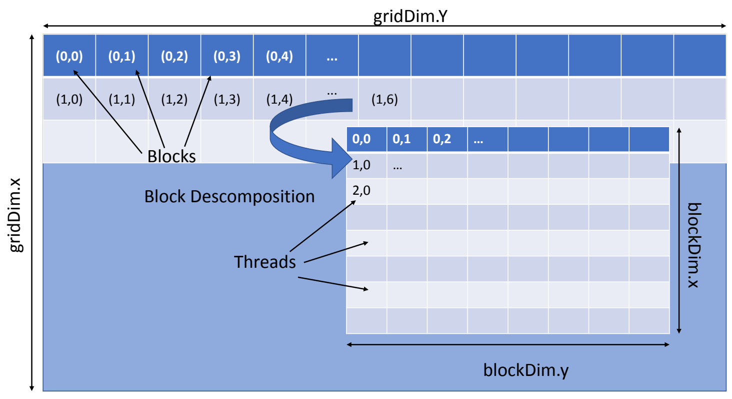


Figure 3.7: Grid and block decomposition in CUDA. For simplicity, we use a 2D grid and block.

kernels must be compiled into binary code by nvcc. Nvcc is a compiler driver that simplifies the process of compiling C++ or PTX code providing a command line to accomplish the different stages of compilation [71].

As it happens in other areas in Software, problems with the same pattern appear frequently. In parallel computing, there has been a continuous effort to derive strategies to face these problems efficiently [172]. We shall explain two of them that we have used in our implementations: tiles and reductions.

3.1.7 Parallel Patterns: Memory

One of the most fundamental problems when optimizing code is the use of memory in an optimal way [70]. There are two ways to improve this: spatial locality with new data structures and temporal locality with ordering loop operations to achieve coalesced memory access [70]. In GPU computing, this problem is more visible because there are many threads concurrently in an algorithm and bad patterns could introduce penalties in the performance. There has been an effort to derive patterns so that algorithm design can address the problems more efficiently.

Spatial Locality

The reorganization of the data to have a better allocation and in this way, better access memory is a very important topic in the GPU computing paradigm. For example, there are many representations to have the best access possible according to different sparse matrices (see [24] for matrix-vector multiplication). These strategies can be applied to other similar problems like our saliency detection algorithm (Section 4.1).

Temporal Locality

On the other hand, temporal locality improves the memory access with the idea to use in a better way the caches, e.g., reducing the stalls. For this goal, some strategies divide the problem into small blocks that could fit in the cache. In this way, improving the final performance when applied to complex calculations in batches (Section 5.1).

3.1.8 Parallel Patterns: Reduction

There are types of algorithms that are not suitable for GPU computing at first glance. This handicap lies in the operations within the algorithms and the difficulty to transform them into an efficient parallel operation. A classic example of this type is the summation of elements of an array. Since the summation is a mathematical operation with associative property[3], then it can be easily decomposed into subtasks, and therefore, it becomes very suitable for computation in parallel (Figure 3.8). However, the coordination needed to accomplish the task makes it difficult for its implementation without any pattern. These kinds of obstacles are solved by parallel reduction.

The parallel reduction is a common and important data parallel primitive, but it is difficult to get the most out of it [266]. The reason for that is these algorithms where parallel reduction is needed, typically, suffer from memory-bound problems,

[3]Bear in mind that in Computer Science is ONLY strict for integers.

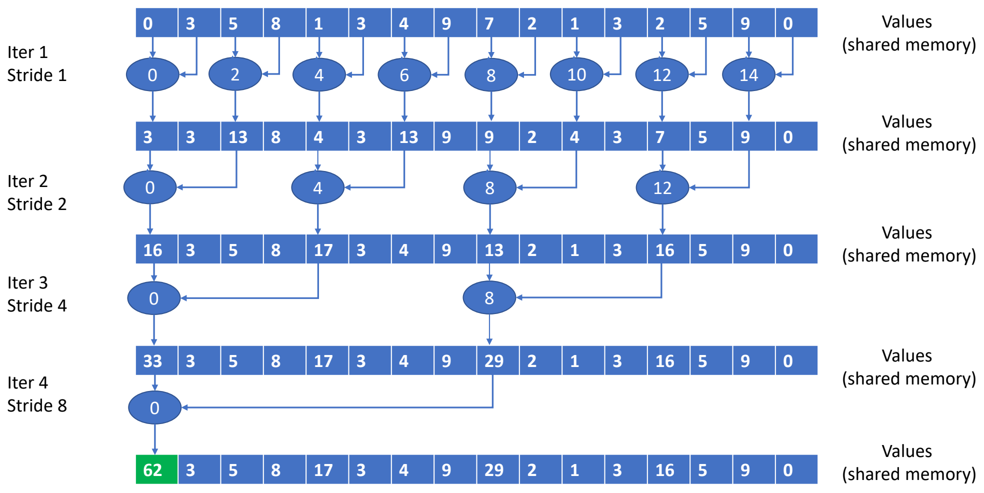


Figure 3.8: Reduction using interleaved addressing. Note that the threads involved in the calculation are halved in each step of the reduction process, in turn, losing computation power. Diagram inspired by [266].

e.g., they lack computation, and the way to access the memory for the small calculations is crucial. Figure 3.8 shows that fewer threads are involved in each step of the reduction, therefore, losing computation power. This is the reason why many libraries on top of NVIDIA implement efficient reduction methods. These reduction methods (scan, sort, reduce, etc.) were encapsulated at the beginning in [114]. After that, some libraries such as Thrust [25], cuBLAS [69] appeared to tackle the problem from a high level perspective. However, sometimes these techniques must be applied inside the kernels to improve algorithms in a more low-level fashion (see CUDA Unbound (CUB) [196]) or implementing your custom solution.

3.1.9 Software Optimization

Software Optimization refers to the process that makes the software work as efficiently as possible [78]. Efficiently has different meanings depending on what is relevant for each application. In this Thesis, we are interested in those optimizations which reduce the time of the overall algorithm (see Efficiency definition in [172]). The optimization makes your code fast and usually unreadable as well. The challenge is to produce optimized code preserving some essential properties for software: object-oriented programming, readability, modularity, and reusability [152]. It has been proved that these concepts can be encapsulated successfully in the same library, for example, in Standard Template Library (STL) [229] or CUDA Unbound (CUB) [196].

Before starting any optimization technique, the choice of a programming language to code the problem is important. Not all programming languages offer the same path for optimization: for example, interpreted languages like Pyhton [224] performs slowly in comparison with other languages because the interpreter must read code, perform the instructions, and update the machine state. On the other hand, Just-in-time (JIT) compilers interpret and generate in run time native code and in this way alleviate these problems (C# [132]), programming language like C++ which generates native code are more interesting from the optimization point of view because they allow having more control of the optimization paths in comparison with Python where the performing model is difficult to figure out. Chapter A.1 illustrates an optimization case for the matrix multiplication problem.

The use of different compilers for C++ on the same operating system has achieved substantial speedups. Therefore, its election is also an important task. Microsoft Visual Studio is a popular platform providing a user-friendly Integrated Development Environment (IDE), but its compiler, **cl**, is not the best choice because of the lack of the latest instruction set to exploit the hardware [91]. Intel compiler, **icc**, can be integrated into Microsoft Visual Studio and it offers good performance in code optimization. Any optimization process must have a reference to compare with and this reference must also provide the expected result. The comparison with the optimization versions is in speed and accuracy. The procedure of optimization makes use of tools to spot the part of the code which occupies more time in the whole process. These tools (profilers) retrieve useful information of the code and present them in a form that it is easier to find the bottlenecks in the code. They are based on different methodology: instrumentation, debugging, time-based, and event-based [91].

Based on these foundations to acquire data and since the program to analyse is not executing alone on the operating system, profilers are sometimes not as accurate as expected, some tools are: VTune for Intel Compilers [131] on CPU and NVIDIA Visual Profiler [206] on GPU.

When optimizing algorithms, we can typically find two kinds of limitations: compute-bound and memory-bound [145]. For algorithms that do complex calculations most of the time in the innermost loop, for example, converting from one colour space into another in Image Processing, the algorithm execution improvement is bounded by the computation (compute-bound). On the other hand, others spend on reading and writing most of the time, but the number of calculations with this data has a very low arithmetic intensity, for instance, summation procedure in large arrays (memory-bound).

Understanding the problem and its bound limitations is an iterative process. Most of the time, after working with the problem for a long time (testing, analysing, and studying the bottlenecks), some good ideas can emerge, discovering new chances, hidden previously with the lack of comprehension of the problem structure, to improve the performance of the code [152]. For example, these new opportunities can be materialized with smarter data organization of the problem.

Software optimization is a complicated task because it depends on many resources that work in cooperation and usually do not have a deterministic output: processor, memory, network, data structure, etc. Although some practical ideas can be followed [27], the experience, the experimentation of different approaches, and knowledge make finally the real work. It is also this experience and knowledge which is applied to optimize the two novels algorithms in Computer Vision in this Thesis.

3.2 Variational and PDE-based Models in Computer Vision

In this manuscript, we propose two novel variational algorithms to solve the automatic and unsupervised saliency detection problem. In the first one, a pure Bayesian modelling approach is considered and an energy functional is constructed and minimized in a graph of characteristics features. While the results were encouraging and comparable with the (variational) state-of-art results in [281], saliency was not explicitly modelled. The successful application of the method strongly depended on pre-calculated data information called *control map*. To overcome this issue, a general non-local model is then proposed. It can be deduced following a PDE-based approach where the dynamics of the governing (transient) PDE equation drives the solution towards stabilization to a stationary state. Notice that in our numerical experiments convexity is preserved and this allows considering a well-posed model in a pure variational setting. The non-convex scenario is also covered by the proposed model. We shall explore it in future work.

Independently from the modelling tool used to justify our formulation (PDE-based or variational techniques), the main goal of our proposal is to promote a binarization of the solution leading, in the graph of characteristics, to a clear, robust solution which is our final saliency map.

This section aims to provide the rationale of the techniques that we shall use to model and solve the automatic saliency detection problem in Computer Vision (Section 3.3).

3.2.1 Bayesian Modelling in Computer Vision

Bayes Theory is a building block technique for modelling artificial vision problems in Computer Vision [97] and a fundamental guide for the design of feasible energy functionals which must be optimized, typically minimized. After a *prior* has been selected, this provides us with a Maximum a Posteriori (MAP) estimation of the solution. The underlying idea is to penalize images that do not meet our *a-priori* hypothesis on the optimal image to be recovered. For example, we can assume *a-priori* that natural images have finite jump discontinuities at the edges and are therefore piece-wise continuous functions. We then have to choose a regularizer (the *a-priori*) that promotes (do not penalize) discontinuous functions. This is introduced in the next section where the Bayesian approach is presented and the associated inverse problems are formalized. Optimization theory is then applied to deduce the governing Euler-Lagrange equations of the model. This motivates and introduces the use of variational techniques in Image Processing as an established and principled way to analyse and solve the proposed models. As a byproduct, we have a mathematical framework in which the models are analysed, discretized, and numerically solved. This is proposed in [10] where a variational model on a graph is considered for automatic saliency detection. Physical process can also be envisaged and exploited to drive fast dynamics and related digital processing. Consequently, we develop further the modelling exercise to include a reaction term in the energy functional and associated Euler-Lagrange equation.

The mathematical approach to image processing based on variational calculus is called continuous to differentiate it from discrete settings where the image is considered, since the beginning, as a matrix. Our solution (the recovered image) is the function that minimizes the energy functional and it is sought for in an infinite-dimensional vector space. Banach spaces of the Lebesgue type must be considered as the natural spaces in which the problem admits a solution. This has interesting consequences when an image processing task is considered. In fact, the choice of the proper regularizer (*prior*) is turned into a problem of regularity and qualitative behaviour of solutions of quasilinear elliptic partial differential equations for which vast literature is available [40, 51].

In the interface between Applied Mathematics and Computer Vision, a great effort arose in the last years for non-local variational calculus [98]. This is interesting in applications because non-local interactions and patterns in the image domain, and patterns and characteristic features in the data space can be conveniently modelled. This introduces the need for differential calculus in continuous manifolds and discrete graphs.

Non-local variational methods and graphs of characteristics are presented at the end of the section to develop our proposed model for automatic saliency detection in natural images. Section 3.2.6 describes fully the theory of discretized differential operators in graphs introduced in [84] and explains how to apply it to our continuous non-local model.

3.2.2 Inverse Problems and Bayesian Modelling

To illustrate the technique which allows the formulation, in a variational setting, of many low-level problems of Computer Vision, we start with the description of one of the most classical and studied general model problem: the *image restoration* (or reconstruction) problem. We shall follow a *local*, point-wise framework. Section 3.3 presents the non-local setting based on the same conceptual steps here considered.

In Computer Vision, there are two ways to describe images and later to process them accordingly [68]. Digital mages (natural or synthetic) are a set of discrete, quantized values representing the gray-scale intensity at each pixel or voxel defining the image domain (a discrete mesh) and its resolution (the number of pixels). These values are stored and collected into a matrix (for gray-scale images) or a tensor (for colour or pseudo-colour and in general multi-channel multi-modality images) and represent our (perturbed) data. On the other hand, images can be described in a continuous setting, [19, 53, 37]. Much like a fluid or an elastic body of deformable material in continuous mechanics, images can be transformed by some physical process, typically heat diffusion and non-linear variants. Viscous fluid models can be found in [41]. Elastic image registration models are reviewed in [14].

These processes are governed by PDE which must be solved. This involves the discretization of the derivatives and the numerical resolution of the PDE which models the physical process considered. The final solution is the reconstructed discrete (digital) image. In such a continuous framework, previous to the discretization step, images are functions defined in a subset $\Omega \subset \mathbb{R}^n$ (the continuous image domain) taking values in $\mathbb{R}^d$ in form $f : \Omega \subset \mathbb{R}^n \mapsto \mathbb{R}^d$ where n represents the dimension of the images and d the number of channels.

Several advantages of the continuous approach can be deduced. The images that our brain processes are continuous. Resolution issues only arise when digital

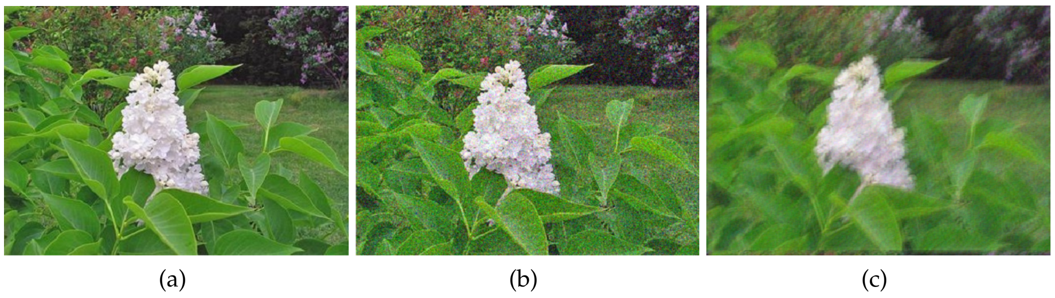


Figure 3.9: (a) Original Image from MSRA10K [58] (b) Noisy data image (input) for the denoising problem (Eq. 3.3) (c) Noisy and blurred data image (input) for the inverse reconstruction problem (Eq. 3.2).

images are mechanically acquired, but they do not when continuous models are considered. Furthermore, the tools in the continuous domain are more abundant and established: functional analysis [40, 51, 12], differential geometry, differential equations [19], convex optimization [35, 233], duality theory [233, 83, 48, 52], etc. Some properties, such as the rotational invariance of an image, mass conservation of the initial noisy data, the maximum principle [233, 222] or the existence of edges preserving regularizing flows are easier to model in the continuous setting because the physics laws models the continuum. Finally, it is often possible to show that the discrete model tends to the continuous one if the discretization parameter goes to zero [68]. This confers extra control on the algorithms used to solve the problem because the qualitative properties of the solution of the continuous model should be reproduced by the numerical solutions. With a view to model saliency in natural images, we shall follow this more general approach.

The degradation of an image occurs basically from image acquisition, for example, a blur caused by the motion of the camera (Figure 3.9c), and some random data transmission errors (Figure 3.9b). This is modelled by the *forward model* for image reconstruction which describes how the observed digital image f is obtained:

$$f = Au + \eta \tag{3.2}$$

where u represents the original data, A is usually a linear operator affecting the signal u (modelling blur, downsampling, integral transforms, etc.), and η is the noise distribution which is usually assumed to be an Additive White Gaussian Noise (AWGN). This last term is dependent on the specific application and image modality. The image processing task is therefore to recover u given f leading to an *inverse problem*.

When A is the identity operator, i.e, $A = I$, we have $Au = u$ and Eq. 3.2 reads

$$f = u + \eta \tag{3.3}$$

This is the forward model for image denoising. To recover the original, clean image u from perturbed data in f is the associated inverse problem. Inverse problems are ubiquitous in Science and, in particular, in Image Processing. Most of the inverse problems such as (3.2) and (3.3) are *ill-posed*. To understand the scope of the statement, we notice that a mathematical problem is well-posed (in the sense of

Hadamard [111]) if and only if the following three conditions are verified: **1)** there exists a solution for the problem (existence), **2)** the solution is unique (uniqueness), and **3)** the solution changes accordingly to the input data (stability or continuous dependency of the solution from the data). When whichever of these conditions is not fulfilled the problem is termed *ill-posed*.

Conditions **1)** and **2)** about existence and uniqueness imply that there exists a bijective map between the space of data and the space of the solutions. Notice that they do not need to be in the same space because some kind of regularity is usually introduced to reduce the noise. In practice, these conditions are seldom met in models (3.2) and (3.3). In fact, following [227], let us assume that f is modelled as a random variable $f \sim \mathcal{N}(u, \sigma_n)$. Then, if we could have a different f from a fixed u varying the σ_n, the recovery process would be a simple mean operation. Unfortunately, the data given is only a single observed image f affected by an unknown noise η. Therefore, there are many possible combinations to produce f varying u and η (*not injectivity*). The restoration problem is ill-posed because of the lack of injectivity. Furthermore, the selection of the operator A could also lead to injectivity problems in applications such as *deblurring*, *compressed sensing*, *inpainting*, etc.

Sometimes it may occur that the problem is not well conditioned, condition **3)**, because there is no continuous dependency of the solution from the data and small changes in the input produce a great change in the output (instability) [19].

The model in (3.2) can be written in form of an energy minimization problem. This will allow highlighting the relationship with Bayesian modelling. Let $\Omega \subset \mathbb{R}^2$ be the domain of the image and let A be a linear operator. Since we have assumed an AWGN as noise model, we can formulate the restoration problem (3.2) in the sense of least-squares as follows:

$$u^* = \arg\min_{u \in X} E(u) = \arg\min_{u \in X} \int_\Omega (Au - f)^2 dx \tag{3.4}$$

where $E(u)$ denotes the energy that we wish to minimize and the arg min function is defined as

$$\arg\min_{u \in X} E(u) = \{u \in X\,;\, E(u) \leq E(v),\, \forall v \in X\}$$

indicating that $u^* \in X$ is the function where the minimum value of the quadratic energy $E(u)$ is attained:

$$\min_{u \in X} E(u) = E(u^*) \leq E(v), \qquad \forall v \in X$$

Notice that the important property that we shall use in the Bayesian model deduction, relating the arg min and arg max operators:

$$\arg\min_{u \in X} E(u) = \arg\max_{u \in X} (-E(u))$$

This allows converting a probability maximization problem (Bayesian framework) into an energy minimization problem (variational framework).

The operator $A : X \to X$ is a linear bounded operator modelling the image degradation and X is the space of functions where the energy is well defined, i.e., bounded. Assuming $f \in L^2(\Omega)$, we see that the energy in (3.4) is well defined in $X = L^2(\Omega)$ which is the space of square integrable functions in the Lebesgue sense. More regularity is usually assumed in Image Processing and data (images) are considered (essentially) bounded functions, i.e., $X = L^\infty(\Omega)$.

A classical approach to deal with ill-posed problems is to add a *regularization* term in Eq. 3.4. This has the effect to shift the spectrum of the A operator. The idea was introduced in a mathematical framework by Tikhonov and Arsenin in 1977 [268]. This technique transforms the ill-posed or ill-conditioned problems using some constraints in the formulation into problems with a high grade of injectivity and stability. In the case of a restoration problem, noise is usually related to high frequencies so the regularizer should be a function that removes those high frequencies and preserve the structures such as edges. In the image domain, we should constrain the gradient of the solution as a faithful descriptor of the edges of the image.

These considerations can be also recovered using a Bayesian approach which we review briefly following [68]. This statistical framework, in fact, allows deriving proper cost functionals for a specific problem modelling the constraints of our problem as a *prior* in the functional (for further details see [97]). Bayes' rule is a powerful tool to model constrained inverse problems assuming prior knowledge on the nature of the data.

Let u be the unknown true image and let f be the perturbed, observed one. Notice that u and f are functions in Ω, but to make the notation simple we use $u = u(\mathbf{x})$ and $f = f(\mathbf{x})$ where $\mathbf{x} \in \Omega$. Then we can write the joint probability for u and f as

$$\mathcal{P}(u, f) = \mathcal{P}(u|f)\mathcal{P}(f) = \mathcal{P}(f|u)\mathcal{P}(u) \tag{3.5}$$

where $\mathcal{P}(u|f)$ is the posterior probability density of u given the data f, $\mathcal{P}(f|u)$ is the *likelihood* that represents what it is expected in the data f for a given u, $\mathcal{P}(u)$ is the *a-priori* probability density (prior knowledge about the expected solution u) which acts as a regularization or constraint, and $\mathcal{P}(f)$ is the probability density of the data. Rewriting the expression (3.5), we obtain Bayes' formula:

$$\mathcal{P}(u|f) = \frac{\mathcal{P}(f|u)\mathcal{P}(u)}{\mathcal{P}(f)} \tag{3.6}$$

The goal is then to find the image which maximizes this probability (3.6). This is called a Maximum a Posteriori (MAP) estimator. It can be formulated as follows:

$$u^* = \arg\max_{u \in X} \mathcal{P}(u|f) = \arg\max_{u \in X} \mathcal{P}(f|u)\mathcal{P}(u) \tag{3.7}$$

we have eliminated $\mathcal{P}(f)$ from 3.7 because it is a constant w.r.t u and does not play any role when the argmax is computed. Assuming a Gaussian distribution centred at 0 and with variance σ_1 for the likelihood (as dictated by 3.2), we have

$$\mathcal{P}(f|u) \quad \propto \quad \frac{1}{\sigma_1\sqrt{2\pi}} exp\left(-\frac{1}{2\sigma_1^2}\int |Au - f|^2 dx\right) \tag{3.8}$$

The crucial step is the election of the *prior*. A natural choice, typically used in signal theory is

$$\mathcal{P}(u) \quad \propto \quad \frac{1}{\sigma_2\sqrt{2\pi}} exp\left(-\frac{1}{2\sigma_2^2}\int |u|^2 dx\right) \tag{3.9}$$

where we denote, following [227], the standard deviations $\sigma_1 = \sigma_1(\mathbf{x})$ and $\sigma_2 = \sigma_2(\mathbf{x})$. As each sample $u_{i,j} = u(\mathbf{x} = (i,j))$ is assumed to be Independent and Identically Distributed (i.i.d.), the standard deviations $\sigma_1(\mathbf{x}) = \sigma_1$ and $\sigma_2(\mathbf{x}) = \sigma_2$ will be therefore considered as constants. Replacing (3.8) and (3.9) in (3.7), we arrive at

$$u^* = \arg\max_u \left\{ \frac{1}{\sigma_1\sqrt{2\pi}} exp\left(-\frac{1}{2\sigma_1^2}\int |Au-f|^2 dx\right) \frac{1}{\sigma_2\sqrt{2\pi}} exp\left(-\frac{1}{2\sigma_2^2}\int |u|^2 dx\right) \right\} \tag{3.10}$$

We can simplify the expression (3.10) by taking the *log* of the *posteriori* distribution because the *log* function is monotonous increasing, and the maximization problems are not the same, but equivalent because the MAP estimation is the same [68]. Finally, changing the sign to the energy functional we convert the maximization optimization problem into the minimization one:

$$u^* = \arg\min_u \left\{ \ln\left(\frac{1}{\sigma_1\sqrt{2\pi}}\right) + \frac{1}{2\sigma_1^2}\int |Au-f|^2 dx + \ln\left(\frac{1}{\sigma_2\sqrt{2\pi}}\right) + \frac{1}{2\sigma_2^2}\int |u|^2 dx \right\} \tag{3.11}$$

Removing the constants in (3.11) that do not depend on the minimization variable u and introducing $\lambda = \sigma_2^2/\sigma_1^2$ we get

$$u^* = \arg\min_{u\in X} \left\{ \lambda \underbrace{\int_\Omega |Au-f|^2 dx}_{\text{Fidelity: F(u)}} + \underbrace{\int_\Omega |u|^2 dx}_{\text{Prior: } \mathcal{P}(u)} \right\} \tag{3.12}$$

The first integral in (3.12) is usually known as the *fidelity term*

$$F(u) = \int_\Omega |Au-f|^2 dx \doteq \|Au-f\|_2^2 \tag{3.13}$$

It gives us a measure of the distance between the degraded solution Au and the data f. The second integral term is the *prior* $\mathcal{P}(u) = J_2(u)$

$$J_2(u) = \int_\Omega |u|^2 dx \doteq \|u\|_2^2$$

which is called the *Tikhonov regularization term*. The λ parameter is a positive real number that models the trade-off between regularization (when we do not believe in data f) and fidelity (we believe in data f). We then have the energy functional

$$E_2(u) = J_2(u) + \lambda F(u) = \|u\|_2^2 + \lambda\|Au-f\|_2^2$$

where we penalize the energy of the solution $\|u\|_2^2$.

This formulation works well when 1D temporal signals are processed but, as observed in [227], the regularized cost functional obtained with the *prior* (3.9) is not

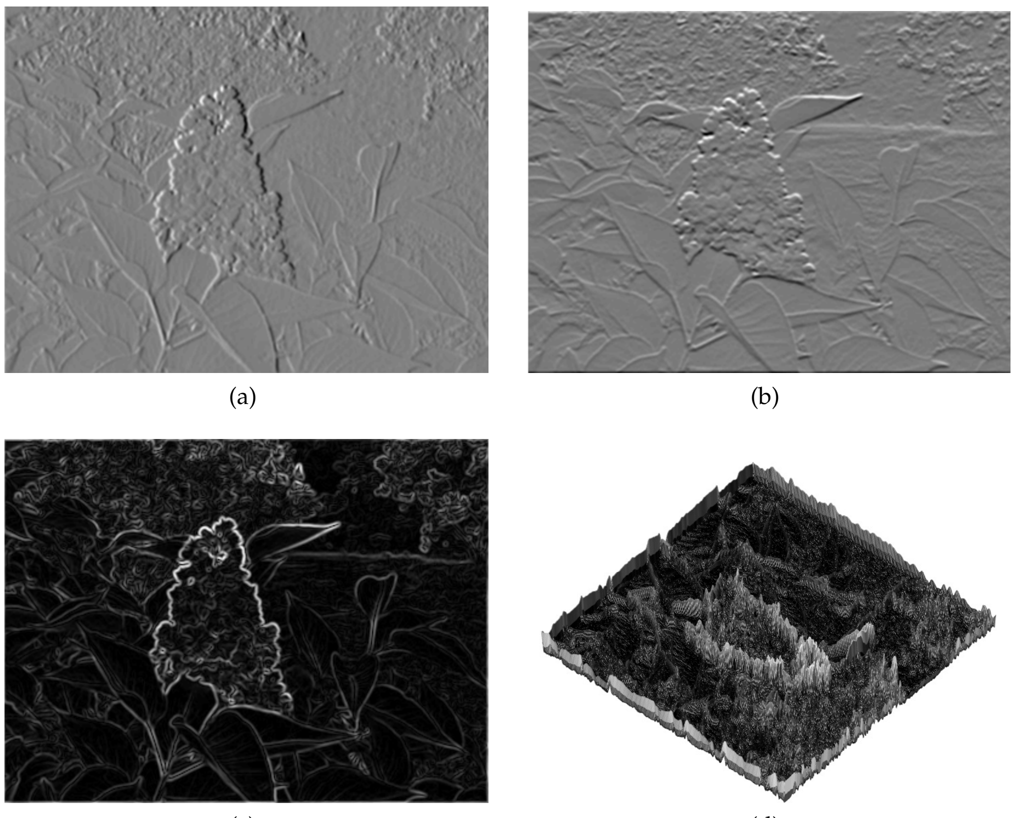

(a) (b) (c) (d)

Figure 3.10: (a) Partial derivative $\partial u/\partial x$ detecting vertical edges (b) Partial derivative $\partial u/\partial y$ detecting horizontal edges (c) The scalar field of the module of the gradient detecting all the fundamental edges (d) 3D representation of (c). The ridges denote the jumps due to the edges.

appropriate for image processing problems. The problem is the assumption made in (3.9) because a zero mean Gaussian distribution implies that the solution with more probability density is $u \equiv 0$. This premise is not the right one because natural images present great variability and are certainly not constant, black images. It is more appropriate to model the prior considering the magnitude of the gradient of the image. In fact, penalization of the energy of the gradient of the image reduces the oscillations in the solution. The formulation is as follows:

$$\mathcal{P}(u) \propto \frac{1}{\sigma_2\sqrt{2\pi}} exp\left(-\frac{1}{2\sigma_2^2}\int |\nabla u|^2 dx\right) \tag{3.14}$$

where

$$\nabla u = \left(\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}\right)$$

is the $2D$ gradient operator, a vector field indicating the direction of the maximal increase of the function. Figure 3.10 shows the partial derivatives of an image. Vertical edges are detected by $\partial u/\partial x$ (Figure 3.10a); correspondingly the horizontal edges are detected by $\partial u/\partial y$ (Figure 3.10b). The full edges detection map is provided by the gradient magnitude $|\nabla u|$ (Figure 3.10c). Low magnitude values of the gradients appear in flat, homogeneous regions of the image. Oscillations are reduced when the energy of the scalar field $|\nabla u|$ is introduced into the energy functional for denoising tasks (Figure 3.10d). The optimization problem with this new *prior* (regularizer) (3.14) reads, in the image domain Ω, as:

$$u^* = \arg\min_u \left\{ \lambda \int_\Omega |Au - f|^2 dx + \int_\Omega |\nabla u|^2 dx \right\} \tag{3.15}$$

We set $K = \nabla$, the gradient operator. The introduction of a general K operator shall be justified later on. This notation is convenient because it will allow writing a unified (local or non-local) formulation of our model problem.

The *prior* $\mathcal{P}(u) = J_2(Ku)$ in (3.14) is called the Tikhonov regularizing term and it is defined as

$$J_2(Ku) = \int_\Omega |\nabla u|^2 dx \doteq \|\nabla u\|_2^2$$

It acts as a regularizer because it penalizes quadratically the energy of $|\nabla u|$, i.e, abrupt, high oscillations of the solution u. We then have the energy functional

$$E_2(u, Ku) = J_2(Ku) + \lambda F(u)$$

Functional analysis and the elliptic regularity theory provide us with a principled framework in which we can understand the effect of each prior. In fact, as in [249], we notice that although the energy functional (3.15) performs much better than (3.12), the domain of this functional is the Sobolev Space [5].

$$W^{1,2}(\Omega) = \{v \in L^2(\Omega) | \nabla v \in [L^2(\Omega)]^n\}, \quad n = 2, 3$$

Using the Morrey embedding theorem [85], we have $W^{1,2}(\Omega) \subset C(\bar{\Omega})$ and this means that the possible solutions of the minimization problem will be continuous functions not allowing jump discontinuities at the edges of the images which are blurred (over-smoothing).

3.2.3 Regularization in Image Processing

Section 3.2.2 has presented the regularization to make the inverse problem solvable by adding some injectivity (Tikhonov), the same concept has been also derived from the Bayesian inference framework. Now, it is time to explore which kind of regularizers are beneficial for the variational formulation in Image Processing. Mathematically, the hint to answer this question lies in the space of functions where the functionals are defined, i.e., which kind of function spaces should be considered for the solution of our problem. Remember that the prior or regularizer indicates without no looking at the image how likely an interpretation is. Considering that a natural image is a piece-wise constant function in a mesh we cannot accept spaces of continuous functions. We need to look for a very non-linear operator to give meaning to a solution in a very weak space.

Rudin, Osher, and Fatemi proposed the very famous ROF model with a new regularizer called the Total Variation (TV) operator in [238]. Since then, the ROF model

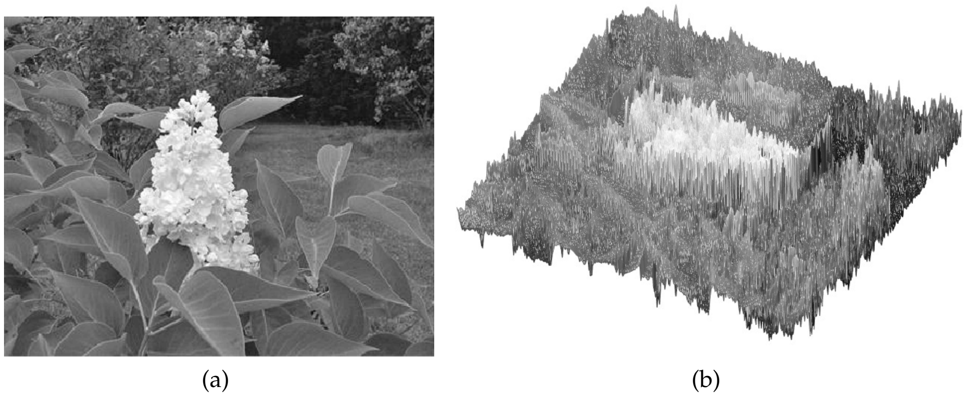


Figure 3.11: (a) Gray-scale image of the public dataset MSRA10K [58] (b) 3D representation of the pixel intensities of (a). The ridges and jumps, characterized by high gradient magnitudes, make the surface clearly discontinuous (piece-wise continuous). The smoothness of the surface corresponds to low gradient magnitudes regions.

has been extensively and successfully applied to many low-level image problems in Computer Vision. In a formal expression, we have

$$u^* = \arg\min_{u\in X}\left\{\lambda\int_\Omega |Au-f|^2dx + \int_\Omega |\nabla u|dx\right\} \tag{3.16}$$

where the abuse of notation in (3.15) lies in the L^1-norm of the gradient. The *prior* $\mathcal{P}(u) = J_1(Ku)$ in (3.16) is

$$J_1(Ku) = \int_\Omega |\nabla u|dx \doteq \|\nabla u\|_1 \tag{3.17}$$

with associated energy functional

$$E_1(u, Ku) = J_1(Ku) + \lambda F(u)$$

This energy is bounded in $W^{1,1}(\Omega)$ which is the space of absolutely continuous functions. We still have over-smoothing which is not admissible because the solution (a natural image) is a clearly discontinuous surface (Figure 3.11). The correct formulation in the case of $p = 1$ uses advanced concepts of measure theory [12] and duality theory [83, 49], and reads (compared with (3.17))

$$u^* = \arg\min_{u\in X}\left\{\lambda\int_\Omega |Au-f|^2dx + \int_\Omega |Du|\right\} \tag{3.18}$$

where Du is the generalized gradient of u. It is also called the *distributional* or *weak* derivative of u, defined by [49]

$$\langle Du, \phi\rangle_{\mathcal{D}'(\Omega,\mathbb{R}^n),\mathcal{D}(\Omega,\mathbb{R}^n)} = -\int_\Omega u(x)\text{div}\phi(x)dx$$

for any vector field $\phi \in \mathcal{D}(\Omega, \mathbb{R}^n)$, i.e., C^∞ with compact support in Ω.

The distributional derivative Du is a vector-valued bounded Radon Measure ,i.e., a Schwartz distribution or generalized function. Distributions generalize the classical notion of functions in mathematical analysis and make it possible to differentiate functions whose derivatives do not exist in the classical sense. The space of bounded Radon measures on Ω is denoted by $\mathcal{M}(\Omega, \mathbb{R}^n)$ and is identified to the dual of $C_0(\Omega, \mathbb{R}^n)$, denoted by $C_0(\Omega, \mathbb{R}^n)'$.

The *prior* is $\mathcal{P}(u) = J(Ku)$, the celebrated Total Variation operator $TV(u)$ which is denoted in literature in many ways

$$J(Ku) = TV(u) = \int_\Omega |Du| \doteq |Du|(\Omega) \tag{3.19}$$

and explicitly defined as

$$TV(u) \equiv \sup_{\phi \in C_c^\infty(\Omega, \mathbb{R}^n)} \left\{ -\int_\Omega u(x) \text{div}\phi(x) dx \; : \; \|\phi\|_\infty \leq 1 \right\} \tag{3.20}$$

The associated energy functional is

$$E(u, Ku) = J(Ku) + \lambda F(u) \tag{3.21}$$

The $TV(u)$ operator is now bounded in a subset of the Lebesgue space $L^1(\Omega)$ called the space of Bounded Radon Measures. This implies that the domain of the functional (3.19) is now

$$X = BV(\Omega) \cap L^2(\Omega) = \{v \in L^1(\Omega) | Dv \in \mathcal{M}(\Omega, \mathbb{R}^n)\} \cap L^2(\Omega)$$

where $BV(\Omega)$ is the space of functions of *Bounded Variation*. Endowed with the norm $||u||_{BV} = ||u||_1 + TV(u)$ it is a Banach space. When $n = 2$ we have, by the Sobolev embedding [12] $BV(\Omega) \to L^{n/(n-1)}(\Omega) = L^2(\Omega)$ that

$$X = BV(\Omega) = \{v \in L^1(\Omega) | TV(v) < +\infty\}.$$

This space contains piece-wise constant functions, so it allows functions (images) with discontinuities. It preserves the structures of the image in form of edges while removing the noise. It is also rotationally invariant. For further details on the properties of the TV, see [50]. Advanced mathematical results about the implementation of the theory of Bounded Variation (BV) functions and related numerical schemes in Image Processing can be found in [170] where the (local) Rician denoising model is considered.

We have then found the correct space in which to work but the choice of the TV operator as a regularizer introduces theoretical and practical difficulties because the energy $E(u, Ku)$ is not differentiable as we further discuss in the next section.

3.2.4 Variational and PDE-Based Models

Computing the maximum a posterior (MAP) estimator is a variational problem [68]. The resolution of the minimization problem (3.18) requires then to find the stationary points of the energy functional which can be minimized using necessary first-order optimality conditions. These conditions are also sufficient when convex functionals are considered. Detailed assumptions and related functional analysis concepts can be found in [19].

By the triangle inequality is possible to show that the $TV(u)$ operator defined in (3.19) is convex, but not strictly. Nevertheless, the quadratic fidelity term $F(u)$ is strictly convex so its sum, which is energy $E(u, Ku)$, is strictly convex.

The existence and uniqueness of the solution to the minimization problem (3.18) can be found in [51]. Central to the variational method is the calculus of the Euler-Lagrange equations of the energy functional through differentiation (see Appendix A.2 for the derivation of the Euler-Lagrange equation of the linear case). When the quadratic prior is considered and in general for all priors of the type:

$$J_p(Ku) = \int_\Omega |\nabla u|^p dx \doteq \|\nabla u\|_p^p, \quad p > 1.$$

the energy functional is differentiable. These operators allow generating a scale-space theory for image restoration in which the regularity of the solution is controlled by the parameter $p > 1$. It determines the functional space the corresponding energy is bounded, which is the Sobolev space $W^{1,p}(\Omega)$. For any $p > 1$, the functional $J_p(Ku)$ is strictly convex and differentiable, and the minimization problem is well-posed. The associated first-order optimality conditions are the Euler-Lagrange equations

$$-\text{div}(|\nabla u|^{p-2}\nabla u) + \lambda A^*(Au - f) = 0, \quad p > 1 \tag{3.22}$$

where A^* is the adjoint operator defined by $\langle Au, v\rangle = \langle u, A^* v\rangle$, $\forall\, u, v \in X$. For $p > 2$, we have a *degenerate* elliptic equation which models the asymptotic state (large time behaviour) of a slow diffusion process. For $1 < p < 2$ we have a *singular* elliptic equation modelling the asymptotic state of a fast diffusion process. In any case, we have an (elliptic) quasilinear PDE. For $p = 2$, we recover the *linear* equation

$$-\Delta u + \lambda A^*(Au - f) = 0 \tag{3.23}$$

Equation (3.23) is still widely used in many low-level image and data processing tasks because it is simple to solve. Nevertheless, when fine image details are of concern the diffusion processes modelled by (3.23) (or (3.22) with $1 < p < 2$) cannot be used because they are aggressive and produce a blurring of the edges, i.e., over-smoothing.

To complete the problem formulation let Ω be the image domain. To solve the elliptic equation (3.22), it is necessary to prescribe feasible boundary conditions. It is common in Image Processing to set the homogeneous Neumann boundary condition (or no flux flow through conditions)

$$|\nabla u|^{p-2}\nabla u \cdot \mathbf{n} = 0 \quad p > 1 \tag{3.24}$$

to preserve the mass of the given data f, when denoising is performed, i.e., $A = I$. Using Green's formulas [105] (for further details see formula (A.3) in Appendix (A.2)), these boundary conditions imply the *mass conservation property*

$$\int_\Omega u dx = \int_\Omega f dx$$

In terms of PDE, the variational method leads to solving the quasilinear elliptic problem

$$P_p(u) = \begin{cases} -\operatorname{div}(|\nabla u|^{p-2}\nabla u) + \lambda A^*(Au - f) = 0, & p > 1 \\ |\nabla u|^{p-2}\nabla u \cdot \mathbf{n} = 0 \end{cases}$$

When the TV operator (3.19) is used as a regularizer and the corresponding energy (3.21) is considered for minimization, the above problems can be avoided (if the right formulation is considered) but a weaker concept of differentiability must be introduced

Definition 3.2.1 (Subgradient) *Let $E : X \to \mathbb{R}$ be a convex proper functional. The subgradient of E at u is defined as:*

$$\partial E(u) := \{u^* \in X^* | E(v) \geq E(u) + < u^*, v - u >, \forall v \in X\}$$

A functional E is said to be subdifferentiable at u if $E(u)$ is finite and the set $\partial E(u)$ is not empty.

A simple example to illustrate the concept of subgradient is the absolute value function $f(z) = |z|$. We set $v = z$ and $u = x$ with $u^* = t$ in (3.2.1). For $x < 0$, we have a unique subgradient $\partial f(x) = \{-1\}$. Similarly $\partial f(x) = \{1\}$ for $x > 0$. At $x = 0$, the subgradient is defined by the inequality $|z| \geq tz$, which is true if and only if $t \in [-1, 1]$. We then have $\partial f(0) = [-1, 1]$: the subdifferential does exist and it is multi-valued.

When $p = 1$, the TV energy functional is not differentiable, but it has a subgradient. Computing the Euler-Lagrange equations leads to the multi-valued problem

$$P_1(u) = \begin{cases} \partial TV(u) + \lambda A^*(Au - f) \ni 0, \\ \dfrac{Du}{|Du|} \cdot \mathbf{n} = 0 \end{cases} \tag{3.25}$$

where the point-wise characterization of the TV operator is [36]

$$-\operatorname{div}\left(\frac{Du}{|Du|}\right) \in \partial TV(u)$$

To simplify, let A be the identity operator, $A = I$. The first-order optimality conditions of the energy minimization problem for image denoising lead to solving the problem

$$P_1(u) = \begin{cases} -\operatorname{div}\left(\dfrac{Du}{|Du|}\right) + \lambda(u - f) = 0, \\ \dfrac{Du}{|Du|} \cdot \mathbf{n} = 0 \end{cases} \tag{3.26}$$

Notice that the gradient vector field ∇u does not appear anymore in the problem formulation being substituted by the distribution Du. It is interesting to have a look at the qualitative properties of the solutions of problem (3.26). This is illustrated by the dynamics of the parabolic problem

$$P_1(u(t)) = \begin{cases} \dfrac{\partial u}{\partial t} = \operatorname{div}\left(\dfrac{Du}{|Du|}\right) - \lambda(u - f), \\ \dfrac{Du}{|Du|} \cdot \mathbf{n} = 0 \\ u(x,0) = f(x) \end{cases} \tag{3.27}$$

which is the gradient descent of the stationary model problem (3.26). The term $-\lambda u$ in (3.27) acts as an absorption term causing a decrease in the solution. This suggests that if we wish to binarize the solution into salient/non-salient regions, it is necessary, or at least convenient, to introduce a reaction term in the equation. The idea is to cause a reaction in the salient zones and absorption in the non-salient, making the solution almost binary.

Problem (3.26) is the *primal problem* associated with the TV operator. It is a formidable, non-linear multi-valued problem. Extending Stampacchia theory [144], it can be shown to be equivalent to a set of variational inequalities that could be solved by using finite elements discretizations.

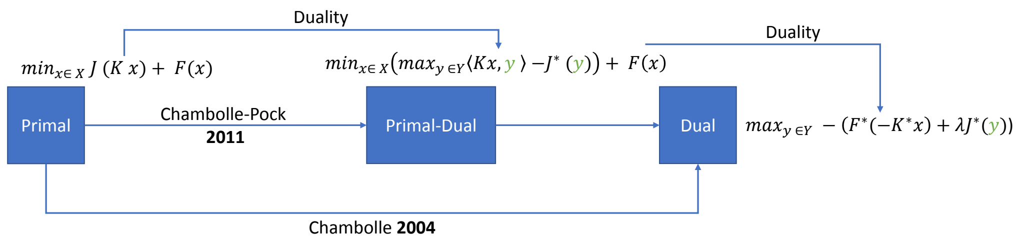
Nevertheless, the problem (3.25) has, in practice, a serious drawback because its formulation must be regularized. In fact, the term $\operatorname{div}(Du/|Du|)$ is numerically highly unstable when $|Du|$ is small or zero and this occurs in the homogeneous regions of the image. To overcome this problem, a regularization is introduced in the form $|\nabla u|_\epsilon = \sqrt{|\nabla u|^2 + \epsilon^2}$. This produces, again, regularized solutions that do not preserve the important features of the image.

An alternative consists of getting back to the energy minimization problem (3.18) and to write it as saddle-point primal-dual problem. To properly understand the primal-dual formulation and its advantages, it is convenient to have some basis of duality theory, which can be found in the book by Rockafellar [233]. Basically, the following considerations hold. Let u be the non-smooth solution of a non-smooth primal minimization problem. Then there exists, under suitable hypothesis, a dual maximization problem for a dual solution p. The maximization problem and the dual solution are both smooth. Both solutions, u and p, satisfy a primal-dual formulation that corresponds to a saddle-point problem with min-max structure.

The basic idea relies on considering the dual form of the non-smooth term (the TV operator) which is written in terms of the dual variable p leaving the fidelity written in terms of the primal variable u as in the primal problem. The solution u of the original primal problem is then recovered from the solution p of the dual problem. This strategy is shown in the next section.

The primal-dual formulation and its local variants are now standard in Image Processing and Computer Vision [64]. The non-local version is more recent in applications. A non-local approach to hyperspectral imagery is proposed in [294]. Variational non-local saliency models in the image domain are presented in [179, 225].

For its resolution, we will follow, as typical in Image Processing, the primal-dual approach proposed by Chambolle and Pock [52] which we adapt to the non-local



Duality
$min_{x\in X}\, J\,(K\,x) + F(x)$
$min_{x\in X}(max_{y\in Y}\langle Kx, y\rangle - J^*(y)) + F(x)$
Duality
Primal
Chamboll-Pock 2011
Primal-Dual
Dual
$max_{y\in Y} - (F^*(-K^*x) + \lambda J^*(y))$
Chambolle 2004

Figure 3.12: There are many algorithms to solve non-smooth convex problems with TV. Some of them consider an exact formulation problem through duality arguments whereas other ones consider approximated regularized formulations [219]. To solve the not differentiability of TV in the primal formulation (3.29), some alternatives have been proposed: primal-dual formulation (3.32) and dual formulation (3.37). The idea is to introduce a new variable (dual y) and it is by construction differentiable so that we can calculate the Euler-Lagrange equations even if x is not differentiable with the new TV (3.20).

calculus framework for the saliency problem (Section 3.3).

3.2.5 Primal-Dual Algorithm

Primal-dual algorithms (PDA) have been introduced in image processing by Zhu and Chan, [292] to overcome the computational challenges which arise when the Primal and Dual ROF are considered. These are non-differentiability at $|\nabla u| = 0$, highly non linearity and spatial stiffness for the primal problem and non-uniqueness due to rank-deficient divergence operator div, extra constraints, and spatial stiffness for the dual problem. Nevertheless, PDA have a long history in Mathematics and their roots can be traced back to the 1950s and development in the 1970s. Often, depending on the problem, they are equivalent or closely related to many other algorithms such operator splitting methods [79, 217, 158], Alternating Direction Method of Multipliers (ADMM) [93, 100], proximal methods [234], or Bregman iterative methods [212, 289].

PDA are descent-type algorithms that alternate between the primal (3.29) and dual (3.37) formulations.

First-order primal-dual algorithms for convex optimization problems with known saddle-point structure are considered in [52] to solve non-smooth convex minimization problems. They allow avoiding any relaxation or regularization of non-differentiable operators such as the TV operator. The general problem presented is in the form of a saddle-point problem for a primal variable u and a dual variable p.

To leave the presentation as simple as possible and to focus directly on the deduction and rationale of the algorithm, we consider a general discretized, finite-dimensional setting where the primal variable is x and the dual variable is y (Figure 3.12). The specific form of the discretized non-local operators is provided in the next section.

Let X, Y be finite-dimensional real vector spaces, with $X = \mathbb{R}^{m\times n}$, the *primal* space, and $Y = \mathbb{R}^{m\times n\times r}$, the *dual* space. In a local approach $r = 2$ and in a non-local discrete graph r is fixed by the number of the directional derivatives between the elements of the graph. Let $K : X \to Y$ be a continuous linear mapping operator with the induced norm:

$$||K|| = max\{||Kx|| : x \in X, \text{ with } ||x|| \leq 1\}$$

Notice that the K operator can be the discretized form of either the local or the non-local gradient operator. In this way, the algorithm can be written in a general setting. For this, we also introduce the K^* operator, which is the adjoint of K:

$$\langle Kx, y \rangle_{X,Y} = \langle x, K^* y \rangle_{X,Y} \tag{3.28}$$

The concept of the adjoint operator is a generalization of the integration by parts formula in multi-variable calculus and functional analysis. The adjoint of a differential operator is another differential operator. When the K operator is the local gradient operator (∇), the K^* adjoint operator is the local divergence operator $-\text{div}$. The Neumann boundary conditions (3.24) ensure that the boundary terms arising in the integration by parts vanish.

We highlight that the adjoint operator acts on the dual variable y through $K^* y$ and the primal variable x is no longer differentiated. This allows considering a regular primal-dual formulation without extra regularization terms. For our analysis, the K operator could be either the local gradient operator (∇) or the non-local gradient operator (∇_w). If $K = \nabla$ then $K^* = -\text{div}$. If $K = \nabla_w$ then $K^* = -\text{div}_w$.

Following [52], we then assume that we have a discretized form of the energy functional to be minimized (compare with the continuous formulation in (3.18) using (3.13) for the fidelity term and (3.19) for the TV operator). We observe that the specific form of the discretized operators K and K^* it is not important because the primal-dual setting encompasses both formulations, the local and the non-local one. Section 3.3 introduces the discretized form of the non-local gradient and divergence operators. We commence looking for the primal variable $x \in X$ which solves the *primal* problem

$$\min_{x \in X} E(x, Kx) = \min_{x \in X} J(Kx) + F(x) \tag{3.29}$$

where the discretized energy terms $J(Kx)$ and $F(x)$ are defined as

$$J(Kx) = |Kx|_1 = \sum_{i,j} |(Kx)_{i,j}|, \qquad F(x) = \frac{\lambda}{2}|x - f|_2^2 = \frac{\lambda}{2} \sum_{i,j} |x_{i,j} - f_{i,j}|^2$$

and f is the corrupted observed image. To apply duality theory, we introduce the Legendre-Fenchel conjugate function:

$$f^*(y) = \sup_{x \in X} \langle y, x \rangle - f(x) \tag{3.30}$$

where x realizes the sup, i.e., the sup is a max if and only if $y \in \partial f(x)$. We also have

$$f(x) = \sup_{y \in Y} \langle x, y \rangle - f^*(y) \tag{3.31}$$

where, as before, the sup is a max if and only if $x \in \partial f^*(y)$. The Legendre-Fenchel identity is

$$y \in \partial f(x) \quad \Longleftrightarrow \quad x \in \partial f^*(y) \quad \Longleftrightarrow \quad f(x) + f^*(y) = \langle x, y \rangle$$

Using (3.31) we have

$$J(Kx) = \max_{y \in Y} \langle Kx, y \rangle - J^*(y)$$

and the *primal-dual*, min-max, saddle-point problem is then:

$$\min_{x \in X} \left(\max_{y \in Y} \langle Kx, y \rangle - J^*(y) \right) + F(x) \tag{3.32}$$

where J^* is the dual functional of J (the TV operator) evaluated at the dual variable $y \in Y$:

$$J^*(y) = \max_{x \in X} \langle y, Kx \rangle - J(Kx)$$

To compute J^*, we define the convex set

$$P = \{ y \in Y \; / \; |y|_\infty \leq 1 \}$$

where $|y|_\infty$ is the discrete maximum norm defined in terms of the euclidean norm

$$|y|_\infty = \max_{i,j} |y_{i,j}|, \quad |y_{i,j}| = |y_{i,j}|_2 = \sqrt{(y^1_{i,j})^2 + (y^2_{i,j})^2} \tag{3.33}$$

We now consider (3.20) which is the continuous (dual) definition of the TV operator

$$TV(u) \doteq \sup_{\phi \in C^\infty_c(\Omega, \mathbb{R}^n)} \left\{ - \int_\Omega u(x) \mathrm{div} \phi(x) dx \; : \; \|\phi\|_\infty \leq 1 \right\}$$

which can be written in discrete form as

$$J(Kx) = TV(x) \doteq \max_{y \in P} \langle Kx, y \rangle \tag{3.34}$$

because using the definition of adjoint and of the set P we have

$$TV(x) \doteq \max_{y \in Y} \{ \langle x, K^* y \rangle \; : \; |y|_\infty \leq 1 \} = \max_{y \in Y} \{ \langle Kx, y \rangle \; : \; |y|_\infty \leq 1 \} = \max_{y \in P} \langle Kx, y \rangle$$

Notice the constraint $y \in P$ in (3.34).

Using the Legendre-Fenchel conjugate function (3.30), we have that $J^*(y) = 0$ when $y \in P$ and $J^*(y) = +\infty$ otherwise. Then J^* is the indicator function I_P of the convex set P

$$J^*(y) = I_P(y) \doteq \begin{cases} 0 & y \in P \\ +\infty & y \notin P \end{cases} \tag{3.35}$$

Using the definition of the adjoint operator (3.28) and substituting into (3.32) the *primal-dual* problem is

$$\min_{x\in X}\left(\max_{y\in Y}\langle x, K^*y\rangle - I_P(y)\right) + \frac{\lambda}{2}|x-f|_2^2 \tag{3.36}$$

The dual variable $y \in Y$ is a solution of the *dual* problem

$$\max_{y\in Y} E^*(y, K^*y) = \max_{y\in Y} -\Big(F^*(-K^*y) + \lambda J^*(y)\Big) \tag{3.37}$$

where $E^*(y, K^*y)$ is the dual functional of $E(x, Kx)$ and F^* is the dual functional of F, the (discretized) fidelity term proposed in (3.13). It can be written in form

$$\max_{y\in Y} E^*(y, K^*y) = \max_{y\in Y} -\left(\frac{1}{2\lambda}|K^*y|_2^2 + \langle f, K^*y\rangle + I_P(y)\right)$$

We can obtain first-order necessary optimality conditions reasoning as follows: notice that by using convexity (and concavity) properties of the energy functionals introduced above these conditions are also sufficient for the existence of a unique min-max $(\hat{x},\hat{y}) \in X\times Y$ solving the primal-dual problem (3.32). We start with the (3.32) problem. Fixed $x = \hat{x}$, dropping the constant term $F(\hat{x})$ we have the maximization problem

$$\max_{y\in Y} E(y) = \max_{y\in Y}\langle K\hat{x}, y\rangle - J^*(y)$$

Stationary points $y \in Y$ are then provided by the equation

$$\partial_y E(y) = K\hat{x} - \partial_y J^*(y) \ni 0$$

where $\partial_y J^*$ is the sub-gradient of the convex function J^*. Getting back to the original (3.32) problem, we fix $y = \hat{y}$ and drop the constant term $J^*(\hat{y})$ to obtain the minimization problem

$$\min_{x\in X} E(x) = \min_{x\in X}\langle Kx, \hat{y}\rangle + F(x)$$

Stationary points $x \in X$ are then provided by the equation

$$\partial_x E(x) = K^*\hat{y} + \partial_x F(\hat{x}) \ni 0$$

where $\partial_x F$ is the sub-gradient of the convex function F. The optimality conditions for the pair $(\hat{x},\hat{y}) \in X\times Y$ are then

$$\big(\partial_y E(y), \partial_x E(x)\big) \ni (0,0)$$

which are the (system) equations (see also the book of Rockafellar [233] for a more rigorous deduction)

$$K\hat{x} \in \partial J^*(\hat{y}), \qquad -K^*\hat{y} \in \partial F(\hat{x}) \tag{3.38}$$

We now argue as follows. From the first equation in (3.38), we deduce by multiplying by τ_d and adding $\hat{y}$ at both terms

$$\hat{y} + \tau_d K\hat{x} \in \hat{y} + \tau_d\partial J^*(\hat{y}) = (I + \tau_d\partial J^*)\,(\hat{y})$$

and the solution $\hat{y}$ is formally given by the implicit equation

$$\hat{y} \in (I + \tau_d \partial J^*)^{-1} (\hat{y} + \tau_d K \hat{x})$$

where the operator $(I + \tau_d \partial J^*)^{-1}$ is called the *resolvent* operator and is generally defined for a convex function G as

$$x = (I + \tau \partial G)^{-1}(y) \doteq \arg\min_x \left(\frac{1}{2\tau} |x - y|_2^2 + G(x) \right) \tag{3.39}$$

Considering the second equation $-K^* \hat{y} \in \partial F(\hat{x})$, we deduce by multiplying by τ_p and adding $\hat{x}$ at both terms

$$\hat{x} - \tau_p K^* \hat{y} \in \hat{x} + \tau_p \partial F(\hat{x}) = \left(I + \tau_p \partial F\right)(\hat{x})$$

and the solution $\hat{x}$ is formally given by the implicit equation

$$\hat{x} \in \left(I + \tau_p \partial F\right)^{-1} (\hat{x} - \tau_p K^* \hat{y})$$

As a result, we have to compute

$$\begin{cases} \hat{y} \in (I + \tau_d \partial J^*)^{-1} (\hat{y} + \tau_d K \hat{x}) \\ \hat{x} \in \left(I + \tau_p \partial F\right)^{-1} (\hat{x} - \tau_p K^* \hat{y}) \end{cases} \tag{3.40}$$

System (3.40) is coupled but it can be iteratively decoupled through the alternate numerical scheme

$$\begin{cases} y^{n+1} = (I + \tau_d \partial J^*)^{-1} (y^n + \tau_d K x^n) \\ x^{n+1} = \left(I + \tau_p \partial F\right)^{-1} \left(x^n - \tau_p K^* y^{n+1}\right) \end{cases} \tag{3.41}$$

This scheme has been deduced in Chambolle and Pock and it can be traced back to the Arrow-Hurwicz method [16]. The key point is the efficient computation of the resolvent operators which define the iterations in (3.41):

$$(I + \tau_d \partial J^*)^{-1}, \qquad \left(I + \tau_p \partial F\right)^{-1}$$

Since J^* is the indicator function of a convex set, see (3.35), the resolvent operator reduces to point-wise Euclidean projectors onto l^2 balls:

$$p = (I + \tau_d \partial J^*)^{-1} (\tilde{p}) \quad \Longleftrightarrow \quad p_{i,j} = \frac{\tilde{p}_{i,j}}{\max(1, |\tilde{p}_{i,j}|_2)}$$

where $|\tilde{p}_{i,j}|_2$ is the Euclidean norm defined in (3.33). Setting $\tilde{p} = y^n + \tau_d K x^n \in Y$ in (3.41) we get the explicit iteration

$$y_{i,j}^{n+1} = \frac{y_{i,j}^n + \tau_d (K x^n)_{i,j}}{\max(1, |y_{i,j}^n + \tau_d (K x^n)_{i,j}|_2)} \tag{3.42}$$

The resolvent operator w.r.t F poses simple pointwise quadratic problems. The solution is

$$u = \left(I + \tau_p \partial F\right)^{-1} (\tilde{u}) \quad \Longleftrightarrow \quad u_{i,j} = \frac{\tilde{u}_{i,j} + \tau_p \lambda f_{i,j}}{1 + \tau_p \lambda} \tag{3.43}$$

In fact, by using the definition of the resolvent operator (3.39) we must solve the simple quadratic problem

$$u = \left(I + \tau_p \partial F\right)^{-1}(\tilde{u}) = \arg\min_u \left(\frac{1}{2\tau_p} |u - \tilde{u}|_2^2 + F(u) \right) =$$
$$= \arg\min_u \left(\frac{1}{2\tau_p} |u - \tilde{u}|_2^2 + \frac{\lambda}{2} |u - f|_2^2 \right)$$

Setting

$$E(u) = \frac{1}{2\tau_p} |u - \tilde{u}|_2^2 + \frac{\lambda}{2} |u - f|_2^2$$

we can calculate its differential

$$\partial E(u) = \frac{1}{\tau_p}(u - \tilde{u}) + \lambda(u - f)$$

and imposing $\partial E(u) = 0$ we get the result (3.43).

Setting $\tilde{u} = x^n - \tau_p K^* y^{n+1} \in Y$ in (3.41), we get the explicit iteration

$$u_{i,j} = \frac{\tilde{u}_{i,j} + \tau_p \lambda f_{i,j}}{1 + \tau_p \lambda} = \frac{x^n_{i,j} - \tau_p K^* y^{n+1}_{i,j} + \tau_p \lambda f_{i,j}}{1 + \tau_p \lambda} \tag{3.44}$$

Considering (3.42) and (3.44) we set $u_{i,j} = x^{n+1}_{i,j}$ and the explicit alternate iterative algorithm to solve system (3.41) is

$$\begin{cases} y^{n+1}_{i,j} = \dfrac{y^n_{i,j} + \tau_d (Kx^n)_{i,j}}{\max(1, |y^n_{i,j} + \tau_d (Kx^n)_{i,j}|_2)} \\ x^{n+1}_{i,j} = \dfrac{x^n_{i,j} - \tau_p K^* y^{n+1}_{i,j} + \tau_p \lambda f_{i,j}}{1 + \tau_p \lambda} \end{cases} \tag{3.45}$$

In the next section, we present the non-local operators K and K^*, which we shall use in the variational model for the automatic saliency detection problem. The general pipeline is as follows.

Given a natural image f, we shall compute a partition of the image in clusters of pixels called superpixels. These clusters are computed using a metric based on some features such as colour and position. This generates a finite-dimensional manifold of characteristics which can be modelled as a weighted graph defined by superpixels (vertexes) and weights for their connections (edges). The data takes (averaged) values in these superpixels and are vector-valued functions causing a dimensional reduction of the problem. Connections are further reduced using a k-nearest neighbours strategy (k-NN) and the non-local operators can be calculated. Notice that the superpixel value of an image refers to feature values not to, only, colour intensity.

This allows obtaining the saliency value of each superpixel at each step of the iterative algorithm. To select the most salient superpixels of the initial partition that define the final binary saliency map, an *hard-thresholding* step is performed. Notice that promoting a binarization of the optimal solution into salient/non-salient superpixels naturally facilitates the choice of the optimal threshold.

A new general variational model is then proposed in Section 3.3 along the lines before presented. We explicitly model saliency detection with a concave quadratic term that promotes binary classification (salient or no salient regions).

3.2.6 Non-local Variational Methods and Graphs

The models and methods described in the previous section can be formulated in a local or non-local framework depending on the notion of derivative we consider. Although the local models and methods perform very well in many low-level image processing problems, especially when the TV operator is used as an edge preserving regularizer, there are some deficiencies in local derivatives when the images possess textures or repetitive structures.

To capture the properties of texture and repetitive structures, non-local methods have appeared in the literature as Yaroslavsky filter [149] or Bilateral filters [269] for denoising problems. In non-local methods, any pixel is related to any pixel in the image. However, due to the computational complexity inherent in the evaluation of all these relationships, a dimensional reduction is typically performed.

Since the success of the seminal paper in denoising with the non-local means in 2005 by Buades et al. [43], there has been an effort to formalize a framework from the variational perspective using the calculus of variations in the non-local setting. We follow the framework proposed in [98] to define the gradient and the divergence operators in the continuous domain.

Let $\Omega \in \mathbb{R}^n$, $x \in \Omega$, $u(x)$ a real function $u : \Omega \to \mathbb{R}$. The non-local derivative for the element x in the direction of y reads:

$$\partial_y u(x) \doteq (u(y) - u(x))\sqrt{w(x,y)}, \quad y, x \in \Omega \tag{3.46}$$

where $w(x,y)$ is a positive and bounded measure of the similarity of two elements in the domain Ω. The non-local gradient $K = \nabla_w$ is a matrix composed of all the partial derivatives (3.46) in the domain

$$\nabla_w u(x) = (\partial_y u(x))_{y\in\Omega} \tag{3.47}$$

Collecting all the contributions in (3.47), we have that

$$\nabla_\omega \mathbf{u} = (\nabla_w u(x))_{x\in\Omega}$$

is a matrix. The divergence operator

$$(div_w \mathbf{v}) : \Omega \times \Omega \to \Omega$$

of a vector field $\mathbf{v} = v(x,y) \in \Omega \times \Omega$ is the continuous adjoint of the non-local gradient operator. We set $K^* = -\text{div}_w$ and in the continuous non-local framework the divergence operator is defined as in [98]

$$(div_w \mathbf{v}) \doteq \int_\Omega (v(x,y) - v(y,x))\sqrt{w(x,y)}dy \tag{3.48}$$

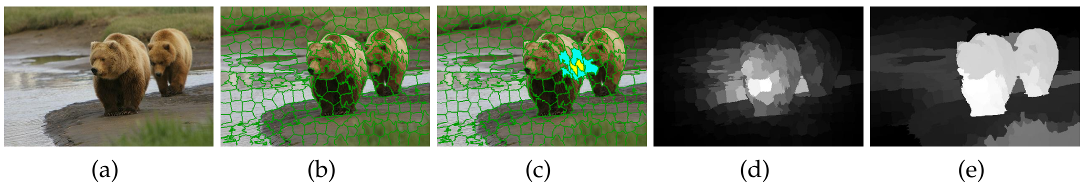


Figure 3.13: The workflow of our proposal consists of: (a) input image from iCoseg dataset [22] (b) over-segmentation by superpixels (SLIC [3]) (c) weights to connect each superpixel among them. A superpixel (yellow) is shown and its k-NN in cyan (d) control map created from the superpixels through colour contrast and location priors (e) variational method and saliency map.

A similar framework in the discrete setting for graphs was formalized in [84]. Given an image and a region partition, an undirected symmetric and weighted graph G can be considered in the space of the regions. The weighted graph $G = (V, E, w)$ consists of a finite set of regions V, the edges E linking regions, and their associated weights w_{pq}, $pq \in E$. These are the foundations of our research which will be further explained (Section 3.3).

3.3 Saliency Variational Model on Graphs

In this section, we describe the core of the algorithm for extracting the salient part in natural images. Chapter 4 presents the implementation and results. Figure 3.13 summarizes the pipeline of our method and presents two conceptual stages: 1) **initialization stage** with the superpixels extraction (Figure 3.13b), calculation of the weights (Figure 3.13c), and generation of the control map which models the likelihood of being salient (Figure 3.13d), and 2) **iterative stage**, with the variational solver for the saliency segmentation (Figure 3.13e). First, we shall present the model to better understand the preliminary calculations.

3.3.1 A General Variational Model for Saliency

A general variational model for saliency segmentation can be formulated as the following energy minimization problem: Given a function f (the data), compute a solution u in the image domain such that it minimizes the energy

$$E(u) = J(u) + \lambda F(u) - H(u) \tag{3.49}$$

where $J(u)$ is the regularizer term (*a-priori* information), $F(u)$ is the fidelity term and $H(u)$ is a saliency term which promotes the binarization of the solution into salient (foreground) and not salient (background) regions. The λ parameter is a positive constant which allows balancing the relative importance of the terms in the energy functional (Section 3.3.4).

This model can be formulated pixel-wise, in the image domain, or in a graph of featured superpixels, the region domain. We have chosen the last approach to reduce the computational time of the whole process yet preserving the fundamental information of the image, i.e., edges, through the superpixel partition.

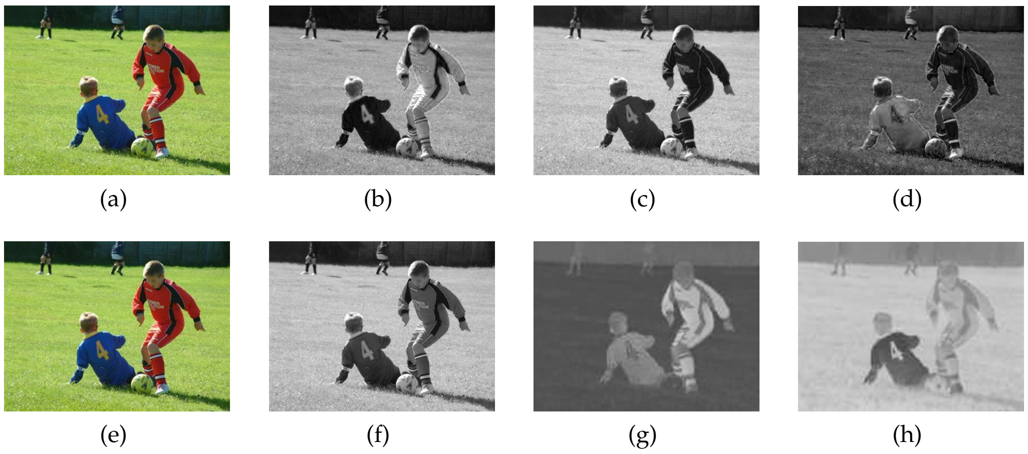


Figure 3.14: (a) and (e) original colour images from MSRA10K dataset [58] (b) red channel from (a) (c) green channel from (a) (d) blue channel from (a) (b) bright channel in CIE $L*a*b$ from (e) (b) green-magenta channel in CIE $L*a*b$ from (e) and (b) yellow-blue channel in CIE $L*a*b$ from (e).

3.3.2 Over-segmentation: Superpixels

The input colour image f is first transformed from RGB into the CIE $L*a*b$ colour space, which is perceptually more interesting for the independence of colour and intensity [58] (Figure 3.14). Then the image is partitioned into regions (*superpixels*). A superpixel is a cluster of pixels connected by some metric. Each pixel is assigned only to one superpixel so that they do not overlap each other. In our approach, the initial partition is created by using the SLIC method (*Simple Linear Iterative Clustering*) [3] (Figure 3.13b) which has been proven to be accurate, computationally efficient, and robust. Other algorithms could have been used, such as Superpixels with Contour Adherence using Linear Path (SCALP) [99], Superpixel Sampling Networks (SSN) [135], etc.

3.3.3 Weights: Adjacency Matrix W

Once the image has been partitioned, the spatial structure in the pixel domain is lost. A finite-dimensional graph can be constructed using the initial data f in the image domain. Let $\mathbf{f}_p$ for $p \in V$ be the vector value in the feature space at a superpixel p which we identify with a vertex of a graph (undirected, symmetric and weighted) $G = (V, E, w)$ where V is the set of vertexes, E is the set of edges, and $w : E \to [0, 1]$ is a weight function. Let $\mathbf{f} = (\mathbf{f}_p)_{p \in V}$ be the matrix-valued collection of the vector values $\mathbf{f}_p$. Norms in the graph are defined as the L^2 analogous

$$|\mathbf{f}|_2^2 = \left(\sum_{p \in V} |\mathbf{f}_p|^2 \right)^{1/2}$$

We briefly present the structure of the graph. Let $\mathcal{H}(V)$ be the Hilbert space of real-valued functions on the vertices of the graph, $\mathbf{f} : V \to \mathbb{R}^n$ and let $\mathcal{H}(E)$ be the Hilbert space of real-valued functions on the edges of the graph, $\mathbf{F} : E \to \mathbb{R}^n$. These spaces are endowed with the scalar products

$$\langle \mathbf{f}, \mathbf{g} \rangle_{\mathcal{H}(V)} = \sum_{p \in V} \mathbf{f}_p \cdot \mathbf{g}_p, \qquad \langle \mathbf{F}, \mathbf{G} \rangle_{\mathcal{H}(E)} = \sum_{p \in V} \sum_{q \in V} \mathbf{F}_{pq} \cdot \mathbf{G}_{pq}$$

The connections $pq \in E$ between superpixels are defined in the feature space by the weight function following [281]

$$\omega_{pq} = \exp\left(-\frac{|\mathbf{f}_p - \mathbf{f}_q|_2^2}{2\sigma^2}\right), \quad \forall\, pq \in E \tag{3.50}$$

where $\mathbf{f}_p$ is a feature normalized vector at superpixel p defined by $\mathbf{f}_p = (r\mathbf{c}_p, \mathbf{l}_p)$, being $\mathbf{c}_p \in \mathbb{R}^3$ the mean of the superpixels for each component in the in CIE $L{*}a{*}b$ colour space and $\mathbf{l}_p \in \mathbb{R}^2$ is the centroid position of the superpixel in the original spatial space. The r is a positive parameter and controls the balance between the two features while σ^2 defines the extent of the non-locality. We experimentally fixed $r = 0.9$ and $\sigma^2 = 0.05$. Superpixels p and q are connected, say $p \sim q$ if and only if $w_{pq} > 0$. All the superpixels are initially connected. The graph G is represented by its adjacency matrix $W^{sp\times sp} = (\omega_{pq})_{pq} \in E$, where sp is the number of superpixels. To reduce computational cost and to exploit local relationships in the feature space, the number of total connections (edges of the graph) of each superpixel is decreased from sp to k-NN (Figure 3.13c). The rest of the weights for a superpixel are set to zero. As a result of this process, we have a sparse weight matrix $W^{sp\times sp}$ which is no longer symmetric.

3.3.4 Regularizer, Fidelity, and Saliency Terms

We fix the notation that is slightly adapted from [84]. Let $\mathbf{u} = (u_p)_{p\in V}$ be a function on the set of vertices V in G representing the solution in the feature domain and let $\partial_q u_p$ be the *weighted partial difference* at a vertex p in the direction of vertex q:

$$\partial_q u_p = \sqrt{w_{pq}}(u_p - u_q) \tag{3.51}$$

where u_p is the value at superpixel p and the weight w_{pq} is calculated by (Eq. 3.50). (3.51) is analogous to the non-local continuous partial derivatives defined in (3.46). These partial differences define the *difference operator* $\mathcal{G}_w : \mathcal{H}(V) \to \mathcal{H}(E)$ for any function $\mathbf{u} \in \mathcal{H}(V)$, with $\mathbf{u} : V \to \mathbb{R}$

$$(\mathcal{G}_w\mathbf{u})(p,q) = \sqrt{w_{pq}}(u_p - u_q) \in \mathcal{H}(E)$$

We set $K = \mathcal{G}_w$ in the primal-dual algorithm (3.45) with $K\mathbf{u} = \mathcal{G}_w\mathbf{u}$ for any $\mathbf{u} \in \mathcal{H}(V)$.

The *weighted gradient operator* is then the vector of all partial differences at superpixel p:

$$\nabla_w u_p = (\partial_q u_p)_{q\in V} \tag{3.52}$$

Collecting all the contributions we have analogously to (3.47) that

$$\nabla_\omega \mathbf{u} = (\nabla_w u_p)_{p\in V} = ((\partial_q u_p)_{q\in V})_{p\in V} \in \mathcal{H}(E)$$

is a matrix defined in the set of edges of the graph. In order to compute the Euler-Lagrange equation of the energy functional in (3.49), we need to define the non-local divergence of a matrix $\mathbf{d} = (d_{pq})_{p,q\in V}$. The continuous formula for a general vectorial field is (3.48).

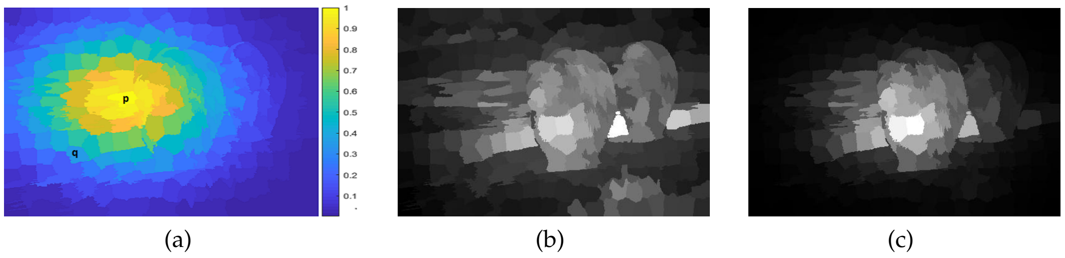


Figure 3.15: (a) Contrast prior (v_p^{con}) is calculated by the difference in colour between superpixel p and q in $\mathbf{f}$ and weighted ($\hat{w}_{pq}$) by the distance of centroids between superpixels p and q (b) control map without object prior projected on the image domain (c) control map with object prior $\mathbf{v}^c = (v_p^c)_{p \in V}$ projected on the image domain.

For any $\mathbf{d} \in \mathcal{H}(E)$ the *adjoint operator* $\mathcal{G}_w^* : \mathcal{H}(E) \to \mathcal{H}(V)$ of the difference operator $\mathcal{G}_w$ can be defined at superpixel $p \in V$ as (see [84])

$$-div_w \mathbf{d}_p = (\mathcal{G}_w^* \mathbf{d})(p) = \sum_{q \in V, qp \in E} \sqrt{w_{pq}} (d_{qp} - d_{pq}) \tag{3.53}$$

where the (dual) functions $d_{pq} \in \mathcal{H}(E)$ are edges functions. We set $K^* = \mathcal{G}_w^*$ in the primal-dual algorithm (3.45) with $K^* \mathbf{d} = \mathcal{G}_w^* \mathbf{d} = -div_w \mathbf{d}$ for any $\mathbf{d} \in \mathcal{H}(E)$. We can now describe the energy functional in the graph (features domain).

Our regularizer $J(\mathbf{u})$ on graphs is defined by the non-local TV operator, denoted $J_{NLTV,w}(\mathbf{u})$, which preserves edges and induces the sparsity of the gradients of saliency maps. The continuous form of this operator for a function (image) u has been introduced in [98] and reads

$$J_{NLTV}(u) = \left(\int_\Omega w(x,y) |u(y) - u(x)|^2 dy \right)^{1/2} dx \tag{3.54}$$

The discrete version of (3.54) can be found in [84] and it is defined as the isotropic l^1 norm of the weighted graph gradient

$$J_{NLTV,w}(\mathbf{u}) = \sum_{p \in V} \left(\sum_{q \in V, pq \in E} w_{pq} |u_q - u_p|^2 \right)^{1/2} \tag{3.55}$$

The fidelity term is defined as in [281]. Using the data f in the image domain, we compute the *control map* ($\mathbf{v}^c$) in the graph generated by the feature domain, where $\mathbf{v}^c = (v_p^c)_{p \in V}$ (Figure 3.13d). Each component is composed of a **contrast prior** (Figure 3.15a)

$$v_p^{con} = \sum_{q \neq p} \hat{w}_{pq} |\mathbf{c}_p - \mathbf{c}_q|_2^2 \tag{3.56}$$

with weights

$$\hat{w}_{pq} = e^{-\frac{|\mathbf{l}_p - \mathbf{l}_q|_2^2}{2\sigma^2}}$$

and an **object prior**

$$v_p^{obj} = e^{-\frac{|\mathbf{l}_p - \bar{\mathbf{l}}|_2^2}{2\sigma^2}}, \tag{3.57}$$

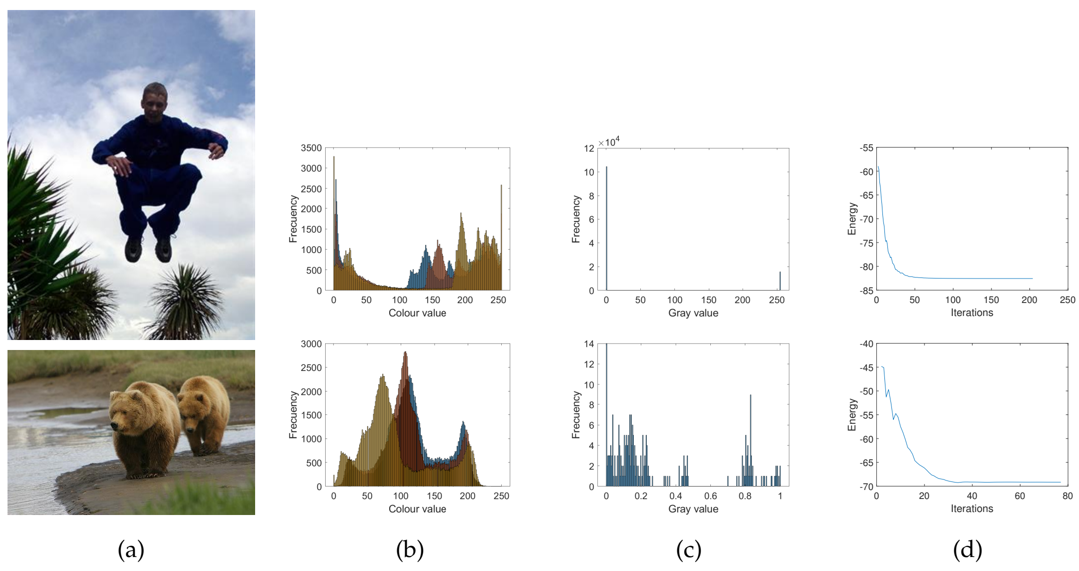


Figure 3.16: We show the effect of the saliency term $H(u)$ in the energy functional. (a) Original images from MSRA10K dataset [58]. In (b) the initial, distributed histogram *per* channel of a given input image (a). (c) The histogram of the binary output (saliency map) of the algorithm in the features domain. (d) The typical, convergent, L-shaped curve of our iterative algorithm during iterations ($sp = 300$ and $\epsilon = 10^{-5}$).

where $\bar{\mathbf{I}}$ are the coordinates of the centre of the image. We can encode these priors in the saliency control map at superpixel p as $v_p^c = v_p^{con} v_p^{obj}$ with $\mathbf{v}^c = (v_p^c)_{p \in V}$ (Figures 3.15b and 3.15c). The fidelity term is

$$F(\mathbf{u}) = \frac{1}{2\alpha}|\mathbf{u} - \mathbf{v}^c|_2^2 = \frac{1}{2\alpha}\sum_{p \in V} |u_p - v_p^c|^2 \tag{3.58}$$

where the positive parameter α models the relative importance of the likelihood and the saliency term.

The saliency term is defined by a concave quadratic energy function -$H(\mathbf{u})$, with:

$$H(\mathbf{u}) = \frac{1}{2\alpha^2}\sum_{p \in V} |1 - \delta u_p|^2, \tag{3.59}$$

and where $\delta > 0$ acts as a threshold in the region domain separating background and saliency (Figures 3.16b and 3.16c). To keep the solution in the (saliency) range $[0, 1]$, we perform a hard truncation at each iteration of the algorithm. It models the probability of each superpixel to be salient. As a result, the value u_p represents a score function estimating the saliency of each superpixel of the graph associated with data f.

3.3.5 Numerical Resolution: Primal-Dual Algorithm

The proposed saliency model generalizes previous attempts [281] where the variational saliency is formulated as a pure denoising problem in the features space. In the asymptotic limit $\alpha \to \infty$, we recover that scenario. In fact, the fidelity term can be retained choosing $\lambda = \alpha\hat{\lambda}$, with $\hat{\lambda}$ real and positive while the saliency term vanishes in the limit. Experimental results can be found in [10]. Replacing the terms in Eq. 3.49 by their analogous Equations 3.55, 3.58, and 3.59, our proposal is the minimization of the following energy:

$$E(\mathbf{u}) = J_{NLTV,w}(\mathbf{u}) + \lambda F(\mathbf{u}) - H(\mathbf{u}) \tag{3.60}$$

We normalize the range of the solution $\mathbf{u}$ in the superpixel domain and then, we project back onto the image domain to obtain the final output u (Figure 3.13e).

We have chosen the primal-dual algorithm because, as we have discussed before, it allows us to solve the primal-dual continuous formulation (3.36) without any kind of regularization. The regularization is inevitably faced when the primal formulation (3.29) is considered and over-smoothing is caused. The resulting minimization problem (Eq. 3.60) is solved by a primal-dual algorithm.

This algorithm encompasses an alternate *maximization* (update dual variable $\mathbf{d}$) and *minimization* (update primal variable $\mathbf{u}$) steps [52, 169]. Both steps are repeated until the energy convergence is reached. As a stopping criterion, we compute the difference between consecutive values of the energy functional (Eq. 3.60) during iterations until this is less than a fixed tolerance ϵ (Figure 3.16d).

In summary, given the k-step solution in terms of the primal and dual variables in the graph $(\mathbf{u}^k, \mathbf{d}^k)$ which correspond to the discrete variables x and y introduced in (3.45), the update is:

– Maximization step: Fixed an ascent dual discretization time τ_d, we compute for every superpixel q

$$\mathbf{d}_q^{k+1} = \frac{\mathbf{d}_q^k + \tau_d \nabla_w u_q^k}{max(1, |\mathbf{d}_q^k + \tau_d \nabla_w u_q^k|_2)} \tag{3.61}$$

and set $\mathbf{d}^{k+1} = (\mathbf{d}_q^{k+1})_{q\in V}$. The non-local gradient $\nabla_w u_q^k$ is calculated in (3.52).

– Minimization step: Fixed a descent primal discretization time τ_p and given $\mathbf{d}^{k+1}$, we compute for every superpixel q

$$u_q^{k+1} = (1 + a\tau_p)u_q^k + \tau_p \left(div_w(\mathbf{d}_q^{k+1}) - b_q \right) \tag{3.62}$$

and set $\mathbf{u}^{k+1} = (u_q^{k+1})_{q\in V}$ where the non-local divergence $div_w(\mathbf{d}_q^{k+1})$ is calculated in (3.53) and

$$a = \frac{\delta^2}{\alpha^2} - \frac{\lambda}{\alpha} \qquad \mathbf{b} = \frac{\delta}{\alpha^2} - \frac{\lambda}{\alpha}\mathbf{v}^c. \tag{3.63}$$

The parameter a and the vector $\mathbf{b}$ defined in (3.63) have been introduced to facilitate the physical interpretation of the model. In fact, the term $a\tau_p u_q^k$ in (3.62) acts as a reaction term (source) when $a > 0$ and an absorption term (sink) when $a < 0$ (as in the ROF model). The forcing term $\mathbf{b} = (b_p)_{p\in V}$ is a changing sign vector driven by the control map $\mathbf{v}^c$ which favours saliency detection. The specific form of (3.63) has been deduced as follows.

The non-local discretized form of the energy functional in (3.60) is

$$\sum_{p\in V}\left(\sum_{q\in V,\,pq\in E} w_{pq}|u_q-u_p|^2\right)^{1/2}+\frac{\lambda}{2\alpha}\sum_{p\in V}|u_p-v^c_p|^2-\frac{1}{2\alpha^2}\sum_{p\in V}|1-\delta u_p|^2 \tag{3.64}$$

The difference of the quadratic functions in (3.64) suggests computing explicitly this difference to obtain a unique quadratic form that encapsulates the physics we introduced into the model. We have

$$\begin{aligned}
&\frac{\lambda}{2\alpha}\sum_{p\in V}|u_p-v^c_p|^2-\frac{1}{2\alpha^2}\sum_{p\in V}|1-\delta u_p|^2=\\
&=\frac{1}{2}\sum_{p\in V}\left[\frac{\lambda}{\alpha}(u_p^2-2u_pv^c_p+{v^c_p}^2)-\frac{1}{\alpha^2}(1-2\delta u_p+\delta^2u_p^2)\right]\\
&=\frac{1}{2}\sum_{p\in V}\left[\left(\frac{\lambda}{\alpha}-\frac{\delta^2}{\alpha^2}\right){u_p}^2+2\left(\frac{\delta}{\alpha^2}-\frac{\lambda}{\alpha}v^c_p\right)u_p+\left(\frac{\lambda}{\alpha}{v^c_p}^2-\frac{1}{\alpha^2}\right)\right]\\
&=\sum_{p\in V}\left(-\frac{1}{2}a{u_p}^2+b_pu_p+c\right)
\end{aligned}$$

The parametric values of λ, α, and δ modify the general behaviour of the model and generate different scenarios to be explored. In this work, we focus on the case $a<0$ where the energy functional is strictly convex and the minimization problem is well-posed.

The gradient descent computing in (3.27) can be justified as follows. We write the Euler-Lagrange equation (3.26) in terms of the non-local dual variable $\mathbf{d}$ with $a=-\lambda<0$ and $b=-\lambda f$ to obtain

$$K^*_w(\mathbf{d}^{k+1}_q)=au^k_q-b_q$$

where $K^*_w(\mathbf{d}^{k+1}_q)$ models the non-local interaction genarating by diffusion. Using forward Euler discretization with step τ_p the analogous gradient descent is

$$\frac{u^{k+1}_q-u^k_q}{\tau_p}=-K^*_w(\mathbf{d}^{k+1}_q)+au^k_q-b_q$$

Multiplying by τ_p, reordering terms, and setting $K_w=-div_w$, we get

$$u^{k+1}_q=(1+a\tau_p)u^k_q+\tau_p\left(div_w(\mathbf{d}^{k+1}_q)-b_q\right)$$

which is the minimization step in (3.45). Notice that the iterative splitting between the primal and dual variable generate a strong data dependency that penalizes the full potential of parallel computation. Algorithm 1 shows the pseudocode of our primal-dual algorithm. We have experimentally fixed $\tau_p=0.3$ and $\tau_d=0.33$ as in [292].

Algorithm 1 Saliency estimation on non-local TV with Saliency Term

```
1: procedure NLTVSalTerm(inputImage,$sp$,$k$,$\lambda$,$\delta$,$\alpha$,$\epsilon$)
2:     Calculate Superpixels over inputImage
3:     Extract superpixel features $\mathbf{f}_p$ for $p \in V$
4:     $W^{sp\times sp} \leftarrow$ Create adjacency matrix using (Eq. 3.50)
5:     Connections in $W^{sp\times sp}$ to $k$-NN
6:     $\mathbf{v}^c \leftarrow$ Calculate controlMap using (Equations 3.56 and 3.57)
7:     Calculate $a$ and $\mathbf{b}$ using (Eq. 3.63)
8:     Initialization: $\mathbf{u} = \mathbf{v}^c$, $\mathbf{d} = 0$
9:     repeat
10:        Compute $\mathbf{d}^{k+1}$ using (Eq. 3.61)
11:        Compute $\mathbf{u}^{k+1}$ using (Eq. 3.62)
12:        Compute $E(\mathbf{u}^{k+1})$ using (Eq. 3.60)
13:    until $|E(\mathbf{u}^k) - E(\mathbf{u}^{k+1})| \leq \epsilon$
14:    return $\mathbf{u}$
15: end procedure
```

3.4 Patch-Based Methods for MRI-CT Synthesis

In this section, we describe the core of the algorithm for generating the pseudo-CT image from MRI images. Chapter 5 presents the implementation and results. CT acquisition modality provides the electron densities necessary to obtain an attenuation map for PET images (see Section 2.3.4 and Eq. 2.3). The synthesis of a pseudo-CT image from MRI images is a natural solution to correct PET when PET/MR multi-modality is considered.

The basic idea is as follows: **1)** generate a pseudo-CT given an input MRI image, **2)** calculate the attenuation maps from the pseudo-CT volume, and **3)** apply the attenuation map to correct the PET image. We are going to focus on the first step producing the pseudo-CT image: synthesis. In this manuscript, synthesis refers to the task of generating an image (target) from a set of images (source) given an input image.

First, we briefly describe the multi-atlas segmentation in biomedical images as a start point to develop our patch-based method to solve the synthesis problem. Multi-atlas segmentation aims to assign segmentation labels to the pixels or voxels of an unlabeled image by using the relationship between the segmentation labels and image intensities observed in images of the atlases [125]. This approach treats segmentation as an image registration problem in which there is a process to propagate correspondences from the atlases to the input image. Since registration-based methods are computationally intensive, some alternatives aim to relax this requirement, such as patch-based strategies. They follow the developments in non-local image denoising [43] as we will see in the next section.

After this, there is a description of the overall workflow of our algorithm and the notation used in the following sections. We explain the way how to evaluate the patch similarity between the input image and the images in the anatomy atlas, non-local self-similarity. We describe how to propagate the labels from the atlases to the target image to produce the final output, group-wise label propagation. Finally, a regularization step is needed to assign a meaningful label to unlabeled voxels.

Contrary to registration-based approaches, patch-based ones might produce no correspondence in certain voxels.

3.4.1 Foundations: Multi-atlas Segmentation

We follow the survey provided by Iglesias et al. [125] to describe the foundations of multi-atlas segmentation. In biomedical image analysis, segmentation refers to group pixels or voxels into biologically meaningful labels according to some criteria, for example, using tissue types or anatomical structures. Medical Imaging enjoys a multitude of general-purpose algorithms and techniques for automatic image segmentation.

Many applications use clinical expert knowledge to label images. These methods are based on the use of *a-priori* acquired data (training images) from the target modality and using the data as an atlas to generate the new data, typically known as atlas-guided segmentation methods and treated as an image registration problem. There are two classes depending on how to combine the training images. The first one is the probabilistic atlas-based segmentation in which atlases are summarized in a probabilistic model. The second one is multi-atlas segmentation (MAS) in which each atlas can potentially contribute to segment the input image.

In this Thesis, we shall focus on MAS methods for our discussion. A MAS algorithm consists of several different steps, steps in *italic* are optional, (Figure 3.17a):

1. Generation of atlases: Offline process to generate the labeled training images (atlases) for the segmentation.

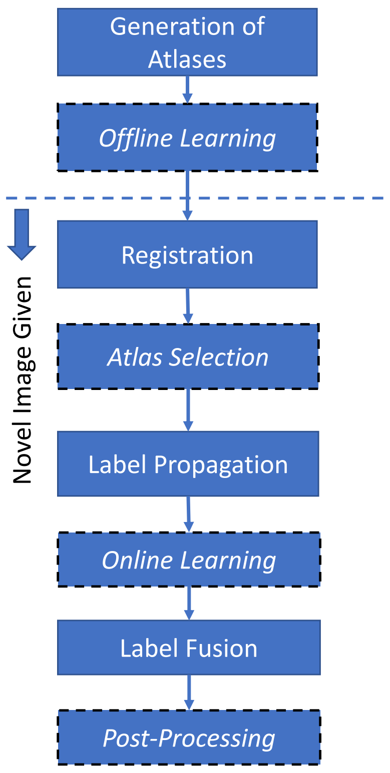
2. *Offline learning*: It aims to analyse the atlases to collect useful information before the segmentation.

3. Registration: Image registration aims to find a spatial transformation (mapping) that associates positions in one image to corresponding positions in one or other images. Image registration consists of the deformation model (rigid or non-rigid), the objective function, typically spatial distance, image intensities, etc., the optimization method, for example, gradient descent. The spatial transformation calculated can then be used to map from the frame of one image to the coordinates of another. In MAS, registration is the step that determines the spatial correspondence between each atlas and the input image. The accuracy of the segmentation depends on how accurate the registration is. More accurate methods usually require more computation.

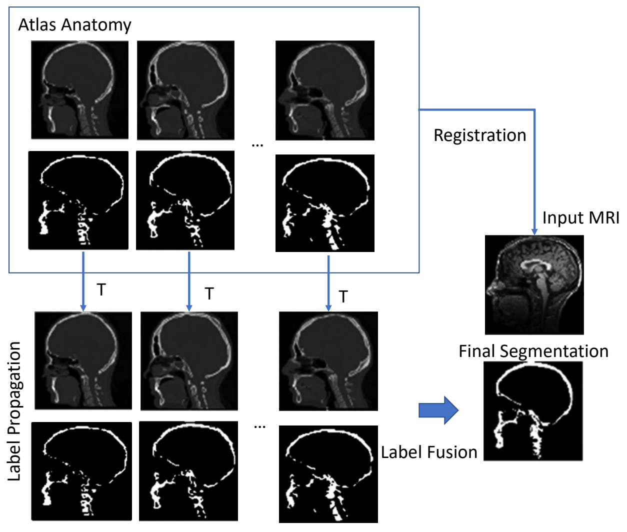
4. *Atlas selection*: Decreasing the number of atlases reduces computational time and discarding irrelevant atlases may improve segmentation accuracy.

5. Label propagation: After registration and possible selection of some atlases, MAS propagates the atlas labels to the input image coordinates space, applying the transformation matrices from registration.

6. *Online learning*: Label fusion algorithms can directly merge propagated labels of the atlases to the input image coordinates. However, some MAS methods try to improve the segmentation by exploiting the relationship between registered atlases and the input image.



(a)



(b)

Figure 3.17: (a) Building blocks of MAS. Dashed blocks may be optional in the process. Inspired by [125]. (b) Workflow for skull estimation through MAS [273]. First, registering the volumes from a multi-atlas CT database to the input image. Second, propagating the corresponding segmentation labels to the input MRI coordinate space. Finally, the segmentation is calculated as a combination of all segmentations in the anatomy atlas by applying a label fusion method, for example, majority voting.

7. Label fusion: It combines propagated atlas labels to infer the final segmentation by using techniques such as major voting (MV), simultaneous truth, and performance level estimation (STAPLE), shape based averaging (SBA), etc. algorithms.

8. *Post-processing*: The result provided by the label fusion step may be the input for another algorithm, for example, using the output as a seed for active contour segmentation.

MAS approach is highly versatile because the main prerequisite is an anatomy atlas, which is a set of pairs including a measured image and the corresponding label map [236]. The key points of this approach (registration-based label propagation) concern the accuracy of the non-rigid registration, the label fusion techniques, the selection of the labeled images, and the labeling errors in primary manual segmentation [236]. For example, a method for complete skull segmentation based only on T1-weighted images of the human head is proposed in [273]. The method uses a pre-alignment and non-rigid registration among volumes and input image (Figure

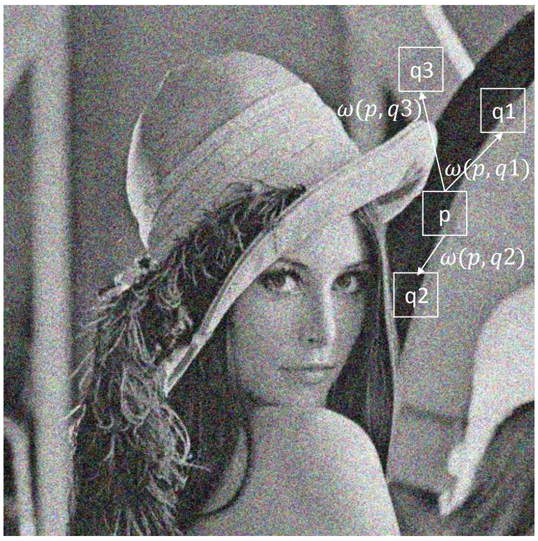


Figure 3.18: Patch similarity description in non-local means (NLM) algorithm. Pixels **q1** and **q2** have a larger weight than **q3** because they are similar based on the intensity gray when comparing the patches. Inspired by [43].

3.17b).

Therefore, each atlas image should align with the input image for MAS algorithm. Registration algorithms are computationally intensive and difficult to parallelize which makes the registration step a bottleneck. Some works have introduced the patch-based idea to MAS [236, 67] for label propagation to identify correspondences among the atlases. These works are based on the denoising algorithm in a non-local framework proposed by Buades et al. [43]. Patch-based methods have shown to be useful to relax the one-to-one constraint existing in non-rigid registration [236, 125].

We briefly describe the classical non-local means (NL-means) to understand the connection between this filter and our proposal. NL-means filter exploits the self-similarity in natural images that every patch has many similar patches in the same image (Figure 3.18). We use a notation slightly different from [43] to keep consistent with the following sections.

Images are defined on the discrete grid I. A patch collection $\mathcal{P}$ is a family $\mathcal{P} = \{\mathcal{P}_\mathbf{x}\}_{\mathbf{x}\in I}$ of subsets of I with $\mathbf{x} = (x, y)^T$ as pixel such that $\forall \mathbf{x} \in I$: 1) $\mathbf{x} \in \mathcal{P}_\mathbf{x}$ and 2) $\mathbf{y} \in \mathcal{P}_\mathbf{x} \Rightarrow \mathbf{x} \in \mathcal{P}_\mathbf{y}$.

For simplicity, a patch (local neighbourhood) has a fixed square dimension. Let v be the discrete noisy image

$$v = \{v(\mathbf{x}) | \mathbf{x} \in I\}$$

The restriction of v to a patch $\mathcal{P}_\mathbf{x}$ is defined as

$$v(\mathcal{P}_\mathbf{x}) = \{v(\mathbf{y}), y \in \mathcal{P}_\mathbf{x}\} \tag{3.65}$$

Buades et al. describe that in principle every pixel $\mathbf{x}$ in the image I is explained as a linear combination of all other pixels, but the algorithm typically reduces these combinations for computation reasons with a search window [43]. Although the method is computationally intensive there exist some optimized versions, for example, on GPU [166]. The estimated denoised value for a pixel is calculated by

$$NL(v)(\mathbf{x}) = \sum_{\mathbf{y}\in I} w(\mathbf{x},\mathbf{y})v(\mathbf{y}) \tag{3.66}$$

In [43], the calculation of similarity of the gray level intensity between patches $\mathcal{P}_{\mathbf{x}}$ and $\mathcal{P}_{\mathbf{y}}$ for pixels $\mathbf{x}$ and $\mathbf{y}$ uses a Gaussian weighted Euclidean distance, $|v(\mathcal{P}_{\mathbf{x}}) - v(\mathcal{P}_{\mathbf{y}})|^2_{2,a}$. This distance is a reliable measure for the comparison of texture patches. Using (3.65), the weight w function reads

$$w(\mathbf{x},\mathbf{y}) = \frac{1}{Z(\mathbf{x})} exp\Big(-\frac{|v(\mathcal{P}_{\mathbf{x}}) - v(\mathcal{P}_{\mathbf{y}})|^2_{2,a}}{h^2}\Big), \tag{3.67}$$

where a is the standard deviation and h controls the decay of the weights as a function of the Euclidean distances. The weights also satisfy the conditions $0 \leq w(\mathbf{x},\mathbf{y}) \leq 1$ and $\sum_{\mathbf{y}} w(\mathbf{x},\mathbf{y}) = 1$. $Z(\mathbf{x})$ is a normalization factor

$$Z(\mathbf{x}) = \sum_{\mathbf{y}\in I} exp\Big(-\frac{|v(\mathcal{P}_{\mathbf{x}}) - v(\mathcal{P}_{\mathbf{y}})|^2_{2,a}}{h^2}\Big)$$

Generally, patch-based methods have been used for medical image denoising [66] and medical segmentation [236, 67]. Ye et al. has adapted this label propagation framework for other tasks [288]. Their proposal is a *modality propagation*, which is the generation of a different image modality by image processing of a different acquired modality.

A fundamental consequence of using patch-based techniques for the synthesis problem is that they are very prone to parallelization, and thus very scalable to a new computing architecture [125]. We will show that the operations involved for calculating a voxel in the synthesis output are independent in the next sections and Chapter 5. Our proposal will be an implementation of modality synthesis on GPU, obtaining synthesis times that are even faster than the acquisition of the real image modality, thus opening the possibility of new clinical applications.

3.4.2 Patch-based Medical Image Modality Propagation

Ye et al. proposed an iterative method for generating T2 and DTI-FA images from T1 MRI images [288]. Our approach is a bit different because we have adapted the ideas in Rousseau et al. [236] for brain labeling and apply them to synthesise a new modality without any iterative scheme.

Figure 3.19 illustrates schematically the workflow of our proposal. The input parameters are an input image I representing the modality from we want to do the synthesis from, an anatomy atlas $\mathcal{A}$ which contains a set of image pairs of two different modalities: $\mathcal{A} = \{(\mathcal{I}^i, L^i)\}^n_{i=1}$. In our case, $\mathcal{I}^i$ are MRI images and L^i CT images labels and $\mathcal{N}$ is the specific neighbourhood. We assume that $\mathcal{I}^i$ and L^i are spatially

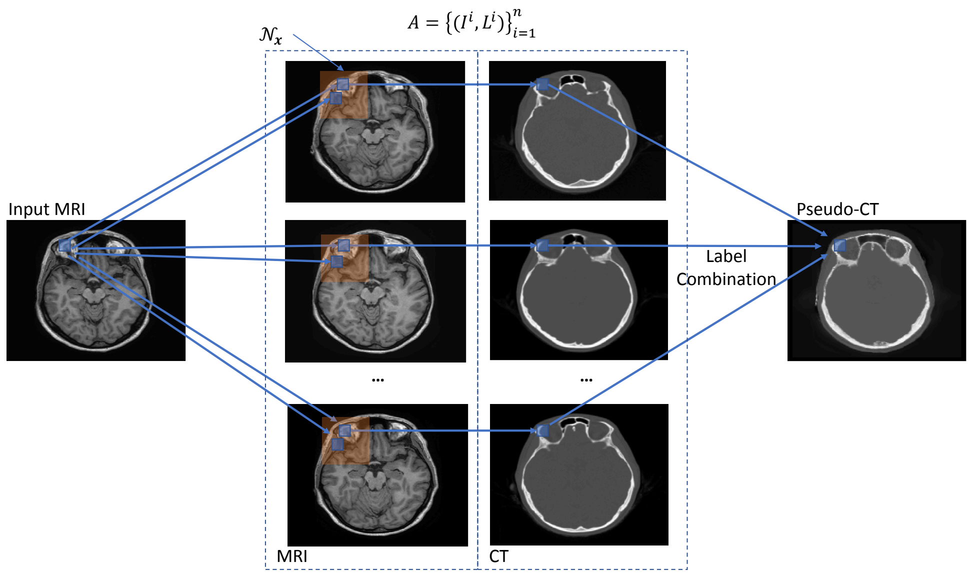


Figure 3.19: Patch-based synthesis workflow. Every voxel in the pseudo-CT is calculated according to a weight combination of source labels in the atlas. Weights measure the similarity between the patch from the image $I(\mathcal{P}_{\mathbf{x}})$ and from the atlases $I^i(\mathcal{P}_{\mathbf{y}})$ in a neighbourhood. For simplicity, only a slice of the volumes is shown.

aligned. We compare the similarity between the input image and the MRI images in the atlas within neighbourhood and combine these similarities to produce the segmentation output ($\hat{L}$) through the labels in the anatomy atlas.

The input image I and the atlas images $\mathcal{I}^i$ and labels L^i as well as the output $\hat{L}$ are volumes. They are defined as follows

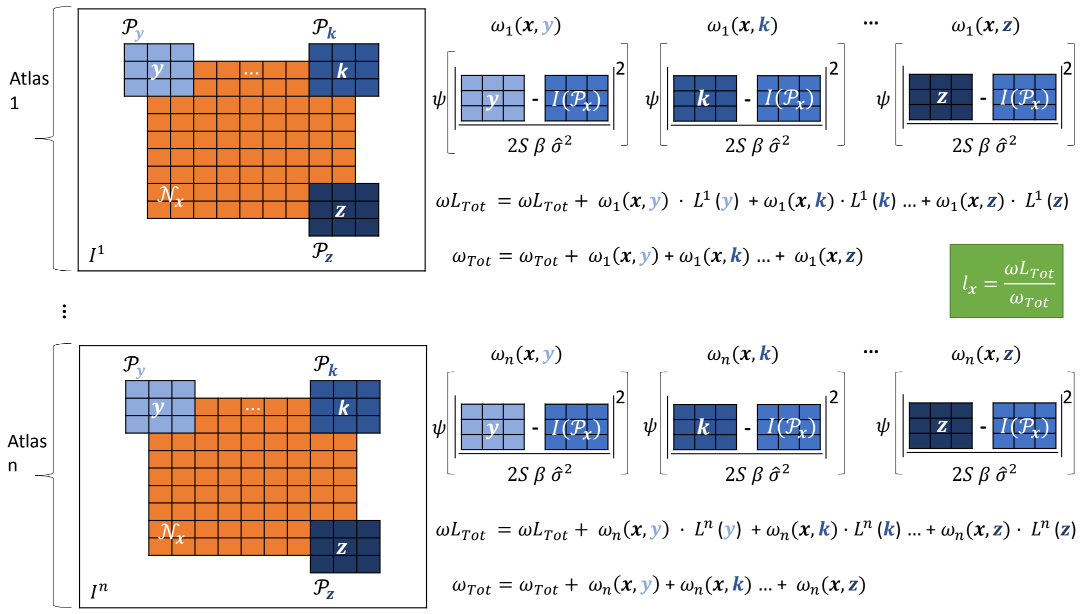
$$u : \Omega_h \to \mathbb{R}$$

where Ω_h is the discrete image domain and $\Omega_h = \{1, \dots N_1\} \times \{1, \dots N_2\} \times \{1, \dots N_3\}$. For simplification in the following formulas, we fix the notation: each voxel $\mathbf{x} \in \Omega_h$ is defined by $\mathbf{x} = (x, y, z)^T$ and the whole volumes are a cubic for simplicity and without loss of generality with dimension N^3. Now, we shall describe the operations in our algorithm.

3.4.3 Non-local Self-Similarity

The weight ω_i is the measurement of non-local similarity at patch level between the input image I and $\mathcal{I}^i$ image from a specific atlas i. Let $\mathbf{x}$ and $\mathbf{y}$ be a voxel in the input image I and the image of the atlas $\mathcal{I}^i$ respectively. The weight calculation reads as follows:

$$\omega_i(\mathbf{x}, \mathbf{y}) = \psi \left(\frac{\sum\limits_{\mathbf{x}' \in \mathcal{P}^I_{\mathbf{x}}, \mathbf{y}' \in \mathcal{P}^{\mathcal{I}^i}_{\mathbf{y}}} (I(\mathbf{x}') - \mathcal{I}^i(\mathbf{y}'))^2}{2S\beta\hat{\sigma}^2} \right) = \psi \left(\frac{|I(\mathcal{P}_{\mathbf{x}}) - \mathcal{I}^i(\mathcal{P}_{\mathbf{y}})|_2^2}{2S\beta\hat{\sigma}^2} \right) \tag{3.68}$$



Atlas 1
Atlas n
$\mathcal{P}_y$
$\mathcal{P}_k$
$\mathcal{P}_z$
$\mathcal{N}_x$
I^1
I^n
$\omega_1(x,y)$
$\omega_1(x,k)$
$\omega_1(x,z)$
$\omega_n(x,y)$
$\omega_n(x,k)$
$\omega_n(x,z)$
$\psi\left[\frac{|y - I(\mathcal{P}_x)|^2}{2S\,\beta\,\hat{\sigma}^2}\right]$
$\omega L_{Tot} = \omega L_{Tot} + \omega_1(x,y)\cdot L^1(y) + \omega_1(x,k)\cdot L^1(k) \ldots + \omega_1(x,z)\cdot L^1(z)$
$\omega_{Tot} = \omega_{Tot} + \omega_1(x,y) + \omega_1(x,k) \ldots + \omega_1(x,z)$
$l_x = \frac{\omega L_{Tot}}{\omega_{Tot}}$
$\omega L_{Tot} = \omega L_{Tot} + \omega_n(x,y)\cdot L^n(y) + \omega_n(x,k)\cdot L^n(k) \ldots + \omega_n(x,z)\cdot L^n(z)$
$\omega_{Tot} = \omega_{Tot} + \omega_n(x,y) + \omega_n(x,k) \ldots + \omega_n(x,z)$

Figure 3.20: The synthesis of a voxel consists of calculating the Euclidean distance $|I(\mathcal{P}_\mathbf{x}) - \mathcal{I}^i(\mathcal{P}_\mathbf{y})|_2^2$ between the input image I patch and each patch in $\mathcal{N}_\mathbf{x}$ from an atlas image $\mathcal{I}^i$. We iterate through all atlases collecting each ωL_{Tot} and ω_{Tot} to produce the output $l_\mathbf{x}$ (rectangle in green) (Eq. 3.69). We set at the beginning for each voxel calculation $\omega L_{Tot} = 0$ and $\omega_{Tot} = 0$. For simplicity, only a slice of the image in the atlas is shown.

where $\mathcal{P}_\mathbf{x}^I$ is a cubic patch of the input volume whose centre is at voxel $\mathbf{x}$ and $\mathcal{P}_\mathbf{y}^{\mathcal{I}^i}$ refers to cubic patch in the atlas volume centred at $\mathbf{y}$ (Figure 3.20). Following [66], the Gaussian weighted Euclidean distance (Eq. 3.67) is replaced by the classical Euclidean distance, decreasing the computational time. Besides, the h^2 parameter has been substituted by $2S\beta\hat{\sigma}^2$ (Eq. 3.68) so that it is independent of the neighbour size. β is a positive real number that influences the difficulty to accept patches with less or more similarity. $\hat{\sigma}$ is the standard deviation of the noise in the images given by Signal Noise Ratio (SNR), but we can expect that images have a good SNR.

The patch size is defined with $S(P \times P \times P)$ number of voxels. The dimension of the patch confers robustness to the similarity measure to capture details and fine structures. We use as transfer function $\psi(r) = e^{-r}$ to keep the values in the range [0,1] being 1 a perfect similarity between patches. However, other sigmoid functions are also possible. The parameters described before are relevant for the quality of the synthesis and Chapter 5 discusses the effect in the result section.

According to the original denoising approach, similarities between patches can be found over an experimental search window [43]. On the contrary, in the context of modality propagation, this similarity search is bounded to the variations of the anatomical structures in a population [236]. This allows us to find good matches in a specific neighbourhood $\mathcal{N}_\mathbf{x}$ of a specific voxel $\mathbf{x}$.

3.4.4 Group-Wise Label Propagation

Once a similarity $w_i(\mathbf{x},\mathbf{y})$ is calculated, we need to combine these weights among atlas and neighbourhood $\mathcal{N}_\mathbf{x}$ to produce the final label for the synthesis output $l_\mathbf{x}$.

The NL-means denoising algorithm relies on the redundancy of any natural image. This means that similar patches can be found in the same image and we can use this fact to calculate a denoised pixel as a weighted average of the pixels in a specific search window. Following this idea, if we assume that patches of the input image I are similar to patches of the anatomy atlas I^i according to a measure distance, then they should refer to similar labels too. Replacing $v(\mathbf{y})$ (Eq. 3.66) by the voxel label $L^i(\mathbf{y})$ in the specific atlas, we can calculate the weighted average of the labels in a $\mathcal{N}_\mathbf{x}$ for all atlases and produce the label output of a voxel as follows:

$$l_\mathbf{x} = \frac{\sum\limits_{i=1}^{n} \sum\limits_{\mathbf{y} \in \mathcal{N}_\mathbf{x}} \omega_i(\mathbf{x}, \mathbf{y}) L^i(\mathbf{y})}{\sum\limits_{i=1}^{n} \sum\limits_{\mathbf{y} \in \mathcal{N}_\mathbf{x}} \omega_i(\mathbf{x}, \mathbf{y})} = \frac{\omega L_{Tot}}{\omega_{Tot}} \tag{3.69}$$

where L^i is the CT image label in atlas i and finally, collecting all the contributions, we have that the volume segmentation is

$$\hat{L} = (l_\mathbf{x})_{\mathbf{x} \in \Omega_h}$$

Figure 3.20 depicts the calculation of a voxel $\mathbf{x}$ in the pseudo-CT output image. This calculation is achieved with the group-wise label propagation (Eq. 3.69) and collecting the sum of square differences among patches in $\mathcal{N}_\mathbf{x}$ with the fixed patch $\mathcal{P}_\mathbf{x}$ from the input image I (Eq. 3.68). We can observe that the algorithm could perform each $l_\mathbf{x}$ in $\hat{L}$ in parallel reducing the overall time of our proposal. Chapter 5 describes how to implement them efficiently and make the most out of these independent operations on multicore and manycore platforms.

3.4.5 Regularization Procedure

Registration-based approaches assume that there is a one-to-one mapping between the input image and the anatomical images in the atlas [236]. However, patch-based methods could produce no correspondence between the patch of the input image I and the patches of the images I^i in the anatomy atlas $\mathcal{A}$. No correspondence means that the group-wise label propagation has not found good similarities and the weights are less than ϵ ($\omega L_{Tot} < \epsilon$ and $\omega_{Tot} < \epsilon$). In those cases, the algorithm associates no value (NaN) to the voxel $\mathbf{x}$ of $\hat{L}$.

The existence of unlabeled voxels in the output image ($\hat{L}$) requires a regularization step to assign a meaningful value to them. These cases are rare in comparison with the volume dimension ($|NaN| \ll N^2$). We have assigned to the unlabeled voxels the value of the median calculated from its neighbourhood, but, for example, inpainting approaches could perform well too. The size of this neighbourhood is $\mathcal{N'}_\mathbf{x}$. The regularization operation reads as follows:

$$\hat{L}(\mathbf{x}) = \begin{cases} \hat{L}(\mathbf{x}), & \text{if} \quad \hat{L}(\mathbf{x}) \neq NaN \\ median_{\mathcal{N'}_\mathbf{x}}(\hat{L}(\mathbf{x})), & \text{otherwise} \end{cases} \tag{3.70}$$

This part of the algorithm will not be optimized because the median procedure requires sorting the elements and the amount of data is not large enough to do in parallel ($|\mathcal{N}'| \ll N$). Algorithm 2 shows the operations involved to accomplish the pseudo-CT image.

Algorithm 2 Pseudo-CT estimation patch-based

```
1: procedure SYTHESISCT($I$,$\mathcal{A}$, $S$,$\hat{\sigma}$,$\beta$,$\mathcal{N}$)
2:     for all $\mathbf{x} \in I$ do
3:         $l_{\mathbf{x}}$ using (Eq. 3.69)
4:     end for
5:     Apply regularization to $\hat{L}$ using (Eq. 3.70)
6:     return synthesis $\hat{L}$
7: end procedure
```

3.5 Algorithm Complexity Analysis

In this section, we are going to analyse the complexity of the algorithms. In this way, we aim to estimate whether the method would experiment with a large improvement or not when using parallel computing techniques.

3.5.1 Saliency Variational Method

Our algorithm consists of two conceptual stages: **1)** initialization stage with superpixels extraction, generation of a control map and features matrix for each superpixel, calculation of the weights, and selection of the k-NN elements **2)** iterative stage, with the variational solver for the saliency segmentation. The solver in the minimization step (Eq. 3.62) is slightly different for NL$TV$ [10] in comparison with our NL$TVSalTerm$, but equivalent in terms of complexity. If the input image has $N = w \times h$, the number of superpixels is sp and the number of k-NN is k, and the number of features f (colour and location). We have the following decomposition:

- Features matrix $\mathcal{O}(N)$
- Control map calculation $\mathcal{O}(sp^2)$
- Weights calculation $\mathcal{O}(sp^2)$
- Primal-Dual saliency. Since it is an iterative part, we present here the complexity for an iteration: $\mathcal{O}(3 \times sp \times sp + sp)$ (non-local gradient + non-local divergence + energy + update)

The number of operations the algorithm must perform can be approximated calculated from the equations 3.50, 3.56, 3.57, 3.61, 3.62, and 3.64

- Features matrix calculation: $N \times f$ summations and sp divisions.
- Weights calculation (Eq. 3.50): For each superpixel calculation, we have $2(f-2)$ multiplications, f subtractions, f multiplications, f summations, 1 division and one exponential operation.
- Control map calculation (Eq. 3.56), (Eq. 3.57): For each superpixel calculation, $(f+1)$ multiplications, f subtractions, $f+1$ summations, 2 division and 2 exponential operations.
- Primal-dual saliency:
  - Maximization (Eq. 3.61): For each superpixel calculation, we have $2sp$ summations, sp substractions, $2sp$ multiplications, sp squares, a division, and sp maximum operations.
  - Minimization (Eq. 3.62): For each superpixel calculation, we have 2 summations, 1 subtraction, 3 multiplications, and non-local divergence (sp summations, sp subtraction, and sp square root).
  - Energy (Eq. 3.64): For each superpixel calculation, we have 4 multiplications, 2 subtractions, 1 square root, 2 summations, sp summations, sp substractions, and $2sp$ multiplications.

Let us assume for our analysis that subtraction, summation, and multiplication consume one-unit time to compute and square root, division, and exponential three units. Finally, the number of operations to be performed in this algorithm is

$$\begin{aligned} Operations &= \underbrace{(N \times f) + 3 \times sp}_{\text{Feature Matrix}} + \underbrace{(5f + 2) \times sp \times sp}_{\text{Weights}} + \underbrace{(3f + 14) \times sp \times sp}_{\text{Control map}} + \\ &= \underbrace{9 \times sp \times sp + 3}_{\text{maximization}} + \underbrace{6 \times sp + 5 \times sp \times sp}_{\text{minimization}} + \underbrace{(11 + 4 \times sp) \times sp}_{\text{energy}} \end{aligned}$$

We also read and store information in memory, we assume single precision (4 bytes). The number of reads/writes:

- Writes = $\underbrace{sp \times f}_{\text{Feature Matrix}} + \underbrace{sp}_{\text{Control map}} + \underbrace{sp \times sp}_{\text{Weights}} + \underbrace{sp}_{\text{primal-dual}}$
- Reads = $\underbrace{sp \times sp}_{\text{Feature matrix}} + \underbrace{2 \times sp \times sp}_{\text{Control map}} + \underbrace{2 \times sp \times sp}_{\text{Weights}} + \underbrace{3 \times sp \times sp + 2 \times sp}_{\text{primal-dual}}$

Taking into consideration that $sp \ll N$ and read/write operations are two orders of magnitude larger than float point operations [285], we can draw from the above calculations that our algorithm will be affected by memory bandwidth bound. The access pattern is good, but there are many positions with zero values that can be not computed. A better organization of the data will be needed to achieve an efficient algorithm. Furthermore, the subtasks exhibit internal summation (see for example the Eq. 3.56), maximum (see for example the Eq. 3.61) operations. These procedures create some dependency between processes. This is the reason why reduction techniques should be used to achieve good performance.

To finalize the iterative part of the algorithm, we need to check whether the criterion is met or not. Typically, the iterative algorithms are not parallel friendly and on GPU Computing the criterion must be evaluated on the host side implying a penalization with the transfer. Finally, we calculate the saliency in the superpixel domain, which means that we have reduced the data to compute. Therefore, GPU computation will be less productive than in the case of MRI-CT Synthesis (Chapter 5). We would expect a gain factor of one order of magnitude.

3.5.2 MRI-CT Synthesis Patch-Based Method

If the problem volume is $N \times N \times N$, the neighbourhood size $K \times K \times K$ and the patch size $P \times P \times P$, and we have A for the number of atlases, the algorithmic complexity would be $\mathcal{O}(A \times P^3 \times K^3 \times N^3)$. Assuming $P \leq K \leq N$, an upper bound would result in $\mathcal{O}(A \times N^9)$. The number of operations the algorithm must perform can be calculated from the equations 3.68 and 3.69. They are as follows:

- Patch similarity: 3 Operations (subtraction, summation, and multiplication) $N^3 \times A \times K^3 \times P^3$ times (see Eq. 3.68) in the dividend part. $2S\beta\hat{\sigma}^2$ does not depend on the data and it calculates offline.

- Propagation: 5 Operations (division, exponential, multiplication, and 2 summations) $N^3 \times A \times K^3$ times (see Eq. 3.69).
- Segmentation: Final output a division operation N^3 times.

We can assume for our analysis that subtraction, summation, and multiplication consume one-unit time to compute and division and exponential three units. Finally, the number of operations to be performed in this algorithm is

$$Operations = \underbrace{3 \times (N^3 \times A \times K^3 \times P^3)}_{\text{Weights}} + \underbrace{9 \times (N^3 \times A \times K^3)}_{\text{Propagation}} + \underbrace{3 \times N^3}_{\text{Final output}}$$

We also read and store information in memory, we assume single precision (4 bytes). The number of reads/writes:

- Writes $W_B = N^3$
- Reads $R_B = A \times N^3 \left\{ \underbrace{K^3}_{\text{Labels}} + \underbrace{(2 \times K^3 \times P^3)}_{\text{Atlas and Input Image}} \right\}$

Taking into consideration that read/write operations are two orders of magnitude larger than float point operations [285], we can draw from the above calculations that our algorithm will be affected by memory bandwidth bound. The reason for this lies in the access pattern in the non-local method which does not fit well in cache policies. Some temporal locality techniques should be implemented for improvement. Apart from that, we can conclude that the calculation of each voxel output is independent of other voxels. This independence calculation for the output voxels and the large data (remember we have volumes) make this algorithm very suitable to accelerate with multicore or GPU computing. Furthermore, the whole algorithm (except for the regularization) is highly parallelizable. Therefore, we would expect a gain factor of two orders of magnitude.

Chapter 4

Saliency Detection in Natural Images

In this chapter, we present the implementation of our proposal to produce the saliency detection in natural images (Section 3.3). This algorithm will be implemented using different paradigms, but the algorithm remains the same. We have implemented a single core solution and two solutions with GPU.

We also present the experiments performed to quantitatively measure the GPU performance compared to a CPU single core solution. We show qualitative and quantitative results that reinforce the idea of our proposal. To arrive at the configurations that we describe here, there has been a lot of experimentation. It is out of the scope of this document to describe all of them, but the rest of the experiments can be found in Github[1]. This chapter is based on the following accepted papers [10, 8].

4.1 Implementation

We can divide our saliency method (Algorithm 1) into two conceptual stages: the initialization stage (lines 2-8) and the iterative stage (lines 9-13). In the next subsections, we shall describe both stages.

4.1.1 Initialization Stage

The weight matrix is very sparse and guides the complete calculations in the iterative stage of the algorithm: non-local gradient (Eq. 3.52) and non-local divergence (Eq. 3.53). Usually, instead of performing the operations in the whole matrix $W^{sp\times sp}$, the matrix is transformed into a more compact one, $W^{sp\times k}$, following the CSR (Compressed Sparse Row) representation.

$$\underbrace{\begin{pmatrix} 5 & 0 & 1 & 0 & 2 \\ 3 & 5 & 0 & 0 & 1 \\ 0 & 2 & 8 & 0 & 1 \\ 1 & 0 & 0 & 1 & 3 \\ 4 & 0 & 9 & 0 & 2 \end{pmatrix}}_{\text{Weights}} \Rightarrow \begin{array}{lcl} \textbf{Rows} & = & [0,3,6,9,12,15] \\ \textbf{Cols} & = & [0,2,4,0,1,4,1,2,4,0,3,4,0,2,4] \\ \textbf{Values} & = & [5,1,2,3,5,1,2,8,1,1,1,3,4,9,2] \end{array}$$

[1]https://github.com/EduardoAlcainBallesteros/VariationalSaliencyDetection

This CSR format can represent any arbitrary sparse pattern and, in particular, our case which is induced by the k-NN dimensional reduction. Notice that the number of non-zero entries (NNZ=$sp \times k$) varies for each different input image. CSR representation consists of three vectors: **Rows** $\in \mathbb{N}_0^{sp+1}$ and **Cols** $\in \mathbb{N}_0^{sp \times k}$ to store the row/column indices of the non-zero entries in the original matrix and **Values** $\in \mathbb{R}^{sp \times k}$ to store the values of the matrix.

This format is commonly used for sparse matrix-vector multiplication [24] (like our non-local gradient operation) because row indices introduce fast memory access to values in the matrix and easy calculation of the number of non-zeros in a specific row. CSR representation reduces drastically the memory for the weight matrix and the number of operations performed in the non-local gradient and non-local divergence. Besides, the optimization in CPU makes use of the vectorization guidelines for the Intel compiler [130] to improve the performance. The GPU implementation starts by migrating this optimized CPU approach into CUDA.

After having partitioned the image domain using SLIC on GPU [230] (Figure 3.13b), we extract the features for each superpixel (colour and location) creating $\mathbf{f}_p$ for $p \in V$ (Section 3.3.3).

This procedure fits well for CPU but not for GPU because some coordination is needed to collect the data and sum them up. Reduction techniques achieve the necessary coordination. A reduction is a class of parallel algorithms that produces a scalar result from a collection [283]. Using an NVIDIA Kepler GPU, we employ the *atomicAdd* operation, as according to [198] it is an optimized implementation since CUDA 7.5. In any case, there are several reduction implementations with different performances depending on the GPU family.

As each feature vector $\mathbf{f}_p$ is normalized, it requires the computation of *min* and *max* of each feature and, again, we perform them at the same time with a reduction kernel.

Algorithm 3 Weights and control map calculation

```
1: procedure ControlMapAndWeights(L,a,b,x,y)
2:     Calculate $w_{pq}$ using (Eq. 3.50)
3:     shared memory $\leftarrow v_{pq}^{con}$ using (Eq. 3.56)
4:     sync
5:     reduce $v_p^{con}$ in shared memory
6:     sync
7:     Calculate $v_p^{obj}$ using (Eq. 3.57)
8:     Calculate $v_p^c = v_p^{con} v_p^{obj}$
9:     return $W^{sp \times sp}$, $\mathbf{v}^c$
10: end procedure
```

Before implementing the numerical solution of our model, we compute the weights (Eq. 3.50) and the control map through (Equations 3.56 and 3.57), which are very suitable and efficient for the CUDA environment (element-wise operations). For optimization purposes, both operations are unified into a single CUDA kernel with grid configuration of $sp \times sp$ (Algorithm 3). L, a, b $\in \mathbb{R}^{sp}$ are the colour components in CIE $L*a*b$ colour space and x, y $\in \mathbb{R}^{sp}$ are the location components.

Note that the *sync* instructions in lines 4 and 6 (Algorithm 3) are synchronization barriers to ensure all threads reach these instructions in their independent evolution through the kernel execution. The control map procedure requires a reduction operation inside the kernel (line 5) because the contrast prior (Eq. 3.56) must be repeated from one superpixel to all superpixels. Again, the control map must also be normalized in the range $[0, 1]$ for the primal-dual algorithm.

The next operation in the initialization stage is the selection of the k-NN values for each superpixel in the $W^{sp \times sp}$ matrix (Algorithm 4). As the matrix is not very large, in CPU is more efficient to sort the values in rows ($\mathcal{O}(Nlog_2N)$) and extract the k-largest values in each row than finding directly the k-largest values in each unsorted row. However, due to insufficient data and few numbers of rows (number of sorts), this approach is unfortunately not very efficient for the GPU. Our strategy is based on a loop of reductions to find the k-largest values in each row until all the rows are completed.

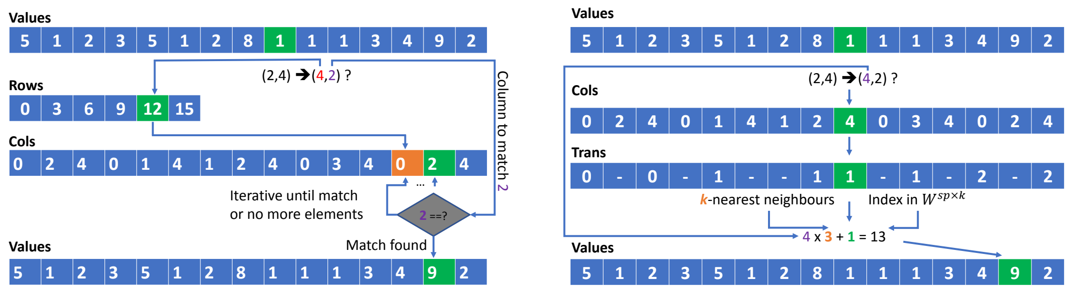
Algorithm 4 Select k-NN values in weights matrix ($W^{sp \times sp}$)

```
1:  procedure CALCULATEK-NNKERNEL(weights,sp,k)
2:      $h = 0$, th = actual thread index
3:      repeat
4:          sdata = row-weights, maxs = row-indices
5:          sync, $j = 0$
6:          repeat
7:              $sdata[maxsFinal[j]] = 0, j = j + 1$
8:          until $j < h$
9:          sync
10:         Find max in sdata & store index in maxsFinal
11:         sync
12:         $h = h + 1$
13:     until $h < k$
14:     if th $\neq$ maxsFinal then
15:         Write zero in actual position in weights
16:     end if
17: end procedure
```

4.1.2 Iterative Part: First GPU Implementation

The iterative part lines 9-13 (Algorithm 1) has data dependencies among its operations and they must be computed one after another. We first split the maximization step (Eq. 3.61) into three kernels (Figure 4.2a): 1) numerator calculation with the non-local gradient of $\mathbf{u}^k$ (**nltvGradient**) (Eq. 3.52). The non-local gradient can be seen as sparse matrix-vector multiplication. The CSR representation makes this operation achieve full coalesced memory access. 2) max-reduction calculation (**Max**). 3) Divide the numerator by the max (**DivByMax**).

We can again split the minimization step (Eq.3.62) into two kernels (Figure 4.2a): 1) the calculation of the non-local divergence (**nltvDivergence**). Although the CSR



(a) (b)

Figure 4.1: Given an initial position (2,4), the selection of its transpose element (4,2) in the CSR format: (a) First GPU implementation. After selecting the row, we iterate through the col vector to match the specific col. (b) Optimized GPU proposal. A lookup table (**Trans**) is used to find directly whether an index in the compact matrix $W^{sp \times k}$ has a transposed element or not.

format also improves this procedure by reducing the number of operations, the access to transposed elements (needed in Eq. 3.53) is not direct and requires an iterative procedure inside the kernel to find the transposed element (Figure 4.1a). 2) Update the solution, which is an element-wise GPU efficient operation (**UpdateUk**). The energy calculation (Eq. 3.64) for the stop criterion exhibits similar GPU data reduction inefficiencies as the extraction of superpixel features and we similarly implement this operation.

4.1.3 Iterative Part: Optimized GPU Version

The first GPU approach solves the data dependencies with the introduction of different kernels. However, as our formulation in superpixels decreases the amount of data to analyse, the time of launching kernels becomes crucial for performance improvement. This data dependency can be solved by using thread synchronization barriers inside kernels while merging them. Therefore, we have merged the maximization and minimization kernels shown in green and purple boxes, respectively (Figure 4.2b).

Apart from the data dependency difficulties, the non-local divergence (Eq. 3.53) implies finding transposed elements in its calculation used by the primal variable update (Eq. 3.62). In the first GPU implementation, a procedure inside the kernel to find the transposed value has been implemented. This procedure is inefficient because most of the values are zero and the loop for finding the match in the column vector is inside the kernel (in the worst case it iterates k times). However, the dual variable (**d**) has only values in the indices where the weights are not zero. Furthermore, the weights matrix is constant during the algorithm. Therefore, we can create an offline lookup table (LUT) with the transposed information (**Trans** $\in \mathbb{Z}^{sp \times k}$). This LUT vector stores the transposed result for each index in the compact matrix $W^{sp \times k}$ where "-" entry means the transposed value is zero and a value $\in [0, k)$ indicates the index of the column where the value in the compact matrix is located (Figure 4.1b).

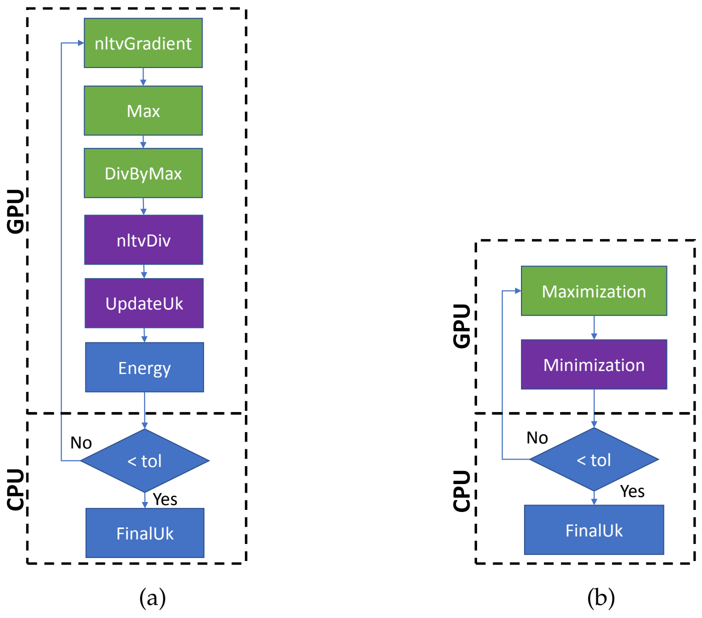


Figure 4.2: These diagrams show the kernels used to calculate the iterative part (Eq. 3.60). (a) First GPU implementation, where each operation in CPU has been implemented as a CUDA kernel and (b) optimized proposal grouping GPU kernels. **Minimization** also calculates the stop criterion (energy). **FinalUk** transfers the final $\mathbf{u}^k$, normalizes data [0,1], and projects the result onto the pixel domain.

4.2 Results

This section provides visual and performance results comparing our non-local TV including saliency term (*NLTVSalTerm*) against different variational approaches.

4.2.1 Platforms

First, we shall describe the platforms used and the way to refer to them along with the discussion of this case. The naming conventions used in the experimental results are as follows:

- The name convention consists of two names with a dash separator between each name, for instance, GPU-Sal+.
- The first name refers to the type of platform: CPU stands for the implementation on CPU and GPU stands for the implementation of the algorithm with CUDA. The devices used are:
  - CPU: Intel Xeon E5-1650v3, 3.5 GHz hexa-core processor, from the 2014 Intel Haswell architecture (Haswell-EP), 1.5MB L2 cache, and 15MB L3 cache with 64GB DDR3 RAM [128]. The CPU uses a Microsoft Windows Server 2012R2 as the operating system.
  - GPU: The Tesla K40c professional platform (with the NVIDIA Kepler GK110 architecture supporting CUDA compute capability 3.5) with 12GB RAM GDDR5, with 15 streaming multiprocessors of 192 CUDA cores each giving a total of 2880 CUDA cores. 5.046 TFlops [201].

- The second name refers to the compiler used in the case of CPU and for the GPU case refers to the implementation choice:
  - CPU: mvcc compiler from Microsoft and icc compiler from Intel.
  - GPU: Sal (first implementation), Sal+ (optimized implementation), and Sal* (without energy check).

We have implemented the code with Visual Studio 2015 Community using the Intel Compiler C++ 19.0 and CUDA 9.0.

4.2.2 DataSets

We have tested our saliency detection algorithm on different public benchmarks:

- **MSRA10K** [58]: This dataset extends **MSRA1K** [155] and has 10000 pixel-wise ground truth annotations images. It contains only one ambiguous salient object in each image with a background clearly different in comparison with the object.
- **ECSSD** [255]: This dataset contains 1000 images from the Internet and five persons marked the ground truth in the images. The authors provide a clear analysis about the properties of their dataset and their importance: multiple salient objects and the intensity difference between background, and foreground is less than others dataset.
- **iCoseg** [22]: This dataset has 643 images from Flickr Website in different situations. Providing pixel-wise ground truth annotation images and images with different degrees of complexity (one or two salient objects).

4.2.3 Evaluation metrics

The evaluation of the algorithms is according to the typical metrics used in Saliency such as Precision (PR), Recall (RC), F-measure, and Mean Absolute Error (MAE) [96]. Let $u : \Omega \subset \mathbb{R}^2 \mapsto \mathbb{R}$ be the estimated saliency map and $g : \Omega \subset \mathbb{R}^2 \mapsto \mathbb{R}$ be the ground truth. To evaluate the estimated saliency map (u) against the ground truth (g) by the models: first, the saliency map must be binarized accordingly to a threshold and secondly applied the evaluation metrics described as follows:

- **Precision**: Precision is also known as quality or exactness. It measures the percentage of selected pixels that belong to the ground truth.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4.1}$$

  where *TP* represents the true-positive cases and *FP* the false positive (elements marked as salient when they are not).

- **Recall**: Recall is also known as sensitivity. It measures the percentage of the salient pixels detected in comparison with the total in the ground truth.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.2}$$

  where *FN* represents the false-negative cases (elements marked as background when they are salient).

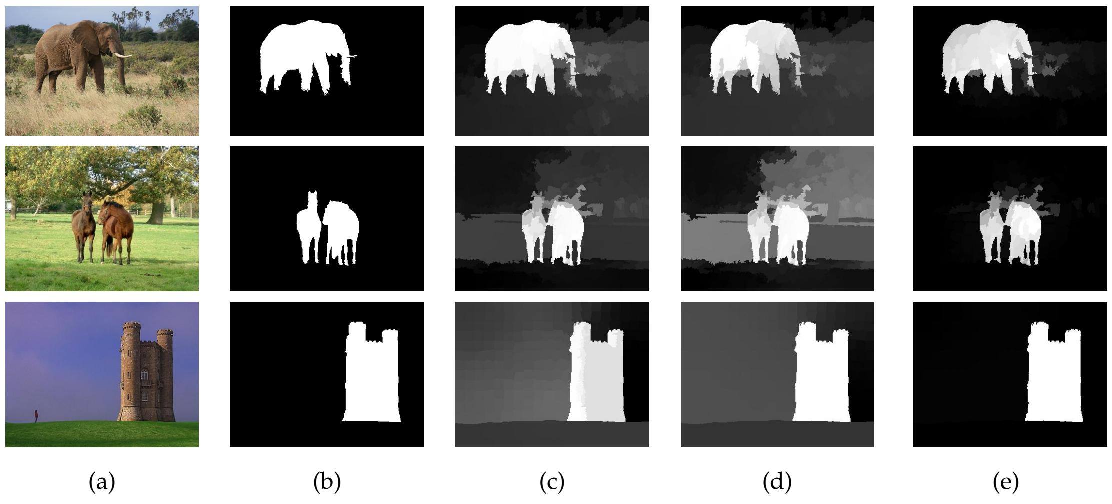


Figure 4.3: Examples of saliency detection in datasets (a) original image, from top to bottom iCoseg [22], ECSSD [287], and MSRA10K [58]. (b) ground truth and saliency maps using: (c) NLL_0 [281], (d) NL$TV$ [10], and (e) NL$TVSalTerm$ [8].

- **F-measure**: Sometimes it is difficult to measure the quality of a model either with precision or recall. A way to alleviate this is by using a harmonic mean between precision and recall measurements. It is common to set $\beta = 0.3$ following [2]

$$F_\beta = \frac{(1+\beta^2)\cdot \text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}} \tag{4.3}$$

- **Mean Absolute Error** (MAE): A complementary measure to assess the validity of the method is MAE because it penalizes the true negative in the output of the algorithm. The range of this measurement is [0,1] where low values of MAE mean that the output has more similarity with the ground truth.

$$MAE = \frac{1}{N}\sum_{\mathbf{x}\in\Omega} |u(\mathbf{x}) - g(\mathbf{x})| \tag{4.4}$$

  where $\mathbf{x} \in \Omega$ and $\mathbf{x} = (x, y)^T$. $N = w \times h$ with w and h width and height of the image respectively.

4.2.4 Qualitative Results

To make a fair comparison among methods non-local non-convex L_0 (NLL_0) [281], non-local convex TV (NL$TV$) [10], and non-local convex TV with saliency term (NLTV-$SalTerm$) [8] in the datasets, we binarize the saliency maps using a threshold starting by 0 up to 255 [2]. In each threshold, we measure the four metrics, described in the previous section, reporting the mean for the whole range $[0, 255]$ for each metric. We use for NLL_0 our own implementation and the parameters are set according to [281], NL$TV$ has the same parameters as in [10] and for the model NL$TVSalTerm$, we set $\lambda = 1$ $\alpha = 1.5$, and $\delta = 0.2$ using (3.63) we get $a = -0.64$. For all methods, we use a fixed number of iterations (50), number of superpixels $sp \in [300, 650]$ and $k = 5$.

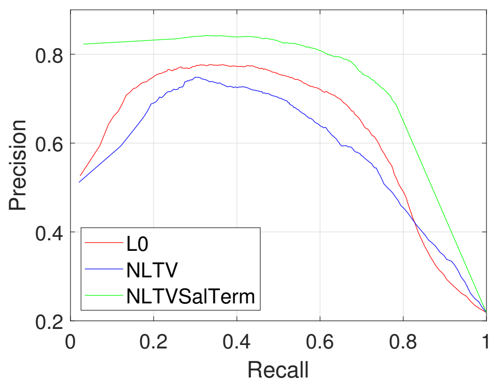
Table 4.1: Quantitative results for the three datasets for the NLL_0, NL$TV$, and NL$TVSalTerm$ methods using Precison (Eq. 4.1), Recall (Eq. 4.2), F-Measure (Eq. 4.3), and MAE (Eq. 4.4). In **bold** the best performance for the specific metric.

iCoseg	NLL_0 [281]			NL$TV$ [10]			NL$TVSalTerm$ [8]		
	300	450	600	300	450	600	300	450	600
Precision	0.678	0.713	0.722	0.653	0.662	0.683	0.805	**0.820**	**0.820**
Recall	0.524	0.526	0.529	0.518	0.506	0.509	**0.557**	0.542	0.524
F-Measure	0.589	0.603	0.606	0.580	0.577	0.587	0.716	**0.718**	0.708
MAE	0.220	0.214	0.207	0.236	0.240	0.225	0.147	0.150	**0.151**
ECSSD	NLL_0 [281]			NL$TV$ [10]			NL$TVSalTerm$ [8]		
	300	450	600	300	450	600	300	450	600
Precision	0.644	0.683	0.700	0.534	0.596	0.635	0.687	0.749	**0.762**
Recall	0.517	**0.522**	0.521	0.505	0.510	0.515	0.469	0.476	0.461
F-Measure	0.544	0.561	0.568	0.479	0.516	0.538	0.601	**0.637**	0.633
MAE	0.265	0.248	0.238	0.329	0.294	0.277	0.178	**0.170**	0.172
MSRA10K	NLL_0 [281]			NL$TV$ [10]			NL$TVSalTerm$ [8]		
	300	450	600	300	450	600	300	450	600
Precision	0.748	0.773	0.787	0.664	0.714	0.742	0.835	0.862	**0.874**
Recall	0.544	0.557	0.559	0.523	0.538	0.542	**0.594**	0.577	0.556
F-Measure	0.630	0.643	0.646	0.582	0.615	0.630	0.748	**0.756**	0.750
MAE	0.209	0.197	0.191	0.248	0.227	0.215	0.119	**0.120**	0.124

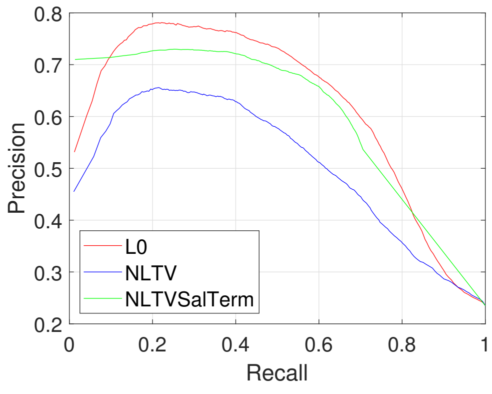
Figure 4.3 shows a visual comparison of the considered methods. We can observe that saliency maps provided by NL$TVSalTerm$ algorithm remove the background from the input images much better than the rest of methods (Figure 4.3e).

4.2.5 Quantitative Results

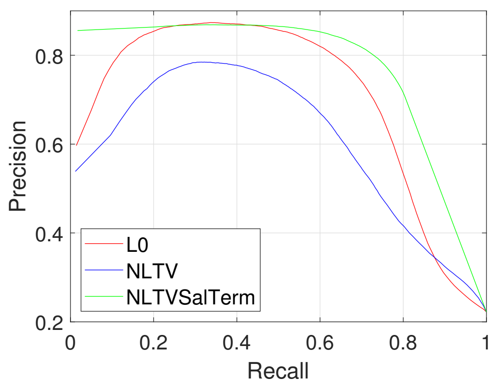
The quantitative comparison shows that NL$TVSalTerm$ algorithm provides the best results for almost all metrics (Table 4.1). Precision is always higher (Figure 4.4) because of the specific saliency term which, through the parameter (δ), increases the percentage of selected pixels that belong to the ground truth (true positive). This mechanism, in turn, restricts the performance of the recall metric, which is very similar in all methods. A complementary measure to assess the validity of the method is MAE because it penalizes the true negatives in the output of the algorithm. In this case, low values of MAE mean that the output has more similarity with the ground truth. NL$TVSalTerm$ has the lowest values for all configurations with a substantial margin (Figure 4.4d) and this behaviour is again related to the effect of the saliency term $H(u)$.



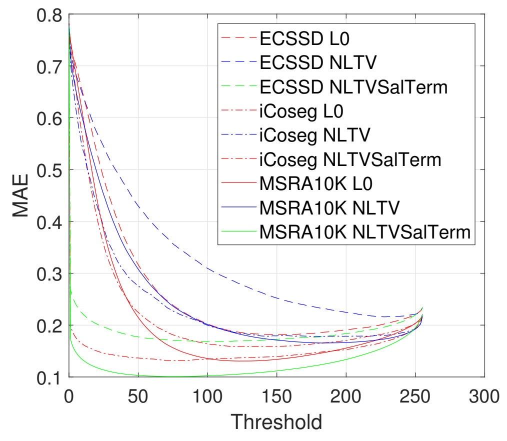
(a)



(b)



(c)



(d)

Figure 4.4: Precision vs Recall curves using 300 superpixels for NLL_0 [281], NLTV[10], and NL$TVSalTerm$ [8]: (a) iCoseg benchmark results (b) ECSSD benchmark results (c) MSRA10K benchmark results and (d) MAE Results for all benchmarks using 300 superpixels as well.

Table 4.2: Timing results (in ms) of the saliency method $NLTVSalTerm$ in CPU for an image [400 × 334] from the MSRA10K dataset and 50 iterations. **CPU-mvcc** stands for Microsoft Visual C++ compiler, **CPU-icc** for Intel C/C++ 19.0 compiler and **Tesla K40C** for Kepler. SP is the number of superpixels used in the algorithm. Speedups against **CPU-mvcc** version solution in parentheses.

	CPU-mvcc			CPU-icc		
SP	SLIC	Sal	Total	SLIC	Sal	Total
300	49.33	6.50	55.83	21.33 (×2.31)	4.25 (×1.53)	25.58 (×2.18)
350	49.62	8.09	57.71	21.41 (×2.32)	5.36 (×1.51)	26.77 (×2.16)
400	49.87	11.56	61.43	21.69 (×2.30)	7.78 (×1.49)	29.47 (×2.08)
450	49.83	14.07	63.90	22.41 (×2.22)	9.57 (×1.47)	31.98 (×2.00)
500	50.18	17.42	67.60	22.30 (×2.35)	11.90 (×1.46)	34.20 (×1.98)
550	50.00	18.09	68.09	21.87 (×2.39)	12.40 (×1.46)	34.27 (×1.99)
600	50.32	22.90	73.22	22.50 (×2.24)	15.78 (×1.45)	38.28 (×1.91)
650	50.16	28.55	78.71	22.35 (×2.24)	19.92 (×1.43)	42.27 (×1.86)

Table 4.3: Timing results (in ms) of the optimized saliency method NL$TVSalTerm$ for an image [400 × 334] and 50 iterations in GPU. SP is the number of superpixels. **GPU-Sal** column represents the saliency of the first GPU implementation. **GPU-Sal+** the optimized GPU version. **GPU-Sal*** represents the saliency results without calculating the energy of the functional. In parentheses total speedups against **CPU-icc** version solution.

GPU Tesla K40C							
SP	SLIC	Sal	Sal+	Total	Total+	Sal*	Total*
300	8.95	22.44 (×0.19)	8.41(×0.51)	31.38	17.36	6.08(×0.70)	14.93
350	8.57	22.87 (×0.23)	8.40(×0.64)	31.49	16.97	6.02(×0.89)	14.51
400	8.66	23.39 (×0.33)	9.02(×0.86)	31.99	17.68	6.17(×1.26)	14.72
450	8.52	23.63 (×0.40)	9.23(×1.04)	32.15	17.75	6.34(×1.51)	14.85
500	8.61	24.61 (×0.48)	9.51(×1.25)	33.25	18.12	6.43(×1.85)	14.90
550	8.16	25.31 (×0.49)	9.57(×1.30)	33.52	17.73	6.52(×1.90)	14.70
600	8.21	25.43 (×0.62)	9.73(×1.62)	33.68	17.94	6.66(×2.37)	14.84
650	8.23	25.18 (×0.79)	9.79(×2.03)	33.48	18.02	6.60(×3.02)	14.78

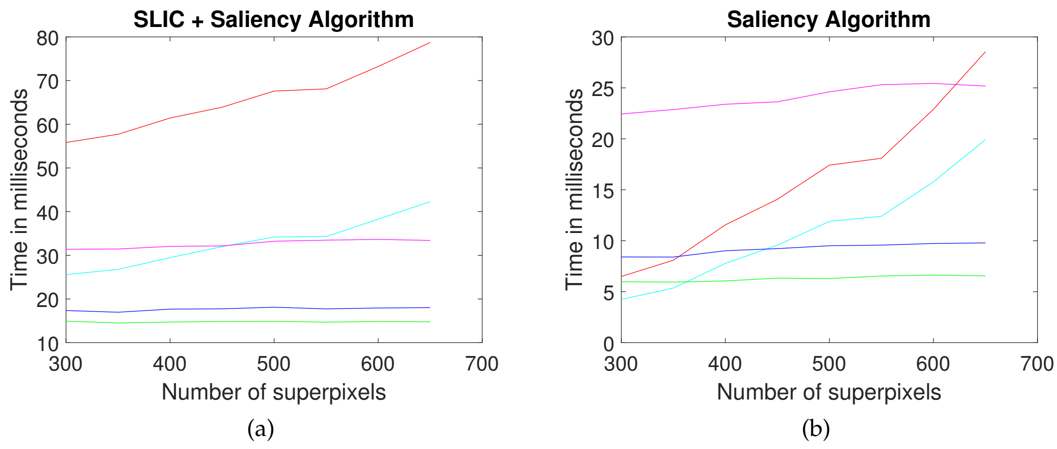


Figure 4.5: CPU-mvcc configuration (red), CPU-icc (cyan), GPU-Sal (magenta), GPU-Sal+ (blue) and, GPU-Sal* (green), varying the number of superpixels [300,650]. Computation times: (a) (SLIC + Saliency Algorithm) (b) Saliency algorithm.

4.2.6 Computational Time

In the different platforms, timing results are the average of running the method 100 times and 50 iterations, and varying the number of superpixels in the range $[300, 650]$. We have used a GPU version of SLIC from [230] to keep a complete GPU solution, but we have only reported and discussed GPU speedups of our contribution (NL$TV$ or NL$TVSalTerm$).

Figure 4.5a shows the total time in milliseconds of different configurations, on CPU and GPU. The Intel C++ compiler (CPU-icc) is twice as fast as Visual C++ compiler (CPU-mvcc) creating more efficient auto-vectorized code using the same -O2 compiler directive and better loop organization improving memory accesses. On the other hand, Table 4.3 shows the computational time for independent methods on GPU: over-segmentation (SLIC) and the variational saliency algorithm.

We report timing comparing the best CPU solution (CPU-icc) (Table 4.2) against three different GPU implementations. The first GPU implementation (Section 4.1.2) performs worse than either CPU-mvcc or CPU-icc version (Figure 4.5b). The second GPU implementation ($Sal+$) is the optimized version (Section 4.1.3) and is about $\times 2.6$ faster than the first GPU implementation. However, this implementation is only faster than CPU-icc when using more than 450 superpixels because the resulting energy is transferred back to the main memory for evaluating the stopping criterion in CPU, although the number of iterations is fixed to 50. Memory transfer in each iteration is one of the historical bottlenecks of GPU implementations and many strategies can be adopted to reduce this penalty.

As we keep a fixed number of iterations of the algorithm for comparison purposes, we report a third GPU approach (*Sal**) avoiding the energy calculation and data transfer from GPU memory to host memory in each iteration. This strategy is reasonable as the implementation is focused on the hardware limitations, reducing the bottleneck, but ensures convergence when compared to an implementation that uses the stopping criterion with fewer iterations. The algorithm reaches an acceptable convergence after about 50-60 iterations (Figure 3.16d).

After removing the energy calculation ($Sal*$), we achieve a benefit of up to $\times 3.02$ when compared to the CPU-icc results for configurations of 650 superpixels and $\times 3.8$ compared to the first GPU version. The results show that the method reaches real-time frame rate capability either on CPU ($\sim$ 25 FPS for 650 superpixels) or on GPU ($\sim$ 60 FPS for 650 superpixels).

The configurations grid for the GPU kernels in the optimized version are as follows:

- Calculation of the features for each superpixel. We use a grid of (sp,sp)
- Calculation of the initial saliency map and weights. We use a grid of (sp,sp)
- Select the k-NN elements in the weight matrix (sp,sp)
- Iterative part Maximization. We use a grid of (sp,k)
- Iterative part Minimization. We use a grid of (1,sp)

4.3 Discussion

The method proposed is a novel non-smooth and non-local variational method on graphs for saliency segmentation. Our solution solves the minimization problem with a primal-dual algorithm which converges quickly to the solution in a few iterations. Furthermore, methods using the primal-dual algorithm (NL$TV$ [10] and NL$TVSalTerm$ [8]) perform better than ADMM in NLL_0 [281] according to time and iterations in this problem.

On the other hand, we have compared quantitatively and qualitatively the methods in three well-known datasets, which have different levels of difficulty. As a result, we can conclude that NL$TVSalTerm$ incorporates the missing piece in the formulation to perform much better than NLL_0 and our previous method NL$TV$. Although the method is based on simple priors, the experimental results show that the method produces very high-quality results removing the background and highlighting the salient part of the image. The quantitative metrics confirm NL$TVSalTerm$ performs better in all of them except for recall that all are very similar.

The saliency term in the model controls the probability of when a superpixel is salient and when it is not. We have demonstrated that the inclusion of this parameter in the model eliminates the background and highlights the salient part. Besides, we obtain an almost binary output in most of the cases. However, this mechanism, in turn, restricts the performance of the recall metric, which is very similar in all methods.

We have also presented an exhaustive performance comparison between CPU and GPU using different configurations, programming tools, technologies, and platforms demonstrating real-time performance either on CPU or GPU, even with computationally intensive configurations. Since we reduce the complexity (we manipulate the saliency in the superpixel domain), the amount of data is not enough to appreciate a significant impact when using GPU computing in this scenario. Therefore, we see better computational results when the number of superpixels is greater than 450. However, we have derived strategies to make efficiently the algorithm (data organization, grid configuration, and reductions) on GPU approaching the optimized CPU-icc computational time with few superpixels ~ 300.

We have achieved real-time capability either on CPU or GPU, even with the handicap of transferring the energy each iteration. Real-time outputs allow us to use this algorithm in video or multi-resolution schemes. Considering that the algorithm converges in a few iterations, we have proposed eliminating the energy transfer and, in this way, improving the performance without losing accuracy.

Chapter 5

MRI-CT Synthesis

In this chapter, we present the implementation of our proposal to generate the pseudo-CT (Section 3.4). This algorithm will be implemented using different paradigms but the algorithm remains the same. The implementation is divided into three sections: single core implementation, multicore implementation on CPU, and GPU implementation. In each implementation, an optimization is added reducing the time substantially.

We also present the experiments performed to quantitatively measure the GPU performance compared to a CPU single and multicore solution. We show qualitative and quantitative results that reinforce the idea of our proposal. Finally, we have run the same algorithm with new hardware and software to illustrate the natural scalability of the method. The results have confirmed that new hardware and software improve the results achieved in Alcaín et al. [9]. To arrive at the configurations that we describe here, there has been a lot of experimentation. It is out of the scope of this document to describe all of them, but the rest of the experiments can be found in Github[1] as well as the source code for each implementation. Since the regularization step is not optimized, we do not describe its implementation (see the source code for further details). This chapter is based on the following accepted papers [9, 271].

5.1 Implementation

5.1.1 Single Core Implementation

The first implementation of the patch-based algorithm is under C++ using only one core. The procedure is divided into four logical blocks:

1. Responsible for iterating through the input image I ($N \times N \times N$) line 2 (Algorithm 5).

2. Responsible for iterating through the atlas $\mathcal{A}$ line 5 (Algorithm 5).

3. Responsible for iterating through the neighbourhood $\mathcal{N}$ ($K \times K \times K$) lines 6 and 12 (Algorithm 5) with the calculation (Eq. 3.69).

4. Responsible for iterating through the patch ($P \times P \times P$) line 7 and 8 (Algorithm 5) and calculating Euclidean distance $|I(\mathcal{P}_{\mathbf{x}}) - I^i(\mathcal{P}_{\mathbf{y}})|_2^2$ where I^i refers to the MRI image in the atlas.

The calculation of the ω_{Tot} and ωL_{Tot} is an accumulative process among the atlas and neighbourhood line 11 and 12 (Algorithm 5). This summation suffers

[1] `https://github.com/EduardoAlcainBallesteros/PseudoCTImaging`

Algorithm 5 Patch-based MRI-CT synthesis approach CPU/C++

```
procedure PATCHBASED($I$,$\mathcal{A}$,$S$,$\hat{\sigma}$,$\beta$,$\mathcal{N}$)
    for all $\mathbf{x}$ in $I$ do
        $\omega_{Tot} = 0$
        $\omega L_{Tot} = 0$
        for all $i$ in $\mathcal{A}$ do
            for all $\mathbf{y}$ in $\mathcal{N}_{\mathbf{x}}$ do
                for all $x'$ in $\mathcal{P}_{\mathbf{x}}^{I}$ and $y'$ in $\mathcal{P}_{\mathbf{y}}^{\mathcal{I}^i}$ do
                    $\omega_i(\mathbf{x},\mathbf{y}) = \omega_i(\mathbf{x},\mathbf{y}) + (I(\mathbf{x}') - \mathcal{I}^i(\mathbf{y}'))^2$
                end for
                $\omega_i(\mathbf{x},\mathbf{y}) = \omega_i(\mathbf{x},\mathbf{y})/2S\beta\hat{\sigma}^2$
                $\omega_{Tot} = \omega_{Tot} + \omega_i(\mathbf{x},\mathbf{y})$
                $\omega L_{Tot} = \omega L_{Tot} + \omega_i(\mathbf{x},\mathbf{y}) \cdot L^i(\mathbf{y})$
            end for
        end for
        if $\omega L_{Tot} < \epsilon$ and $\omega_{Tot} < \epsilon$ then
            $l_x = NaN$
        else
            $l_x$ using (Eq. 3.69)
        end if
    end for
end procedure
```

from rounding problems. This behaviour can be avoided with Kahan summation algorithm [118] or in a much simpler way using double for internal calculations. Our algorithm uses the second alternative. For each implementation, we have set $\epsilon = 2.2204460492503131 \times 10^{-16}$ for our algorithm.

Therefore, the routine is coded with ten loops (Algorithm 5). Figure 5.1a shows the computation decomposition diagram.

5.1.2 Multicore Implementation on CPU

As the codification of our routine presents a structure of nested *for*s and the calculation in the innermost loop is independent, we can accelerate the algorithm using threads on a multicore machine. To accomplish this, we have used the OpenMP technology [72]. OpenMP offers many compiler directives in the standard, but our algorithm needs *#pragma omp parallel for* to launch the threads on different cores transparently and keep the scalability when new hardware is available. Apart from this, we need to declare some private variables to assure that the summation process is thread-safe *private*(ω_{Tot},ωL_{Tot}).

The *parallel for* compiler directive is coded at the outermost loop line 2 (Algorithm 6) because thread context switches in CPU when no necessary reduces performance. In this configuration, a thread computes several voxels generating the proper MRI-CT synthesis output. Machines with simultaneous multithreading (Hyper-Threading) capabilities can set the number of threads as twice much time as the number of cores in the CPU and obtain in theory around 30% improvement [167].

Algorithm 6 Patch-based MRI-CT synthesis approach CPU/C++ OpenMP

```
1: procedure PATCHBASED($I$,$\mathcal{A}$,$S$,$\hat{\sigma}$,$\beta$,$\mathcal{N}$)
2:     #pragma omp parallel for private($\omega_{Tot}$,$\omega L_{Tot}$)
3:     for all $\mathbf{x}$ in $I$ do
4:         $\omega_{Tot} = 0$
5:         $\omega L_{Tot} = 0$
6:         for all $i$ in $\mathcal{A}$ do
7:             for all $\mathbf{y}$ in $\mathcal{N}_{\mathbf{x}}$ do
8:                 for all $x'$ in $\mathcal{P}^{I}_{\mathbf{x}}$ and $y'$ in $\mathcal{P}^{\mathcal{I}^i}_{\mathbf{y}}$ do
9:                     $\omega_i(\mathbf{x},\mathbf{y}) = \omega_i(\mathbf{x},\mathbf{y}) + (I(\mathbf{x}') - \mathcal{I}^i(\mathbf{y}'))^2$
10:                end for
11:                $\omega_i(\mathbf{x},\mathbf{y}) = \omega_i(\mathbf{x},\mathbf{y})/2S\beta\hat{\sigma}^2$
12:                $\omega_{Tot} = \omega_{Tot} + \omega_i(\mathbf{x},\mathbf{y})$
13:                $\omega L_{Tot} = \omega L_{Tot} + \omega_i(\mathbf{x},\mathbf{y}) \cdot L^i(\mathbf{y})$
14:            end for
15:        end for
16:        if $\omega L_{Tot} < \epsilon$ and $\omega_{Tot} < \epsilon$ then
17:            $l_x = NaN$
18:        else
19:            $l_x$ using (Eq. 3.69)
20:        end if
21:    end for
22: end procedure
```

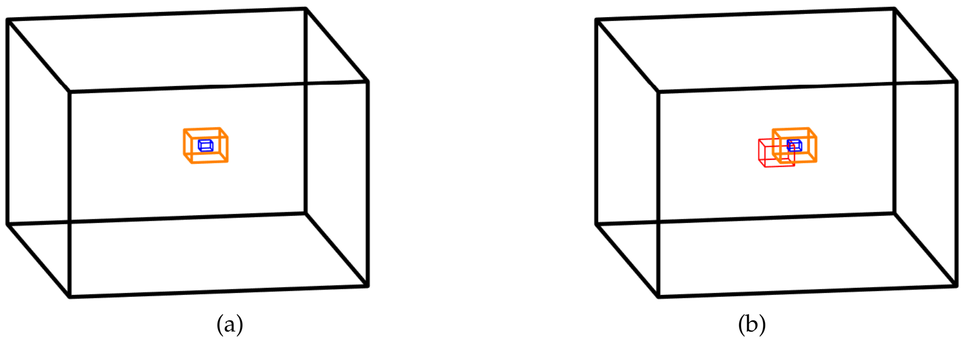


Figure 5.1: Computation decomposition: (a) On CPU the black cube represents the entire input volume, the orange cube represents the neighbourhood $\mathcal{N}_{\mathbf{x}}$ (K×K×K) and the blue cube is the considered image patch $I(\mathcal{P}_{\mathbf{x}})$ (P×P×P). (b) On GPU a new element (block of threads in red) has been added to the decomposition.

5.1.3 GPU Implementation

The first implementation in the GPU computing with NVIDIA CUDA model eliminates the first block of loops see line 2 (Algorithm 5). The kernel launches a 3D grid, which will be executed as many threads in parallel as the hardware allows (according to the resources each thread needs). Notice that voxels $\mathbf{x}$ are threads in the CUDA grid. Our CUDA kernel contains the calculation needed for performing a voxel from MRI-CT synthesis output. Therefore, threads calculate each voxel output independently. Algorithm 7 shows the pseudocode of a preliminary implementation only using device global memory (called *CUDA-GPU-GM* standing *Global Memory*). Figure 5.1b shows the computation decomposition diagram.

Algorithm 7 Patch-based MRI-CT synthesis kernel (CUDA-GPU-GM)

```
1:  procedure PATCHBASEDKERNEL($I$,$\mathcal{A}$,$S$,$\hat{\sigma}$,$\beta$,$\mathcal{N}$)
2:      $\omega_{Tot} = 0$
3:      $\omega L_{Tot} = 0$
4:      for all $i$ in $\mathcal{A}$ do
5:          for all $\mathbf{y}$ in $\mathcal{N}(\mathbf{x})$ do
6:              for all $\mathbf{y}$ in $\mathcal{N}_{\mathbf{x}}$ do
7:                  for all $x'$ in $\mathcal{P}^{I}_{\mathbf{x}}$ and $y'$ in $\mathcal{P}^{\mathcal{I}^i}_{\mathbf{y}}$ do
8:                      $\omega_i(\mathbf{x},\mathbf{y}) = \omega_i(\mathbf{x},\mathbf{y}) + (I(\mathbf{x}') - \mathcal{I}^i(\mathbf{y}'))^2$
9:                  end for
10:                 $\omega_i(\mathbf{x},\mathbf{y}) = \omega_i(\mathbf{x},\mathbf{y})/2S\beta\hat{\sigma}^2$
11:                 $\omega_{Tot} = \omega_{Tot} + \omega_i(\mathbf{x},\mathbf{y})$
12:                 $\omega L_{Tot} = \omega L_{Tot} + \omega_i(\mathbf{x},\mathbf{y}) \cdot L^i(\mathbf{y})$
13:             end for
14:         end for
15:     end for
16:     if $\omega L_{Tot} < \epsilon$ and $\omega_{Tot} < \epsilon$ then
17:         $l_x = NaN$
18:     else
19:         $l_x$ using (Eq. 3.69)
20:     end if
21: end procedure
```

This first implementation accesses global memory frequently to perform the calculations. Furthermore, the cache system on the GPU does not deal well with this memory access pattern in 3D (it is not coalesced in the CUDA terminology). We can draw that there are two types of accesses in our algorithm for performing a voxel calculation:

1. Input image patch: For a specific voxel during the synthesis process, we need to load $A \times K^3 \times P^3$ memory addresses from the input volume for the similarity patch calculation. These values are used repeatedly in the same thread to calculate each atlas-patch similarity. We can assume that inside the kernel these accesses are fixed.

2. Atlas patches: we also need to load $A \times K^3 \times P^3$ memory addresses from the atlas volume to calculate the similarity and weight for each voxel against each atlas. However, other threads in the grid load similar memory addresses with a slight offset.

Algorithm 8 Patch-based MRI-CT synthesis kernel (CUDA-GPU-GM2)

```
1:  procedure PATCHBASEDKERNEL($I$,$\mathcal{A}$,$S$,$\hat{\sigma}$,$\beta$,$\mathcal{N}$)
2:      $\omega_{Tot} = 0$
3:      $\omega L_{Tot} = 0$
4:      $localArray \leftarrow$ globalMemory($\mathcal{P}_{\mathbf{x}}^{I}$)
5:      for all $i$ in $\mathcal{A}$ do
6:          for all $\mathbf{y}$ in $\mathcal{N}_{\mathbf{x}}$ do
7:              for all $x'$ in $\mathcal{P}_{\mathbf{x}}^{I}$ and $y'$ in $\mathcal{P}_{\mathbf{y}}^{\mathcal{I}^i}$ do
8:                  $\omega_i(\mathbf{x},\mathbf{y}) = \omega_i(\mathbf{x},\mathbf{y}) + (localarray(\mathbf{x}') - \mathcal{I}^i(\mathbf{y}'))^2$
9:              end for
10:             $\omega_i(\mathbf{x},\mathbf{y}) = \omega_i(\mathbf{x},\mathbf{y})/2S\beta\hat{\sigma}^2$
11:             $\omega_{Tot} = \omega_{Tot} + \omega_i(\mathbf{x},\mathbf{y})$
12:             $\omega L_{Tot} = \omega L_{Tot} + \omega_i(\mathbf{x},\mathbf{y}) \cdot L^i(\mathbf{y})$
13:         end for
14:     end for
15:     if $\omega L_{Tot} < \epsilon$ and $\omega_{Tot} < \epsilon$ then
16:         $l_x = NaN$
17:     else
18:         $l_x$ using (Eq. 3.69)
19:     end if
20: end procedure
```

According to these accesses, we can do two incremental optimizations to the first implementation. To reduce the stalls producing by reading the patch input image during the label calculation, we can load this patch in a local array of each thread at the beginning because the patch is relatively small $P = 3$ (3×3×3) and its indexes are fixed in a voxel output. NVIDIA CUDA compiler (nvcc) is responsible for loading this data directly onto registers. Registers are the fastest memory accesses in the graphics card because they are close to the processor. With this optimization, we let the compiler treat the patch information like registers, which have the fastest access (Algorithm 8).

It is also important to emphasize that there are some limitations in the use of registers and they are governed by the number of threads per block. Therefore, the more threads we configure in the block, the fewer registers we can use in the kernel. This optimization version is called (CUDA-GPU-GM2).

The second optimization is based on shared memory (Figure 5.2a). We do not have enough registers to load the whole $\mathcal{N}_{\mathbf{x}}$ as we did previously for the patch. Furthermore, the elements in $\mathcal{N}_{\mathbf{x}}$ are used by other threads ($\mathcal{N}_{\mathbf{x}} \cap \mathcal{N}_{\mathbf{y}} \neq \emptyset$ if $|\mathbf{x} - \mathbf{y}|_2 < Radius_{\mathcal{N}}$) and it could be beneficial to have fast access for both. Therefore, we should store as many elements in the shared memory to calculate a synthesis output for each thread block. It means that we must assure that each thread within a block can calculate a new voxel $l_{\mathbf{x}}$ (Eq. 3.69) having fast access to needed elements.

It is impossible to load all the elements at once because the amount of memory is considerably large and we must do this for each atlas in the dictionary lines 6-7 (Algorithm 9). Therefore, we need to partition the data into subsets (tiles) such that each one fits into the shared memory to compute a single voxel in a number of steps (Algorithm 9).

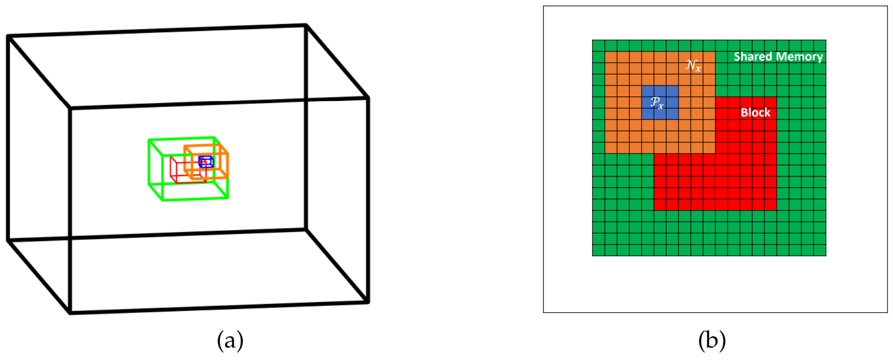


Figure 5.2: (a) Computation decomposition for CUDA-GPU-SM: (a) the black cube represents the entire input volume, the green cube represents the CUDA shared memory which a thread block needs (in red). Finally, the orange cube represents the neighbourhood $\mathcal{N}_\mathbf{x}$ and the blue cube represents the considered image patch $I(\mathcal{P}_\mathbf{x})$ (P×P×P voxels) from input image I. (b) Projection of (a) onto the Y plane.

Each thread has the task to load a part of the needed memory collaboratively. Therefore, the computation cannot start unless all the threads have loaded their part line 6 what we accomplish with a synchronization barrier in line 7 (Algorithm 9). In the same sense, the calculation for the next atlas cannot start if the previous one has not finished yet (synchronization in line 16). Global memory accesses are replaced by shared memory improving the overall performance of the algorithm substantially. The size of shared memory depends on the block grid, radius of the neighbourhood, and radius of the window patch.

$$SharedMemory = (Block_{Dim} + 2 \times Radius_{\mathcal{N}} + 2 \times Radius_{Patch})^3$$

Figure 5.2b illustrates the elements involved in the last optimization. Each cell represents a thread in the grid, the green part represents the whole shared memory, the red cells are the block of threads, the orange part is the neighbourhood, and the blue part is the patch.

5.2 Results

5.2.1 Platforms

First, we shall describe the platforms used as well as the way to refer to them along this section. The naming conventions used in the experimental results are as follows:

- The name convention consists of three names with a dash separator between each name, for instance, CUDA-K40C-SM.
- The first name refers to the type of platform: CPU stands for the implementation on CPU and CUDA stands for the implementation of the algorithm with NVIDIA toolkit on GPU.
- The second name refers to the physical device where experiments are conducted:

Algorithm 9 Patch-based MRI-CT synthesis kernel (CUDA-GPU-SM)

```
procedure PATCHBASEDKERNEL($I$,$\mathcal{A}$,$S$,$\hat{\sigma}$,$\beta$,$\mathcal{N}$)
    $\omega_{Tot} = 0$
    $\omega L_{Tot} = 0$
    $localArray \leftarrow$ globalMemory($\mathcal{P}_I(\mathbf{x})$)
    for all $i$ in $\mathcal{A}$ do
        $sh_i \leftarrow$ load data onto shared memory from $I^i(N_{\mathbf{x}})$
        $sync$
        for all $\mathbf{y}$ in $\mathcal{N}_{\mathbf{x}}$ do
            for all $x'$ in $\mathcal{P}^I_{\mathbf{x}}$ and $y'$ in $\mathcal{P}^{\mathcal{I}^i}_{\mathbf{y}}$ do
                $\omega_i(\mathbf{x},\mathbf{y}) = \omega_i(\mathbf{x},\mathbf{y}) + (localArray(\mathbf{x}') - sh_i(\mathbf{y}'))^2$
            end for
            $\omega_i(\mathbf{x},\mathbf{y}) = \omega_i(\mathbf{x},\mathbf{y})/2S\beta\hat{\sigma}^2$
            $\omega_{Tot} = \omega_{Tot} + \omega_i(\mathbf{x},\mathbf{y})$ using (Eq. 3.68)
            $\omega L_{Tot} = \omega L_{Tot} + \omega_i(\mathbf{x},\mathbf{y}) \cdot L^i(\mathbf{y})$
        end for
        $sync$
    end for
    if $\omega L_{Tot} < \epsilon$ and $\omega_{Tot} < \epsilon$ then
        $l_x = NaN$
    else
        $l_x$ using (Eq. 3.69)
    end if
end procedure
```

- **Xeon v3**: Intel Xeon E5-1650v3, 3.5 GHz hexa-core processor, from the 2014 Intel Haswell architecture (Haswell-EP), 1.5MB L2 cache, and 15MB L3 cache with 64GB DDR3 RAM [128]. The CPU uses a Microsoft Windows Server 2012R2 as the operating system.
- **Xeon v4**: Intel Xeon E5 2698v4 2.2 GHz 20 cores [129]. The CPU uses a Ubuntu 18.04.4 LTS (Bionic Beaver) `https://releases.ubuntu.com/18.04/` Desktop Linux as operating system.
- **K40C**: The Tesla K40c professional platform (with the NVIDIA Kepler GK110 architecture supporting CUDA compute capability 3.5) with 12GB RAM GDDR5, with 15 streaming multiprocessors of 192 CUDA cores each giving a total of 2880 CUDA cores. 5.046 TFlops [201].
- **DGX**: DGX station (with the NVIDIA Volta 4 Tesla V100 architecture supporting CUDA compute capability 7.0) 480 TFlops (GPU FP16), GPU Memory 64 GB total NVIDIA Tensor Cores 2560 NVIDIA CUDA© Cores 20480 hosted on Xeon v4 [199].

• The final name is the software implementation choice: No threads (1C), with threads using OpenMP (6C) and the number of threads is 12, global memory in CUDA (GM), global memory local array in CUDA (GM2), and CUDA using shared memory (SM).

We have implemented the code with Visual Studio 2015 Community using the Intel Compiler C++ 16.0, OpenMP 2.0, and CUDA 6.5.

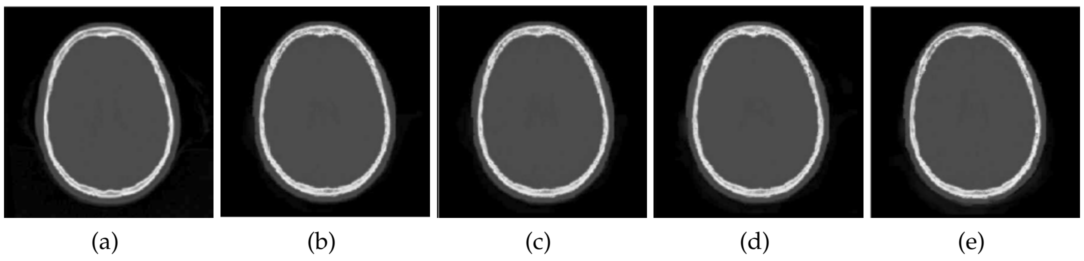


Figure 5.3: Ground truth CT and pseudo-CT volumes synthesized using atlas of different sizes and a neighbourhood of size $11 \times 11 \times 11$. (a) CT Ground Truth (b) 18 Atlas pseudo-CT (c) 13 Atlas pseudo-CT (d) 8 Atlas pseudo-CT (e) 3 Atlas pseudo-CT.

5.2.2 Dataset

MRI-CT Dataset

The MRI-CT dataset contains 19 MRI-CT volume pairs from healthy volunteers. A General Electric Signa HDxt 3.0T MR scanner using the body coil for excitation and an 8-channel quadrature brain coil for reception was used to acquire the head MRI volumes. Subjects were positioned supine. Imaging was performed using an isotropic 3DT1w SPGR sequence with a TR=10.024ms, TE=4.56ms, TI=600ms, NEX=1, acquisition matrix=288 $\times$ 288, resolution=1 $\times$ 1 $\times$ 1 mm, flip angle=12°.

CT images were acquired on a Siemens Somatom Sensation 16 CT scanner (Siemens, Erlangen) with matrix=512 $\times$ 512, resolution=0.48 $\times$ 0.48mm, slice thickness=0.75mm, PITCH=0.7mm, acquisition angle=0°, voltage=120kV, radiation intensity=200mA.

MRI T2-T1 Dataset

The MRI T2-T1 dataset contains 8 MRI T2 and T1 volume pairs from healthy volunteers. MRI images of the head were acquired on a General Electric Signa HDxt 3.0T MR scanner using the body coil for excitation and an 8-channel quadrature brain coil for reception. Subjects were positioned supine. Imaging was performed using an isotropic 3DT1w SPGR sequence with a TR = 10.944 ms, TE = 4.776 ms, TI = 600 ms, NEX = 1, acquisition matrix = 288 $\times$ 288, resolution= 1 $\times$ 1 $\times$ 1 mm, flip angle = 12°, and 3DT2w CUBE sequence with a TR =3000 ms, TE = 90.341 ms, TI = 0ms, NEX = 1, acquisition matrix = 256 $\times$ 256, resolution= 1 $\times$ 1 $\times$ 1 mm, flip angle = 90°.

Image Pre-processing

After acquiring the images, there is a pre-processing step using 3D Slicer built-in modules. This pre-processing is composed of MRI bias correction (N4 ITK MRI bias correction), rigid registration (general registration BRAINS) to align all the images, and normalization of the gray-scale values (ITK-based histogram matching).

5.2.3 Qualitative Results

Synthesis experiments were performed with the MRI-CT database, using different neighbourhood sizes and a different number of volumes in the atlas, to evaluate the

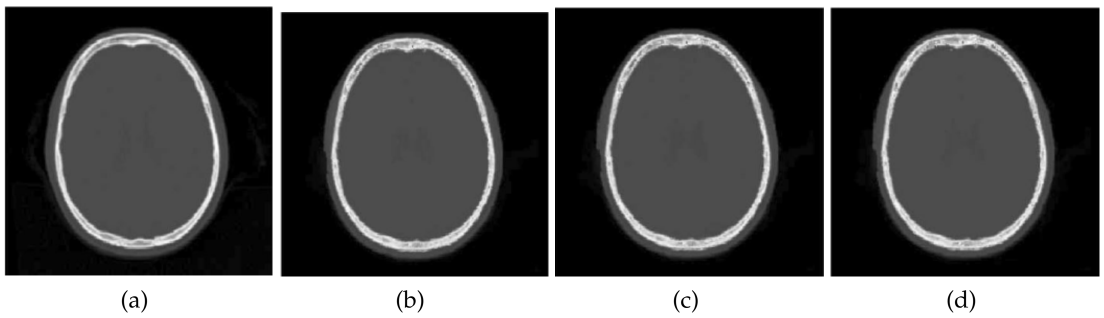


Figure 5.4: Ground truth CT and pseudo-CT volumes synthesized using different neighbourhood sizes with an atlas of 10 volumes: (a) Ground truth CT (b) $11 \times 11 \times 11$ pseudo-CT (c) (b) $9 \times 9 \times 9$ pseudo-CT (c) $7 \times 7 \times 7$ pseudo-CT.

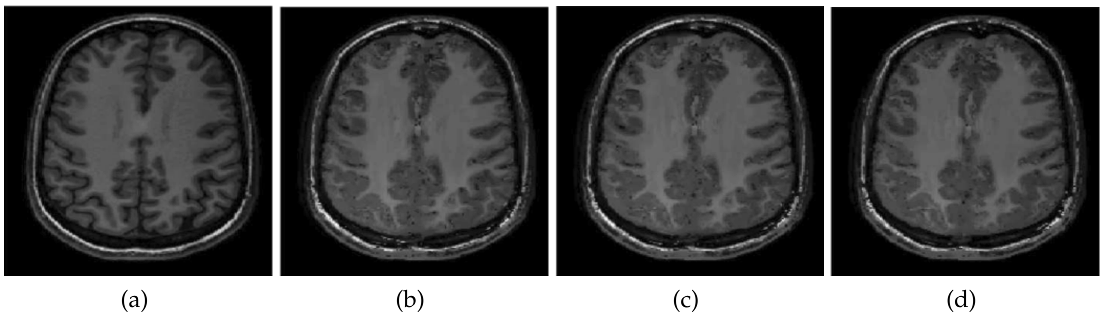


Figure 5.5: T1 MRI image generation (pseudo-T1) from T2 MRI images using different neighbourhood sizes. (a) Ground Truth T1 (b) $11 \times 11 \times 11$ pseudo-T1 (c) $9 \times 9 \times 9$ pseudo-T1 (d) $7 \times 7 \times 7$ pseudo-T1.

visual quality of the results. First, we show the accuracy of the algorithm to generate the synthesis image. A second goal is to prove that the method can produce good results with the reduced size of the atlas and neighbourhood. In this sense, we have set three neighbourhoods where the likelihood of finding similarities decreases ($11 \times 11 \times 11$, $9 \times 9 \times 9$, $7 \times 7 \times 7$). Furthermore, we have prepared a wide range of atlas sizes (1-18) to be able to determine when the improvement does not increase any further. For all our experiments, we have fixed $P = 3$, $\beta = 1$ and $\sigma = 1$. The volumes presented in these experiments are always $222 \times 222 \times 112$ voxels.

Figure 5.3 shows the ground truth and the resulting estimations for several atlas sizes, using a neighbourhood of size $11 \times 11 \times 11$. Figure 5.4 shows the ground truth and the resulting estimations with different sizes of the neighbourhood, using an atlas of 10 volumes. The comparison between the patient-specific CT and the pseudo-CT shows how our method can approximate the ground truth, delimiting the skull contours and differentiating air from the bone.

Figure 5.5 shows the results of applying the pipeline to obtain T1 MRI images from T2 acquisitions. In this case, the volumes are also very well defined. Enlarging the neighborhood allows us to capture some more detail in the image, but for our applications, all the reconstructions seem good enough.

Table 5.1: NCC results for different neighbourhoods and atlases as a quantitative measure of the registration.

	Neighbourhood Size		
# atlases	7×7×7	9×9×9	11×11×11
1	0.8918 ± 0.0242	0.9020 ± 0.0209	0.9055 ± 0.0194
2	0.9070 ± 0.0116	0.9144 ± 0.0088	0.9174 ± 0.0076
3	0.9169 ± 0.0087	0.9219 ± 0.0077	0.9236 ± 0.0075
4	0.9196 ± 0.0087	0.9236 ± 0.0078	0.9252 ± 0.0077
5	0.9235 ± 0.0072	0.9269 ± 0.0067	0.9281 ± 0.0066
6	0.9254 ± 0.0060	0.9287 ± 0.0056	0.9302 ± 0.0056
7	0.9257 ± 0.0052	0.9284 ± 0.0050	0.9294 ± 0.0051
8	0.9279 ± 0.0050	0.9304 ± 0.0048	0.9313 ± 0.0047
9	0.9281 ± 0.0053	0.9305 ± 0.0053	0.9313 ± 0.0052
10	0.9292 ± 0.0049	0.9314 ± 0.0047	0.9324 ± 0.0048
11	0.9301 ± 0.0048	0.9322 ± 0.0046	0.9331 ± 0.0048
12	0.9302 ± 0.0049	0.9322 ± 0.0050	0.9329 ± 0.0052
13	0.9306 ± 0.0044	0.9326 ± 0.0046	0.9332 ± 0.0048
14	0.9317 ± 0.0048	0.9334 ± 0.0048	0.9340 ± 0.0048
15	0.9317 ± 0.0049	0.9334 ± 0.0049	0.9341 ± 0.0051
16	0.9321 ± 0.0052	0.9338 ± 0.0051	0.9343 ± 0.0051
17	0.9325 ± 0.0046	0.9341 ± 0.0047	0.9346 ± 0.0047
18	0.9327 ± 0.0047	0.9344 ± 0.0048	0.9349 ± 0.0049

5.2.4 Quantitative Results

To obtain a quantitative measure of the quality of the synthesized volumes, we use the normalized cross-correlation (NCC) as a metric of similarity. The normalized cross-correlation between image I_1 and I_2 reads:

$$NCC = \frac{1}{N} \sum_{\mathbf{x} \in \Omega} \frac{(I_1(\mathbf{x}) - \mu_1)(I_2(\mathbf{x}) - \mu_2)}{\sigma_1 \sigma_2} \tag{5.1}$$

Using the MRI-CT database, we tested the effect of the size of the atlas on the quality of the image reconstruction. Figure 5.6 illustrates the result of the mean correlation of the reconstructed images of the complete database, for three sizes of the neighbourhood. As can be seen, the mean NCC plateaus when more than 5 volumes are present in the atlas. It is worth noting that correlation measures are very high, even in the worst-case scenario (1 volume in the atlas, neighbourhood of $7 \times 7 \times 7$).

Table 5.1 shows the detailed results of a leave-one-out experiment. Taking every image pair in the atlas in turn, we synthesized the CT using an increasing number of the remaining volumes as an atlas. The table shows the mean NCC and its standard deviation of the resulting image for every neighbourhood size and some atlas size.

Table 5.2: Computational time (in seconds) using different neighbourhood size ($\mathcal{N}$) and number of atlases (#) for the tested platforms: CPU-XeonV3-1C, CPU-XeonV3-6C, CUDA-K40C-GM, CUDA-K40C-GM2, and CUDA-K40C-SM (Section 5.2.1). Speedups against the single core solution in parentheses. For CUDA-K40C-SM times are the average of 100 executions. For an input image with 222×222×112 voxels.

		CPU		CUDA		
$\mathcal{N}$	#	**XeonV3-1C**	**XeonV3-6C**	**K40C-GM**	**K40C-GM2**	**K40C-SM**
11	1	593.6	123.0 (×4.8)	18.4 (×32.2)	11.1 (×53.5)	7.8 (×76.5)
×	4	2385.1	494.1 (×4.8)	73.5 (×32.5)	44.2 (×54.0)	30.9 (×77.2)
11	6	3334.1	660.3 (×5.0)	110.3 (×30.2)	66.2 (×50.4)	46.4 (×71.9)
×	10	5544.0	1080.9 (×5.1)	183.7 (×30.2)	110.5 (×50.2)	77.4 (×71.6)
11	14	7832.9	1538.1 (×5.1)	257.3 (×30.4)	154.8 (×50.6)	108.5 (×72.2)
	18	10406.1	2350.7 (×4.4)	330.8 (×31.5)	199.0 (×52.3)	139.6 (×74.6)
9	1	323.6	60.5 (×5.4)	10.7 (×30.1)	6.58 (×49.9)	3.28 (×98.6)
×	4	1305.5	242.2 (×5.4)	42.7 (×30.6)	25.7 (×50.8)	17.3 (×75.5)
9	6	1971.8	363.0 (×5.4)	63.9 (×30.9)	38.6 (×51.1)	26.0 (×76.0)
×	10	3299.1	610.7 (×5.4)	106.6 (×30.9)	64.3 (×51.3)	43.3 (×76.3)
9	14	4645.6	849.8 (×5.4)	149.3 (×31.1)	90.10 (×51.5)	60.5 (×76.6)
	18	5974.3	1113.8 (×5.3)	191.8 (×31.1)	115.9 (×51.5)	77.91 (×76.6)
7	1	158.1	29.6 (×5.4)	5.20 (×30.4)	3.16 (×50.0)	2.12 (×74.5)
×	4	636.1	117.2 (×5.4)	20.5 (×31.0)	12.4 (×51.3)	8.23 (×77.3)
7	6	955.1	176.7 (×5.4)	30.8 (×31.0)	18.6 (×51.4)	12.3 (×77.6)
×	10	1612.4	342.2 (×4.7)	51.3 (×31.5)	30.9 (×52.1)	20.5 (×78.7)
7	14	2278.7	475.2 (×4.7)	71.7 (×31.7)	43.2 (×52.6)	28.6 (×79.4)
	18	2933.5	532.9 (×5.5)	92.3 (×31.7)	55.6 (×52.7)	36.8 (×79.5)

5.2.5 Computational Time

Figure 5.7a shows timing results (in seconds in *log* scale) for different number of atlases and problem configurations (neighbourhoods of $11 \times 11 \times 11$, $9 \times 9 \times 9$, and $7 \times 7 \times 7$ voxels) using different platforms. The XeonV3-1C shows the single-threaded CPU version performance and the XeonV3-6C is a 12-threaded CPU version, the K40C-GM is the basic GPU version using only CUDA global memory with a grid configuration of $8 \times 8 \times 10$, the K40C-GM2 is the global memory GPU version, but using registers for accessing the image patch. This version keeps the same grid configuration as before. Finally, K40C-SM is the shared memory GPU version with a grid configuration of $10 \times 10 \times 10$. Figure 5.7b shows the speedup among versions against the base XeonV3-1C.

Table 5.2 displays the computational time in seconds for all configurations. We can draw that the multithreaded solution **XeonV3-6C** achieves a speedup of ×5 over

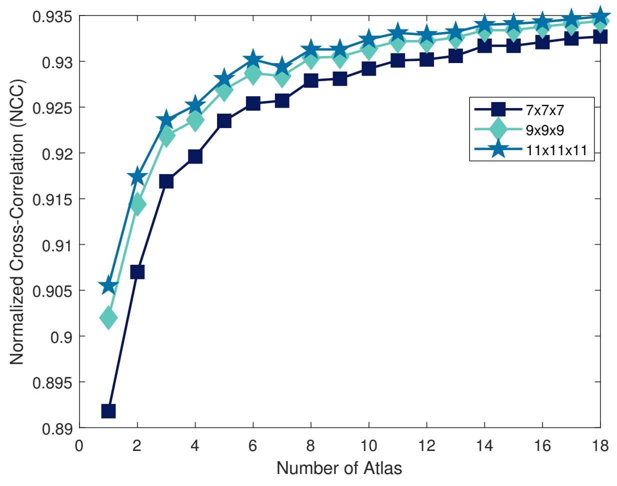


Figure 5.6: Mean normalized cross-correlation measures of synthesized images compared with the ground truth for the MRI-CT database, for a different number of volumes in the anatomy atlas and different sizes of the neighbourhood.

the single core configuration. This improvement is more notable with the first implementation on GPU **K40C-GM** reaching $\sim \times 30$ when compared with the implementation with a single core. The first optimization **K40C-GM2** on GPU improves $\times 1.67$ in comparison with the initial implementation on GPU and accumulated a speedup of $\times 50$. The solution using shared memory **K40C-SM** has gained almost another $\times 1.42$ relative to the previous version and with a $\times 74.6$ for the most intensive computation 18 atlases and $11 \times 11 \times 11$ neighbourhood.

A real-time solution for this synthesis problem can be achieved by relaxing the problem configuration. In this sense, a neighbourhood size of $7 \times 7 \times 7$ and 4 atlases, we obtain 10.6 minutes using a single core solution (**XeonV3-1C**) and almost 2 minutes in the multi-threaded version (**XeonV3-6C**) to 8.23 seconds using the shared memory version of the CUDA solution (**K40C-SM**). It is important to highlight that in certain applications with this configuration, we generate images that meet the quality expectations of practitioners.

5.2.6 Computational Time Update

The times presented in the section above are from the paper [9]. Furthermore, we have compiled the code for Intel C++ 19.0 on the platform XeonV3 (previously Intel C++ 16.0) and with CUDA 9.0 for GPU. We have also included the results of the GPU V100 (DGX with XeonV4) with CUDA 9.0 (previously CUDA 6.5) (Figure 5.7c). Table 5.3 compares the results from the new platform for GPU with the previous platform updated with compiler and software.

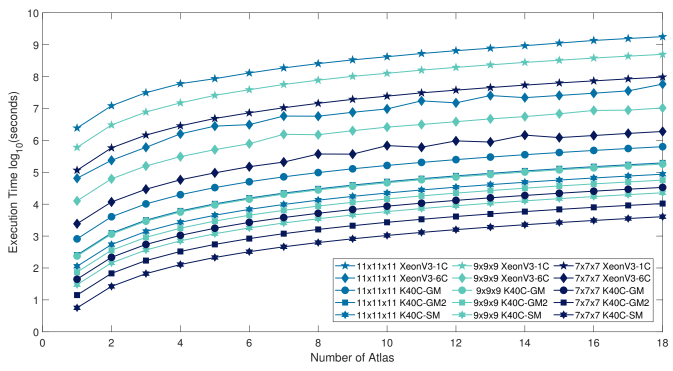
The new platform, DGX, has a CPU (XeonV4) with a slower clock cycle (2.2 GHz) than the previous platform XeonV3 (3.5 GHz), see Section 5.2.1. In this sense, a single core solution (sequential) runs faster on XeonV3 than XeonV4 platform. Therefore, we present the new results with the new compiler on XeonV3 as the best results for the sequential configuration. When comparing both tables 5.2 and 5.3, we can observe that the single core solution (**XeonV3-1C**) has been improved by $\times 1.60$. Multicore solution on the platform XeonV3 performs better than on XeonV4 because

Table 5.3: Computational time update (in seconds) using different neighbourhood size ($\mathcal{N}$) and number of atlases (#) for the tested platforms: CPU-XeonV3-1C, CPU-XeonV3-6C, CUDA-K40C-SM, CUDA-DGX-SM (Section 5.2.1). Speedups against the single core solution in parentheses. For CUDA-K40C-SM and CUDA-DGX-SM times are the average of 100 executions. For an input image with 222×222×112 voxels.

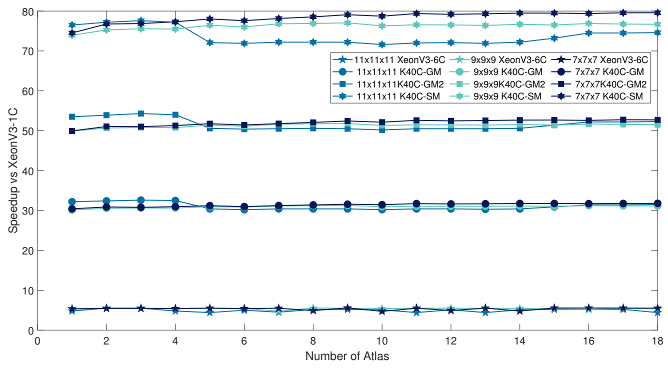
		CPU		CUDA	
$\mathcal{N}$	#	**XeonV3-1C**	**XeonV3-6C**	**K40C-SM**	**DGX-SM**
11	1	349.4	98.2 (×3.55)	6.38 (×54.8)	1.08 (×323.4)
×	4	1424.7	466.6 (×3.05)	25.29 (×56.3)	4.12 (×345.9)
11	6	2144.6	701.0 (×3.06)	37.96 (×56.5)	6.15 (×348.6)
×	10	3570.5	973.6 (×3.67)	63.34 (×56.4)	10.30 (×346.6)
11	14	5021.8	1626.5 (×3.09)	88.75 (×56.6)	14.35 (×350.0)
	18	6467.7	1762.0 (×3.67)	114.18 (×56.7)	18.43 (×351.0)
9	1	198.5	64.6 (×3.07)	3.59 (×55.3)	0.33 (×197.3)
×	4	804.7	217.9 (×3.69)	14.10 (×57.1)	1.17 (×186.8)
9	6	1223.3	398.1 (×3.07)	21.13 (×57.9)	1.72 (×231.3)
×	10	2045.4	538.7 (×3.80)	35.27 (×58.0)	2.86 (×188.6)
9	14	2868.4	764.2 (×3.75)	49.39 (×58.1)	3.97 (×192.6)
	18	3703.8	1214.3 (×3.05)	63.52 (×58.3)	5.09 (×238.8)
7	1	97.8	27.5 (×3.55)	1.77 (×55.3)	0.19 (×143.7)
×	4	389.1	107.6 (×3.62)	6.78 (×57.4)	0.60 (×178.7)
7	6	589.6	161.8 (×3.64)	10.13 (×58.2)	0.88 (×184.2)
×	10	1001.3	265.3 (×3.77)	16.83 (×59.5)	1.43 (×185.6)
7	14	1416.4	367.8 (×3.85)	23.54 (×60.2)	1.97 (×186.9)
	18	1820.3	582.8 (×3.12)	30.25 (×60.2)	2.52 (×231.3)

in addition to the clock cycle, the memory access in the method does not produce faster code when using more cores with OpenMP. The scalability of the method is confirmed because the **DGX-SM** solution performs the best among the solutions (Figure 5.7d) and it has gained ×6 against the best previous GPU solution (**K40C-SM**) and with a peak of ×351.0 when compared with the single core solution of the previous platform with compiler and software update in the worst scenario, 18 atlases and a neighbourhood 11 × 11 × 11.

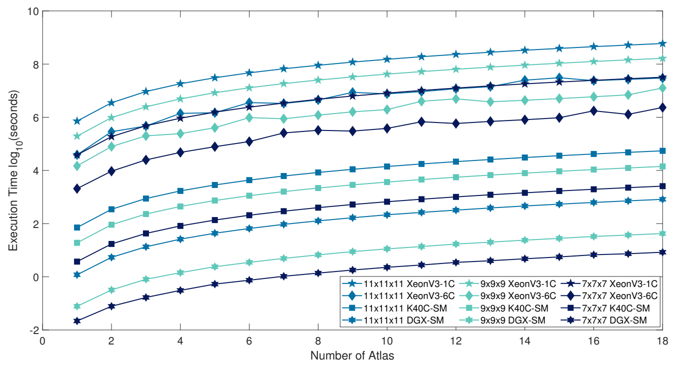
Indeed, if we review the hardware capability in CUDA for execution (Section 3.1.5), we can see that the CWD assigns blocks of threads on the vacated SM in the graphic card (Figure 3.5). Volta architecture has more SMs available [199] than Kepler [201], which, in turn, improves the performance substantially. In addition to that, there are other features in this architecture that speed up the overall algorithm (better cache system, divergence, thread warp execution model, etc.) (see for further details [204]).



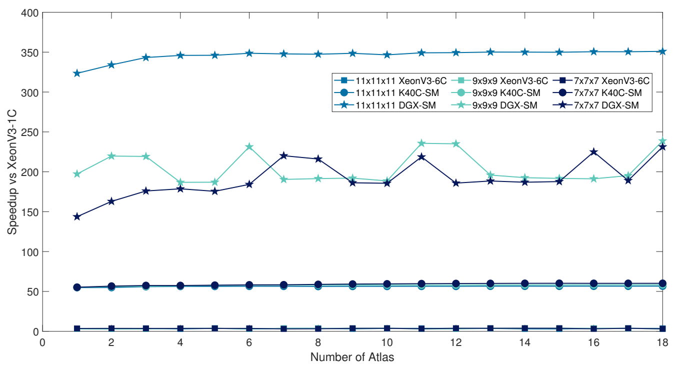
(a)



(b)



(c)



(d)

Figure 5.7: (a) Results of all configurations from [9] (b) speedup of all configurations from [9] (c) Update results with the new platform DGX and XeonV3 with new compiler and software (d) speedup comparison of the best update results and DGX.

5.3 Discussion

Firstly, the attenuation correction, what PET/MR multimodality needs, can be obtained with accuracy from the synthetic CT (see [271] for a quantitative evaluation of our method in the correction for PET images). Secondly, there is a minimization of patient ionization because our proposal eliminates the patient-specific CT acquisition while keeping a good CT estimation in comparison with the ground truth. Thirdly, the factor of reducing acquisition time as well as cost in equipment when using PET/MR is an important advantage because the synthesis images contain detailed information of the soft tissues.

The synthesis problem treated here is computationally intensive with a lot of data to process which suits perfectly for multicore and manycore programming paradigm (SIMD). Our implementation has achieved a remarkable speedup in comparison with a single core solution gaining $\times 74.6$ (**K40C-SM**) in the worst configuration published in [9]. Furthermore, we have revisited the algorithm with new hardware, updated software, and compiler, and another important speedup has been reached (Table 5.3) confirming the scalability of the method in new architectures.

A study has been conducted to find a compromise between accuracy and "real-time" in Medical Imaging. After our experiments, we can draw that a configuration with a neighbourhood of $9 \times 9 \times 9$ and 10 atlases is a good compromise to obtain good quality in the image (NCC = 0.93) (Eq. 5.1) and a computational burden of fewer than three seconds (**DGX-SM**), which is far lower than typical image reconstruction times in clinical MRI scanners. The neighbourhood determines the likelihood to find similarities at a specific voxel surrounding. In the anatomic variability of the structures that we study, it does not matter how we enlarge the neighbourhood after a specific limit ($11 \times 11 \times 11$), the image no longer improves because the matches are far away from where we search. The size of the atlas allows generalizing the variability of our samples. However, the inclusion of new atlases does not produce improvements because this information is already contained. Therefore, the generalization is completed at that point.

It is noteworthy that the results described here are for complete volumes. In specified applications, it may be necessary to compute only several slices, allowing to reduce even further the overall time.

Overall, the results are quite promising because the GPU implementation is capable of returning outputs in order of seconds with high image quality (NCC = 0.93). The results presented in [271] demonstrated that attenuation correction in PET images using patient-specific CT volume or our pseudo-CT volume produces similar quantitative results. In addition to this result, the short computational time via GPU is desirable property when PET/MR systems as a potential protocol for attenuation correction are considered. Finally, we have also demonstrated that the method can synthesise other modalities like T1 MRI images from T2 acquisitions.

Chapter 6

Conclusions and Future Work

In this chapter, we present the main contributions of this Thesis and provide the answers to the initial hypotheses stated in (Section 1.2). Furthermore, we indicate possible future work directions based on the research presented in this Thesis. Finally, we summarize the list of publications, in journals and conferences, as well as talks produced from this research.

6.1 Principal Contributions and Conclusions

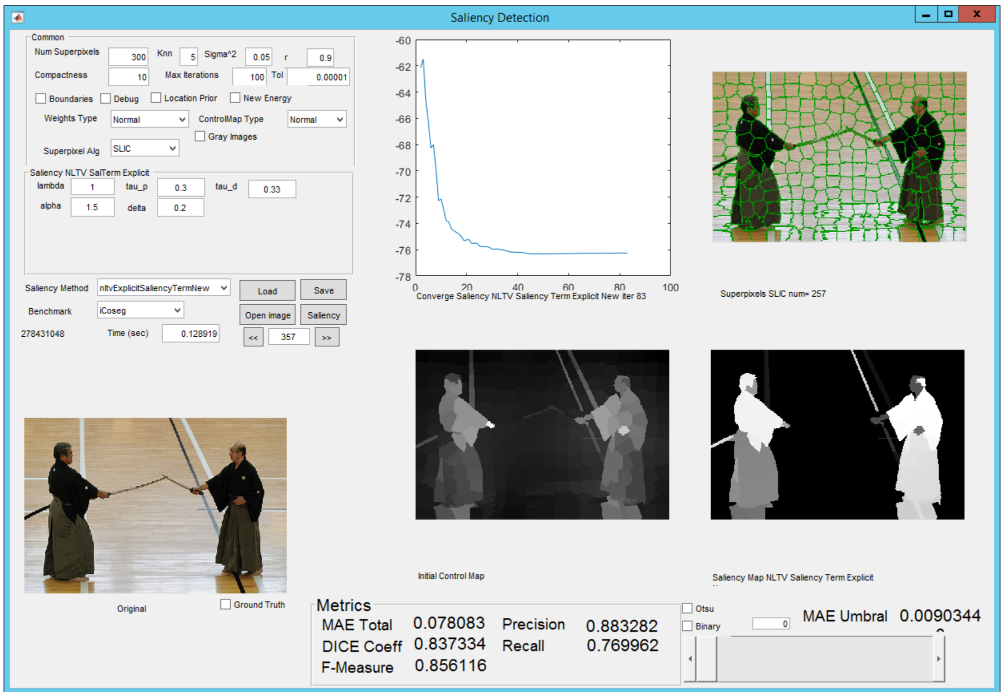
6.1.1 Principal Contributions

The contributions of this Thesis can be classified into 1) modelling and numerical resolution, 2) computational optimization. We have used classical techniques that have been proved to be powerful tools: variational and PDE methods, and patch-based methods. Furthermore, our methods are based on a non-local framework. Non-local approaches also typically introduce a highly intensive computation demand. To tackle this, we have proposed efficient computation solutions on CPU or GPU. Our source code is freely available on Github[1] so that the investigations presented in this Thesis can be used for further research.

Chapter 3 describes the methods proposed for both problems: saliency detection and MRI-CT synthesis. For the saliency detection problem, we have derived two novel formulations using Total Variation regularization in a non-local framework on graphs. The first one follows a pure Bayesian approach and the second one is based on modelling specific PDE dynamics. These dynamics are characterized by absorption-reaction flows, which allow separating salient and no salient regions as in a classification problem. An automatic binarization output is possible with this new formulation. Both formulations are convex but non-smooth. This can be dealt with a smooth primal-dual formulation which is efficiently solved with a primal-dual algorithm converging to a min-max, saddle point solution.

On the other hand, the patch-method is also discussed to synthesise CT images from MRI data as a natural way to correct PET images in PET/MR modality. We also describe the scalability of this method when computing in parallel architectures like GPUs. Therefore, we expect better improvements with better hardware or software. Furthermore, our proposal achieves an accurate estimation of a pseudo-CT with similar accuracy in comparison with the patient-specific CT and avoids the necessity of a registration step. Finally, the combination of short computation time and

[1]The saliency detection code `https://github.com/EduardoAlcainBallesteros/VariationalSaliencyDetection` and Code for the pseudo-CT problem `https://github.com/EduardoAlcainBallesteros/PseudoCTImaging`



Common

Num Superpixels	300	Knn	5	Sigma^2	0.05	r	0.9
Compactness	10	Max Iterations	100	Tol	0.00001		

☐ Boundaries ☐ Debug ☐ Location Prior ☐ New Energy

Weights Type: Normal — ControlMap Type: Normal — ☐ Gray Images

Superpixel Alg: SLIC

Saliency NLTV SalTerm Explicit

lambda	1	tau_p	0.3	tau_d	0.33
alpha	1.5	delta	0.2		

Saliency Method: nltvExplicitSaliencyTermNew

Benchmark: iCoseg

278431048 Time (sec) 0.128919

357

Metrics

MAE Total	0.078083	Precision	0.883282
DICE Coeff	0.837334	Recall	0.769962
F-Measure	0.856116		

☐ Otsu ☐ Binary 0 MAE Umbral 0.0090344

Figure 6.1: The user-friendly interface in Matlab for testing several variational saliency approaches. Some of the functionality provided is to load different datasets, display metrics, results, ground truth, etc.

an accurate estimation of the new modality makes the approach of much interest to clinical protocols.

Chapter 4 presents the implementation and results of the saliency problem. To reduce computation complexity, we have decreased the size of the problem with superpixels. Our solution is in the superpixel domain (graph). The resulting numerical algorithms have been implemented on CPU and GPU platforms. The computation of the non-local gradient performs efficiently after organising the data using CSR format for a sparse matrix-vector. We have also extended this concept for the computation of the non-local divergence to find the transposed elements in the compact matrix $W^{sp \times k}$ with a $\times 2$ speedup for this operation. The selection of the k-NN elements in the W matrix is an operation that fits well on CPU implementation. On the contrary, for the GPU platform, the same idea takes far more time than for the CPU platform. We have formulated this operation so that it can fit better in the GPU paradigm. The saliency term proposed has improved the results considerably according to the metrics (Precision, Recall, F-Measure, and MAE). It forces almost binary solutions in the features space while maintaining an equivalent computational complexity to our previous approach [10]. The algorithms are capable of real-time performance for both platforms.

We have also designed a user-friendly interface in Matlab for prototyping purposes (Figure 6.1). Images can be loaded from several datasets with ease. Different algorithms are rapidly compared through the quick evaluation of their metrics (MAE, Precision, Recall, F-Measure, and Dice coefficient). The parameters, which control the algorithms, can be modified in the user interface and the intermediate steps from the algorithms are displayed (superpixels extraction, control map, convergence, and the saliency map).

Chapter 5 presents the implementation and the results for the pseudo-CT from MR T1-weighted images problem. Our proposal has achieved good accuracy using the metric normalized cross-correlation (NCC = 0.935) when compared with the patient-specific CT. Computational intense and time consuming is one of the major drawbacks in algorithms using patch-based methods [56]. We have implemented the solution on CPU single core and multicore. Since each voxel computation is independent and very intensive in our patch-based method, OpenMP as a parallel programming language is a natural way to improve the algorithm on the CPU platform. To the best of our knowledge, we have implemented the first pure GPU solution for pseudo-CT synthesis. We have developed three configurations on the GPU. The shared memory configuration (CUDA-K40C-SM) has achieved $\sim \times 74$ speedup against the single core configuration. In the last part of this Thesis, we have run experiments on a much better platform (NVIDIA DGX Station with NVIDIA V100 GPUs). The speedup reached is $\sim \times 6$ against the best configuration confirming the scalability of the method and the use of parallel architectures for it.

After the main contributions have been presented, we shall now address the hypotheses formulated in this manuscript (Section 1.2).

Hypothesis One: *Variational methods can be successfully applied to the automatic saliency detection problem for natural images. This is based on the fact that variational methods have shown to be powerful modelling tools for many low-level image processing tasks providing, state-of-the-art results among classical, unsupervised methods. Based on the works of [179, 281] we then assume that, if conveniently modified, classical variational models can also address the saliency detection problem. Since the variational methods rely on unsupervised clues (contrary to the recent deep learning approaches based on data processing and training), in this work, we wish to explore the possibility to provide unsupervised, automatic segmentation (classification) state-of-the-art results.*

Our proposed models have achieved good results (Section 4.2) in comparison with other variational methods. The new saliency term introduced in our formulation has brought either qualitative or quantitative improvements while keeping the same computation complexity. The dynamics of the model promote binarization in the saliency maps. This simplifies the recovering of the salient regions (Figure 3.16c). In this sense, we can answer affirmatively to our hypothesis because we have introduced a key term in the variational formulation so that the salient discrimination has more control and removes the background.

However, deep learning approaches are the state-of-the-art methods for saliency detection in the datasets we considered. Saliency is a difficult concept to describe because it is highly dependent on the human interpretation of each scene. Deep learning in saliency approaches usually uses pre-trained networks for object classifications, localization, etc. via transfer learning. In this way, these approaches can aggregate semantic information so that the algorithms can discriminate whole objects in the images and mark them as salient (Figure 6.2c). We have proposed a model based on priors without a learning process (unsupervised) and this semantic interaction is complicated to capture (Figure 6.2d). New variational formulations

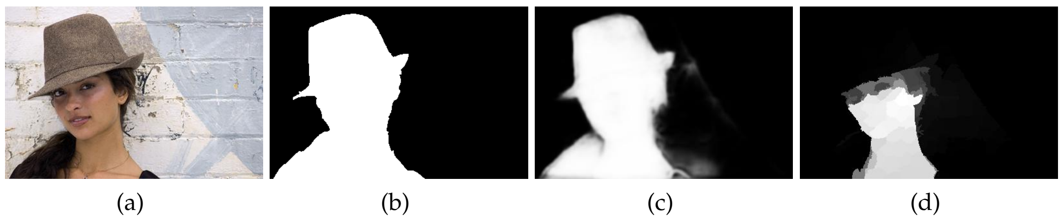


Figure 6.2: (a) Image from the public dataset MSRA10K [58] (b) Ground truth (c) saliency map by a deep learning approach from [290] (d) our approach. Notice that machine learning approach can take the hat in the saliency map output whereas our output eliminates it.

should include this semantic information via machine learning approaches.

Hypothesis Two: *Iterative methods such as the numerical schemes used to solve the proposed variational models introduce some challenges when optimizing them computationally. The combination of parallel techniques either on GPU or CPU and a numerical scheme that converges rapidly to the solution can be capable of real-time processing video.*

The iterative nature of the algorithm implies some difficulties in using parallel architectures. Although the operations of the algorithm are independent, it exhibits data dependencies among them in the primal-dual algorithm. Each operation in the solver, e.g., non-local gradient, divergence, and update does not have much data to compensate for the iterative nature. Apart from this, inside the method, many operations require reduction techniques (max, min, sum, etc.). Reduction operations do not perform well when not much data is available and this is particularly critical for GPU.

However, we have achieved a very optimized version (33 fps) on the CPU platform. In our solution for GPU, we have merged steps of the algorithm in the same kernels to reduce the time for kernel calls and implemented several reductions inside kernels. Our best GPU configuration has reached 62 fps, but we expect much better computational performance increasing the number of superpixels, which leads to the use of multi-resolution. Table 4.1 suggests that precision increases but not the recall. Combining different superpixel resolutions can be accomplished with GPU according to the computational times (Table 4.3). Furthermore, the primal-dual algorithm converges in a few iterations while keeping the accuracy (Figure 3.16d).

We can answer affirmatively nevertheless the results are not ideal for GPU where the reduction of data has been a handicap.

Hypothesis Three: *Patch-based methods in multi-atlas have been applied successfully as a segmentation technique [236]. They can also be applied to generate other image modalities like CT. In this sense, PET/MR modality can use synthetic CT for the attenuation correction. This approach can reach accuracy results in comparison to the patient-specific CT when using an atlas composed of human head images. In this sense, clinical protocols could incorporate these methods.*

We have proposed an algorithm based on multi-atlas using patches to synthesise a CT image from a set of MRI images. The extraction of attenuation maps can be accomplished from this synthetic CT and can correct PET images in the end for a PET/MR multimodality. Our solution has achieved good results in comparison with the gold standard for the human brain images that we have.

We can positively answer the hypothesis. We think based on the results in Chapter 5 that the proposal could be applied in clinical scenarios for PET/MR scanners such as Neurology allowing co-localization of lesions in the head or and neck, Oncology identifying tumours in the head, and Paediatrics to reduce radiation exposition for children in cancer therapy responses to name a few. Nevertheless, more tests must be conducted so that practitioners accept this solution.

Hypothesis Four: *Patch-based methods in multi-atlas consist of independent tasks. Using computing parallel techniques either on CPU or GPU can improve the performance substantially. Furthermore, some parameters in the patch-based algorithm can be relaxed to reduce time without losing accuracy.*

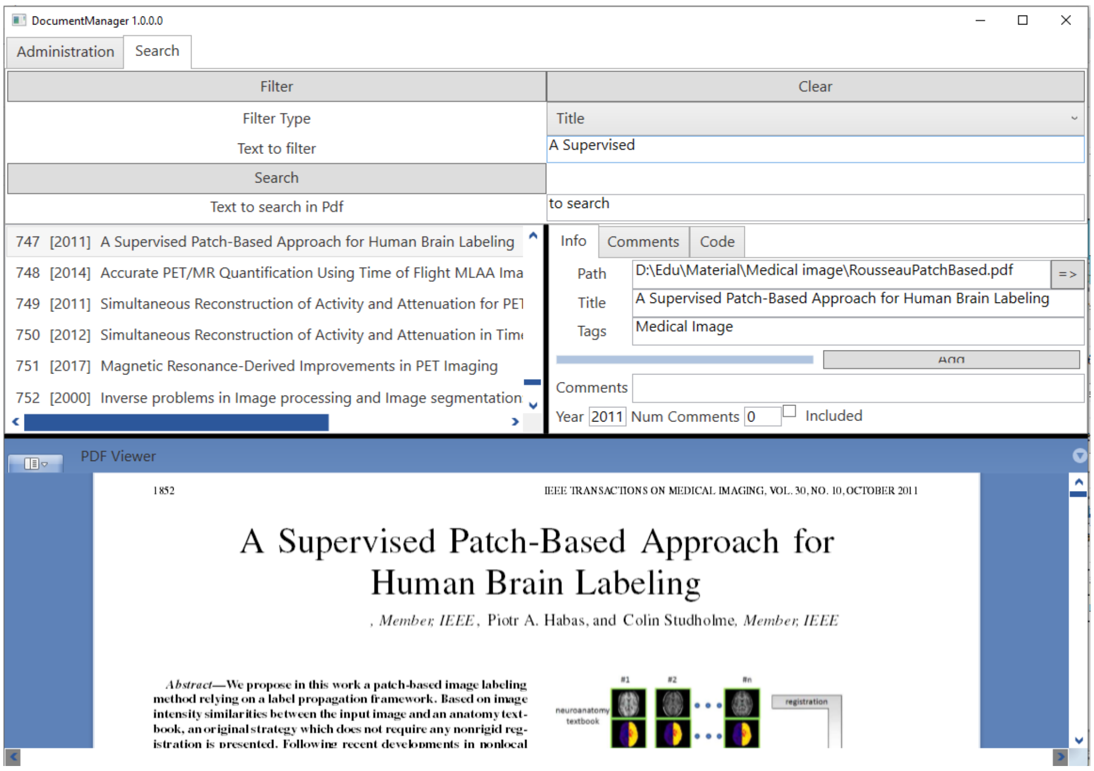
To address this hypothesis, we compare the time needed to create an image by a CT scanner and measure how far our pseudo-CT is either in time and accuracy.

However, it is not that simple because there are other fundamental aspects to consider. CT uses ionizing radiation, which is harmful to the body. Creating pseudo-CT eliminates this problem. Furthermore, the time for generating the real CT volume must consider the preparation (patient from one bed to another) and study procedure time. For these reasons, we think that our proposal would be of much interest if we were sufficiently close in NCC measurement (accuracy) to the real CT and the time were a bit faster than the whole process to acquire the CT volume.

One of the difficulties of using non-local approaches like ours is the high computational demand (remember the complexity of the algorithm $\mathcal{O}(A \times N^9)$). Although the method has a parallel nature, which has materialized in a speedup of two orders of magnitude higher against a single core implementation, the reduction of the time could be derived by two strategies: trying to improve the computation with new optimizations or reducing the number of the atlas ($\mathcal{A}$) or neighbourhood ($\mathcal{N}$) or both for each execution. It could be possible to reduce the overall time. However, we do not expect anymore a speedup gain like what we have achieved previously with new optimizations in the algorithm.

On the other hand, in the anatomic variability of the structures that we study, it does not matter how much we enlarge the neighbourhood ($\mathcal{N}$) after a specific limit, the image no longer improves because the matches are far away from where we search. The size of the atlas ($\mathcal{A}$) allows generalizing the variability of our samples. However, when the generalization is reached, adding new atlases do not produce improvements because this information is already contained.

We answer affirmatively to this question based on our experimental results (Chapter 5). A configuration with a neighbourhood of $\mathcal{N} = 9 \times 9 \times 9$ and 10 atlases produces excellent accuracy (NCC = 0.93) in our pseudo-CT images with a time of 43 seconds. In the last part of this Thesis, the algorithm has been revisited with new hardware (NVIDIA DGX Station with NVIDIA V100 GPUs) then the selected configuration can be further improved using the best configuration in our experiments


DocumentManager 1.0.0.0

Administration | Search

Filter | Clear

Filter Type	Title
Text to filter	A Supervised
Search	
Text to search in Pdf	to search

747 [2011] A Supervised Patch-Based Approach for Human Brain Labeling
748 [2014] Accurate PET/MR Quantification Using Time of Flight MLAA Ima
749 [2011] Simultaneous Reconstruction of Activity and Attenuation for PE1
750 [2012] Simultaneous Reconstruction of Activity and Attenuation in Time
751 [2017] Magnetic Resonance-Derived Improvements in PET Imaging
752 [2000] Inverse problems in Image processing and Image segmentation

Info | Comments | Code

Path	D:\Edu\Material\Medical image\RousseauPatchBased.pdf
Title	A Supervised Patch-Based Approach for Human Brain Labeling
Tags	Medical Image
Comments	

Year 2011 Num Comments 0 Included

PDF Viewer

1852 IEEE TRANSACTIONS ON MEDICAL IMAGING, VOL. 30, NO. 10, OCTOBER 2011

A Supervised Patch-Based Approach for Human Brain Labeling

, *Member, IEEE*, Piotr A. Habas, and Colin Studholme, *Member, IEEE*

Abstract—We propose in this work a patch-based image labeling method relying on a label propagation framework. Based on image intensity similarities between the input image and an anatomy textbook, an original strategy which does not require any nonrigid registration is presented. Following recent developments in nonlocal

Figure 6.3: Document Manager application to index documents and visualize them.

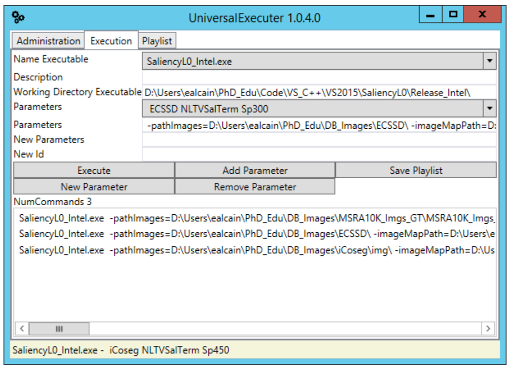
$\mathcal{N} = 11 \times 11 \times 11$ and 18 atlases (NCC = 0.935) spending half of the time (18 seconds). The nature of the proposed method promotes data independence reinforcing the idea to design algorithms in parallel to make the most out of these multicore and manycore architectures.

	Scanner CT	our pseudo-CT
Time	$\sim$ 2-4 min	18 seconds
Accuracy **NCC** (Eq. 5.1)	1	0.935

6.1.2 Tools for Research

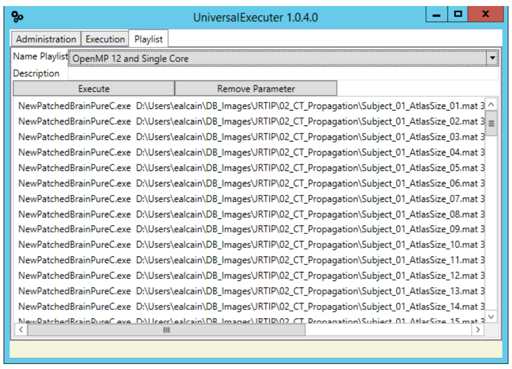
The organization of the information is a fundamental task to access it quickly and selectively. During the years of a PhD, reading papers, book chapters, technical reports, and Theses is a daily task, as well as many experiments, are carried out for several purposes. Promptly we realized that the available free tools (such as Zotero and Mendeley) do not provide the characteristics we needed to organize the relevant literature. People usually associate details in the document with the precise information they read previously in it, but the title of the document is difficult to memorize. To search for titles seems to be pointless in this case. However, if you could have a way to visualize the documents while browsing through them, it would be very helpful.

There may also be a need to run simulations on datasets and collect the results for proving hypotheses, for generating statistics in articles, etc. Implementing functionality inside the code and combining it in a series of scripts can typically address the problem. Some of these requirements are: to have a list of programs to execute



```
Name Executable              SaliencyL0_Intel.exe
Description
Working Directory Executable D:\Users\ealcain\PhD_Edu\Code\VS_C++\VS2015\SaliencyL0\Release_Intel\
Parameters                   ECSSD NLTVSalTerm Sp300
Parameters                   -pathImages=D:\Users\ealcain\PhD_Edu\DB_Images\ECSSD\ -imageMapPath=D:
New Parameters
New Id
NumCommands 3
SaliencyL0_Intel.exe  -pathImages=D:\Users\ealcain\PhD_Edu\DB_Images\MSRA10K_Imgs_GT\MSRA10K_Imgs_
SaliencyL0_Intel.exe  -pathImages=D:\Users\ealcain\PhD_Edu\DB_Images\ECSSD\ -imageMapPath=D:\Users\e
SaliencyL0_Intel.exe  -pathImages=D:\Users\ealcain\PhD_Edu\DB_Images\iCoseg\img\ -imageMapPath=D:\Us
SaliencyL0_Intel.exe -  iCoseg NLTVSalTerm Sp450
```

(a)



```
Name Playlist OpenMP 12 and Single Core
Description
NewPatchedBrainPureC.exe  D:\Users\ealcain\DB_Images\JRTIP\02_CT_Propagation\Subject_01_AtlasSize_01.mat 3
NewPatchedBrainPureC.exe  D:\Users\ealcain\DB_Images\JRTIP\02_CT_Propagation\Subject_01_AtlasSize_02.mat 3
NewPatchedBrainPureC.exe  D:\Users\ealcain\DB_Images\JRTIP\02_CT_Propagation\Subject_01_AtlasSize_03.mat 3
NewPatchedBrainPureC.exe  D:\Users\ealcain\DB_Images\JRTIP\02_CT_Propagation\Subject_01_AtlasSize_04.mat 3
NewPatchedBrainPureC.exe  D:\Users\ealcain\DB_Images\JRTIP\02_CT_Propagation\Subject_01_AtlasSize_05.mat 3
NewPatchedBrainPureC.exe  D:\Users\ealcain\DB_Images\JRTIP\02_CT_Propagation\Subject_01_AtlasSize_06.mat 3
NewPatchedBrainPureC.exe  D:\Users\ealcain\DB_Images\JRTIP\02_CT_Propagation\Subject_01_AtlasSize_07.mat 3
NewPatchedBrainPureC.exe  D:\Users\ealcain\DB_Images\JRTIP\02_CT_Propagation\Subject_01_AtlasSize_08.mat 3
NewPatchedBrainPureC.exe  D:\Users\ealcain\DB_Images\JRTIP\02_CT_Propagation\Subject_01_AtlasSize_09.mat 3
NewPatchedBrainPureC.exe  D:\Users\ealcain\DB_Images\JRTIP\02_CT_Propagation\Subject_01_AtlasSize_10.mat 3
NewPatchedBrainPureC.exe  D:\Users\ealcain\DB_Images\JRTIP\02_CT_Propagation\Subject_01_AtlasSize_11.mat 3
NewPatchedBrainPureC.exe  D:\Users\ealcain\DB_Images\JRTIP\02_CT_Propagation\Subject_01_AtlasSize_12.mat 3
NewPatchedBrainPureC.exe  D:\Users\ealcain\DB_Images\JRTIP\02_CT_Propagation\Subject_01_AtlasSize_13.mat 3
NewPatchedBrainPureC.exe  D:\Users\ealcain\DB_Images\JRTIP\02_CT_Propagation\Subject_01_AtlasSize_14.mat 3
```

(b)

Figure 6.4: (a) Execution configurations for different programs (b) Playlist functionality: a batch of executions. In this case, test for patch-based algorithm on single core and threads.

with different parameters, batch execution, e.g., create a custom list and run it, the possibility to save this list for further executions, for example, for a congress or to be able to record and display the parameters used in every experiment in a user-friendly interface.

To cover the requirements and functionalities we have exposed above, we have designed two applications for Windows. Figures 6.3 and 6.4 illustrate the layout of the applications designed. The tools are constantly in development with new functionalities. The *DocumentManager* (Figure 6.3) has helped us to index more than 800 documents. Its functionality is limited, but the PDF viewer is implemented and provides a very helpful mechanism to find the papers, books, etc. Furthermore, we can list the documents included in the Thesis document and can annotate interesting ideas for each document, code, etc. On the other hand, *UniversalExecuter* is the application to run our implementations and from others for comparison. Figure 6.4b shows some random executions in a batch. Figure 6.4a shows the layout where we can see a previous configurable batch with the execution of our saliency algorithm (playlist). The program, UniversalExecuter, has most of the functionality that we have described before.

6.1.3 Conclusions

The emphasis of this Thesis has been to bring parallel programming techniques in modern architectures to relevant problems in Computer Vision. The use of these techniques allows having real-time applications which are of much interest in Computer Vision nowadays. Parallel programming techniques have also become a fundamental building block for many areas outside Computer Vision since there is always a desire for performance improvement in algorithms and the performance growth driven by miniaturization has slowed down. Therefore, the alternatives, to satisfy these demands of many scientific areas, should come from parallel architecture design, parallel software, or parallel algorithms.

In our work, we have proposed a new algorithm and a model to solve relevant and different problems in Computer Vision using parallel programming techniques. The link was computation what has provided us with another perspective to design algorithms or models for future challenges: scalability.

The two methods produce good results in a short computational time and good quantitative results. However, their algorithms have different nature. The MRI-CT synthesis algorithm is naturally parallel, while the saliency detection is an iterative method. Based on the results of both methods, we have observed in the MRI-CT synthesis problem that scalability has produced a substantial impact on the performance with new hardware. The parallel nature in the algorithms has an advantage over not parallel ones when the computational time is an essential requirement. This advantage reinforces the idea to rethink the tools in mathematics, above all, iterative methods, to match in a better way parallel architectures. Therefore, we encourage mathematicians to explore/research new methods to make the most out of the parallel paradigm. Works like the presented here, where the collaboration was close, can help in making visible the advantages when adapting models to parallel architectures. Without a doubt, modeling, computing, or thinking in parallel is the present and the future.

Computation also evolves to satisfy the demands which arise in the new problems. Remember that GPU computation was a pure parallel model at the beginning, but it has integrated ways to express many algorithms that were not possible before. Implementing or, at least, trying to code solutions for problems that are not that parallel at first glance has helped others in thinking differently. For example, it would be of much interest a mechanism to check converge criteria without transferring data on GPU for iterative methods like variational formulations. As we can see, it is a duplex communication model-computation. Furthermore, user-friendly interface tools, frameworks, etc., also contribute towards easy communication among parts.

Finally, we have developed some tools for research (with limited functionality) that we think could help others. In this sense, we will publish the tools, *DocumentManager* and *UniversalExecuter*, for research purposes soon.

6.2 Future work

This work has provided a new model and an algorithm to solve efficiently two computer vision problems. In this sense, we shall propose future work in two directions: firstly, how to improve the model and the algorithm described in this Thesis, and secondly, how to optimize our implementations further.

6.2.1 Saliency Detection in Natural Images

- **Model:** There are several directions to try to improve the model. First, the control map used for the variational method should be more sophisticated using some object proposals or length penalty. With this idea, we aim to create a control map (fidelity term) more robust which can capture more complex scenes.

  Secondly, Section 3.3.5 has presented a formulation that is strictly convex because we have chosen the parameters to be in that way ($a < 0$). Nevertheless, the modelization with non-convex energies usually produces much better results [55]. Our idea is to change the parameters to make the formulation non-convex and use different numerical resolutions such as an inertial proximal algorithm for non-convex optimization (iPiano) [208], proximal point algorithm [262], or inexact primal-dual algorithm [228].

Thirdly, the energy profile has an L-shaped form (Figure 3.16d) which suggests using multigrid methods to calculate the saliency in a fine to coarse to fine multi-scale paradigm.

All our experiments have been carried out with a fixed δ-parameter, which was empirically set to 0.2. Figure 6.5 illustrates the effect of varying the δ-parameter. If a proper value is selected, then we have a clear binarization of the image. Based on this observation, we should try to estimate the optimal values of any image in the dataset. Nowadays, deep learning approaches can automatically provide the best parameters (see, for example, [226]). In future work, we propose designing a convolutional neural network (Figure 6.6) to obtain optimal parameters (α, δ, and λ) for a given input image.

Lastly, the real-time capability of the algorithm leads naturally to two strategies. The first one is to apply a multiresolution scheme with the superpixels. The second one is to adapt the algorithm to video sequences where the new formulations should include new pieces like the correlation between frames.

- **Optimization**: Unlike the synthesis of the CT problem, the saliency exhibits fewer data and the execution of kernels is fundamental for improving the algorithm on GPU. The primal-dual method consists of two different parts: maximization (Eq. 3.61) and minimization (Eq. 3.62). We believe that these two parts can be merged in only one kernel with synchronization directives and achieving a substantial speedup.

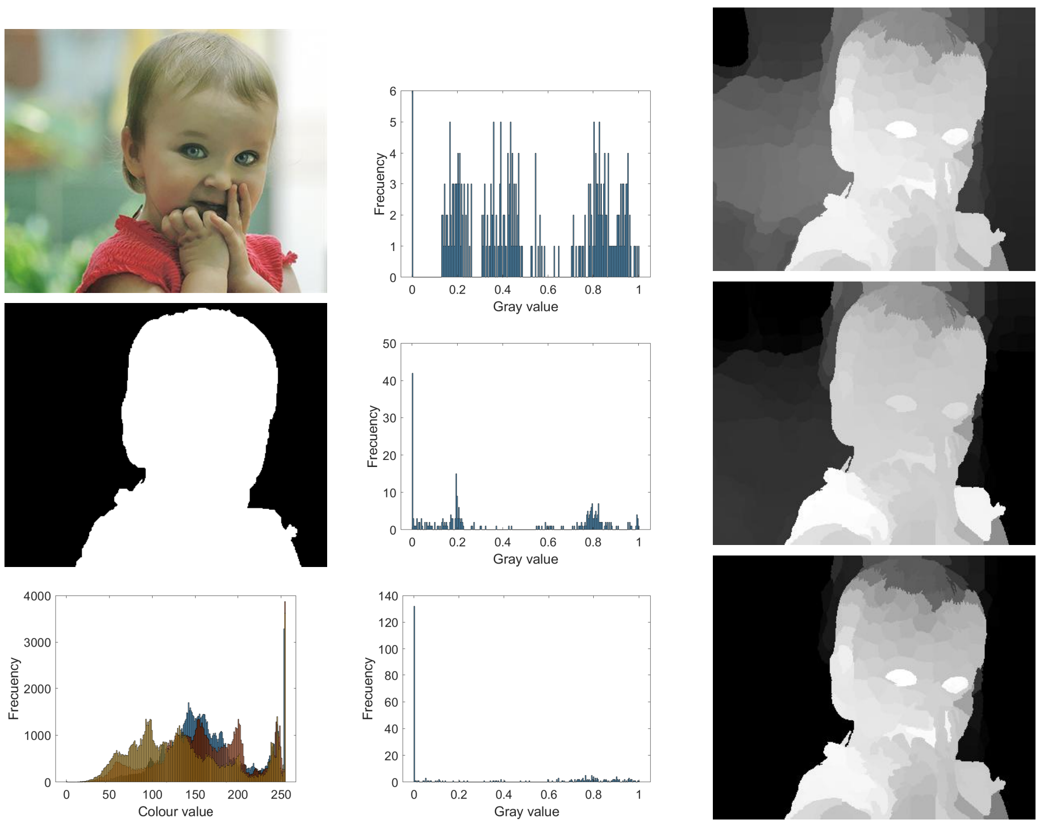
  Secondly, the transposed elements used for the divergence do not exhibit a good access pattern. We have solved this problem partly, but some research should conduct to find better data representation to improve performance.

  Finally, the subtask to leave the k-NN elements for a superpixel in the weight matrix $W^{sp \times sp}$ decreases in performance when k is large. We should derive a more sophisticated algorithm to perform well even if the k is large.

6.2.2 MRI-CT Synthesis

- **Algorithm:** To measure the similarity among patches in our proposal, we use an intensity metric following [66], which is computationally less intensive than the original Gaussian weighted Euclidean distance in [43]. These metrics are sensitive to the overall intensity. There are some alternatives to improve the robustness of the patch similarity.

  We could consider metrics that measure non-intensity differences, such as normalized mutual information, correlation, or geodesic distance. These metrics



(a) (b) (c)

Figure 6.5: (a) Input image from MSRA10K dataset [58] (top), ground truth (middle) and colour histogram (bottom) (b) saliency map histograms with $\delta = 0.1$, $\delta = 0.2$ and $\delta = 0.3$ respectively from top to bottom (c) saliency map accordingly to the δ-parameters.

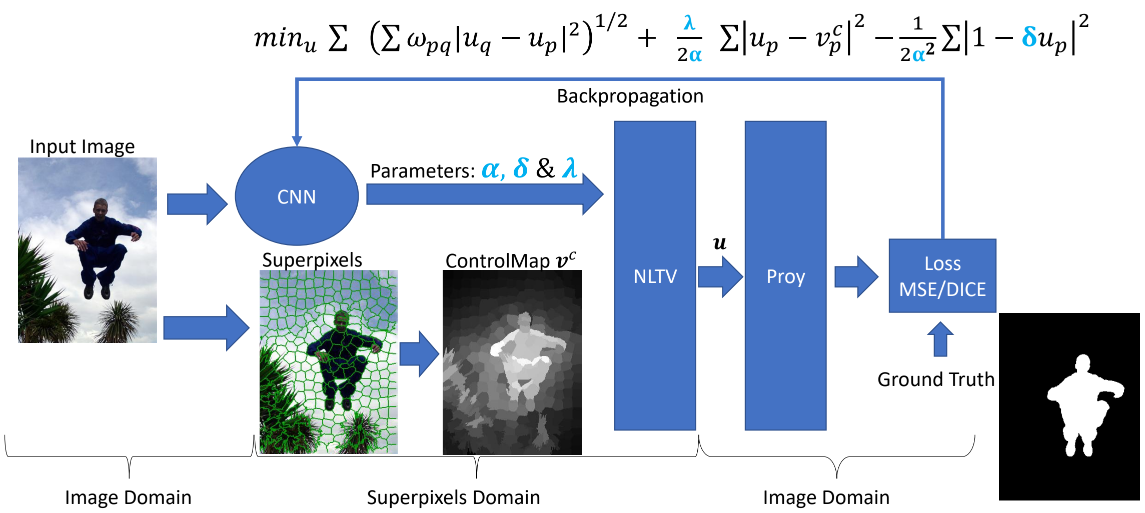

$min_u \sum \left(\sum \omega_{pq}|u_q - u_p|^2\right)^{1/2} + \frac{\lambda}{2\alpha} \sum|u_p - v_p^c|^2 - \frac{1}{2\alpha^2}\sum|1 - \delta u_p|^2$
Backpropagation
Input Image
CNN
Parameters: α, δ & λ
Superpixels
ControlMap v^c
NLTV
u
Proy
Loss MSE/DICE
Ground Truth
Image Domain
Superpixels Domain
Image Domain

Figure 6.6: Proposed idea to learn the parameters (α, δ, λ) for our variational model NL$TVSalTerm$. Input image from the MSRA10K dataset [58].

may produce better matching among patches in exchange for computational time.

On the other hand, since the method has demonstrated good scalability (Section 5.2.6), it will allow us to consider alternative algorithms that can use the output of a first iteration to feed the process several times and refine the result. This idea is under investigation to publish a new paper.

The data considered in this Thesis are from the human head. Extending the proposal to the whole body is desirable, but it is a difficult task. The method relies on *a-priori* information coded in the atlas. In the case of the head, we have demonstrated that a population dictionary with (MRI, CT) pairs can be constructed and used for synthesizing the unknown modality. However, anatomy variability increases when we consider the whole body. There are some problems to address, such as the size of the subject, pathology, and body mass index. For these reasons, dictionaries that could encode this variability of the entire population accurately are complicated to obtain. Furthermore, more complex registration procedures are required in other regions of the body (rigid and non-rigid registrations).

After the knowledge obtained in ill-posed problems from the saliency detection, it may be interesting to approach the correction problem from the emission data perspective so that the difficulties regarding the anatomical variability may be solved in the whole body [244]. This methodology also exhibits a large amount of data to be accelerated by multicore and manycore architectures.

- **Optimization**: The implementation has achieved a remarkable result because the method is highly parallelizable. However, there is still room for improvements. Firstly, the algorithm performs the calculations in double precision. When necessary and appropriate, we can execute the calculations in single precision and achieve a speedup of ($\sim \times 2$).

  Secondly, the problem is bandwidth bound or memory bound, e.g., more memory accesses in comparison with computations. Now, the algorithm produces very optimized results. It is time to investigate the way that the elements are loaded from global memory to achieve the maximum of coalesced memory access.

  Lastly, when considering new large problems (better voxel resolution), the multi-GPUs paradigm could allow us to achieve real-time capability. However, the synchronization among GPUs and how to calculate the load for each one are tasks for further research.

6.3 Publication Summary

- **Eduardo Alcaín**, Ana I. Muñoz, Emanuele Schiavi, and Antonio S. Montemayor *A Non-Smooth Non-Local Variational Approach to Saliency Detection in Real Time* Journal of Real-Time Image Processing (JRTIP) September 2020 doi: https://doi.org/10.1007/s11554-020-01016-4 **IF-2020: 1.968** (Q3)

- **Eduardo Alcaín**, Ana I. Muñoz, Iván Ramírez, and Emanuele Schiavi *A non local non convex approach to saliency detection* HONOM: VII European Workshop on High Order Numerical Methods for Evolutionary PDEs, 1-5 April, 2019, Madrid, Spain.

- **Eduardo Alcaín**, Ana I. Muñoz, Iván Ramírez, and Emanuele Schiavi *Modelling sparse saliency maps on manifolds numerical results and applications* SEMA SIMAI Springer Series book series (SEMA SIMAI, volume 18) Recent Advances in Differential Equations and Applications pp 157-175 Best selected contribution to the CEDYA/CMA'17 2019 https://doi.org/10.1007/978-3-030-00341-8_10

- **Eduardo Alcaín**, Ana I. Muñoz, Iván Ramírez, and Emanuele Schiavi *Modelling sparse saliency maps on manifolds numerical results and applications* XXV Conference on differential equations and applications 26-30 June 2017, Cartagena, Spain

- **Eduardo Alcaín**, Ángel Torrado-Carvajal, Antonio S. Montemayor, and Norberto Malpica *Real-time patch-based medical image modality propagation by GPU computing*. Journal of Real-Time Image Processing (JRTIP) 2017 https://doi.org/10.1007/s11554-016-0568-0 **IF-2017: 1.574** (Q3)

- Ángel Torrado-Carvajal, **Eduardo Alcaín**, Antonio S. Montemayor, Joaquin L. Herraiz, Juan A. Hernandez-Tamames, Yves Rozenholc, E. Adalsteinsson, L. L. Wald, and Norberto Malpica *Fast pseudo-CT synthesis from MRI T1-weighted images using a patch-based approach* Proceedings of SPIE, 11th International Symposium on Medical Information Processing and Analysis (SIPAIM), 17 November 2015, Cuenca, Ecuador, https://doi.org/10.1117/12.2210963

- Ángel Torrado-Carvajal, Joaquin L. Herraiz, **Eduardo Alcaín**, Antonio S. Montemayor, Juan A. Hernandez-Tamames, Yves Rozenholc, and Norberto Malpica *Fast Patch-Based Pseudo-CT Synthesis from T1-Weighted MR Images for PET/MR Attenuation Correction* Journal of Nuclear Medicine January 2016 https://doi.org/doi:10.2967/jnumed.115.156299 **IF-2016: 6.646** (Q1)

- Ángel Torrado-Carvajal, **Eduardo Alcaín**, Antonio S. Montemayor, and Norberto Malpica *A Fast Patch-Based Approach for Pseudo-CT Generation from MRI T1-Weighted Images: A Potential Solution for PET/MR Attenuation Correction* International Society for Magnetic Resonance in Medicine (ISMRM) May 30 - June 5 2015, Toronto, Canada Accepted Poster ID 6590 http://www.ismrm.org/15/Accepted_Abstracts.pdf

- **Eduardo Alcaín**, Jorge F. Villena, Athanasios G. Polimeridis, Emanuele Schiavi, and Antonio S. Montemayor *Accelerated FFT-JVIE Solvers for Electromagnetic Analysis in MRI* Poster at GPU Technology Conference (GTC) 2014, San Jose, USA https://on-demand.gputechconf.com/gtc/2014/poster/pdf/P4193_MRI_FFT_JVIE_MATLAB.pdf

Appendix A

Appendix

Table A.1: Different time results for the matrix multiplication algorithm in three languages: Python (3.5.2), C# (4.5.2 Framework), C++ (Microsoft Visual Studio), MKL stands for Math Kernel Library from Intel. After C++ is selected as the language for coding, we show the incremental optimizations. From implementation **C++ Intel** to **MKL**, the results displayed are the average of 10 executions. Matrix size 4096 $\times$ 4096.

Implementation	Run Time (s)	Relative Speedup	Absolute Speedup
Python	17380.8	$\times$1.0	$\times$1.0
C#	645.9	$\times$26.9	$\times$26.9
C++	230.0	$\times$2.8	$\times$75.5
C++ Intel	210.2	$\times$1.0	$\times$82.6
C++ O2	45.7	$\times$4.5	$\times$379.5
C++ Threads	6.1	$\times$7.5	$\times$2849.3
C++ Tiling	4.2	$\times$1.4	$\times$4138.3
C++ Oblivious AVX2	3.2	$\times$1.3	$\times$5431.5
MKL	0.63	$\times$5.0	$\times$27588.7

A.1 Software Optimization: Matrix Multiplication

Matrix multiplication represents a clear example of an optimization process: easy to understand the operations involved (add and multiply) and the speedup achieved at the end of the optimization process reinforces the idea of performance in Computer Science. We implement the classical algorithm with $\mathcal{O}(n^3)$ and use a square matrix. For illustration purposes, the dimensions of the matrix are large enough to see the difference among optimizations (4096 $\times$ 4096) and the elements in the matrix are in double precision. We start by selecting one programming language and, after this, we apply successive optimization techniques. We have used for our implementation: Python 3.5.2, Visual Studio 2015 (C# .NET 4.5.2) and Intel C++ Compiler 19.0, and the experiments have been carried out on Intel© Xeon© Processor E5-1650 v3 [128]. These optimizations are inspired by the lecture [152] and the following manuscripts [80, 92].

A.1.1 Choice of the Programming Language

The selection of the programming language is an important step to prepare an optimization process. There are different criteria to select it depending on the requirements, for example, portability. Here, we focus on performance. We have implemented the matrix multiplication algorithm in three languages: Python, C#, and C++. The same algorithm is implemented in the three languages and no optimization has been applied (Table A.1). The language C++ achieves the best time without any optimization. So in the next sections, we shall use C++ language to apply the optimizations to the matrix multiplication algorithm.

A.1.2 Choice of the Compiler

Once the language has been decided, we should choose one of the many compilers for compiling C++ code. We have compared the compiler provided by Intel **icc** and the one provided Microsoft **cl**. The Intel compiler **icc** outperforms **cl** (Table A.1).

A.1.3 Compilation Improvement: Flags

Compilation flags apply some optimizations to the code by the compiler transparently. Until now, we have used the flag -O0 (no optimization). Using the optimization flag -O2, we improve ($\times$4.5) without no changing the code. This kind of optimization allows the compiler to apply auto-vectorization transparently. This optimization is **C++ 02** (Table A.1).

A.1.4 Multicore Improvement: OpenMP

We have used only one core to compute the matrix multiplication so far, but the machine has 6 cores and Hyper-Threading technology [128]. In theory, we could use up to 12 threads to compute the matrix multiplication algorithm. We set 12 threads for a multithreaded solution by OpenMP [72]. This optimization is **C++ Threads** (Table A.1).

A.1.5 Memory Improvement: Tiling

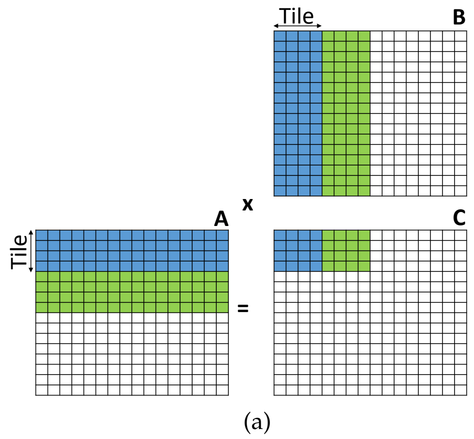
Until now, we have implemented strategies without modifying the source code. Then, it is time to reconsider why we are still far from the MKL library performance (Table A.1). The memory access pattern in our implementation creates too many cache misses.

```
for (int i = 0; i < n; i++){
 for (int k = 0; k < n; k++){
  for (int j = 0; j < n; j++){
     C[(i*n)+ j] += A[(i*n) + k] * B[(k*n) + j];
  }
 }
}
```

We have cache misses because, for each row calculation in C, we need to retrieve from memory a whole row from A and the whole matrix for the column element in matrix B. We can use a block cache technique (tiling) to improve the temporal locality of the operations (Figure A.1a). The code implementing this technique is:

```
for (int i = 0; i < n; i += tile){
 for (int j = 0; j < n; j += tile){
  for (int k = 0; k < n; k += tile){
   for (int ib = 0; ib < tile; ib++) {
    for (int kb = 0; kb < tile; kb++) {
     for (int jb = 0; jb < tile; jb++) {
      C[(i*n + ib*n) + j + jb] += A[(i*n + ib*n) + k + kb] * B[(k + kb)*n + j + jb];
         }
    }
   }
  }
 }
}
```

The tile parameter should be the cache line size divided by *sizeof(double)* [80] and we set tile=64 after some experimentation. This optimization is **C++ Tiling** (Table A.1).



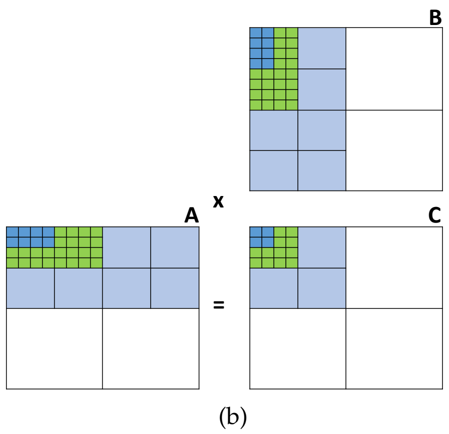


Figure A.1: (a) Tiling technique workflow in matrix multiplication (b) Cache oblivious dividing the matrix multiplication into 4 blocks and multiplying each block recursively.

A.1.6 Memory Improvement: Oblivious + AVX2

To accelerate the matrix algorithm, we divide the problem into subproblems (*cache-oblivious*) and use Cilk [31] (nested parallelism). In this sense, we can compute them in parallel and achieve good performance for the cache, and use the cores in the machine (Figure A.1b). Furthermore, we apply vectorization techniques for the base case [130]. We have set THRESHOLD=64 after experimentation. This optimization is **C++ Oblivious AVX2** (Table A.1).

```
void rec_mult(const double *A, const double *B, double *C,
              const int n, const  int rowsize) {

if (n <= THRESHOLD) {
  mm_baseAVX2(A, B, C, n, rowsize);
}else{
 int d11 = 0;
 int d12 = n / 2;
 int d21 = (n / 2) * rowsize;
 int d22 = (n / 2) * (rowsize + 1);
 cilk_spawn        rec_mult(A + d11, B + d11, C + d11, n / 2, rowsize);
 cilk_spawn        rec_mult(A + d12, B + d21, C + d11, n / 2, rowsize);
 cilk_spawn        rec_mult(A + d11, B + d12, C + d12, n / 2, rowsize);
 rec_mult(A + d12, B + d22, C + d12, n / 2, rowsize);

 cilk_sync;
 cilk_spawn rec_mult(A + d21, B + d11, C + d21, n / 2, rowsize);
 cilk_spawn rec_mult(A + d22, B + d21, C + d21, n / 2, rowsize);
 cilk_spawn rec_mult(A + d21, B + d12, C + d22, n / 2, rowsize);
 rec_mult(A + d22, B + d22, C + d22, n / 2, rowsize);
 cilk_sync ;
 }
}
```

A.2 Calculus of Variations in Image Processing

This appendix is based on the lecture notes of *Fundamentos Matemáticos* at the Rey Juan Carlos University for the master of *Visión artificial* [249] and *Variational Methods for Computer Vision* at TU München [68].

A.2.1 Euler-Lagrange Equations for the Denoising Problem

We describe a detailed derivation of the Euler-Lagrange equations in image processing. To illustrate the derivation, we consider the denoising problem (3.3) which is about recovering the original signal from an observed noisy sample and for simplicity, the linear case is assumed (3.23).

Let $\Omega \subset \mathbb{R}^n$ with $n \in \mathbb{N}$ be an open bounded domain of boundary $\partial\Omega$. Let X be a function space, say $X = \{u(\mathbf{x}) \;/\; u : \Omega \to \mathbb{R}\}$, and let E be a functional defined on X, i.e., a mapping of the function space X onto the set of real numbers, say $E : X \to \mathbb{R}$. In short, a functional is a function that takes an element $u(\mathbf{x})$ of the function space X as an input and returns a scalar. When we say that E is well defined in X we mean that E is bounded on X, i.e., $E(u) < +\infty$ for any $u \in X$.

We consider the minimization problem

$$\min_{u\in X} E(u) = \min_{u\in X} ||\nabla u||_2^2 + \lambda ||u - f||_2^2 \tag{A.1}$$

where X is the Sobolev space $X = H^1(\Omega)$ because, by definition, if $u \in H^1(\Omega)$ then $||u||_2^2 < +\infty$, $||\nabla u||_2^2 < +\infty$ and the functional $E(u)$ is bounded, i.e., well defined in the space $H^1(\Omega)$.

This variational model is strictly convex because the fidelity term is strictly convex and the regularizer is convex. Their sum is strictly convex and therefore, the solution is unique. To find the minimum, we should solve the necessary first-order conditions of the null gradient, which determines the stationary points of the functional. Because of the strict convexity, these conditions are also sufficient to obtain a (global) minimum. Nevertheless, it is not correct to say *the gradient of the functional* because in an infinite-dimensional space there are infinite directions to consider. We then introduce the concept of Gâteaux differential of a functional, [19] as a generalization of the concept of directional derivatives in multi-variable calculus. This implies to perturb the energy of the functional in all directions defined by functions $v \in X$ and, passing to the limit, to obtain the Gâteaux differential of the functional

$$\langle \partial E(u), v \rangle, \qquad \forall\, v \in X$$

We shall describe the whole procedure.

Let X an infinite-dimensional vector space and $E : X \to \mathbb{R}$ a functional defined in X. If the limit is finite:

$$\langle \partial E(u), v \rangle = \lim_{\alpha\to 0} \frac{E(u + \alpha v) - E(u)}{\alpha} \tag{A.2}$$

then we say that E is Gâteaux differentiable and

$$\partial_v E(u) = \langle \partial E(u), v \rangle$$

is the directional derivative of E in the point u in the direction v. We calculate the limit for $E(u)$ in (A.1) as follows:

$$\begin{aligned}
E(u+\alpha v) - E(u) &= \underbrace{||\nabla(u+\alpha v)||_2^2 + \lambda||(u+\alpha v) - f||_2^2}_{E(u+\alpha v)} - \underbrace{||\nabla u||_2^2 - \lambda||u - f||_2^2}_{E(u)} \\
&= \int_\Omega |\nabla(u+\alpha v)|^2 dx - \int_\Omega |\nabla u|^2 dx \\
&\quad + \lambda \int_\Omega (u+\alpha v - f)^2 dx - \lambda \int_\Omega (u-f)^2 dx
\end{aligned}$$

Now we can consider each term separately. First

$$\begin{aligned}
\int_\Omega |\nabla(u+\alpha v)|^2 dx &= \int_\Omega |\nabla u + \alpha \nabla v|^2 dx \\
&= \int_\Omega |\nabla u|^2 dx + 2\alpha \int_\Omega \nabla u \nabla v dx + \alpha^2 \int_\Omega |\nabla v|^2 dx
\end{aligned}$$

and

$$\begin{aligned}
\lambda \int_\Omega (u+\alpha v - f)^2 dx &= \lambda \int_\Omega (u - f + \alpha v)^2 dx \\
&= \lambda \int_\Omega (u-f)^2 dx + 2\lambda\alpha \int_\Omega (u-f) v dx + \lambda\alpha^2 \int_\Omega v^2 dx
\end{aligned}$$

then using these partial results

$$\begin{aligned}
\int_\Omega |\nabla(u+\alpha v)|^2 dx - \int_\Omega |\nabla u|^2 dx &= 2\alpha \int_\Omega \nabla u \nabla v dx + \alpha^2 \int_\Omega |\nabla v|^2 \\
\lambda \int_\Omega (u+\alpha v - f)^2 dx - \lambda \int_\Omega (u-f)^2 dx &= 2\lambda\alpha \int_\Omega (u-f) v dx + \lambda\alpha^2 \int_\Omega v^2 dx
\end{aligned}$$

Plugging these partial calculations into Eq. A.2 and dividing by α

$$\begin{aligned}
\frac{E(u+\alpha v) - E(u)}{\alpha} &= 2 \int_\Omega \nabla u \nabla v dx + \alpha \int_\Omega |\nabla v|^2 dx \\
&\quad + 2\lambda \int_\Omega (u-f) v dx + \lambda\alpha \int_\Omega v^2 dx
\end{aligned}$$

taking the limit $\alpha \to 0$, the energy terms of the test function v vanish

$$\alpha \int_\Omega |\nabla v|^2 dx + \lambda\alpha \int_\Omega v^2 dx \to 0,$$

because $\|v\|_2^2 < +\infty$ and $\|\nabla v\|_2^2 < +\infty$. We then have

$$\langle \partial E(u), v \rangle = 2 \int_\Omega \nabla u \nabla v dx + 2\lambda \int_\Omega (u-f) v dx$$

Using Green's formulas [105]

$$\int_{\Omega} \nabla u \nabla v dx = -\int_{\Omega} \Delta u v \, dx + \int_{\partial\Omega} \nabla u \cdot \mathbf{n} v \, dx \tag{A.3}$$

and homogeneous Neumann boundary condition (also termed, in fluid mechanics, the *no flux (flow) across the boundary* condition):

$$\nabla u(\mathbf{x}) \cdot \mathbf{n} = 0, \qquad \mathbf{x} \in \partial\Omega$$

we obtain

$$\int_{\partial\Omega} \nabla u \cdot \mathbf{n} v \, dx = 0, \quad \forall \, v \in X \tag{A.4}$$

hence

$$\int_{\Omega} \nabla u \nabla v dx = -\int_{\Omega} \Delta u v dx \tag{A.5}$$

Using formula (A.5), we have

$$\begin{aligned} \langle \partial E(u), v \rangle &= 2\int_{\Omega} \nabla u \nabla v dx + 2\lambda \int_{\Omega} (u-f) v dx \\ &= -2\int_{\Omega} \Delta u v dx + 2\lambda \int_{\Omega} (u-f) v dx \\ &= 2\int_{\Omega} (-\Delta u + \lambda(u-f)) v dx \end{aligned}$$

where Δu is the Laplacian operator defined as the divergence of the gradient operator

$$\Delta u = div(\nabla u) = \frac{\partial^2 u}{\partial x_1^2} \ldots + \frac{\partial^2 u}{\partial x_n^2} \tag{A.6}$$

for $x = (x_1 \ldots x_n) \in \Omega \subset \mathbb{R}^n$. The notation $div(\nabla u) = \nabla \cdot \nabla u$ is also widely used.

Imposing the first-order necessary condition to cancel the gradient

$$\langle \partial E(u), v \rangle = 0, \qquad \forall \, v \in X$$

we get

$$\langle \partial E(u), v \rangle = 2\int_{\Omega} (-\Delta u + \lambda(u-f)) v dx = 0, \forall \, v \in X$$

and applying the Fundamental Lemma of Variational Calculus we deduce the Euler-Lagrange equation of the functional to be minimized

$$-\Delta u + \lambda(u-f) = 0, \qquad \text{on } \Omega \tag{A.7}$$

which is complemented with the homogeneous Neumann boundary conditions (A.4) along the boundary $\partial\Omega$.

A.2.2 A General Formula

As a general case, we shall consider the following kind of integral functionals:

$$E(u) = \int_\Omega \mathcal{L}(\mathbf{x}, u, \nabla u) d\mathbf{x}, \tag{A.8}$$

where $\mathbf{x} \in \Omega \in \mathbb{R}^n$ and $\mathcal{L}$ is called the *lagrangian*.

The Euler-Lagrange equation for this kind of functional is as follows (see [68] for details)

$$\frac{\partial \mathcal{L}}{\partial u}(\mathbf{x}, u, \nabla u) - \sum_{j=1}^{n} \frac{\partial}{\partial x_j}\Big(\frac{\partial \mathcal{L}}{\partial p_j}(\mathbf{x}, u, \nabla u)\Big) = 0 \tag{A.9}$$

Given an energy functional in (A.8), we can use the general formula (A.9) to illustrate how to calculate the Euler-Lagrange equation of our denoising functional (A.1). Set $\mathbf{x} = (x, y)^T$ and let

$$\mathcal{L}(\mathbf{x}, u, \nabla u) = \frac{1}{2}|\nabla u|^2 + \frac{\lambda}{2}(u - f)^2$$

We introduce a variable $\mathbf{p}$ to denote the gradient term

$$\mathbf{p} = (p_1, p_2)^T = (\nabla u)^T = (u_x, u_y)$$

with magnitude

$$|\mathbf{p}|^2 = |\nabla u|^2 = \Big(\sqrt{u_x^2 + u_y^2}\Big)^2 = u_x^2 + u_y^2$$

We can rewrite the lagrangian $\mathcal{L}$ in terms of $\mathbf{p}$ as:

$$\mathcal{L}(\mathbf{x}, u, \nabla u) = \mathcal{L}(\mathbf{x}, u, \mathbf{p}) = \frac{1}{2}|\mathbf{p}|^2 + \frac{\lambda}{2}(u - f)^2 = \frac{1}{2}(p_1^2 + p_2^2) + \frac{\lambda}{2}(u - f)^2$$

Applying formula (A.9) written in terms of $\mathbf{p}$ we have the Euler-Lagrange equation

$$\frac{\partial \mathcal{L}}{\partial u}(\mathbf{x}, u, \mathbf{p}) - \frac{\partial}{\partial x}\Big(\frac{\partial \mathcal{L}}{\partial p_1}(\mathbf{x}, u, \mathbf{p})\Big) - \frac{\partial}{\partial y}\Big(\frac{\partial \mathcal{L}}{\partial p_2}(\mathbf{x}, u, \mathbf{p})\Big) = 0$$

We compute the partial derivatives as

$$\frac{\partial \mathcal{L}}{\partial u}(\mathbf{x}, u, \mathbf{p}) = \lambda(u - f), \qquad \frac{\partial \mathcal{L}}{\partial p_1}(\mathbf{x}, u, \mathbf{p}) = p_1, \qquad \frac{\partial \mathcal{L}}{\partial p_2}(\mathbf{x}, u, \mathbf{p}) = p_2$$

Substituting, the Euler-Lagrange equation is

$$\lambda(u - f) - \frac{\partial p_1}{\partial x} - \frac{\partial p_2}{\partial y} = 0$$

and getting back to the original variables we have

$$\lambda(u - f) - \frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = 0$$

which is (A.7) by (A.6)

$$-\Delta u + \lambda(u-f) = 0$$

The $p-$laplacian operator

In the general case, we set

$$\mathcal{L}_p(\mathbf{x}, u, \nabla u) = \frac{1}{p}|\nabla u|^p + \frac{\lambda}{2}(u-f)^2, \qquad p > 1.$$

Then

$$\mathcal{L}_p(\mathbf{x}, u, \nabla u) = \mathcal{L}_p(\mathbf{x}, u, \mathbf{p}) = \frac{1}{p}|\mathbf{p}|^p + \frac{\lambda}{2}(u-f)^2 = \frac{1}{p}(p_1^2 + p_2^2)^{p/2} + \frac{\lambda}{2}(u-f)^2$$

The partial derivatives are

$$\frac{\partial \mathcal{L}}{\partial u}(\mathbf{x}, u, \mathbf{p}) = \lambda(u-f),$$

and

$$\frac{\partial \mathcal{L}}{\partial p_1}(\mathbf{x}, u, \mathbf{p}) = p_1(p_1^2 + p_2^2)^{p/2-1}, \qquad \frac{\partial \mathcal{L}}{\partial p_2}(\mathbf{x}, u, \mathbf{p}) = p_2(p_1^2 + p_2^2)^{p/2-1}$$

The Euler-Lagrange equation is

$$\frac{\partial \mathcal{L}}{\partial u}(\mathbf{x}, u, \mathbf{p}) - \frac{\partial}{\partial x}\Big(\frac{\partial \mathcal{L}}{\partial p_1}(\mathbf{x}, u, \mathbf{p})\Big) - \frac{\partial}{\partial y}\Big(\frac{\partial \mathcal{L}}{\partial p_2}(\mathbf{x}, u, \mathbf{p})\Big) = 0$$

Substituting

$$\lambda(u-f) - \frac{\partial}{\partial x}\left(p_1(p_1^2 + p_2^2)^{p/2-1}\right) - \frac{\partial}{\partial y}\left(p_2(p_1^2 + p_2^2)^{p/2-1}\right) = 0$$

Using the divergence operator

$$\lambda(u-f) - \operatorname{div}\left((p_1^2 + p_2^2)^{p/2-1}(p_1, p_2)\right) = 0$$

and observing that

$$(p_1^2 + p_2^2)^{p/2-1}(p_1, p_2) = (p_1^2 + p_2^2)^{(p-2)/2}(p_1, p_2) = |\mathbf{p}|^{p-2}\mathbf{p}$$

we have

$$\lambda(u-f) - \operatorname{div}\left(|\mathbf{p}|^{p-2}\mathbf{p}\right) = 0$$

Getting back to the original variable

$$\lambda(u-f) - \operatorname{div}\left(|\nabla u|^{p-2}\nabla u\right) = 0$$

which is the $p-$laplacian equation

$$-\Delta_p u + \lambda(u-f) = 0, \quad \text{on } \Omega, \qquad \forall\, p > 1$$

associated to the $p-$laplacian operator

$$\Delta_p u = \operatorname{div}\left(|\nabla u|^{p-2}\nabla u\right).$$

Bibliography

[1] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: http://tensorflow.org/.

[2] R. Achanta, S. Hemami, F. Estrada, and S. Susstrunk. "Frequency-tuned salient region detection". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 1597–1604. DOI: 10.1109/CVPR.2009.5206596.

[3] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. "SLIC Superpixels Compared to State-of-the-Art Superpixel Methods". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 34.11 (Nov. 2012), pp. 2274–2282. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2012.120.

[4] Atanu Acharya et al. "Supercomputer-Based Ensemble Docking Drug Discovery Pipeline with Application to Covid-19". In: (July 2020). DOI: 10.26434/chemrxiv.12725465.

[5] R.A. Adams. *Sobolev Spaces. Adams*. Pure and applied mathematics. Academic Press, 1975.

[6] Alfred V. Aho and John E. Hopcroft. *The Design and Analysis of Computer Algorithms*. 1st. USA: Addison-Wesley Longman Publishing Co., Inc., 1974. ISBN: 0201000296.

[7] E. Ajkunic, H. Fatkic, E. Omerovic, K. Talic, and N.Nosovic. "A comparison of five parallel programming models for C++". In: *2012 Proceedings of the 35th International Convention MIPRO*. 2012, pp. 1780–1784.

[8] E. Alcaín, A. Muñoz, E. Schiavi, and A. S. Montemayor. "A non-smooth non-local variational approach to saliency detection in real time". In: *Journal of Real-Time Image Processing* (Sept. 2020). DOI: 10.1007/s11554-020-01016-4.

[9] E. Alcaín, A. Torrado-Carvajal, A.S. Montemayor, and N. Malpica. "Real-time patch-based medical image modality propagation by GPU computing". In: *Journal of Real-Time Image Processing (JRTIP)* 13 (Feb. 2017), pp. 193–204. ISSN: 1861-8219. DOI: 10.1007/s11554-016-0568-0.

[10] Eduardo Alcaín, Ana I. Muñoz, Iván Ramírez, and Emanuele Schiavi. *Modelling Sparse Saliency Maps on Manifolds: Numerical Results and Applications*. SEMA SIMAI Springer Series book series (SEMA SIMAI, volume 18) Recent Advances in Differential Equations and Applications pp 157-175 Best selected contribution to the CEDYA/CMA'17 https://doi.org/10.1007/978-3-030-00341-8_10. 2019.

[11] Amjad Ali and Khalid Saifullah Syed. "An Outlook of High Performance Computing Infrastructures for Scientific Computing". In: *Advances in Computers*. Elsevier, 2013, pp. 87–118. DOI: 10.1016/b978-0-12-408089-8.00003-3.

[12] L. Ambrosio, N. Fusco, and D. Pallara. *Functions of Bounded Variation and Free Discontinuity Problems*. Oxford Science Publications. Clarendon Press, 2000. ISBN: 9780198502456.

[13] Gene M. Amdahl. "Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities". In: *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*. AFIPS '67 (Spring). Atlantic City, New Jersey: Association for Computing Machinery, 1967, pp. 483–485. ISBN: 9781450378956. DOI: 10.1145/1465482.1465560.

[14] Asmita AMoghe and Jyoti Singhai. "Image Registration: A Review of Elastic Registration Methods Applied to Medical Imaging". In: *International Journal of Computer Applications* 70.7 (May 2013), pp. 6–11. DOI: 10.5120/11972-7827.

[15] Daniel Andreasen, Koen Van Leemput, Rasmus H. Hansen, Jon A. L. Andersen, and Jens M. Edmund. "Patch-based generation of a pseudo CT from conventional MRI sequences for MRI-only radiotherapy of the brain". In: *Medical Physics* 42.4 (Mar. 2015), pp. 1596–1605. DOI: 10.1118/1.4914158.

[16] K.J. Arrow et al. *Studies in Linear and Non-linear Programming*. Stanford mathematical studies in the social sciences. Stanford University Press, 1958. ISBN: 9780804705622.

[17] Krste Asanović et al. *The Landscape of Parallel Computing Research: A View from Berkeley*. Tech. rep. UCB/EECS-2006-183. EECS Department, University of California, Berkeley, Dec. 2006. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html.

[18] Eric Aubanel. *Elements of Parallel Computing*. 1st. Chapman and Hall/CRC, 2016. ISBN: 1498727891.

[19] Gilles Aubert and Pierre Kornprobst. *Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations (Applied Mathematical Sciences)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387322000.

[20] Shai Avidan and Ariel Shamir. "Seam Carving for Content-Aware Image Resizing". In: *ACM SIGGRAPH 2007 Papers*. SIGGRAPH '07. San Diego, California: Association for Computing Machinery, 2007, 10–es. ISBN: 9781450378369. DOI: 10.1145/1275808.1276390.

[21] Syed Lal Badshah, Nasir Ahmad, Ashfaq Ur Rehman, Khalid Khan, Asad Ullah, Abdulrhman Alsayari, Abdullatif Bin Muhsinah, and Yahia N. Mabkhot. "Molecular docking and simulation of Zika virus NS3 helicase". In: *BMC Chemistry* 13.1 (May 2019). DOI: 10.1186/s13065-019-0582-y.

[22] Dhruv Batra, Adarsh Kowdle, Devi Parikh, Jiebo Luo, and Tsuhan Chen. "iCoseg: Interactive co-segmentation with intelligent scribble guidance". In: *CVPR*. 2010, pp. 3169–3176. DOI: 10.1109/CVPR.2010.5540080.

[23] Marcin B. Bauza et al. "Realization of Industry 4.0 with high speed CT in high volume production". In: *CIRP Journal of Manufacturing Science and Technology* 22 (2018), pp. 121–125. ISSN: 1755-5817. DOI: 10.1016/j.cirpj.2018.04.001.

[24] Nathan Bell and Michael Garland. *Efficient Sparse Matrix-Vector Multiplication on CUDA*. NVIDIA Technical Report NVR-2008-004. NVIDIA Corporation, Dec. 2008.

[25] Nathan Bell and Jared Hoberock. "Thrust: Productivity-Oriented Library for CUDA". In: *Astrophysics Source Code Library* 7 (Dec. 2012), pp. 12014–.

[26] Gino Bella, Nicola Rossi, Salvatore Filippone, Stefano Ubertini, Studi Roma, and Tor Vergata. “Using OpenMP on a hydrodynamic lattice-Boltzmann code”. In: (Jan. 2002).

[27] Jon Louis Bentley. *Writing Efficient Programs*. USA: Prentice-Hall, Inc., 1982. ISBN: 0139702512.

[28] Amy Berrington de González and Sarah Darby. “Risk of cancer from diagnostic X-rays: Estimates for the UK and 14 other countries”. In: *Lancet* 363 (Feb. 2004), pp. 345–51. DOI: 10.1016/S0140-6736(04)15433-0.

[29] Thomas Beyer et al. “A Combined PET/CT Scanner for Clinical Oncology”. In: *Journal of Nuclear Medicine* 41.8 (2000), pp. 1369–1379. URL: http://jnm.snmjournals.org/content/41/8/1369.short.

[30] F. Bloch. “Nuclear Induction”. In: *Phys. Rev.* 70 (7-8 Oct. 1946), pp. 460–474. DOI: 10.1103/PhysRev.70.460.

[31] Robert D. Blumofe, Christopher F. Joerg, Bradley C. Kuszmaul, Charles E. Leiserson, Keith H. Randall, and Yuli Zhou. “Cilk: An Efficient Multithreaded Runtime System”. In: *Journal of Parallel and Distributed Computing* 37.1 (1996), pp. 55–69. ISSN: 0743-7315. DOI: https://doi.org/10.1006/jpdc.1996.0107.

[32] A. Borji and L. Itti. “State-of-the-Art in Visual Attention Modeling”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.1 (2013), pp. 185–207. DOI: 10.1109/TPAMI.2012.89.

[33] Ali Borji. “Saliency Prediction in the Deep Learning Era: Successes and Limitations”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019), pp. 1–1. DOI: 10.1109/tpami.2019.2935715.

[34] Ali Borji, Ming-Ming Cheng, Huaizu Jiang, and Jia Li. “Salient Object Detection: A Survey”. In: *CoRR* abs/1411.5878 (2014). arXiv: 1411.5878. URL: http://arxiv.org/abs/1411.5878.

[35] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. USA: Cambridge University Press, 2004. ISBN: 0521833787.

[36] K. Bredies and M. Holler. “A Total Variation–Based JPEG Decompression Model”. In: *SIAM Journal on Imaging Sciences* 5.1 (Jan. 2012), pp. 366–393. DOI: 10.1137/110833531.

[37] K. Bredies and D. Lorenz. *Mathematische Bildverarbeitung: Einführung in Grundlagen und moderne Theorie*. Vieweg+Teubner Verlag, 2010. ISBN: 9783834810373.

[38] David J. Brenner and Eric J. Hall. “Computed Tomography — An Increasing Source of Radiation Exposure”. In: *New England Journal of Medicine* 357.22 (2007). PMID: 18046031, pp. 2277–2284. DOI: 10.1056/NEJMra072149.

[39] Sebastian Breß, Max Heimel, Norbert Siegmund, Ladjel Bellatreche, and Gunter Saake. “GPU-Accelerated Database Systems: Survey and Open Challenges”. In: *Transactions on Large-Scale Data- and Knowledge-Centered Systems XV: Selected Papers from ADBIS 2013 Satellite Events*. Ed. by Abdelkader Hameurlain, Josef Küng, Roland Wagner, Barbara Catania, Giovanna Guerrini, Themis Palpanas, Jaroslav Pokorný, and Athena Vakali. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 1–35. ISBN: 978-3-662-45761-0. DOI: 10.1007/978-3-662-45761-0_1.

[40] H. Brezis, K. Chang, S. Li, and P.H. Rabinowitz. *Topological Methods, Variational Methods And Their Applications - Proceedings Of The Icm2002 Satellite Conference On Nonlinear Functional Analysis*. World Scientific Publishing Company, 2003. ISBN: 9789814486767.

[41] Morten Bro-Nielsen and Claus Gramkow. "Fast Fluid Registration of medical images". In: *Visualization in Biomedical Computing*. Ed. by Karl Heinz Höhne and Ron Kikinis. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 265–276. ISBN: 978-3-540-70739-4. DOI: 10.1007/BFb0046964.

[42] G.L. Brownell and C.A. Burnham. "MGH positron camera". In: *NEREM* (1972).

[43] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. "A Review of Image Denoising Algorithms, with a New One." In: *Multiscale Modeling & Simulation* 4.2 (2005), pp. 490–530. DOI: 10.1137/040616024.

[44] Ian Buck. "Stream Computing on Graphics Hardware". PhD thesis. Stanford University Graphics Lab, Sept. 2006. URL: http://graphics.stanford.edu/~ianbuck/thesis.pdf.

[45] Ninon Burgos et al. "Attenuation Correction Synthesis for Hybrid PET-MR Scanners: Application to Brain Studies". In: *IEEE Transactions on Medical Imaging* 33 (July 2014). DOI: 10.1109/TMI.2014.2340135.

[46] C. A. Burnham and G. L. Brownell. "A Multi-Crystal Positron Camera". In: *IEEE Transactions on Nuclear Science* 19.3 (June 1972), pp. 201–205. ISSN: 1558-1578. DOI: 10.1109/TNS.1972.4326726.

[47] Theofilos Chamalis and Aristidis Likas. "Region merging for image segmentation based on unimodality tests". In: *3rd International Conference on Control, Automation and Robotics (ICCAR)*. Apr. 2017. DOI: 10.1109/ICCAR.2017.7942722.

[48] Antonin Chambolle. "An Algorithm for Total Variation Minimization and Applications". In: *Journal of Mathematical Imaging and Vision* 20.1–2 (Jan. 2004), pp. 89–97. ISSN: 0924-9907. DOI: 10.1023/B:JMIV.0000011325.36760.1e.

[49] Antonin Chambolle. *Inverse problems in Image processing and Image segmentation: some mathematical and numerical aspects*. ICTP lecture notes. Abdus Salam International. Centre for Theoretical Physics, 2001. ISBN: 9789295003040.

[50] Antonin Chambolle, Vicent Caselles, Matteo Novaga, Daniel Cremers, and Thomas Pock. "An introduction to Total Variation for Image Analysis". working paper or preprint. Nov. 2009. URL: https://hal.archives-ouvertes.fr/hal-00437581.

[51] Antonin Chambolle and Pierre-Louis Lions. "Image recovery via total variation minimization and related problems". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 76.2 (Apr. 1997), pp. 167–188. ISSN: 0945-3245. DOI: 10.1007/s002110050258.

[52] Antonin Chambolle and Thomas Pock. "A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging". In: *Journal of Mathematical Imaging and Vision* 40.1 (May 2011), pp. 120–145. ISSN: 1573-7683. DOI: 10.1007/s10851-010-0251-1.

[53] Antonin Chambolle and Thomas Pock. "An introduction to continuous optimization for imaging". In: *Acta Numerica* 25 (2016), pp. 161–319. DOI: 10.1017/S096249291600009X.

[54] Tony F. Chan and Jianhong (Jackie) Shen. "Variational image inpainting". In: *Communications on Pure and Applied Mathematics* 58.5 (2005), pp. 579–619. DOI: 10.1002/cpa.20075.

[55] Rick Chartrand. "Exact Reconstruction of Sparse Signals via Nonconvex Minimization". In: *IEEE Signal Processing Letters* 14.10 (Oct. 2007), pp. 707–710. DOI: 10.1109/lsp.2007.898300.

[56] Yasheng Chen and Hongyu An. "Attenuation Correction of PET/MR Imaging". In: *Magnetic Resonance Imaging Clinics of North America* 25.2 (May 2017), pp. 245–255. DOI: 10.1016/j.mric.2016.12.001.

[57] Yasheng Chen et al. "Probabilistic Air Segmentation and Sparse Regression Estimated Pseudo CT for PET/MR Attenuation Correction". In: *Radiology* 275.2 (May 2015), pp. 562–569. DOI: 10.1148/radiol.14140810.

[58] Ming-Ming Cheng, Niloy J. Mitra, Xiaolei Huang, Philip H. S. Torr, and Shi-Min Hu. "Global Contrast based Salient Region Detection". In: *IEEE TPAMI* 37.3 (2015), pp. 569–582. DOI: 10.1109/TPAMI.2014.2345401.

[59] Ming-Ming Cheng, Ziming Zhang, Wen-Yan Lin, and Philip H. S. Torr. "BING: Binarized Normed Gradients for Objectness Estimation at 300fps". In: *IEEE CVPR*. 2014. DOI: 10.1109/CVPR.2014.414.

[60] Z.H Cho. "Circular ring transverse axial positron". In: *Reconstruction Tomography in Diagnostic Radiology and Nuclear Medicine* 363 (Feb. 1975), pp. 345–51.

[61] *Cisco Global Forecast Highlights 2020*. https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2020_Forecast_Highlights.pdf. Accessed: 2020-10-13. 2020.

[62] Lyndon Clarke, Ian Glendinning, and Rolf Hempel. "The MPI Message Passing Interface Standard". In: *Programming Environments for Massively Parallel Distributed Systems*. Ed. by Karsten M. Decker and René M. Rehmann. Basel: Birkhäuser Basel, 1994, pp. 213–218. ISBN: 978-3-0348-8534-8.

[63] Dorin Comaniciu and Peter Meer. "Mean Shift: A Robust Approach Toward Feature Space Analysis." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 24.5 (2002), pp. 603–619. DOI: 10.1109/34.1000236.

[64] Institute of Computer Graphics and Vision at TU Graz. *Publications of Thomas Pock's group*. https://www.tugraz.at/institutes/icg/research/team-pock/publications/. Accessed: 2020-10-07. 2020.

[65] Shane Cook. *CUDA Programming: A Developer's Guide to Parallel Computing with GPUs*. 1st. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2012. ISBN: 9780124159334.

[66] P. Coupe, P. Yger, S. Prima, P. Hellier, C. Kervrann, and C. Barillot. "An Optimized Blockwise Nonlocal Means Denoising Filter for 3-D Magnetic Resonance Images". In: *IEEE Transactions on Medical Imaging* 27.4 (Apr. 2008), pp. 425–441. DOI: 10.1109/tmi.2007.906087.

[67] Pierrick Coupé, José V. Manjón, Vladimir Fonov, Jens Pruessner, Montserrat Robles, and D. Louis Collins. "Patch-based segmentation using expert priors: Application to hippocampus and ventricle segmentation". In: *NeuroImage* 54.2 (Jan. 2011), pp. 940–954. DOI: 10.1016/j.neuroimage.2010.09.018.

[68] Professor Daniel Cremers. *Variational Methods for Computer Vision*. WS 2013/14, TU München https://vision.in.tum.de/teaching/ws2013/vmcv2013. Accessed: 2016-04-13.

[69] *cuBLAS NVIDIA Documentation*. http://docs.nvidia.com/cuda/cublas/. Accessed: 2019-09-30. 2019.

[70] *CUDA C++ Best Practices Guide*. https://docs.nvidia.com/cuda/cuda-c-best-practices-guide/index.html. Accessed: 2019-05-24. 2019.

[71] *CUDA C++ Programming Guide*. https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html. Accessed: 2018-04-24. 2018.

[72] L. Dagum and R. Menon. "OpenMP: an industry standard API for shared-memory programming". In: *IEEE Computational Science and Engineering* 5.1 (Jan. 1998), pp. 46–55. DOI: 10.1109/99.660313.

[73] R Damadian. "Tumor detection by nuclear magnetic resonance". In: *Science (New York, N.Y.)* 171.3976 (Mar. 1971), pp. 1151–1153. ISSN: 0036-8075. DOI: 10.1126/science.171.3976.1151.

[74] Sana Damani, Daniel R. Johnson, Mark Stephenson, Stephen W. Keckler, Eddie Yan, Michael McKeown, and Olivier Giroux. "Speculative Reconvergence for Improved SIMT Efficiency". In: *Proceedings of the 18th ACM/IEEE International Symposium on Code Generation and Optimization*. CGO 2020. San Diego, CA, USA: Association for Computing Machinery, 2020, pp. 121–132. ISBN: 9781450370479. DOI: 10.1145/3368826.3377911.

[75] Birendra Kishore Das. *Positron Emission Tomography A guide for Clinicians*. Berlin, Heidelberg: Springer India, 2015. ISBN: 978-81-322-2097-8.

[76] R. H. Dennard, F. H. Gaensslen, H. Yu, V. L. Rideout, E. Bassous, and A. R. LeBlanc. "Design of ion-implanted MOSFET's with very small physical dimensions". In: *IEEE Journal of Solid-State Circuits* 9.5 (1974), pp. 256–268. DOI: 10.1109/JSSC.1974.1050511.

[77] J. Diaz, C. Muñoz-Caro, and A. Niño. "A Survey of Parallel Programming Models and Tools in the Multi and Many-Core Era". In: *IEEE Transactions on Parallel and Distributed Systems* 23.8 (2012), pp. 1369–1386. DOI: 10.1109/TPDS.2011.308.

[78] Oxford Online Dictionary. *Definition optimize (computing)*. https://www.oxfordlearnersdictionaries.com/definition/english/optimize. Accessed: 2020-06-07. 2020.

[79] Jim Douglas and H. H. Rachford. "On the numerical solution of heat conduction problems in two and three space variables". In: *Transactions of the American Mathematical Society* 82.2 (Feb. 1956), pp. 421–421. DOI: 10.1090/s0002-9947-1956-0084194-4.

[80] Ulrich Drepper. "What Every Programmer Should Know About Memory". In: 2007.

[81] Abraham Duarte, Ángel Sánchez, Felipe Fernández, and Antonio S. Montemayor. "Improving Image Segmentation Quality through Effective Region Merging using a Hierarchical Social Metaheuristic". In: *Pattern Recognition Letters* 27.11 (2006), pp. 1239–1251. DOI: 10.1016/j.patrec.2005.07.022.

[82] Joe Duffy. *Concurrent programming on Windows*. Reading, Massachusetts: Addison-Wesley, 2008.

[83] Ivar Ekeland and Roger Témam. *Convex Analysis and Variational Problems*. Society for Industrial and Applied Mathematics, Jan. 1999. DOI: 10.1137/1.9781611971088.

[84] Abderrahim. Elmoataz, Matthieu. Toutain, and Daniel. Tenbrinck. "On the p-Laplacian and ∞-Laplacian on Graphs with Applications in Image and Data Processing". In: *SIAM Journal on Imaging Sciences* 8.4 (2015), pp. 2412–2451. DOI: 10.1137/15M1022793.

[85] Lawrence C. Evans. *Partial differential equations*. Providence, R.I.: American Mathematical Society, 2010. ISBN: 9780821849743 0821849743.

[86] Fang Xu and K. Mueller. "Accelerating popular tomographic reconstruction algorithms on commodity PC graphics hardware". In: *IEEE Transactions on Nuclear Science* 52.3 (2005), pp. 654–663. DOI: 10.1109/TNS.2005.851398.

[87] P. Favaro and S. Soatto. "A geometric approach to shape from defocus". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.3 (Mar. 2005), pp. 406–417. DOI: 10.1109/TPAMI.2005.43.

[88] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. "Efficient Graph-Based Image Segmentation". In: *International Journal of Computer Vision* 59.2 (Sept. 2004), pp. 167–181. DOI: 10.1023/b:visi.0000022288.19776.77.

[89] Arlindo R. Galvão Filho, Lauro C. Martins de Paula, Clarimar José Coelho, Telma Woerle de Lima, and Anderson da Silva Soares. "CUDA parallel programming for simulation of epidemiological models based on individuals". In: *Mathematical Methods in the Applied Sciences* 39.3 (Apr. 2015), pp. 405–411. DOI: 10.1002/mma.3490.

[90] M. J. Flynn. "Some Computer Organizations and Their Effectiveness". In: *IEEE Transactions on Computers* C-21.9 (Sept. 1972), pp. 948–960. ISSN: 2326-3814. DOI: 10.1109/TC.1972.5009071.

[91] Agner Fog. *Best C++ compiler for Windows*. https://www.agner.org/optimize/blog/read.php?i=1015.

[92] Agner Fog. *Optimizing software in C++ An optimization guide for Windows, Linux,and Mac platforms*. https://www.agner.org/optimize/optimizing_cpp.pdf.

[93] Daniel Gabay and Bertrand Mercier. "A dual algorithm for the solution of nonlinear variational problems via finite element approximation". In: *Computers & Mathematics with Applications* 2.1 (1976), pp. 17–40. ISSN: 0898-1221. DOI: https://doi.org/10.1016/0898-1221(76)90003-1.

[94] Michael Gadermayr and Andreas Uhl. "Dual-Resolution Active Contours Segmentation of Vickers Indentation Images with Shape Prior Initialization". In: *Image and Signal Processing*. Ed. by Abderrahim Elmoataz, Driss Mammass, Olivier Lezoray, Fathallah Nouboud, and Driss Aboutajdine. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 362–369. ISBN: 978-3-642-31254-0. DOI: 10.1007/978-3-642-31254-0_41.

[95] Matthieu Garrigues and Antoine Manzanera. "Video++, a Modern Image and Video Processing C++ Framework". In: *Conference on Design & Architectures for Signal & Image Processing*. Madrid, Spain, Oct. 2014. URL: https://hal.archives-ouvertes.fr/hal-01118295.

[96] Lee Gayoung, Tai Yu-Wing, and Kim Junmo. “Deep Saliency with Encoded Low level Distance Map and High Level Features”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016. DOI: 10.1109/CVPR.2016.78.

[97] Stuart Geman and Donald Geman. “Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 6.6 (Nov. 1984), pp. 721–741. ISSN: 0162-8828. DOI: 10.1109/TPAMI.1984.4767596.

[98] Guy. Gilboa and Stanley. Osher. “Nonlocal Operators with Applications to Image Processing”. In: *Multiscale Modeling & Simulation* 7.3 (2009), pp. 1005–1028. DOI: 10.1137/070698592.

[99] R. Giraud, Vinh-Thong Ta, and N. Papadakis. “SCALP: Superpixels with Contour Adherence using Linear Path”. In: *2016 23rd International Conference on Pattern Recognition (ICPR)*. 2016, pp. 2374–2379.

[100] R. Glowinski and A. Marroco. “Sur l’approximation, par éléments finis d’ordre un, et la résolution, par pénalisation-dualité d’une classe de problèmes de Dirichlet non linéaires”. fr. In: *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique* 9.R2 (1975), pp. 41–76. URL: http://www.numdam.org/item/M2AN_1975__9_2_41_0.

[101] S. Goferman, L. Zelnik-Manor, and A. Tal. “Context-Aware Saliency Detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.10 (2012), pp. 1915–1926. DOI: 10.1109/TPAMI.2011.272.

[102] Micah Goldblum, Jonas Geiping, Avi Schwarzschild, Michael Moeller, and Tom Goldstein. “Truth or backpropaganda? An empirical investigation of deep learning theory”. In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=HyxyIgHFvr.

[103] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[104] Anatoliy Granov, Andrey Stanzhevskiy, and Thomas Schwarz. *Positron Emission Tomography*. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2013. ISBN: 978-3-642-21119-5.

[105] G. Green. *An essay on the application of mathematical analysis to the theories of electricity and magnetism. Facsimile-Druck*. English. Berlin. Mayer und Müller. IX + 72 S. 4° (1889). 1889.

[106] Professor Rolf Gruetter. *Fundamentals of biomedical imaging*. Physics - master program, 2019-2020 at the Ecole Polytechnique Fédérale de Lausanne (EPFL). https://edu.epfl.ch/coursebook/en/fundamentals-of-biomedical-imaging-PHYS-438. Accessed: 2020-04-03.

[107] Carlos A. S. J. Gulo, Henrique F. de Arruda, Alex F. de Araujo, Antonio C. Sementille, and João Manuel R. S. Tavares. “Efficient parallelization on GPU of an image smoothing method based on a variational model”. In: *Journal of Real-Time Image Processing* 16.4 (Aug. 2019), pp. 1249–1261. ISSN: 1861-8219. DOI: 10.1007/s11554-016-0623-x.

[108] C. Guo and L. Zhang. “A Novel Multiresolution Spatiotemporal Saliency Detection Model and Its Applications in Image and Video Compression”. In: *IEEE Transactions on Image Processing* 19.1 (2010), pp. 185–198. DOI: 10.1109/TIP.2009.2030969.

[109] Jinlian Guo, Tao Mei, Falin Liu, and Xian-Sheng Hua. "AdOn: An Intelligent Overlay Video Advertising System". In: Jan. 2009, pp. 628–629. DOI: 10.1145/1571941.1572049.

[110] John L. Gustafson. "Reevaluating Amdahl's Law". In: *Commun. ACM* 31.5 (May 1988), pp. 532–533. ISSN: 0001-0782. DOI: 10.1145/42411.42415.

[111] J. Hadamard. "Sur les problèmes aux dérivées partielles et leur signification physique." In: *Princeton university bulletin* 13.28 (1902), pp. 49–52.

[112] Ameer Haj-Ali, Nesreen Ahmed, Ted Willke, Sophia Shao, Krste Asanovic, and Ion Stoica. "NeuroVectorizer: End-to-End Vectorization with Deep Reinforcement Learning". In: *Proceedings of the 2020 International Symposium on Code Generation and Optimization*. CGO 2020. San Diego, USA: ACM, 2020. DOI: 10.1145/3373120.

[113] Mark J. Harris. *Real-Time Cloud Simulation and Rendering*. Thesis TR03-040. University of North Carolina, 2003.

[114] Mark Harris and John Owens. *CUDA Data Parallel Primitives Library*. https://github.com/cudpp/cudpp. Accessed: 2020-09-11. 2020.

[115] John L. Hennessy and David A. Patterson. *Computer Architecture, Sixth Edition: A Quantitative Approach*. 6th. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2017. ISBN: 0128119055.

[116] Maurice Herlihy and Nir Shavit. *The Art of Multiprocessor Programming*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008. ISBN: 0123705916.

[117] hgpu.org. *High performance computing on graphics processing units: hgpu.org*. hgpu.org/. 2018.

[118] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. 2nd. USA: Society for Industrial and Applied Mathematics, 2002. ISBN: 0898715210.

[119] M. Hofmann et al. "MRI-Based Attenuation Correction for PET/MRI: A Novel Approach Combining Pattern Recognition and Atlas Registration". In: *Journal of Nuclear Medicine* 49.11 (Oct. 2008), pp. 1875–1883. DOI: 10.2967/jnumed.107.049353.

[120] Q. Hou, M. Cheng, X. Hu, A. Borji, Z. Tu, and P. Torr. "Deeply Supervised Salient Object Detection with Short Connections". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017, pp. 5300–5309. DOI: 10.1109/CVPR.2017.563.

[121] G. N. Hounsfield. "Computerized transverse axial scanning (tomography): Part 1. Description of system". In: *The British Journal of Radiology* 46.552 (1973). PMID: 4757352, pp. 1016–1022. DOI: 10.1259/0007-1285-46-552-1016.

[122] F. Huang, J. Qi, H. Lu, L. Zhang, and X. Ruan. "Salient Object Detection via Multiple Instance Learning". In: *IEEE Transactions on Image Processing* 26.4 (2017), pp. 1911–1922. DOI: 10.1109/TIP.2017.2669878.

[123] W. Hwu, K. Keutzer, and T. G. Mattson. "The Concurrency Challenge". In: *IEEE Design Test of Computers* 25.4 (2008), pp. 312–320. DOI: 10.1109/MDT.2008.110.

[124] IBM. *Power 4 The First Multi-Core, 1GHz Processor*. https://www.ibm.com/ibm/history/ibm100/us/en/icons/power4/. Accessed: 2020-10-06. 2020.

[125] Juan Eugenio Iglesias and Mert R. Sabuncu. "Multi-atlas segmentation of biomedical images: A survey". In: *Medical Image Analysis* 24.1 (2015), pp. 205–219. ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2015.06.012.

[126] Intel. *IA-32 Architectures Optimization Reference Manual*. 2007.

[127] Intel. *Intel Pentium 4 Processor Extreme Edition Supporting Hyper-Threading Technology*. http://www.intel.com/products/processor/pentium4htxe/index.htm. 2002.

[128] Intel. *Intel© Xeon© Processor E5-1650 v3*. https://ark.intel.com/content/www/us/en/ark/products/82765/intel-xeon-processor-e5-1650-v3-15m-cache-3-50-ghz.html.

[129] Intel. *Intel© Xeon© Processor E5-2698 v4*. https://ark.intel.com/content/www/us/en/ark/products/91753/intel-xeon-processor-e5-2698-v4-50m-cache-2-20-ghz.html.

[130] Intel. *Programming Guidelines for Vectorization*. https://software.intel.com/sites/default/files/m/4/8/8/2/a/31848-CompilerAutovectorizationGuide.pdf.

[131] *Intel VTune Profiler*. https://software.intel.com/en-us/vtune. Accessed: 2020-05-13. 2019.

[132] ECMA International. *Standard ECMA-334 C# Language Specification*. https://www.ecma-international.org/publications/standards/Ecma-334.htm. Accessed: 2020-06-07. 2017.

[133] L. Itti, C. Koch, and E. Niebur. "A model of saliency-based visual attention for rapid scene analysis". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.11 (Nov. 1998), pp. 1254–1259. DOI: 10.1109/34.730558.

[134] David Izquierdo-Garcia et al. "An SPM8-Based Approach for Attenuation Correction Combining Segmentation and Nonrigid Template Formation: Application to Simultaneous PET/MR Brain Imaging". In: *Journal of nuclear medicine : official publication, Society of Nuclear Medicine* 55 (Oct. 2014). DOI: 10.2967/jnumed.113.136341.

[135] Varun Jampani, Deqing Sun, Ming-Yu Liu, Ming-Hsuan Yang, and Jan Kautz. "Superpixel Sampling Networks". In: *Computer Vision – ECCV 2018*. Ed. by Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss. Cham: Springer International Publishing, 2018, pp. 363–380. ISBN: 978-3-030-01234-2. DOI: 10.1007/978-3-030-01234-2_22.

[136] H. Jang, A. Park, and K. Jung. "Neural Network Implementation Using CUDA and OpenMP". In: *2008 Digital Image Computing: Techniques and Applications*. 2008, pp. 155–161. DOI: 10.1109/DICTA.2008.82.

[137] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. "Caffe: Convolutional Architecture for Fast Feature Embedding". In: *Proceedings of the 22nd ACM International Conference on Multimedia*. MM '14. Orlando, Florida, USA: Association for Computing Machinery, 2014, pp. 675–678. ISBN: 9781450330633. DOI: 10.1145/2647868.2654889.

[138] Jianbo Shi and J. Malik. "Normalized cuts and image segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.8 (Aug. 2000), pp. 888–905. ISSN: 1939-3539. DOI: 10.1109/34.868688.

[139] Christopher Joerg and Bradley C. Kuszmaul. "Massively Parallel Chess". In: *In Proceedings of the Third DIMACS Parallel Implementation Challenge, Rutgers*. 1994.

[140] Tilke Judd, Krista Ehinger, Frédo Durand, and Antonio Torralba. "Learning to Predict Where Humans Look". In: *IEEE International Conference on Computer Vision (ICCV)*. 2009. DOI: 10.1109/ICCV.2009.5459462.

[141] Subramanian Kartheeswaran and Daniel Dharmaraj Christopher Durairaj. "A data-parallelism approach for PSO-ANN based medical image reconstruction on a multi-core system". In: *Informatics in Medicine Unlocked* 8 (2017), pp. 21–31. ISSN: 2352-9148. DOI: 10.1016/j.imu.2017.05.001.

[142] Vincent Keereman, Yves Fierens, Tom Broux, Yves De Deene, Max Lonneux, and Stefaan Vandenberghe. "MRI-based attenuation correction for PET/MRI using ultrashort echo time sequences". English. In: *Journal of Nuclear Medicine* 51.5 (May 2010), pp. 812–818. ISSN: 0161-5505. DOI: 10.2967/jnumed.109.065425.

[143] John H. Kelm, Daniel R. Johnson, William Tuohy, Steven S. Lumetta, and Sanjay J. Patel. "Cohesion: An Adaptive Hybrid Memory Model for Accelerators". In: *IEEE Micro* 31.1 (Jan. 2011), pp. 42–55. ISSN: 0272-1732. DOI: 10.1109/MM.2011.8.

[144] D. Kinderlehrer and G. Stampacchia. *An Introduction to Variational Inequalities and Their Applications*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 2000. ISBN: 9780898714661.

[145] David B. Kirk and Wen-mei W. Hwu. *Programming Massively Parallel Processors: A Hands-on Approach*. Morgan Kaufmann Publishers Inc., 2010.

[146] Christof Koch and Shimon Ullman. "Shifts in Selective Visual Attention: Towards the Underlying Neural Circuitry". In: (1987). Ed. by Lucia M. Vaina, pp. 115–141. DOI: 10.1007/978-94-009-3833-5_5.

[147] George Alex Koulieris, George Drettakis, Douglas Cunningham, and Katerina Mania. "C-LOD: Context-aware Material Level-of-Detail applied to Mobile Graphics". In: *Computer Graphics Forum* (2014). ISSN: 1467-8659. DOI: 10.1111/cgf.12411.

[148] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Curran Associates, Inc., 2012, pp. 1097–1105. DOI: 10.1145/3065386.

[149] Yaroslavsky L.P. *Digital Picture Processing, an Introduction.* Berlin: Springe-Verlag, 195.

[150] P. LAUTERBUR. "Image Formation by Induced Local Interactions: Examples Employing Nuclear Magnetic Resonance". In: *Nature* 242 (5394 1973), pp. 190–191. DOI: 10.1038/242190a0.

[151] A. E. Lefohn, J. M. Kniss, C. D. Hansen, and R. T. Whitaker. "A streaming narrow-band algorithm: interactive computation and visualization of level sets". In: *IEEE Transactions on Visualization and Computer Graphics* 10.4 (2004), pp. 422–433. DOI: 10.1109/TVCG.2004.2.

[152] Charles Leiserson and Julian Shun. *6.172 Performance Engineering of Software Systems*. Fall 2018. Massachusetts Institute of Technology: MIT OpenCourseWare https://ocw.mit.edu License: Creative Commons BY-NC-SA. Accessed: 2020-06-10.

[153] Jia Li and Wen Gao. *Visual Saliency Computation*. Springer International Publishing, 2014. DOI: 10.1007/978-3-319-05642-5. URL: https://doi.org/10.1007/978-3-319-05642-5.

[154] Xinyi Li, Yinchuan Li, Hongyang Yang, Liuqing Yang, and Xiao-Yang Liu. "DP-LSTM: Differential Privacy-inspired LSTM for Stock Prediction Using Financial News". In: (Dec. 2019).

[155] Z. Li, X. Wu, and S. Chang. "Segmentation using superpixels: A bipartite graph partitioning approach". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. June 2012, pp. 789–796. DOI: 10.1109/CVPR.2012.6247750.

[156] Calvin Lin and Larry Snyder. *Principles of Parallel Programming*. 1st. USA: Addison-Wesley Publishing Company, 2008. ISBN: 0321487907.

[157] E. Lindholm, J. Nickolls, S. Oberman, and J. Montrym. "NVIDIA Tesla: A Unified Graphics and Computing Architecture". In: *IEEE Micro* 28.2 (2008), pp. 39–55. DOI: 10.1109/MM.2008.31.

[158] P. L. Lions and B. Mercier. "Splitting Algorithms for the Sum of Two Nonlinear Operators". In: *SIAM Journal on Numerical Analysis* 16.6 (1979), pp. 964–979. ISSN: 00361429. DOI: 10.2307/2156649. URL: http://www.jstor.org/stable/2156649.

[159] Risheng Liu, Junjie Cao, Zhouchen Lin, and Shiguang Shan. "Adaptive Partial Differential Equation Learning for Visual Saliency Detection". In: *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*. CVPR '14. USA: IEEE Computer Society, 2014, pp. 3866–3873. ISBN: 9781479951185. DOI: 10.1109/CVPR.2014.494.

[160] T. Liu, J. Sun, N. Zheng, X. Tang, and H. Shum. "Learning to Detect A Salient Object". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. 2007, pp. 1–8. DOI: 10.1109/CVPR.2007.383047.

[161] Peter Longhurst, Kurt Debattista, and Alan Chalmers. "A GPU Based Saliency Map for High-Fidelity Selective Rendering". In: *Proceedings of the 4th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*. AFRIGRAPH '06. Cape Town, South Africa: Association for Computing Machinery, 2006, pp. 21–29. ISBN: 1595932887. DOI: 10.1145/1108590.1108595.

[162] G. Maicas, A. I. Muñoz, G. Galiano, A. Ben Hamza, and E. Schiavi. "Spectral Shape Analysis of the Hippocampal Structure for Alzheimer's Disease Diagnosis". In: *Trends in Differential Equations and Applications*. Ed. by Francisco Ortegón Gallego, María Victoria Redondo Neble, and José Rafael Rodríguez Galván. Cham: Springer International Publishing, 2016, pp. 17–32. ISBN: 978-3-319-32013-7. DOI: 10.1007/978-3-319-32013-7_2.

[163] P Mansfield. "Multi-planar image formation using NMR spin echoes". In: *Journal of Physics C: Solid State Physics* 10.3 (Feb. 1977), pp. L55–L58. DOI: 10.1088/0022-3719/10/3/004.

[164] S. Marat, M. Guironnet, and D. Pellerin. "Video summarization using a visual attention model". In: *2007 15th European Signal Processing Conference*. 2007, pp. 1784–1788. DOI: 10.5281/zenodo.40569.

[165] W. R. Mark, R. S. Glanville, K. Akeley, and M. J. Kilgard. “Cg: A System for Programming Graphics Hardware in a C-like Language”. In: *ACM SIGGRAPH ’03, ACM New York. NY. USA* 22.3 (June 2003), pp. 896–907. DOI: `10.1145/1201775.882362`.

[166] Adrián Márques and Alvaro Pardo. “Implementation of Non Local Means Filter in GPUs”. In: *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. Ed. by José Ruiz-Shulcloper and Gabriella Sanniti di Baja. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 407–414. ISBN: 978-3-642-41822-8. DOI: `10.1007/978-3-642-41822-8_51`.

[167] D. Marr et al. *Hyper-Threading Technology Architecture and Microarchitecture. Intel Technology Journal*, Volume 06, Issue 01 February 14, 2002. 2002.

[168] Luis Marti-Bonmati, Ramon Sopena, Paula Bartumeus, and Pablo Sopena. “Multimodality imaging techniques”. In: *Contrast media & molecular imaging* (2010), pp. 180–189. DOI: `10.1002/cmmi.393`.

[169] A. Martín, J. Garamendi., and E. Schiavi. “Two efficient primal-dual algorithms for MRI Rician denoising”. In: *Computational Modelling of Objects Represented in Images - Fundamentals, Methods and Applications III, Third International Symposium, CompIMAGE 2012, Rome, Italy, September 5-7, 2012*. 2012, pp. 291–296. DOI: `10.1201/b12753-54`.

[170] A. Martín, E. Schiavi, and S. Segura de León. “On 1-Laplacian Elliptic Equations Modeling Magnetic Resonance Image Rician Denoising”. In: *Journal of Mathematical Imaging and Vision* 57.2 (July 2016), pp. 202–224. DOI: `10.1007/s10851-016-0675-3`.

[171] Axel Martinez-Möller, Michael Souvatzoglou, Gaspar Delso, Ralph Bundschuh, Sibylle Ziegler, Nassir Navab, Markus Schwaiger, and Stephan Nekolla. “Tissue Classification as a Potential Approach for Attenuation Correction in Whole-Body PET/MRI: Evaluation with PET/CT Data”. In: *Journal of nuclear medicine : official publication, Society of Nuclear Medicine* 50 (May 2009), pp. 520–6. DOI: `10.2967/jnumed.108.054726`.

[172] Timothy Mattson, Beverly Sanders, and Berna Massingill. *Patterns for Parallel Programming*. First. Addison-Wesley Professional, 2004. ISBN: 0321228111.

[173] Michael D. McCool, Zheng Qin, and Tiberiu S. Popa. “Shader Metaprogramming”. In: HWWS ’02. Saarbrucken, Germany: Eurographics Association, 2002, pp. 57–68. ISBN: 1581135807.

[174] Michael McCool, Stefanus Du Toit, Tiberiu Popa, Bryan Chan, and Kevin Moule. “Shader Algebra”. In: *ACM Trans. Graph.* 23.3 (Aug. 2004), pp. 787–795. ISSN: 0730-0301. DOI: `10.1145/1015706.1015801`.

[175] Michael McCool, James Reinders, and Arch Robison. *Structured Parallel Programming: Patterns for Efficient Computation*. 1st. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2012. ISBN: 9780123914439.

[176] S. F. McGinn and R. E. Shaw. “Parallel Gaussian elimination using OpenMP and MPI”. In: *Proceedings 16th Annual International Symposium on High Performance Computing Systems and Applications*. 2002, pp. 169–173. DOI: `10.1109/HPCSA.2002.1019151`.

[177] Tao Mei, Xian-Sheng Hua, Linjun Yang, and Shipeng Li. “VideoSense: towards effective online video advertising”. In: *Proceedings of the 15th international conference on Multimedia - MULTIMEDIA ’07*. ACM Press, 2007. DOI: `10.1145/1291233.1291467`.

[178] Richard Membarth, Frank Hannig, Jürgen Teich, Mario Körner, and Wieland Eckert. "Frameworks for Multi-core Architectures: A Comprehensive Evaluation Using 2D/3D Image Registration". In: *Architecture of Computing Systems - ARCS 2011*. Ed. by Mladen Berekovic, William Fornaciari, Uwe Brinkschulte, and Cristina Silvano. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 62–73. ISBN: 978-3-642-19137-4. DOI: 10.1007/978-3-642-19137-4_6.

[179] Yi Zhan Meng Li and Lidan Zhang. "Nonlocal Variational Model for Saliency Detection". In: *Mathematical Problems in Engineering* vol. 2013, Article ID 518747, 7 pages (2013). DOI: 10.1155/2013/518747. eprint: 518747.

[180] Duane Merrill. "Allocation-oriented Algorithm Design with Application to GPU Computing". PhD thesis. Department of Computer Science, University of Virginia, Dec. 2011. URL: https://research.nvidia.com/publication/allocation-oriented-algorithm-design-application-gpu-computing-phd-dissertation.

[181] Microsoft. *Directx Developer Site*. http://msdn.microsoft.com/directx/. Accessed: 2020-07-21. 2020.

[182] Sparsh Mittal and Jeffrey S. Vetter. "A Survey of CPU-GPU Heterogeneous Computing Techniques". In: *ACM Computing Surveys* 47.4 (July 2015), pp. 1–35. DOI: 10.1145/2788396.

[183] Hope Mogale. "High Performance Canny Edge Detector using Parallel Patterns for Scalability on Modern Multicore Processors". In: *CoRR* abs/1710.07745 (2017). arXiv: 1710.07745. URL: http://arxiv.org/abs/1710.07745.

[184] G. E. Moore. "Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp.114 ff." In: *IEEE Solid-State Circuits Society Newsletter* 11.3 (2006), pp. 33–35. DOI: 10.1109/N-SSC.2006.4785860.

[185] G. E. Moore. "Progress in digital integrated electronics [Technical literaiture, Copyright 1975 IEEE. Reprinted, with permission. Technical Digest. International Electron Devices Meeting, IEEE, 1975, pp. 11-13.]" In: *IEEE Solid-State Circuits Society Newsletter* 11.3 (2006), pp. 36–37. DOI: 10.1109/N-SSC.2006.4804410.

[186] Christian Müller-Horvat et al. "Prospective comparison of the impact on treatment decisions of whole-body magnetic resonance imaging and computed tomography in patients with metastatic malignant melanoma". In: *European journal of cancer (Oxford, England : 1990)* 42 (Mar. 2006), pp. 342–50. DOI: 10.1016/j.ejca.2005.10.008.

[187] Aaftab Munshi, Benedict Gaster, Timothy G. Mattson, James Fung, and Dan Ginsburg. *OpenCL Programming Guide*. 1st. Addison-Wesley Professional, 2011. ISBN: 9780321749642.

[188] Kengo Nakajima and Hiroshi Okuda. "Parallel Iterative Solvers for Unstructured Grids Using an OpenMP/MPI Hybrid Programming Model for the GeoFEM Platform on SMP Cluster Architectures". In: *High Performance Computing*. Ed. by Hans P. Zima, Kazuki Joe, Mitsuhisa Sato, Yoshiki Seo, and Masaaki Shimasaki. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 437–448. ISBN: 978-3-540-47847-8. DOI: 10.1007/3-540-47847-7_40.

[189] Bharath K. Navalpakkam, Harald Braun, Torsten Kuwert, and Harald H. Quick. “Magnetic Resonance–Based Attenuation Correction for PET/MR Hybrid Imaging Using Continuous Valued Attenuation Maps”. In: *Investigative Radiology* 48.5 (May 2013), pp. 323–332. DOI: 10.1097/rli.0b013e318283292f.

[190] A. Nguyen, J. Yosinski, and J. Clune. “Deep neural networks are easily fooled: High confidence predictions for unrecognizable images”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 427–436. DOI: 10.1109/CVPR.2015.7298640.

[191] J. Nickolls. “Scalable parallel programming with CUDA introduction”. In: *2008 IEEE Hot Chips 20 Symposium (HCS)*. 2008, pp. 1–9. DOI: 10.1109/HOTCHIPS.2008.7476518.

[192] C. Nieuwenhuis and D. Cremers. “Spatially Varying Color Distributions for Interactive Multi-Label Segmentation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.5 (2013), pp. 1234–1247. DOI: 10.1109/TPAMI.2012.183.

[193] Roland Norcen and Andreas Uhl. “High performance JPEG 2000 and MPEG-4 VTC on SMPs using OpenMP”. In: *Parallel Computing* 31.10 (2005). OpenMP, pp. 1082–1098. ISSN: 0167-8191. DOI: doi.org/10.1016/j.parco.2005.03.013.

[194] J. Nuyts, G. Bal, F. Kehren, M. Fenchel, C. Michel, and C. Watson. “Completion of a Truncated Attenuation Image From the Attenuated PET Emission Data”. In: *IEEE Transactions on Medical Imaging* 32.2 (2013), pp. 237–246. DOI: 10.1109/TMI.2012.2220376.

[195] NVIDIA. *Compute Unified Device Architecture programming guide 1.1*. Technical Report, Nvidia. 2007.

[196] NVIDIA. *Cuda Unbound*. https://nvlabs.github.io/cub/.

[197] NVIDIA. *NVIDIA Ampere White paper*. https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/nvidia-ampere-architecture-whitepaper.pdf. 2010.

[198] NVIDIA. *NVIDIA Developer blog: Optimized Filtering with Warp-Aggregated Atomics*. https://devblog.nvidia.com/. Accessed: 2019-08-20.

[199] NVIDIA. *NVIDIA DGX STATION PERSONAL AI SUPERCOMPUTER*. https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/dgx-station/368040-DGX-Station-DS-R11.pdf.

[200] NVIDIA. *NVIDIA Fermi White paper*. https://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf. 2009.

[201] NVIDIA. *NVIDIA Kepler White paper*. https://www.nvidia.com/content/PDF/product-specifications/GeForce_GTX_680_Whitepaper_FINAL.pdf. 2010.

[202] NVIDIA. *NVIDIA Maxwell White paper*. https://www.nvidia.com/content/PDF/product-specifications/GeForce_GTX_680_Whitepaper_FINAL.pdf. 2010.

[203] NVIDIA. *NVIDIA Pascal White paper*. https://www.nvidia.com/en-us/data-center/resources/pascal-architecture-whitepaper/. 2010.

[204] NVIDIA. *NVIDIA Turing White paper*. https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf. 2010.

[205] John Ashley at NVIDIA blog. *STAC-A3 Accelerated backtesting hedge funds*. https://blogs.nvidia.com/blog/2019/05/13/accelerated-backtesting-hedge-funds/. Accessed: 2020-10-21. 2020.

[206] *Nvidia-visual-profiler*. https://developer.nvidia.com/nvidia-visual-profiler. Accessed: 2020-05-24. 2020.

[207] P. Ochs, A. Dosovitskiy, T. Pock, and T. Brox. "An iterated L1 Algorithm for Non-smooth Non-convex Optimization in Computer Vision". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2013. URL: http://lmb.informatik.uni-freiburg.de/Publications/2013/ODB13.

[208] Peter Ochs, Yunjin Chen, Thomas Brox, and Thomas Pock. "iPiano: Inertial Proximal Algorithm for Nonconvex Optimization". In: *SIAM Journal on Imaging Sciences* 7.2 (2014), pp. 1388–1419. DOI: 10.1137/130942954.

[209] openacc.org. *The OpenACC© Application Programming Interface Version 3.0*. https://www.openacc.org/sites/default/files/inline-images/Specification/OpenACC.3.0.pdf. Accessed: 2020-10-13. 2020.

[210] OpenMP.org. *OpenMP API Specification 5.0*. https://www.openmp.org/specifications/. 2018.

[211] OpenMP.org. *Specifications - OpenMP*. https://www.openmp.org/specifications/. 2002.

[212] Stanley Osher, Martin Burger, Donald Goldfarb, Jinjun Xu, and Wotao Yin. "An Iterative Regularization Method for Total Variation-Based Image Restoration". In: *Multiscale Modeling and Simulation* 4.2 (Jan. 2005), pp. 460–489. DOI: 10.1137/040605412. URL: https://doi.org/10.1137/040605412.

[213] John D. Owens, David Luebke, Naga Govindaraju, Mark Harris, Jens Krüger, Aaron E. Lefohn, and Timothy J. Purcell. "A Survey of General-Purpose Computation on Graphics Hardware". In: *Computer Graphics Forum* 26.1 (2007), pp. 80–113. DOI: 10.1111/j.1467-8659.2007.01012.x.

[214] Mansfield P. and Maudsley A. "Medical imaging by NMR". In: *The British Journal of Radiology* 50 (591 Oct. 1977), pp. 188–194. DOI: 10.1259/0007-1285-50-591-188.

[215] *Parallel Thread Execution ISA Version 4.0*. https://docs.nvidia.com/cuda/parallel-thread-execution/index.html. Accessed: 2020-09-30. 2020.

[216] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[217] D. W. Peaceman and Jr. H. H. Rachford. "The Numerical Solution of Parabolic and Elliptic Differential Equations". In: *Journal of the Society for Industrial and Applied Mathematics* 3.1 (Mar. 1955), pp. 28–41. DOI: 10.1137/0103003.

[218] Bernd Pichler, Martin Judenhofer, and Christina Pfannenberg. "Multimodal imaging approaches: PET/CT and PET/MRI". In: *Handbook of experimental pharmacology* 185 (Feb. 2008), pp. 109–32. DOI: 10.1007/978-3-540-72718-7_6.

[219] T. Pock, M. Unger, D. Cremers, and H. Bischof. “Fast and exact solution of Total Variation models on the GPU”. In: *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. 2008, pp. 1–8. DOI: 10.1109/CVPRW.2008.4563099.

[220] Thomas Pock, Markus Grabner, and Horst Bischof. “Real-Time Computation of Variational Methods on Graphics Hardware”. English. In: *Proceedings 12th Computer Vision Winter Workshop*. 2007, pp. 67–74.

[221] V. Porpodas and T. M. Jones. “Throttling Automatic Vectorization: When Less is More”. In: *2015 International Conference on Parallel Architecture and Compilation (PACT)*. 2015, pp. 432–444. DOI: 10.1109/PACT.2015.32.

[222] M.H. Protter and H.F. Weinberger. *Maximum Principles in Differential Equations*. Partial differential equations. Springer New York, 1999. ISBN: 9780387960685. URL: https://www.springer.com/gp/book/9780387960685.

[223] E. M. Purcell, H. C. Torrey, and R. V. Pound. “Resonance Absorption by Nuclear Magnetic Moments in a Solid”. In: *Phys. Rev.* 69 (1-2 Jan. 1946), pp. 37–38. DOI: 10.1103/PhysRev.69.37.

[224] Python. *Python Software Foundation. Python Language Reference, version 2.7.* http://www.python.org. Accessed: 2020-06-07. 2020.

[225] I. Ramírez, G. Galiano, and E. Schiavi. *Non-convex non-local reactive flows for saliency detection and segmentation. Journal of Computational and Applied Mathematics*. e-print: https://doi.org/10.1016/j.cam.2020.112873. 2020. arXiv: 1805.09408 [cs.CV].

[226] I. Ramírez, A. Martín, and E. Schiavi. “Optimization of a variational model using deep learning: An application to brain tumor segmentation”. In: *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*. 2018, pp. 631–634. DOI: 10.1109/ISBI.2018.8363654.

[227] Iván Ramírez. *Variational and Deep Learning Methods in Computer Vision*. Thesis. 2018.

[228] Julian Rasch and Antonin Chambolle. “Inexact first-order primal–dual algorithms”. In: *Computational Optimization and Applications* 76.2 (Mar. 2020), pp. 381–430. DOI: 10.1007/s10589-020-00186-y.

[229] James Reinders. *Intel threading building blocks - outfitting C++ for multi-core processor parallelism*. O’Reilly, 2007. ISBN: 978-0-596-51480-8. URL: http://www.oreilly.com/catalog/9780596514808/index.html.

[230] C. Y Ren, V. A. Prisacariu, and I. D Reid. “gSLICr: SLIC superpixels at over 250Hz”. In: *ArXiv e-prints* (Sept. 2015). eprint: 1509.04232.

[231] J.S. Robertson, R.B. Marr, M. Rosenblum, V. Radeka, and Y.L. Yamamoto. “32-Crystal positron transverse section detector”. In: *Tomographic Imaging in Nuclear Medicine,* (Feb. 1973). Ed. by Freedman GS, pp. 142–153.

[232] Matthew D. Robson and Graeme M. Bydder. “Clinical ultrashort echo time imaging of bone and other connective tissues”. In: *NMR in Biomedicine* 19.7 (2006), pp. 765–780. DOI: 10.1002/nbm.1100.

[233] R. Tyrrell Rockafellar. *Convex analysis*. Princeton Mathematical Series. Princeton, N. J.: Princeton University Press, 1970.

[234] R. Tyrrell Rockafellar. “Monotone Operators and the Proximal Point Algorithm”. In: *SIAM Journal on Control and Optimization* 14.5 (Aug. 1976), pp. 877–898. DOI: 10.1137/0314056. URL: https://doi.org/10.1137/0314056.

[235] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. ““GrabCut”: Interactive Foreground Extraction Using Iterated Graph Cuts”. In: *ACM SIGGRAPH 2004 Papers*. SIGGRAPH ’04. Los Angeles, California: Association for Computing Machinery, 2004, pp. 309–314. ISBN: 9781450378239. DOI: 10.1145/1186562.1015720.

[236] Studholme C. Rousseau F. Habas P.A. “A supervised patch-based approach for human brain labeling”. In: *IEEE Transactions on Medical Imaging* 30.10 (2011), pp. 1852–1862. DOI: 10.1109/TMI.2011.2156806.

[237] S. Roy, W.-T. Wang, A. Carass, J. L. Prince, J. A. Butman, and D. L. Pham. “PET Attenuation Correction Using Synthetic CT from Ultrashort Echo-Time MR Imaging”. In: *Journal of Nuclear Medicine* 55.12 (Nov. 2014), pp. 2071–2077. DOI: 10.2967/jnumed.114.143958.

[238] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. “Nonlinear Total Variation Based Noise Removal Algorithms”. In: *Proceedings of the Eleventh Annual International Conference of the Center for Nonlinear Studies on Experimental Mathematics: Computational Issues in Nonlinear Science: Computational Issues in Nonlinear Science*. Los Alamos, New Mexico, USA: Elsevier North-Holland, Inc., 1992, pp. 259–268.

[239] M. Rumpf and R. Strzodka. “Nonlinear Diffusion in Graphics Hardware”. In: *Data Visualization 2001*. Ed. by David S. Ebert, Jean M. Favre, and Ronald Peikert. Vienna: Springer Vienna, 2001, pp. 75–84. ISBN: 978-3-7091-6215-6. DOI: 10.1007/978-3-7091-6215-6_9.

[240] K. Rungraung and P. Uthayopas. “A high performance computing for AOM stock trading order matching using GPU”. In: *2013 International Computer Science and Engineering Conference (ICSEC)*. 2013, pp. 43–46. DOI: 10.1109/ICSEC.2013.6694750.

[241] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[242] *S21760 CUDA New Features And Beyond*. https://developer.download.nvidia.com/video/gputechconf/gtc/2020/presentations/s21760-cuda-new-features-and-beyond.pdf. Accessed: 2020-05-24. 2020.

[243] S. Saini, H. Jin, R. Hood, D. Barker, P. Mehrotra, and R. Biswas. “The impact of hyper-threading on processor resource utilization in production applications”. In: *2011 18th International Conference on High Performance Computing*. 2011, pp. 1–10. DOI: 10.1109/HiPC.2011.6152743.

[244] A. Salomon, A. Goedicke, B. Schweizer, T. Aach, and V. Schulz. “Simultaneous Reconstruction of Activity and Attenuation for PET/MR”. In: *IEEE Transactions on Medical Imaging* 30.3 (2011), pp. 804–813. DOI: 10.1109/TMI.2010.2095464.

[245] Diogo Santos-Martins, Leonardo Solis-Vasquez, Andreas Koch, and Stefano Forli. “Accelerating AutoDock4 with GPUs and Gradient-Based Local Search”. In: (Aug. 2019). DOI: 10.26434/chemrxiv.9702389.v1. URL: https://chemrxiv.org/articles/Accelerating_AutoDock4_with_GPUs_and_Gradient-Based_Local_Search/9702389.

[246] S. R. Sathe and D. D. Shrimankar. "Parallelization of DNA Sequence Alignment Using OpenMP". In: *Proceedings of the 2011 International Conference on Communication, Computing amp; Security*. ICCCS '11. Rourkela, Odisha, India: Association for Computing Machinery, 2011, pp. 200–203. ISBN: 9781450304641. DOI: 10.1145/1947940.1947983.

[247] Nadathur Satish, Mark Harris, and Michael Garland. "Designing Efficient Sorting Algorithms for Manycore GPUs". In: IPDPS '09. USA: IEEE Computer Society, 2009, pp. 1–10. ISBN: 9781424437511. DOI: 10.1109/IPDPS.2009.5161005.

[248] Tao B. Schardl, I-Ting Angelina Lee, and Charles E. Leiserson. "Brief Announcement: Open Cilk". In: *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures*. SPAA '18. Vienna, Austria: Association for Computing Machinery, 2018, pp. 351–353. ISBN: 9781450357999. DOI: 10.1145/3210377.3210658.

[249] Professor Emanuele Schiavi. *Fundamentos Matemáticos Lectures*. Computer Vision Masters at Rey Juan Carlos University https://mastervisionartificial.es/asignaturas/fundamentos-matematicos. Accessed: 2020-05-04.

[250] Thomas Schiwietz, Ti-chiun Chang, Peter Speier, and Rüdiger Westermann. "MR image reconstruction using the GPU". In: vol. 6142. 2006, 61423T-61423T–12. DOI: 10.1117/12.652223.

[251] Matthias Schloegl, Martin Holler, Andreas Schwarzl, Kristian Bredies, and Rudolf Stollberger. "Infimal convolution of total generalized variation functionals for dynamic MRI". In: *Magnetic Resonance in Medicine* 78.1 (2017), pp. 142–155. ISSN: 1522-2594. DOI: 10.1002/mrm.26352.

[252] See-Mode. *Augmented Vascular Analysis to prevent strokes*. https://www.see-mode.com/product. 2020.

[253] Shubhabrata Sengupta, Mark Harris, Yao Zhang, and John D. Owens. "Scan Primitives for GPU Computing". In: *Proceedings of the 22nd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware*. GH '07. San Diego, California: Eurographics Association, 2007, pp. 97–106. ISBN: 9781595936257. DOI: 10.5555/1280094.1280110.

[254] X. Shen and Y. Wu. "A unified approach to salient object detection via low rank matrix recovery". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pp. 853–860. DOI: 10.1109/CVPR.2012.6247758.

[255] J. Shi, Q. Yan, L. Xu, and J. Jia. "Hierarchical Image Saliency Detection on Extended CSSD". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.4 (2016), pp. 717–729. DOI: 10.1109/TPAMI.2015.2465960.

[256] Isaac Shiri, Pardis Ghafarian, Parham Geramifar, Kevin Ho-Yin Leung, Mostafa Ghelichoghli, Mehrdad Oveisi, Arman Rahmim, and Mohammad Reza Ay. "Direct attenuation correction of brain PET images using only emission data via a deep convolutional encoder-decoder (Deep-DAC)". In: *European Radiology* 29.12 (June 2019), pp. 6867–6879. DOI: 10.1007/s00330-019-06229-1.

[257] Sai Srivatsa R and R Venkatesh Babu. "Salient Object Detection via Objectness Measure". In: *Image Processing (ICIP), 2015 IEEE International Conference on*. IEEE. 2015.

[258] statista.com. *Worldwide data created*. https://www.statista.com/statistics/871513/worldwide-data-created/. Accessed: 2020-09-05. 2020.

[259] Robert Stoffey and Judith Mizrachi. "MRI in Practice, 4th ed. MRI in Practice, 4th ed. By Catherine Westbrook , Carolyn Kaut Roth , and John Talbot . West Sussex, UK : Wiley-Blackwell , 456 pp. , 2011 .(ISBN: 978-1444337433)". In: *American Journal of Roentgenology* 198 (May 2012), W501–W501. DOI: 10.2214/AJR.11.8252.

[260] E. Strekalovskiy and D. Cremers. "Real-Time Minimization of the Piecewise Smooth Mumford-Shah Functional". In: *European Conference on Computer Vision (ECCV)*. 2014, pp. 127–141. DOI: 10.1007/978-3-319-10605-2_9.

[261] P. Suetens. *Fundamentals of Medical Imaging*. Cambridge medicine. Cambridge University Press, 2009. ISBN: 9780521519151.

[262] Wenyu Sun, Raimundo Sampaio, and M.A.B. Candido. "Proximal point algorithm for minimization of DC function". In: *Journal of Computational Mathematics* 21 (July 2003).

[263] United States Geological Survey. *The Water in You: Water and the Human Body*. https://www.usgs.gov/special-topic/water-science-school/science/water-you-water-and-human-body.

[264] William H. Sweet. "The Uses of Nuclear Disintegration in the Diagnosis and Treatment of Brain Tumor". In: *New England Journal of Medicine* 245.23 (1951). PMID: 14882442, pp. 875–878. DOI: 10.1056/NEJM195112062452301.

[265] Konstantin B. Tarmyshov and Florian Müller-Plathe. "Parallelizing a Molecular Dynamics Algorithm on a Multiprocessor Workstation Using OpenMP". In: *Journal of Chemical Information and Modeling* 45.6 (2005). PMID: 16309301, pp. 1943–1952. DOI: 10.1021/ci050126l.

[266] Mark Harris NVIDIA Developer Technology. *Optimizing Parallel Reduction in CUDA*. https://developer.download.nvidia.com/assets/cuda/files/reduction.pdf. Accessed: 2020-08-26. 2020.

[267] *Thread Build Blocks Intel*. https://www.threadingbuildingblocks.org/documentation. Accessed: 2020-22-02.

[268] A. N. Tikhonov and V. Y. Arsenin. "Solutions of Ill-Posed Problems". In: *SIAM Review* 21.2 (1979), pp. 266–267. DOI: 10.1137/1021044.

[269] C. Tomasi and R. Manduchi. "Bilateral filtering for gray and color images". In: *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*. 1998, pp. 839–846. DOI: 10.1109/ICCV.1998.710815.

[270] top500.org. *Roadrunner with PowerXCell*. https://www.top500.org/lists/top500/2008/06/. Accessed: 2020-10-03. 2020.

[271] Angel Torrado-Carvajal, Joaquin L Herraiz, Eduardo Alcain, Antonio S Montemayor, Lina Garcia-Cañamaque, Juan A Hernandez-Tamames, Yves Rozenholc, and Norberto Malpica. "Fast Patch-Based Pseudo-CT Synthesis from T1-Weighted MR Images for PET/MR Attenuation Correction in Brain Studies". In: *Journal of nuclear medicine : official publication, Society of Nuclear Medicine* 57.1 (Jan. 2016), pp. 136–143. ISSN: 0161-5505. DOI: 10.2967/jnumed.115.156299.

[272] Angel Torrado-Carvajal et al. "Dixon-VIBE Deep Learning (DIVIDE) Pseudo-CT Synthesis for Pelvis PET/MR Attenuation Correction". In: *Journal of Nuclear Medicine* 60.3 (Aug. 2018), pp. 429–435. DOI: 10.2967/jnumed.118.209288.

[273] Angel Torrado-Carvajal et al. “Multi-atlas and label fusion approach for patient-specific MRI based skull estimation”. In: *Magnetic Resonance in MedicineMagnetic Resonance in Medicine* (June 2015), n/a–n/a. DOI: 10.1002/mrm.25737.

[274] David W. Townsend. “Dual-Modality Imaging: Combining Anatomy and Function”. In: *Journal of Nuclear Medicine* 49.6 (2008), pp. 938–955. DOI: 10.2967/jnumed.108.051276.

[275] A M Treisman and G Gelade. “A feature-integration theory of attention”. In: *Cognit Psychol* 12.1 (Jan. 1980), pp. 97–136. DOI: 10.1016/0010-0285(80)90005-5.

[276] D. M. Tullsen, S. J. Eggers, and H. M. Levy. “Simultaneous multithreading: Maximizing on-chip parallelism”. In: *Proceedings 22nd Annual International Symposium on Computer Architecture*. 1995, pp. 392–403.

[277] L. Vincent and P. Soille. “Watersheds in digital spaces: an efficient algorithm based on immersion simulations”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13.6 (1991), pp. 583–598. DOI: 10.1109/34.87344.

[278] Vasily Volkov. “Understanding Latency Hiding on GPUs”. PhD thesis. EECS Department, University of California, Berkeley, Aug. 2016. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-143.html.

[279] G. Wagenknecht, H.J. Kaiser, F.M. Mottaghy, and H. Herzog. “MRI for attenuation correction in PET: methods and challenges”. English. In: *Magnetic Resonance Materials in Physics Biology and Medicine* 26.1 (Feb. 2013), pp. 99–113. ISSN: 0968-5243. DOI: 10.1007/s10334-012-0353-4.

[280] Dali Wang, Michael W. Berry, and Louis J. Gross. “A Parallel Structured Ecological Model for High End Shared Memory Computers”. In: *OpenMP Shared Memory Parallel Programming*. Ed. by Matthias S. Mueller, Barbara M. Chapman, Bronis R. de Supinski, Allen D. Malony, and Michael Voss. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 107–118. ISBN: 978-3-540-68555-5. DOI: 10.1007/978-3-540-68555-5_9.

[281] Yiyang Wang, Risheng Liu, Xiaoliang Song, and Zhixun Su. “Saliency Detection via Nonlocal L_0 Minimization”. In: *Computer Vision – ACCV 2014*. Ed. by Daniel Cremers, Ian Reid, Hideo Saito, and Ming-Hsuan Yang. Cham: Springer International Publishing, 2015, pp. 521–535. ISBN: 978-3-319-16808-1. DOI: 10.1007/978-3-319-16808-1_35.

[282] Tien-Hsiung Weng, Yi-Siang Chen, Huimin Lu, Mario Donato Marino, and Kuan-Ching Li. “On parallelisation of image dehazing with OpenMP”. In: *International Journal of Embedded Systems* 11.4 (2019), pp. 427–439. DOI: 10.1504/IJES.2019.100859.

[283] N. Wilt. *Cuda Handbook: A Comprehensive Guide to Gpu Programming*. CreateSpace Independent Publishing Platform, 2017. ISBN: 9781548845162.

[284] Mason Woo, Jackie Neider, Tom Davis, and Dave Shreiner. *OpenGL programming guide: the official guide to learning OpenGL, version 1.2*. Addison-Wesley Longman Publishing Co., Inc., 1999.

[285] Wm. A. Wulf and Sally A. McKee. “Hitting the Memory Wall: Implications of the Obvious”. In: *SIGARCH Comput. Archit. News* 23.1 (Mar. 1995), pp. 20–24. ISSN: 0163-5964. DOI: 10.1145/216585.216588.

[286] J. Yan, M. Zhu, H. Liu, and Y. Liu. "Visual Saliency Detection via Sparsity Pursuit". In: *IEEE Signal Processing Letters* 17.8 (2010), pp. 739–742. DOI: 10.1109/LSP.2010.2053200.

[287] Q. Yan, L. Xu, J. Shi, and J. Jia. "Hierarchical Saliency Detection". In: *2013 IEEE Conference on Computer Vision and Pattern Recognition*. June 2013, pp. 1155–1162. DOI: 10.1109/CVPR.2013.153.

[288] Dong Hye Ye, Darko Zikic, Ben Glocker, Antonio Criminisi, and Ender Konukoglu. "Modality Propagation: Coherent Synthesis of Subject-Specific Scans with Data-Driven Regularization". In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2013*. Ed. by Kensaku Mori, Ichiro Sakuma, Yoshinobu Sato, Christian Barillot, and Nassir Navab. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 606–613. ISBN: 978-3-642-40811-3.

[289] Xiaoqun Zhang, Martin Burger, and Stanley Osher. "A Unified Primal-Dual Algorithm Framework Based on Bregman Iteration". In: *Journal of Scientific Computing* 46.1 (Aug. 2010), pp. 20–46. DOI: 10.1007/s10915-010-9408-8. URL: https://doi.org/10.1007/s10915-010-9408-8.

[290] Ting Zhao and Xiangqian Wu. "Pyramid Feature Attention Network for Saliency detection". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019. DOI: 10.1109/CVPR.2019.00320.

[291] Huming Zhu, Yanfei Wu, Pei Li, Duo Wang, Wei Shi, Peng Zhang, and Licheng Jiao. "A Parallel Non-Local Means Denoising Algorithm Implementation with OpenMP and OpenCL on Intel Xeon Phi Coprocessor". In: *Journal of Computational Science* 17 (July 2016). DOI: 10.1016/j.jocs.2016.07.001.

[292] Mingqiang Zhu and Tony Chan. *An efficient primal-dual hybrid gradient algorithm for total variation image restoration*. Tech. rep. 2008.

[293] Wangjiang Zhu, Shuang Liang, Yichen Wei, and Jian Sun. "Saliency Optimization from Robust Background Detection". In: *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*. CVPR '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 2814–2821. ISBN: 978-1-4799-5118-5. DOI: 10.1109/CVPR.2014.360.

[294] Wei Zhu, Victoria Chayes Alex, Re Tiard Stephanie Sanchez, Devin Dahlberg, Da Kuang, Andrea Bertozzi, Stanley Osher, and Dominique Zosso. *Nonlocal Total Variation with Primal Dual Algorithm and Stable Simplex Clustering in Unsupervised Hyperspectral Imagery Analysis*. 2015.

[295] C. Lawrence Zitnick and Piotr Dollár. "Edge Boxes: Locating Object Proposals from Edges". In: *Computer Vision – ECCV 2014*. Ed. by David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars. Cham: Springer International Publishing, 2014, pp. 391–405. ISBN: 978-3-319-10602-1. DOI: 10.1007/978-3-319-10602-1_26.