Alma Mater Studiorum · Università di Bologna

Corso di Laurea magistrale in Matematica

Left Atrial Appendage Analysis: 2D Segmentation and Imaging Approaches

Supervisors:
Chiar.ma Prof. Serena Morigi
Chiar.ma Prof. Cristiana Corsi

Candidate student:
Anna Terzi

Anno Accademico 2024/2025

Introduction

Stroke risk assessment in patients with atrial fibrillation represents one of the most relevant challenges in both clinical and scientific contexts. In recent years, the digital twin approach, namely, the construction of personalized digital models based on clinical and imaging data, has opened new perspectives in the study of atrial dynamics and thrombus formation. In particular, the works "A digital twin approach for stroke risk assessment in Atrial Fibrillation Patients" and "A Proof of Concept for Computational Fluid Dynamic Analysis of the Left Atrium in Atrial Fibrillation on a Patient Specific Basis" have shown how computational fluid dynamics techniques, applied to cardiac geometries reconstructed from CT images, can provide advanced tools for thromboembolic risk assessment.

However, one of the main limitations of these studies is related to the use of CT images. While CT provides high spatial resolution, it also presents some critical issues: low temporal resolution (about 10 frames per cardiac cycle), the need to artificially simulate AF dynamics, and the invasive nature of the procedure, which makes it unsuitable for longitudinal studies. To overcome these limitations, the present work proposes replacing CT data with echocardiographic acquisitions, both transthoracic (TTE) and transesophageal (TEE). Ultrasound, in fact, offers much higher temporal resolution (up to 50 frames per cycle), allows direct image acquisition in atrial fibrillation conditions without additional simulations, and represents a non-invasive modality more suitable for repeated follow-ups in fragile populations.

The project presented in this thesis stems from the idea of replicating the procedures originally developed for CT-based studies, but replacing CT images with echocardiographic data. This methodological choice introduces several new challenges, primarily related to the intrinsic characteristics of ultrasound images, which are affected by speckle noise and exhibit lower contrast compared to CT. As a result, specific preprocessing steps are required to improve image quality and make subsequent analyses feasible.

For this reason, a part of this work is dedicated to denoising and filtering techniques aimed at reducing speckle noise and enhancing the visualization of relevant anatomical structures. Following this stage, the focus shifts to the two-dimensional segmentation of

the left atrial appendage using advanced variational methods.

The core of this thesis focuses on the study and application of different two dimensional variational segmentation methods in order to compare their effectiveness in the processing of echocardiographic images. Among the models analyzed are "Convex non-convex image segmentation", "Unsupervised Multiphase Segmentation: a phase balancing model" and "Frame based segmentation for medical images", which represent advanced mathematical approaches for the accurate reconstruction of anatomical structures. The purpose of this analysis is to evaluate the performance of these techniques, highlighting their potential and limitations in a realistic clinical context.

Finally, a direct comparison is presented between the three-dimensional segmentation obtained using the variational methods described in the previous chapters and the three-dimensional segmentation produced by the ITK-SNAP software, which performs automatic segmentation based on intensity and region-growing techniques. The aim of this comparison is to assess whether the algorithms implemented in MATLAB can serve as a valid alternative to commercial or open-source software for the segmentation of the left atrial appendage from transesophageal echocardiographic images.

Contents

Chapter 1

Preliminaries

In this first chapter we introduce the studies from which we started: "A digital twin approach for stroke risk assessment in Atrial Fibrillation Patients", [2], and "A Proof of Concept for Computational Fluid Dynamic Analysis of the Left Atrium in Atrial Fibrillation on a Patient Specific Basis", [1].

Before doing that, we present an introduction of the atrial fibrillation.

1.1 Atrial fibrillation

Atrial fibrillation (AF) is the most diffused arrhythmia in the entire population, its prevalence increases with age and is often associated with symptoms that can impair or reduce the quality of life. During atrial fibrillation the atrial electrical activity is completely disorganized and it does not correspond to efficient mechanical activity. The atrioventricular node receives lots of impulses from the atrium, it filters them and it transmits a limited number of them to the ventricle: it corresponds to the detected heart rate during the electrocardiogram. Variability in the atrioventricular conduction makes the ventricles contract irregularly. The irregular and rapid contraction of the heart chambers determines the reduction in the blood volume expelled at each systole, which alters the blood supply to the organs and sometimes causes symptoms of heart failure.

Atrial fibrillation can be classified in:

- Paroxysmal: episodes that resolve spontaneously within one week.
- Persistent: episodes lasting more than seven days and requiring medical intervention to restore normal rhythm.
- Permanent: the arrhythmia is continuous and no attempt is made, or attempts have failed, to restore sinus rhythm.

Atrial fibrillation is associated with a five-fold increase in the risk of cerebrovascular events and it is responsible for $15 - 18\%$ of all strokes: progressing episodes of AF lead to a remodeling and reshaping of the left atrium and left atrial appendage and it favors blood stasis and, consequently, risk of stroke. It has been noted that cardiovascular stroke associated with AF is due to left atrial appendage (LAA) thrombi and that this is the most common site for thrombus formation in the setting of AF. For this reason, the aim of this study is to build a patient-specific model using hemodynamic information on the left atrium and the left atrial appendage, which could clarify the hemodynamic implications of AF on a patient-specific basis.

In this context it is now important to understand LAA anatomy and functionality. We will start by describing some important anatomical aspects.

1.2 Structural and functional characterization of the LAA

LAA is a small pouch of muscular tissue that extends out from your left atrium. It is in a groove between that chamber and the one below it. In most hearts, the LAA extends between the anterior and the lateral walls of the left atrium, and its tip is directed anterosuperiorly, overlapping the left border of the right ventricular outflow tract or the pulmonary trunk and the main stem of the left coronary or the circumflex artery. The external appearance of the LAA is that of a slightly flattened tubular structure with crenellations, often with one or more bends and terminating in a pointed tip. There are considerable variations in its size, shape and relationship with adjacent cardiac and extracardiac structures. An increased number of lobes was associated with the presence of a thrombus independent of clinical risk and blood stasis. It has been studied, using multidetector computed tomography (MDCT) and cardiac magnetic resonance (CMR), that the shapes of the LAA in patients with drug-refractory AF were classified into 4 morphological types, with “chicken wing” being the most common (48%), followed by “cactus” (30%), “windsock” (19%), and “cauliflower”. (3%).

The function of the left atrial appendage is to manage the amount of blood in your heart: it can be seen as a spare blood, helping regulate blood flow inside the LA. It does this by releasing natriuretic peptides (proteins) when blood volume is high enough to make your LAA walls stretch too much. These peptides go into your bloodstream and encourage your kidneys to get rid of more salt and water (in pee). The peptides also dilate your blood vessels. These actions can lower your blood pressure.

Normal contraction of the LAA during sinus rhythm (SR) and adequate blood flow within the LAA lower the risk of formation of thrombi inside its cavity. Thrombus

formation is more likely to occur within the LAA when reduced contractility and stasis ensue. During AF there is a decrease in LAA contractility and function, manifest as a decrease in Doppler velocities and dilation of the LAA. The remodeling process associated with AF causes the LAA to function as a static pouch, predisposing to stagnation and thrombosis. LAA thrombi are present in up to 14% of patients with acute (< 3 days) AF. Moreover, thrombus formation may develop even in patients with AF who are receiving therapeutic anticoagulation therapy. A transesophageal echocardiography study found that 1.6% of patients treated with anticoagulation for 1 month had echocardiographic evidence of an LAA thrombus.

Computational fluid dynamics (CFD) simulations are essential for building a detailed, patient-specific profile, complementing standard clinical data. These simulations allow for an in-depth examination of blood flow within the heart, particularly in the LAA, and help identify areas where blood may stagnate, thereby increasing the risk of thrombus formation.

Given these considerations I report below a study based on Computed Tomography images.

1.3 Data analysis

In the study, "A digital twin approach for stroke risk assessment in Atrial Fibrillation Patients", it has been proposed the development of a digital twin of the LA and the LAA. This digital model would be based on patient-specific data and would simulate blood flow in real time to accurately estimate the risk of thrombus formation. By incorporating detailed anatomical information, individualized flow parameters, and the biomechanical behavior of atrial walls, the digital twin would act as an integrated platform for personalized stroke risk evaluation. This method has the potential to significantly improve the precision of risk assessment and support more customized therapeutic decisions, ultimately enhancing clinical outcomes for patients.

Thirty individuals were chosen among one hundred patients enrolled and were subjected to a Contrast-Enhanced Computed Tomography while they were in sinus rhythm, reconstructing ten volumes to cover a full cardiac cycle. For each patient it has been assessed the blood velocity fields at both the ostium and within the LAA, metrics such as time average wall shear stress (TAWSS), oscillatory shear index (OSI), relative residence time (RRT) and endothelial cell activation potential (ECAP) in the LAA.

- Wall shear stress measures the retarding force per unit area that the inner wall of the left atrium applies to the adjacent flowing blood. It represents the tangential force exerted by the blood movement on the surface of the endothelial cells and so

it can provide valuable information about areas tat promote thrombus deposition and growth. Since the heartbeat is periodic, it can be considered an average metric,

$$\text{TAWSS} = \frac{1}{T}\int_0^T \|\text{WSS}\|_2 dt,$$

where $\text{WSS} = \vec{\mu} - (\mu \cdot \vec{n})\vec{n}$, in which $\vec{\mu}$ is the viscous stress vector and $\vec{n}$ is the unit normal vector extending from the fluid to the wall.

- The oscillatory shear index,

$$\text{OSI} = \frac{1}{2}\left(1 - \frac{\| \int_0^T \vec{WSS} dt\|_2}{\int_0^T \|\vec{WSS}\|_2 dt}\right)$$

characterizes the deviation of the WSS vector from its average direction; a higher OSI value indicates greater oscillation in the flow. It is elevated in areas where the WSS fluctuates significantly throughout the cardiac cycle. The directional changes in flow primarily contribute to the higher OSI values.

- The relative residence time,

$$\text{RRT} = \frac{1}{(1 - 2\text{OSI}) * TAWSS}$$

represents the duration that blood spends close to the left atrial wall. It describes the accumulation of inflammatory cells, which can lead to degradation of the LA wall, potentially resulting in enlargement and particle deposition. This metric highlights areas of the vessel wall exposed to low shear stress and fluctuating flow patterns. Under these hemodynamic conditions, blood tends to remain near the wall for longer periods. This prolonged residence time can promote the accumulation of particles and potentially initiate an inflammatory response in the endothelial layer.

- The endothelial cell activation potential

$$\text{ECAP} = \frac{\text{OSI}}{\text{TAWSS}}$$

is used to identify areas of the wall that experience both high OSI and low TAWSS: it measures the degree of "thrombogenic susceptibility" of the wall. Higher values of the ECAP index are associated with conditions of elevated OSI and reduced TAWSS, indicating increased endo thelial susceptibility. This measure is commonly used in studies assessing thrombogenic risk within the LAA.

- To assess blood stasis, the residence time T_r is defined as the duration that blood particles remain within the LA chamber.

1.3.1 Results and discussion

The study found that blood flow velocities in the left atrium and left atrial appendage were significantly higher in control subjects compared to those with atrial fibrillation. Control participants had average LA velocities of $0,5m/s$, while PAR-AF and PER-AF groups had lower velocities, $0.2m/s$ and $0.07m/s$. Similar trends were observed in the LAA and at the LAA ostium, where control subjects consistently showed higher flow speeds. These findings suggest more efficient blood circulation and better cardiac function in individuals without AF, potentially reducing their risk of thrombus formation.

The analysis showed that control subjects had significantly higher TAWSS values at the LAA ostium compared to AF patients, indicating more effective blood flow. In contrast, AF subjects (PAR-AF and PER-AF) had lower TAWSS values, suggesting impaired flow dynamics.

OSI, which measures changes in flow direction, was higher in AF patients, but the difference was not statistically significant. This indicates more chaotic blood flow in AF, especially at the LAA tip.

RRT, indicating how long blood stays near the wall, was much higher in AF groups, pointing to slower, more stagnant flow conditions that may promote clot formation and inflammation.

Finally, ECAP values, a metric for thrombogenic potential, were also significantly higher in AF patients, further highlighting their increased risk for thrombus development in the LAA.

Secondary flow patterns, less restricted than primary ones, can greatly vary and influence blood movement near the LAA ostium, impacting residence time in that region. However, within the left atrial body, T_r remains fairly consistent across all groups at about 1 second (one cardiac cycle), even in AF patients, rarely exceeding 2 cycles. By contrast, residence time in the LAA shows notable differences: in healthy individuals, T_r remains around 1 cycle; in paroxysmal AF, T_r can reach 5 cycles at the LAA tip; in persistent AF, it is even longer. Additionally, AF patients show high variability in T_r within the proximal LAA, a pattern seen consistently across all individuals in the AF groups. A linear regression analysis was used to explore the relationship between residence time, T_r, and two key hemodynamic indicators: TAWSS and ECAP in the LAA.

The results show:

1. An inverse correlation between T_r and TAWSS: higher TAWSS values, associated with higher blood velocity and better washout, correspond to shorter residence times.

2. A positive correlation between T_r and ECAP: regions with higher ECAP, indicating greater thrombogenic potential, also exhibit longer residence times.

3. These findings suggest that in AF patients, the LAA is particularly sensitive to hemodynamic changes, emphasizing the need for careful monitoring to prevent thrombus formation.

This study demonstrated the feasibility of using computational fluid dynamics to quantitatively assess hemodynamic parameters associated with stroke risk in patients with atrial fibrillation, offering a framework for personalized risk evaluation. From the measurement that has been performed, patients with AF showed lower Wall Shear Stress (WSS) and higher values of Oscillatory Shear Index (OSI), Relative Residence Time (RRT), and Endothelial Cell Activation Potential (ECAP): these indicate slower and more oscillatory blood flow in the LAA, which can trigger endothelial activation and promote thrombosis. Furthermore, regression analysis revealed a negative correlation between WSS and T_r: higher flow velocities promote better blood washout and a positive correlation between ECAP and T_r, areas with greater thrombogenic potential show longer residence times.

From this research, we can conclude that the use of CFD offers a comprehensive and individualized approach to stroke risk assessment in AF, moving beyond population-based scores toward personalized medicine that could improve prevention and treatment outcomes.

1.4 Methods

It follows a schematic summary of the workflow developed in this research. Firstly, the use of the dynamic CT imaging to reconstruct the moving patient-specific three-dimensional atrial anatomy over the cardiac cycle. Computed tomography dynamic acquisition was performed in sinus rhythm using a 64-slice multi detector CT scanner and volumetric CT images were reconstructed for a total of ten phases from ventricular end-diastole. The processed patient data were then provided as input to the CFD solver; a representative MV flow rate of intracardiac pulsed wave (PW) Doppler in AF patients was used to set boundary conditions for the CFD simulations.

Then the definition of the anatomical model of the left atrium and the left atrial appendage was performed. In order to identify the LA and the LAA, the first volume of the CT dynamic acquisition was processed. A specific volume of interest containing the left atrium was manually selected. Within this volume, the LA was automatically identified in each eco image, and then a 3D reconstruction of the LA was generated.

To segment the LA in 2D, an adaptive thresholding method was used based on the histogram of each image. First of all, an eco image is acquired and we obtain a scan in which the LA and other anatomical structures have different grey level. Then we compute the histogram, that is the distribution of the grey level of the pixels in the image; the peak of the histogram corresponds to the region of interest. In order to compute the adaptive threshold, the maximum pixel intensity within the LA is determined as int_{max}; two thresholds, th_{down} and th_{up}, are established to define the range of accepted gray level values for the LA,

$$\text{th}_{\text{down}} = \text{int}_{\text{max}} q * \text{int}_{\text{max}}, \quad \text{th}_{\text{up}} = \text{int}_{\text{max}} + q * \text{int}_{\text{max}}$$

The constant q depended on the brightness level of int_{max}: if int_{max} was in the highest third of intensity values, $q = 5\%$, otherwise, $q = 3\%$. Gray level values between th_{down} and th_{up} were used to perform a first rough segmentation of the LA. A binary mask is generated by selecting pixels with intensity values between th_{down} and th_{up}, allowing the LA to be distinguished from surrounding structures. Since in the top slices of the axial acquisition the LAA might be disconnected from the LA chamber, we automatically detected these slices in which the two biggest connected regions corresponding to the LA and the LAA were selected. After this procedure, to refine the segmentation, the method applied morphological operations to separate regions and smooth the contours. Once applied all the steps described previously, the 3D anatomy was then obtained by stacking the two-dimensional segmentation and five cut planes were applied to the four PVs and the MV to define inflows and outflow boundary subsets of the anatomical model for the CFD simulation. This final anatomical model was used as the input for the labeling and volume mesh generation algorithm.

The next phase involved the computation of the left atrium deformation throughout the cardiac cycle. The deformation of the LA throughout the time instants of the cardiac cycle was computed by applying a 3D nonrigid image registration step of the eco volumes. Image registration is a technique that aligns two or more images to compare them and to analyze the variation over time. Rigid registration allows global transformations such as translations and rotations without changing the shape of the organ; nonrigid registration allows local variations of the shape, adapting to the physiological deformations of the organ over time. A LA reference image is selected, and a soft recording algorithm is applied to map each subsequent volume to the reference volume. This step allows to track the deformation of the LA over time. After recording, the changes in shape and volume of the left atrial appendage are analyzed, creating a dynamic model of its deformation. In this way, it is possible to compute the displacement $s_{i \to i+1}(x)$ between two successive eco volumes $Im_i(x)$ and $Im_{i+1}(x)$. Before the application of the nonrigid registration

step, we decided to perform an affine transformation which is a type of geometric transformation that preserves points, straight lines, and planes, and maintains parallelism, but it does not necessarily preserve distances or angles. An affine transformation can involve: translation, scaling, rotation, shearing and reflection and it is useful in order to roughly align the images and to prepare them for the nonrigid registration. Subsequently the result obtained was combined with the nonrigid transformation based on B-spline model. It was defined as

$$T(x) = x + \sum_{x_k \in \mathcal{N}_x} p_k \beta^3 \left(\frac{x - x_k}{\sigma} \right) \tag{1.1}$$

with x_k the control points, $\beta^3(x)$ the cubic multidimensional B-spline polynomial, p_k the B-spline coefficient vectors (the control point displacements), σ represents the spacing of the B-spline control points, and $\mathcal{N}_x$ the set of all control points within the compact support of the B-spline at x. The control points x_k were defined on a regular grid, overlayed on the fixed image. The control point grid was defined by the amount of space between the control points $\sigma = (\sigma_1, \ldots, \sigma_{d_{imm}})$, which could be different for each direction. The parameters in (1.1) were chosen for optimizing tracking quality as assessed visually. Moreover, we employed the mean square difference as the image registration measure similarity. Following this step, the global displacement of a general eco volume at time i with respect to the reference volume (time 0) was computed by increasing the successive interframe displacements:

$$s_{i \to 0}(x) = s_{i \to n-1}(x) \circ s_{i-1 \to 0}(x),$$

with $s_{0 \to 0}(x) = 0$. Therefore, considering x_0 the position of a mesh vertex at time 0, the new position x_i at time of the position i was calculated in this way

$$x_i = x_0 + s_{n \to 0}(x_0).$$

The motion field was then used to propagate the LA computational domain on the entire cardiac cycle and to simulate LA motion in AF condition. Moreover, to improve the temporal resolution, which refers to the ability to capture details over time and a high temporal resolution ensures a more continuous and accurate representation of the LA deformation in the cardiac cycle, we applied the Fourier interpolation to the displacement field. It describes the movement of the left atrial appendage points during contraction and relaxation. It is a function that assigns a shift vector over time to each point of the LA mesh. Fourier interpolation is a technique that uses the Fourier transform to reconstruct a continuous signal from discrete data. It consists in the following steps:

(i) Discrete displacement data of the LA are acquired at a few time instances.

(ii) The Fourier Transform is applied to represent the signal in the frequency domain.

(iii) The missing frequencies are interpolated, and the signal is reconstructed in the time domain using the Inverse Fourier Transform.

(iv) A continuous and smoother motion of the LA over time is obtained.

This procedure has many advantages: it increases temporal resolution without the need to acquire more data, it generates a more realistic simulation of the heartbeat and it avoids aliasing effects and distortions in the time-domain data. Thus, thanks to the Fourier interpolation, the motion was reconstructed in a smooth and continuous way, improving the simulation of atrial appendage contraction.

Therefore,we were able to recover a continuous and periodic function from the discrete data available. The reconstruction of a continuous displacement function was necessary to ensure the stability of the CFD numerical model.

Finally, the computational fluid dynamic model was computed. The fluid domain was governed by the incompressible Navier-Stokes equations in the Arbitrary Lagrangian-Eulerian (ALE) formulation, which accounts for the motion of the LA walls during the cardiac cycle. To solve the equations, a Finite Element approach was used, with the velocity and pressure fields as the primary unknowns. The blood was modeled as a Newtonian fluid, and appropriate boundary conditions were applied at various parts of the LA such as the mitral valve and pulmonary veins.

A pseudo-parabolic velocity profile was imposed at the pulmonary veins to mimic inflow conditions, computed by solving Laplacian equations. A natural-type boundary condition was introduced at the outflow boundary (mitral valve) to reduce nonphysical backflow instabilities by penalizing outward velocity.

To enhance numerical stability, especially near the fluid-structure interface, the motion of the LA geometry was incorporated, and interpolation techniques were used to maintain mesh quality. Temporal and spatial refinement was applied for accurate results.

Lastly, the flow rate profiles at each pulmonary vein were obtained from 4D flow MRI data, and simulations were personalized for each patient in sinus rhythm or atrial fibrillation.

1.5 Internship

During my internship, I was integrated into the host institution and had the opportunity to be actively involved in the activities of the Bioimaging research group, collaborating closely with the scientific team on various topics in the cardiovascular

field. In particular, I contributed by applying my mathematical knowledge to analyze the biomedical problems presented to me. This experience was very different from the purely theoretical world of mathematics I am used to; for the first time, I found myself in a laboratory setting where I could put my knowledge into practice while also learning new skills.

The aim of the internship project was to create a computational fluid dynamics model, based on the Navier-Stokes equations, of the left atrium and auricle, starting from transthoracic and transesophageal echocardiography images, rather than CT scans.

Specifically, I participated in:

- Advanced data analysis of medical imaging techniques, including computed tomography (CT), magnetic resonance imaging (MRI), and echocardiography, aimed at building personalized anatomical models based on the patient's specific characteristics;

- Development of computational models to support the diagnosis and clinical management of patients with cardiac arrhythmias, using numerical simulations and physiopathological modeling tools;

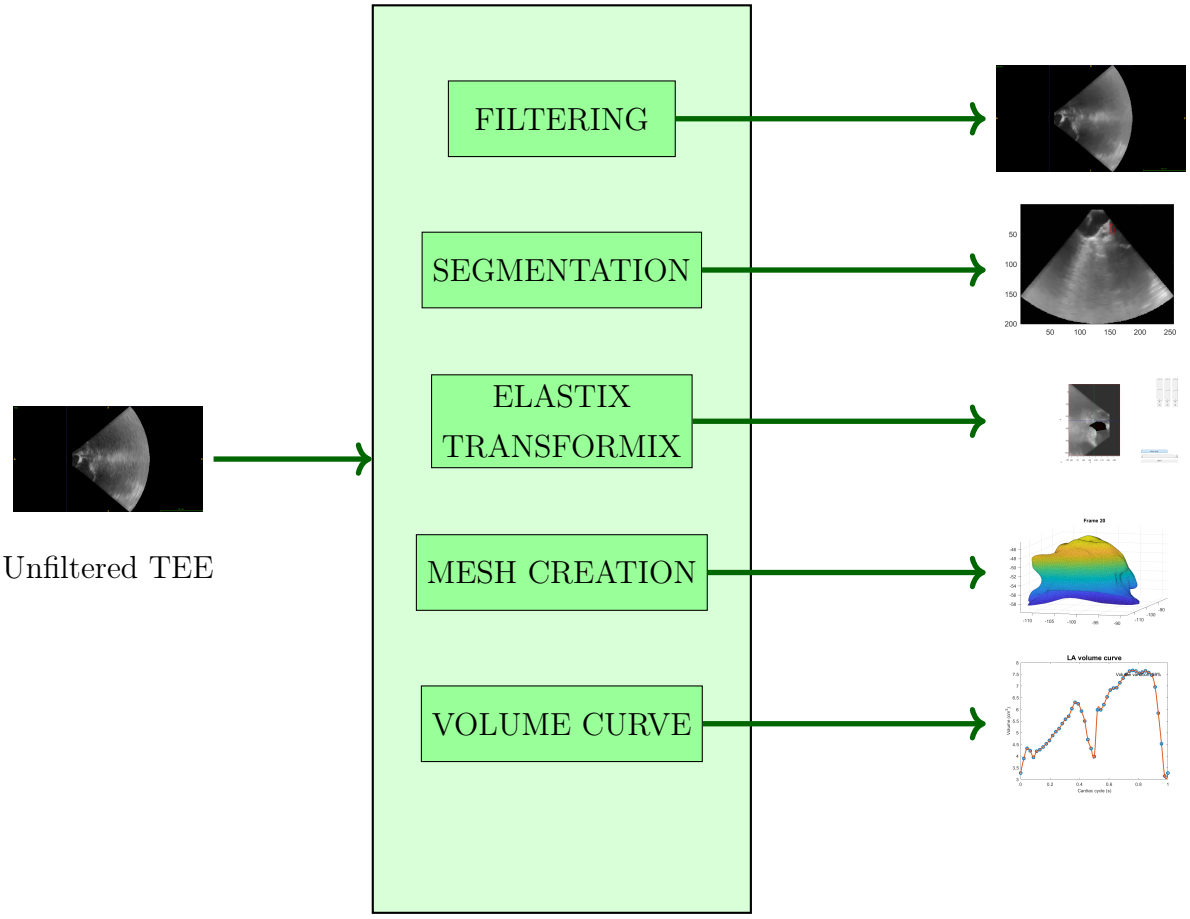
- Design and implementation of algorithms and codes for the modeling of echocardiographic images, with the goal of extracting relevant structural and functional information;

- Application of signal processing techniques, with particular focus on denoising methods and enhancing the quality of ultrasound images, to optimize visualization and diagnostic analysis;

- Use of advanced software tools such as MATLAB, ITK-SNAP, MeshLab, and ParaView.

First, I focused on studying ultrasound images and their associated noise in order to identify appropriate denoising techniques for the type of data I was working with. Then, I examined the workflow previously developed for CT images, adapting various MATLAB functions to handle ultrasound data. Building upon codes already implemented by the team, I worked on modifying and extending them for the new data type, with particular attention to implementing alternative methods for 3D segmentation, such as the Malladi approach.

Below is the workflow diagram of my work and the following chapter provides a detailed explanation of each step shown in the diagram.



Unfiltered TEE

Chapter 2

CT vs Ultrasound Imaging: A Comparative Study

There are some limitations in the studies referenced above.

1. Dynamic imaging data were utilized to derive patient-specific anatomical details and motion models during sinus rhythm and the developed workflow was evaluated under two different conditions: SR and atrial fibrillation. Unfortunately, the CT scanner available at the hospital that was involved in this study did not allow time resolved images during AF episodes, so to simulate LA motion in AF, they modeled atrial contraction by employing a random displacement applied independently to each vertex of the anatomical LA model and consisting in a sinusoidal function at a frequency of 4 Hz, defined considering the typical frequency of atrial fibrillatory, multiplied by a random factor from an uniform probability density function from 0 to 1. This aimed to replicate the disorganized and unsynchronized contractions characteristic of AF.

2. The results of the model in the two patients analyzed were not compared to experimental clinical measurements, which were not available. The aim for the future is to test the model on patients in which personalized intracardiac Doppler flow measurements are available.

3. The approach used in the study does not use patient-specific boundary conditions for the MV and PVs, the flowrates for the CFD model boundary conditions imposition can be directly computed for each patient.

4. This study does not include long-term follow-up data, limiting our understanding of how these hemodynamic parameters may evolve over time and how they correlate with actual stroke outcomes in patients with atrial fibrillation. Longitudinal studies

would be beneficial for assessing the predictive value of these parameters and for informing long-term management strategies for these patients.

5. The relatively small sample size of 30 patients may restrict the generalizability of the findings. Larger size multi-center studies are necessary to validate these results across diverse populations and to confirm that the observed trends hold true across various patient demographics and clinical settings

To address one of the limitations mentioned above, specifically the use of CT imaging data, we aim to repeat this study using ultrasound data instead. This choice is motivated by several important advantages associated with echocardiographic imaging. First, ultrasound acquisitions typically provide a significantly higher temporal resolution, often capturing up to 50 frames per cardiac cycle, compared to the 10 frames commonly available from cardiac CT. This allows for a more detailed and realistic representation of cardiac motion.

Second, echocardiographic recordings, both transthoracic (TTE) and transesophageal (TEE), can be performed while the patient is in atrial fibrillation, thus eliminating the need to artificially simulate AF-induced motion in the left atrium. This enhances the physiological accuracy of the motion model.

Third, ultrasound is a less invasive imaging modality compared to CT, making it more suitable for longitudinal and repeatable studies, especially in vulnerable patient populations.

Overall, the use of echocardiographic data is expected to improve both the physiological fidelity and clinical applicability of the developed computational framework.

2.1 CT vs Ultrasound Imaging

In this section, we present a comparative analysis between computed tomography and ultrasound imaging, focusing on their respective characteristics, the types of noise typically encountered in each modality, and the denoising techniques adopted. This comparison aims to highlight the strengths and limitations of both imaging methods in the context of their application to cardiac modeling.

Before proceeding, I will introduce the concepts of noise and signal-to-noise ratio.

2.1.1 Noise

In an image, noise refers to unwanted variations in brightness or color information, appearing as random speckles or distortions that obscure details and degrade image

quality. It's essentially a form of digital artifacting, similar to film grain in analog photography. The causes can be the following:

(i) **Low light**: Images captured in low light often have more noise due to the sensor needing to amplify the signal, which also amplifies the noise.

(ii) **High ISO settings**: Increasing the ISO on your camera makes the sensor more sensitive to light, but it also amplifies noise.

(iii) **Sensor limitations**: All digital sensors have a certain level of inherent noise.

(iv) **Heat**: Heat generated by the camera sensor can also contribute to noise.

The noise model is described by $b = N(Kx)$, where N is the noise operator (typically of probabilistic nature).

Definizione 2.1 (White noise)**.** *A discrete random image n is said to be a white noise matrix if its pixels each have a probability distribution with zero mean and finite variance, and are statistically independent (statistically uncorrelated). The covariance matrix for vector n is a diagonal matrix (a scaled multiple of the identity matrix), each element $c_{ii} = \sigma_i^2 \cdot \sigma_i^2$ is the variance of component n_i.*

Noise can be of different types: additive, multiplicative, impulsive salt and pepper and Poisson.

ADDITIVE NOISE is a type of noise that is added to an image, resulting in a corrupted image that is the sum of the original, uncorrupted image and the noise:

$$b_i = (Ax)_i + n_i, \quad i \in \Omega \equiv \{1, 2, \dots, d\}$$

- **Additive White Gaussian Noise (AWGN)**: $n_i \sim G(0, \sigma)$ where
$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$
with $\mu \in \mathbb{R}$ is the mean, $\sigma \in \mathbb{R}_{++}$ is the standard deviation, also called noise level.

- **Additive White Uniform Noise (AWUN)**: $n_i \sim U(0, \sigma)$ where
$$p(x) = \begin{cases} \frac{1}{2\sqrt{3}\sigma} & \text{for } x \in [\mu - \sqrt{3}\sigma, \mu + \sqrt{3}\sigma] \\ 0 & \text{for } x \notin [\mu - \sqrt{3}\sigma, \mu + \sqrt{3}\sigma] \end{cases}$$
with $\mu \in \mathbb{R}$ is the mean, $\sigma \in \mathbb{R}_{++}$ is the standard deviation.

- **Additive White Laplacian Noise (AWLN)**: $n_i \sim L(0, \sigma)$ where
$$p(x) = \frac{\sqrt{2}}{2\sigma} \exp\left(-\sqrt{2}\frac{|x-\mu|}{\sigma}\right)$$
with $\mu \in \mathbb{R}$ is the mean, $\sigma \in \mathbb{R}_{++}$ is the standard deviation.

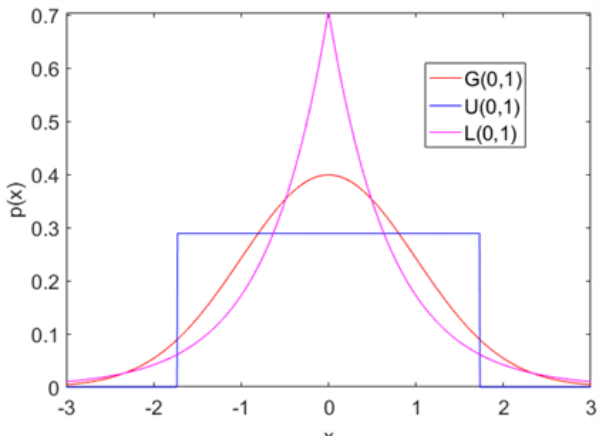


Figure 2.1: *Additive noise model*

MULTIPLICATIVE NOISE refers to an unwanted random image that gets multiplied into some relevant image during capture, transmission, or other processing. Multiplicative noise is a type of image-dependent noise in which the noise amplitude scales with the intensity of the image:

$$b_i = (Ax)_i \times n_i, \quad i \in \Omega \equiv \{1, 2, \ldots, d\}.$$

Unlike additive noise, which is independent of the image, multiplicative noise complicates processing due to its dependence on the underlying image.

- **Multiplicative White Gaussian Noise (MWGN)**: $n_i \sim G(1, \sigma)$ where

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

  with $\mu \in \mathbb{R}$ is the mean, $\sigma \in \mathbb{R}_{++}$ is the standard deviation, also called noise level.

- **Multiplicative White Uniform Noise (MWUN)**: $n_i \sim U(1, \sigma)$ where

$$p(x) = \begin{cases} \frac{1}{2\sqrt{3}\sigma} & \text{for } x \in [\mu - \sqrt{3}\sigma, \mu + \sqrt{3}\sigma] \\ 0 & \text{for } x \notin [\mu - \sqrt{3}\sigma, \mu + \sqrt{3}\sigma] \end{cases}$$

  with $\mu \in \mathbb{R}$ is the mean, $\sigma \in \mathbb{R}_{++}$ is the standard deviation.

- **Multiplicative White Laplacian Noise (MWLN)**: $n_i \sim L(1, \sigma)$ where

$$p(x) = \frac{\sqrt{2}}{2\sigma} \exp\left(-\sqrt{2}\frac{|x-\mu|}{\sigma}\right)$$

  with $\mu \in \mathbb{R}$ is the mean, $\sigma \in \mathbb{R}_{++}$ is the standard deviation.

Impulsive Salt and Pepper Noise (ISPN) is a form of noise sometimes seen on digital images. For black and white or grayscale images, it presents as sparsely occurring white and black pixels, giving the appearance of an image sprinkled with salt and pepper.

$$b_i = \begin{cases} n_i & \text{if } i \in \Omega_0 \subseteq \Omega \\ (Ax)_i & \text{if } i \in \Omega \setminus \Omega_0 \end{cases}$$

with $n_i \in \{0, 255\}$ and $\text{Prob}(n_i = 0) = \text{Prob}(n_i = 255) = 0.5$

Poisson noise is a basic form of uncertainty associated with the measurement of light, inherent to the quantized nature of light and the independence of photon detections. Its expected magnitude is signal dependent and constitutes the dominant source of image noise except in low-light conditions.

$$b_i \text{ realization of } B_i \sim \text{POISS}((Ax)_i), \quad i \in \Omega \equiv \{1, 2, \ldots, d\}$$

where $p(x|\mu) = \frac{\mu^x \exp(-\mu)}{x!} \quad \mu \in \mathbb{R}_+,\ x \in \mathbb{N}$ and $E[X] = Var[X] = \mu$

2.1.2 SNR

The Signal-to-Noise Ratio, commonly abbreviated as SNR, is a measure used to quantify the level of a desired signal relative to the level of background noise, it is used in data processing to characterize data quality. In simpler terms, it is a metric that measures how much of what you want to hear or see is present compared to what you don't want. Higher SNR values indicate a clearer signal with less interference from noise, which is essential for the quality of communication and data transmission.

SNR is typically expressed in decibels (dB), a logarithmic unit that quantifies the ratio between the difference in signal power and noise power:$\text{SNR (dB)} = 10 \times \log_{10}\left(\frac{\text{Potenza segnale}}{\text{Potenza rumore}}\right)$ and for the image we can express it as

$$\text{SNR(b\_exact, b)} := 10 \log_{10}\left(\frac{\sum_w \sum_h (\text{b\_exact(x,y)})^2}{\sum_w \sum_h (\text{b\_exact(x,y)-b(x,y)})^2}\right)$$

where b is the perceived image and b_exact is the real one. Higher SNR value indicates a clearer signal, as it means that the signal power is much greater than the noise power.

Thus, in order to obtain ultrasound images with better resolution, we have to reduce the noise and improve the SNR.

2.1.3 CT imaging

Computed Tomography is a radiological imaging technique that allows for the reconstruction of cross-sectional (tomographic) and three-dimensional images of anatomical structures, generated by computer-based analysis of the attenuation of an X-ray beam

as it passes through a section of the body. It is a non-invasive technique, as it allows for the acquisition of internal structural information without causing any damage to the sample under investigation. However, despite the significant dose reductions achieved with modern CT technology, CT scans still represent the largest source of artificial radiation exposure in the general population and approximately 70% of the total radiation dose in medical practice. Therefore, the benefits of each CT procedure must be carefully weighed against its potential risks by both physicians and patients.

During a CT scan, a narrow beam of X-rays is directed at the body and rapidly rotated around the body. The radiation detector typically consists of 4 to 64 or more rows of sensors that record the radiation passing through the body. Depending on the amount absorbed in a particular tissue, a different amount of X-rays will pass through and exit the body. The data collected by the sensors represent a series of radiographic measurements acquired from multiple angles around the subject. However, these measurements are not displayed directly; they are sent to a computer, which processes them into two-dimensional "slice-like" images (cross-sectional views) of the body. The computer can also reconstruct three-dimensional images based on the acquired slices.

In computed tomography, image noise refers to random variations in pixel intensity that do not correspond to actual anatomical structures. It can significantly impact image quality, especially when working with low-dose scans. The primary sources of noise in CT imaging include:

(i) **Quantum noise**: this is the most significant source of noise in CT images. It arises from the random nature of X-ray photon arrival at the detector. When a low radiation dose is used, fewer photons are detected, and the statistical fluctuations in their number become more pronounced. As a result, the image appears grainier and less sharp. Quantum noise follows a Poisson distribution with standard deviation $\sim \sqrt{N}$, with N being the detected photons, and becomes more prominent as photon count decreases. Quantum noise typically appears as image graininess, which can significantly degrade image quality. It reduces the visibility of fine anatomical details, decreases the contrast between adjacent structures, and may even obscure small abnormalities such as nodules or lesions. These effects are especially critical in clinical contexts where accurate visualization of subtle features is essential for diagnosis.

(ii) **Electronic noise**: this type of noise is generated by the electronic components of the detector and the data acquisition system. While it was more problematic in earlier CT scanners, it is generally less relevant in modern systems, thanks to advances in hardware design and signal processing.

(iii) **Reconstruction noise**: this noise is introduced during the image reconstruction phase. It can occur when basic algorithms or poorly optimized filters are used. In particular, reconstruction noise may distort fine anatomical details, especially if the raw data is limited or of low quality.

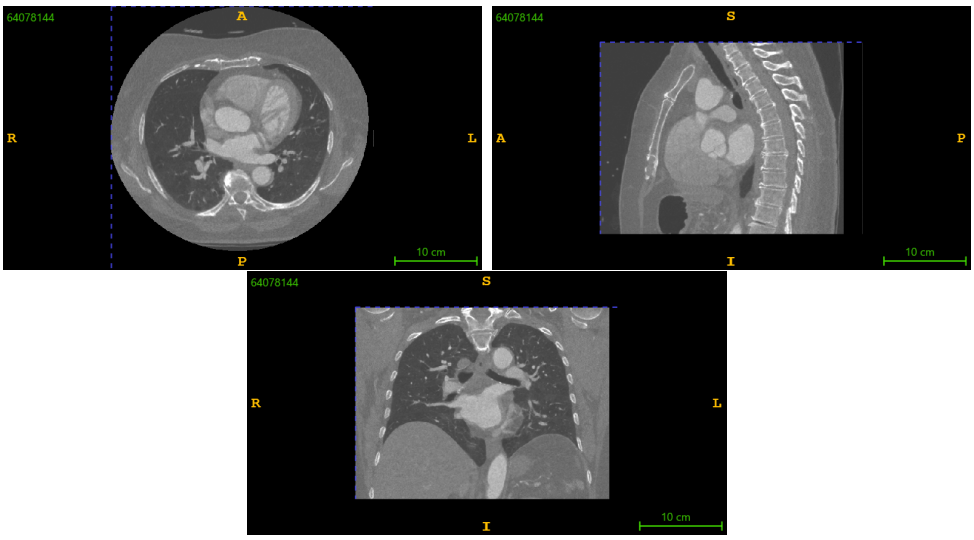


Figure 2.2: *Axial view, Sagittal view, Coronal view*

2.1.4 Eco imaging

Ultrasound is a medical diagnostic imaging technique that does not use ionizing radiation but instead employs ultrasound waves and is based on the principle of echo emission and the transmission of ultrasonic waves. Unlike the radiation used in radiology, ultrasound waves are harmless. For this reason, no special precautions are needed, and the examination can be performed on any patient as many times as necessary. This makes ultrasound a safe, non-invasive, and widely accessible diagnostic tool. It is often used as a basic or screening examination before resorting to more complex imaging techniques such as CT, MRI, or angiography.

Diagnostic ultrasound is based on the transmission of high-frequency sound waves through the body, and their reflected echoes are analyzed by a computer to create high-resolution tomographic images of organs, tissues, and blood flow. The image displayed on the monitor is the result of the interaction between the ultrasound waves and the tissues. The frequency of the ultrasound waves used typically ranges from approximately 2 MHz to 15 MHz. The frequency is selected based on the understanding that higher frequencies provide greater image resolution but penetrate less deeply into the body. Today, every ultrasound machine is equipped with so-called real-time probes, in which ultrasound waves are emitted and received sequentially in different directions through mechanical

or electronic modulation of the probe. These waves are generated by a crystal that utilizes the piezoelectric effect and is housed in a probe placed in direct contact with the patient's skin. The same probe is capable of detecting the returning echoes, which are then processed by a computer and displayed on a monitor. By adjusting the emitting aperture of the probe, it is possible to modify the beam's angle and thereby control the depth over which the ultrasound beam can be considered parallel.

Echocardiography is a technique used to study the heart and the flow of blood through its valves using ultrasound waves. This method provides valuable information about the heart's contractility, the morphology of its valves, and the blood flow within its chambers. There are various methods of performing echocardiography, transthoracic echocardiography and transesophageal echocardiography.

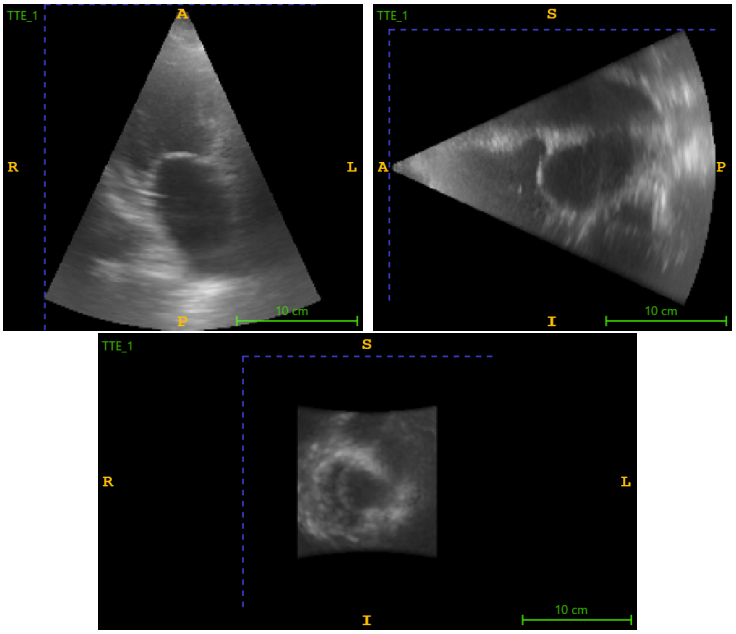


Figure 2.3: *Transthoracic Echocardiography*

Ultrasound images are affected by a type of multiplicative noise called *speckle noise*, which appears as a grainy texture superimposed on the image. It is not an external artifact or technical error, but rather a consequence of the interaction between ultrasound waves and biological tissues, especially with microscopic, heterogeneous structures like muscle fibers or cells.

Speckle arises from the wave nature of ultrasound and results from constructive and destructive interference among reflected echoes. It is a coherent, quasi-random phenomenon that occurs when coherent acoustic pulses interact with rough or inhomogeneous tissue surfaces. Many tissues contain irregular distributions of ultrasound scatterers, causing varying degrees of discontinuity that contribute to speckle.

Speckle is typically modeled as multiplicative Rayleigh noise, characterized by a con-

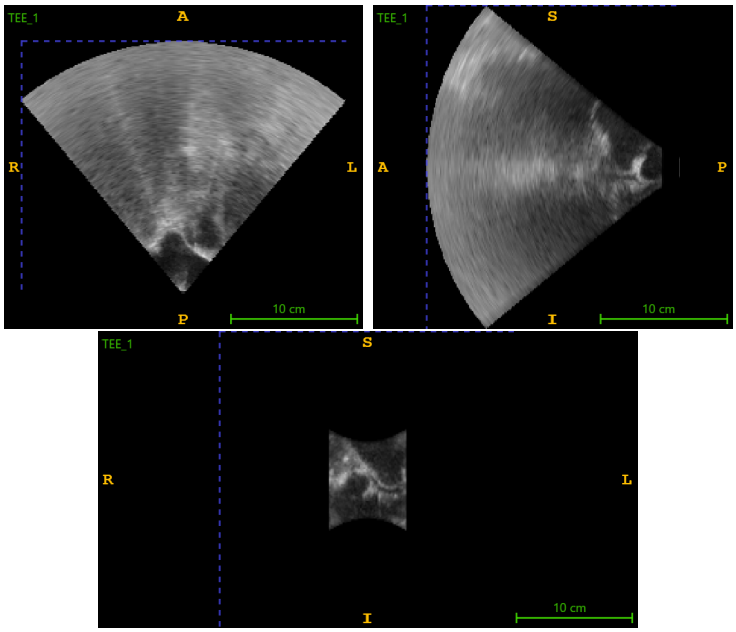


Figure 2.4: *Transesophageal Echocardiography*

stant signal-to-noise ratio. Given an observed signal b resulting from the multiplication of a clean signal component Ax and noise n, we have $b = Ax \times n$. The Rayleigh distribution is given by

$$f(b) = \begin{cases} \frac{b}{\sigma^2} \exp\left(\frac{-b^2}{2\sigma^2}\right) & \text{if } b > 0 \\ 0 & \text{otherwise} \end{cases}$$

with mean $\sigma\sqrt{\pi/2}$ and variance $\sigma^2(2 - \pi/2)$.

Although ultrasound imaging is widely used in medical diagnostics due to its non-invasive nature, low cost, and real-time capabilities, speckle noise poses a significant challenge. It reduces image contrast, obscures fine details, lowers SNR, and complicates image interpretation and processing.

Because speckle affects the variance of pixel values (rather than their mean), it creates a textured appearance that can distort important features. Therefore, it is often necessary to apply specialized speckle filtering techniques that suppress noise while preserving edges and structural details, in order to enhance image quality for further analysis or clinical use.

2.2 Denoising procedure

When comparing image noise in CT scans and ultrasounds, it's important to understand that ultrasound images generally have more noise than CT images. Ultrasound images are affected by a type of noise called speckle, which is a coherent, multiplicative noise caused by the interference of ultrasound waves reflecting off microscopic structures

in the tissue. This speckle noise appears as a grainy texture on the image and can reduce image contrast and the ability to see fine details. On the other hand, CT images are often filled with quantum noise, even if it is often depicted as additive noise, which is both random and additive. This noise comes from statistical fluctuations in the number of detected X-ray photons. Compared to ultrasound, CT images are generally cleaner and less noisy, especially when the radiation dose is sufficient.

The speckle noise in ultrasound is difficult to completely eliminate because it is an inherent physical effect of how ultrasound interacts with tissue. It also depends heavily on factors like the operator's technique, the position of the probe, and the patient's anatomy.

In summary, while ultrasound images tend to be noisier than CT images, ultrasounds offer important advantages. They don't use ionizing radiation, they are more affordable and allow for real-time imaging, making them a valuable diagnostic tool despite the increased noise.

Now I will outline the procedures we followed in processing ultrasound images, with a particular focus on transesophageal echocardiography. The workflow we implemented aligns with that employed in the reference studies above.

We started from the file `'0001 US 0012.seq.nrrd'` (our ultrasound), a NRRD file with dimensions $[46, 200, 200, 200]$, where 46 represents the time points, we scaled each 3D volume to correct the spatial proportions and obtain isotropic voxels. As a result, we obtained a file with dimensions $[46, 200, 257, 256]$. We filtered every slice of the 46 volumes to reduce noise.

We have already said that the typical noise in ultrasound images is called speckle noise. It is a multiplicative noise named so because it modulates the original image intensity by multiplying it by a factor related to wave interference.

Therefore, we used filters specifically designed to remove speckle noise. We applied various combinations of denoising filters such as imnlmfilt, histeq, imadjust, and adapthisteq. The final combination used was: **imnlmfilt**(volume(:,:,i), 'DegreeOfSmoothing', 7, 'SearchWindowSize', 21, 'ComparisonWindowSize', 7), followed by **adapthisteq**(uint8(volume(:,:,i)), 'ClipLimit', 0.001, 'NumTiles', [1616]).

We applied several filters that allowed us to enhance the resolution of the ultrasound images. The main issue we encountered concerned the boundaries: the tissues separating the different heart chambers were difficult to distinguish and poorly defined, due to speckle noise and the limitations of the ultrasound device used for acquisition. To improve the segmentation of the left atrial appendage, we introduced various filtering techniques.

- `imnlmfilt` applies a filter based on the "non-local means" method to the image

really efficient for speckle noise. "Non-local means" is a method used for denoising for digital image. Unlike classical filters, which use only the nearest pixels, NLM exploits the similarity between even distant regions of the image. The main idea is the following: if in the image there are similar areas, even if far away, then it is useful use them to make a weighted average and reduce noise without blurring too much detail.

The input arguments are "DegreeOfSmoothing", which is a positive number indicating the smoothing level of the image, if this value is not specified then it is considered the default value that is the standard deviation of noise estimated from the image; "SearchWindowSize", which is an odd-valued positive integer indicating the size of the area around each pixel used to search for similar regions and it affects the quality of denoising (the higher the value, the better the result) and the execution time (the higher the value, the slower the processing); "ComparisonWindowSize", which is an odd-valued positive integer indicating the size of the image region around each pixel that is used for comparison with other regions by computing similarity weights.

- `Anisotropic diffusion` is a filtering technique that reduces noise while preserving important edges in the image, based on a partial differential equation (PDE). In practice, the filter diffuses the image values selectively, avoiding the blurring of sharp edges: the filter is applied in a non-uniform way, high diffusion in homogeneous areas of the image (to reduce noise) and little or no diffusion near edges.

- `imadjust` maps the intensity values in grayscale image. It saturates the bottom 1% and the top 1% of all pixel values: it means that that pixels with values lower than the first are set to 0 and those with values higher than the 99th percentile are set to 1. The function linearly maps pixel values between the saturation limits to values between 0 and 1, i.e. all intermediate values are scaled proportionally. Since it expands the useful tone range this operation increases the contrast of the output image.

  The input arguments can be "[low_in, high_in]-Contrast limits for input image", "[low_out,high_out]-Contrast limits for output image" and "γ". The first is used to linearly map intensity values in the input image, scaling the range between low_in and high_in to the interval $[0, 1]$. The second is used when it is useful to map intensity values, not to the interval $[0, 1]$, but between low_out and high_out. Lastly, γ is a non-negative scalar specifying the shape of curve describing the relationship of input and output values. If γ is less than 1, then imadjust weights the mapping toward higher (brighter) output values. If γ is greater than 1, then

imadjust weights the mapping toward lower (darker) output values.

- `histeq` is used to enhance the contrast of grayscale images through histogram equalization. It transforms the grayscale input image so that the histogram of the output grayscale image with n bins (input argument) is approximately flat. "Flat" means that the gray levels are more evenly distributed, resulting in improved contrast, especially in dark or overexposed areas. When n is much smaller than the number of gray levels in the input image (e.g. 256), the output image will have an even flatter distribution, but with less details.

  An input argument is "hgram", the target histogram. It transforms the grayscale input image so that the histogram of the output grayscale image approximately matches the target histogram hgram. The number of bins in the histogram of the output image is equal to the length of hgram.

- `adapthisteq` uses a technique called Contrast-Limited Adaptive Histogram Equalization (CLAHE), that is Adaptive Histogram Equalization (AHE) and Contrast-Limited (CL): instead of equalizing the histogram of the entire image as histeq does, equalization is performed locally on small regions (blocks) of the image; this allows for improved contrast in specific areas, highlighting local details and in order to avoid excessively amplifying noise (which sometimes happens with traditional AHE), CLAHE limits the amplified contrast by setting a maximum threshold for how much the contrast can be increased in each region.

  The input arguments are "NumTiles-Number of tiles" that is the number of rectangular contextual regions (tiles) into which adapthisteq divides the image, specified as a $2-$element vector of positive integers; "ClipLimit-Contrast enhancement limit", number in the range $[0, 1]$, is a contrast factor that prevents over-saturation of the image specifically in homogeneous areas, these areas are characterized by a high peak in the histogram of the particular image tile due to many pixels falling inside the same gray level range; "NBins" that corresponds to the number of histogram bins used to build a contrast enhancing transformation, specified as a positive integer, higher values result in greater dynamic range at the cost of slower processing speed; "Range-Range of output data", "Distribution" that describes the desired histogram shape (uniform, rayleigh or exponential), and "α" that is a distribution parameter, specified as a nonnegative number.

2.3 Segmentation

Once the filtered 3D ultrasound volume was obtained, it was imported into ITK-SNAP, an interactive software application that allows users to navigate three-dimensional medical images, manually delineate anatomical regions of interest, and perform automatic image segmentation, where automatic segmentation was performed, with optional manual refinement, to extract the 3D segmentation of the left atrial appendage (LAA). The automatic segmentation in ITK-SNAP is based on the Active Contour Model, implemented through a variant of the Level Set Method, a mathematical technique used to represent curves or surfaces evolving over time, commonly applied to the segmentation of 2D or 3D image data. This method, classified as a deformable geometric model, relies on the theory of hypersurface evolution, where surfaces adapt according to geometric properties extracted from the image. Since the evolution is independent of any specific parameterization, evolving curves and surfaces are implicitly represented as level sets of a higher-dimensional function. This implicit representation allows the model to naturally handle complex topological changes, such as merging or splitting of regions, during the segmentation process. Let us consider the curve $\Gamma(s,t) : [0, L(t)] \times [0, T) \to \mathbb{R}^2$ with s is the arc-length of the curve at time t. The time evolution of the curve is given by

$$\Gamma_t = V, \quad \Gamma(s,0) = \Gamma_0(s)$$

$$V = V_n n + V_t t, \quad \Gamma_t = (V \cdot n)n + (V \cdot t)t, \ \Gamma(s,0) = \Gamma_0(s)$$

Using the Epstein-Gage theorem, which states that the tangential component of the

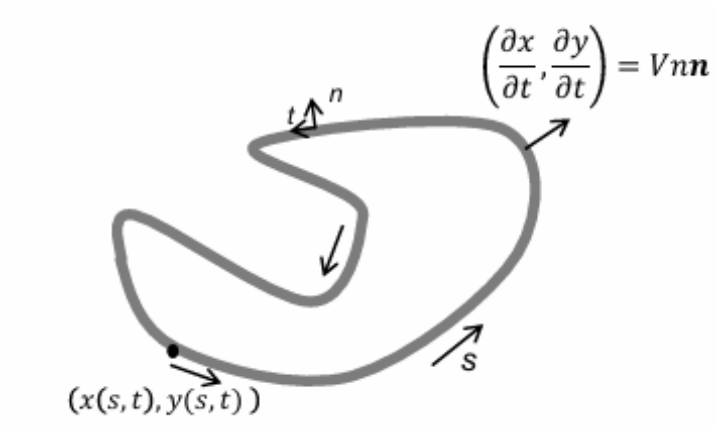


Figure 2.5: *curva*

velocity vector affects only the parametrization and not the geometric shape of the curve, we obtain: $\Gamma_t = V_n n$ and $\begin{cases} x_t = Vn\frac{y_S}{(x_s^2+y_s^2)^{1/2}} \\ y_t = Vn\frac{x_S}{(x_s^2+y_s^2)^{1/2}} \end{cases}$

The underlying idea is to represent the curve $\Gamma(s,t)$ implicitly by a level set function

$$\phi(x,y,t) : \mathbb{R}^2 \times [0,T) \to \mathbb{R}.$$

In this framework, the zero level set $\phi(x,y,t) = 0$ corresponds to the set of points lying on the curve $\Gamma(s,t)$. Therefore, the evolution of the curve Γ in time t is captured by the zero level set of the function ϕ at time t.

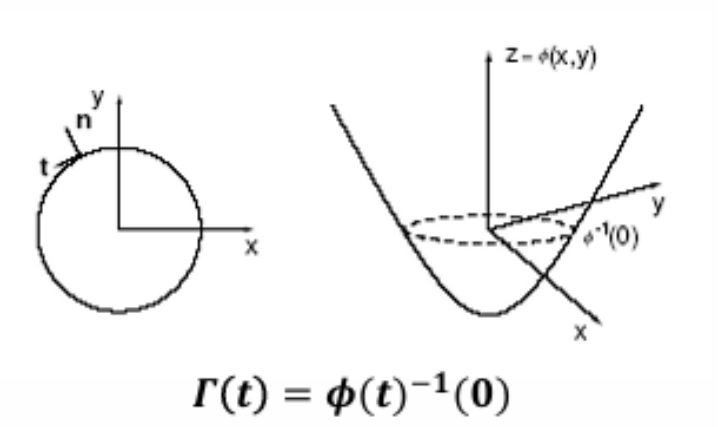


Figure 2.6

The main question is how the function $\phi(t)$ must evolve in time so that its zero level set accurately follows the motion of the curve $\Gamma(t)$. The evolution of the level set function is governed by the *level set equation*:

$$\phi_t + V \cdot \nabla\phi = 0.$$

Since $V = V_n n + V_t t$, we have

$$\phi_t + (V_n n + V_t t) \cdot \nabla\phi = 0.$$

The tangent vector t is orthogonal to $\nabla\phi$, so $t \cdot \nabla\phi = 0$. Therefore, the level set equation becomes

$$\phi_t + V_n n \cdot \nabla\phi = 0.$$

Computing the dot product

$$n \cdot \nabla\phi = \frac{\nabla\phi}{|\nabla\phi|} \cdot \nabla\phi = \frac{|\nabla\phi|^2}{|\nabla\phi|} = |\nabla\phi|$$

we obtain

$$\phi_t + V_n n \cdot \nabla\phi = 0 \Rightarrow \phi_t + V_n|\nabla\phi| = 0.$$

This leads to the Eulerian formulation of the level set evolution

$$\begin{cases} \phi_t + V_n|\nabla\phi| = 0 \\ \phi(x,y,0) = \phi_0(x,y) \end{cases}$$

where $\phi_0(x, y)$) is defined such that $\phi_0^{-1}(0) = \Gamma_0$. The initial condition is

$$\phi(x, y, 0) = \begin{cases} -d & \text{if } (x, y) \text{ is inside the contour} \\ 0 & \text{if } (x, y) \text{ lies on the contour} \\ d & \text{if } (x, y) \text{ is outside the contour} \end{cases}$$

where d is a signed distance function. By evaluating ϕ at a generic point (x_0, y_0), one can determine its relative position to the interface:

- If $\phi(x_0, y_0) < 0$ then (x_0, y_0) lies inside the interface;
- If $\phi(x_0, y_0) > 0$ then (x_0, y_0) lies outside the interface;
- If $\phi(x_0, y_0) = 0$ then (x_0, y_0) lies on the interface.

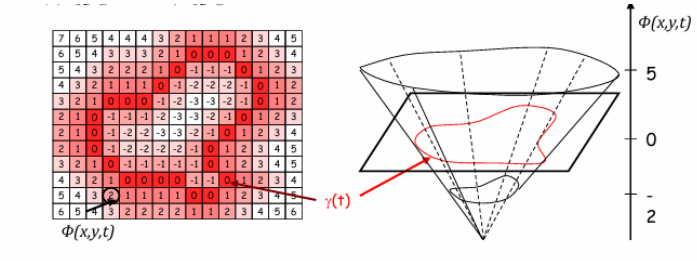


Figure 2.7

The curve evolution may be: at constant speed, curvature-driven, motion-field driven, or geodesic curvature-driven.

In particular, ITK-SNAP uses the Malladi-Sethian model, a topologically flexible approach based on the interaction of three terms:

- A **balloon term** $g(-|\nabla\phi|)$, responsible for uniform expansion or contraction of the surface;
- A **curvature term** $g\epsilon K|\nabla\phi|$ inducing motion by mean curvature;
- An **advection term** $\nu\nabla\phi\nabla g$, which pushes the contour toward image edges, following the gradient of an edge indicator function g.

This leads to the level set evolution equation

$$\begin{cases} \phi_t = g(\epsilon K - 1)|\nabla\phi| + \nu\nabla\phi \cdot \nabla g & \text{in } \Omega\times]0, \infty[\\ \phi(x, y, t) = \min(\phi_0) & \text{in } \partial\Omega\times]0, \infty[\\ \phi(x, y, t) = \phi_0(x, y) & \text{in } \Omega \end{cases}$$

The first two terms describe a weighted motion by g that smooths and inflates the interface. The inflation term is active away from the object boundaries where $g = 1$ The third term drives the level set toward the object boundaries. Parameters ϵ and ν control the relative influence of curvature and advection. The discrete update rule is

$$\phi_{ij}^{n+1} = \phi_{ij}^{n} + \Delta t \left\{ \begin{array}{l} g_{ij}\left(\epsilon K_{ij}^{n} - 1\right)\left((D_{ij}^{x})^2 + (D_{ij}^{y})^2\right)^{1/2} \\ + \left[\nu\left(\max(g_{ij}^{x}, 0)D_{ij}^{-x} + \min(g_{ij}^{x}, 0)D_{ij}^{+x}\right)\right. \\ \left. \quad + \max(g_{ij}^{y}, 0)D_{ij}^{-y} + \min(g_{ij}^{y}, 0)D_{ij}^{+y}\right] \end{array} \right\}$$

2.4 Post-segmentation processing

Once the segmentation was obtained, we used the following codes in MATLAB: `nifti_read`, `DicomtoFsl`, `main_convert2nii`, `converti_nii`, `dilate_segmentation`.

(i) `nifti_read`: is used to load, visualize, process, and save a 3D segmentation of the TEE ultrasound stored in NIfTI format. A filtered 3D image and its corresponding segmentation are loaded; the code then aligns the segmented volume correctly with the 3D image, displays them through an image, and finally saves the result (creating a binary volume Vol_fin, where each voxel with value > 0 becomes 1, and all other values become 0).

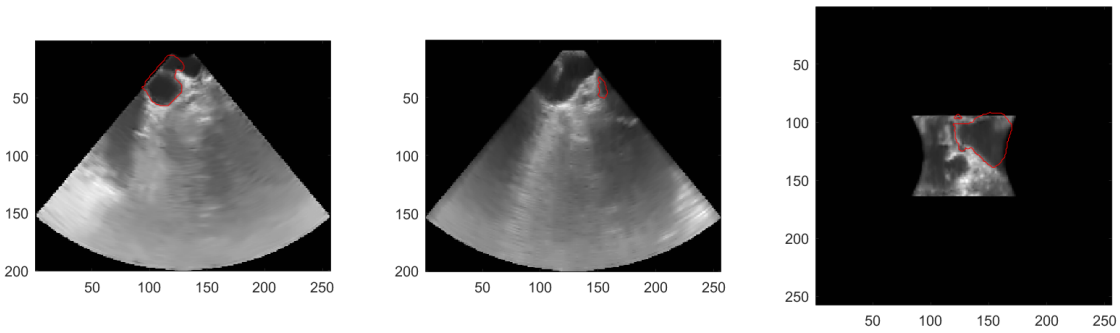


Figure 2.8: *Segmentation*

(ii) `DicomtoFsl` is used to convert DICOM image series into the NIfTI format compatible with FSL, which is commonly used in medical image processing. The software

used for the conversion is MRIConvert, via its command-line utility mcverter. The purpose of the code is to convert DICOM series (acquired from TEE ultrasound) into NIfTI (.nii) format, save the results into a structure of 46 organized folders, and use mcverter to perform the conversion into FSL-compatible NIfTI format. As a result, it creates 46 folders (named TEE_1, TEE_2, etc.), each containing the volume in hdr format.

(iii) `main_convert2nii` is used to convert medical images from the Analyze format (.hdr) to the NIfTI format (.nii), using an external software called Convert3D (c3d). The goal of the code is to: read .hdr files from a directory, sort them based on the sequence name (TEE_1, TEE_2, ...), convert them to .nii (NIfTI format) using c3d, and save the new .nii files into a separate folder. Convert3D (c3d) is a command-line tool for converting between various medical image formats (DICOM, NIfTI, HDR).

(iv) `converti_nii` has the aim to apply a volumetric segmentation to a medical image (NIfTI), visualize it, save it, and automatically crop it across all TEE images, preparing the data for subsequent analysis by focusing only on the Region of Interest (ROI). The code saves the segmentation mask in NIfTI format and performs a crop by calculating the minimum and maximum coordinates of the segmented region and adding safety margins. Finally, it crops all TEE images, producing cropped images named: "image.crop1.nii.gz", "image.crop2.nii.gz", etc.

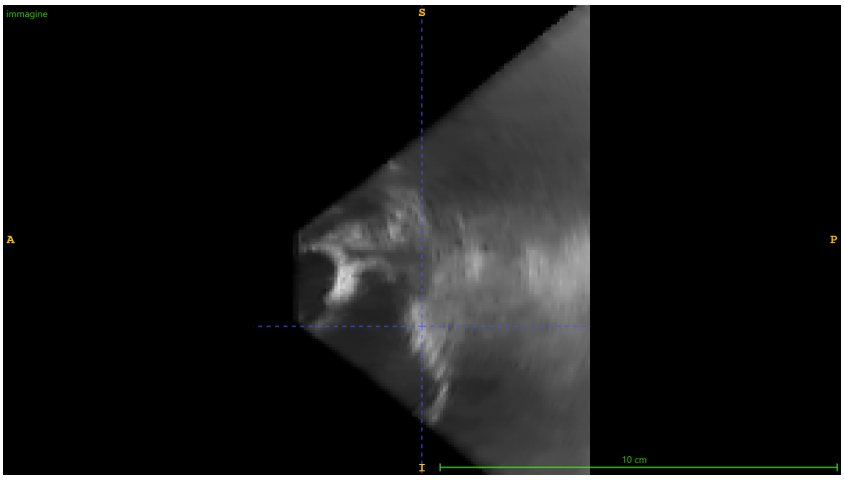


Figure 2.9: *Crop della TEE*

(v) `dilate_segmentation` has the aim to clean and binarize a segmented mask, apply a 3D dilation (i.e., expand the segmented area), and save the result as a new mask `mask.nii.gz`. To perform the dilation, each active voxel is expanded by 1 in all directions using a $3 \times 3 \times 3$ cube. This improves anatomical coverage.

2.5 Elastix and Transformix

The goal is to construct a time-varying deformable mesh to describe the motion of an anatomical structure (in our case, the left atrial appendage) based on volumetric data acquired from transesophageal echocardiography images. After data preparation, mask generation and segmentation, we will focus on the registration, and deformation of 3D medical images.

To accomplish this, we will use the software tools ELASTIX and TRANSFORMIX.

(i) **Elastix (registration)** is an open-source framework for medical image registration. It can be interpreted as the mathematical formulation of the registration process. Image registration is, in fact, an important tool in the field of medical imaging. Combination of patient data, mono- or multi- modal, often yields additional clinical information not apparent in the separate images. For this purpose, the spatial relation between the images has to be found. Image registration is the task of finding a spatial one-to-one mapping from voxels in one image to voxels in the other image.

Elastix is used to align two images (typically 3D, such as ultrasound volumes) through geometric transformations and similarity optimization. One image is considered the *moving image* $I_M(x)$, and the other the *fixed image* $I_F(x)$. Registration is the problem of finding a displacement $u(x)$ that makes $I_M(x + u(x))$ spatially aligned to $I_F(x)$, thus the goal is to transform the moving image so that it aligns as closely as possible with the fixed image.

The transformation is defined as a mapping from the fixed image to the moving image, $T : \Omega_F \subset \mathbb{R}^d \to \Omega_M \subset \mathbb{R}^d$ where d is the images dimension, Ω_F and Ω_M the spatial domain. The quality of alignment is defined by a distance or similarity

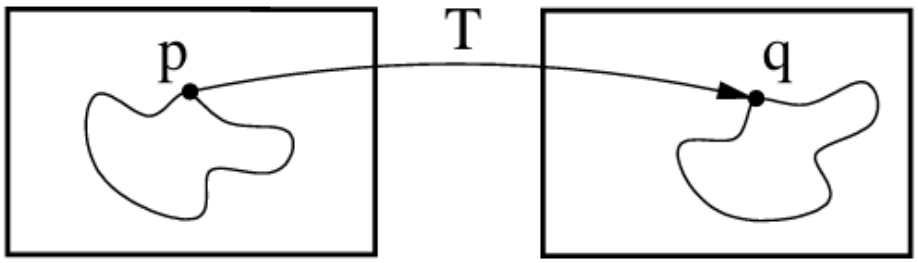


Figure 2.10: *Image registration*

measure S, such as the sum of squared differences (SSD), the correlation ratio,

or the mutual information (MI) measure. Because this problem is ill-posed for nonrigid transformations $\mathcal{T}$, a regularisation or penalty term $\mathcal{P}$ is often introduced that constrains $\mathcal{T}$.

Commonly, the registration problem is formulated as an optimisation problem in which the cost function $\mathcal{C}$ is minimised w.r.t. $\mathcal{T}$:

$$\hat{T} = \text{argmin}_T \mathcal{C}(T; I_F, I_M), \tag{2.1}$$

with

$$\mathcal{C}(T; I_F, I_M) = -\mathcal{S}(T; I_F, I_M) + \gamma \mathcal{P}(T) \tag{2.2}$$

where γ weighs similarity against regularity.

To solve the above minimisation problem, there are basically two approaches: parametric and nonparametric and Elastix is based on the parametric one. In parametric methods, the number of possible transformations is limited by introducing a parametrization (model) of the transformation. The original optimization problem thus becomes

$$\hat{T}_\mu = \text{argmin}_{T_\mu} \mathcal{C}(T_\mu; I_F, I_M) \tag{2.3}$$

where the subscript μ indicates that the transform has been parameterized. The vector μ contains the values of the "transformation parameters". For example, when the transformation is modeled as a 2D rigid transformation, the parameter vector μ contains one rotation angle and the translations in x and y direction.

We rewrite this equation as

$$\hat{\mu} = \text{argmin}_\mu \mathcal{C}(\mu; I_F, I_M) \tag{2.4}$$

From this equation it becomes clear that the original problem (2.1) has been simplified. Instead of optimizing over a space of functions $\mathcal{T}$, we now optimize over the elements of μ. Figure 2.11 shows the general components of a parametric reg-

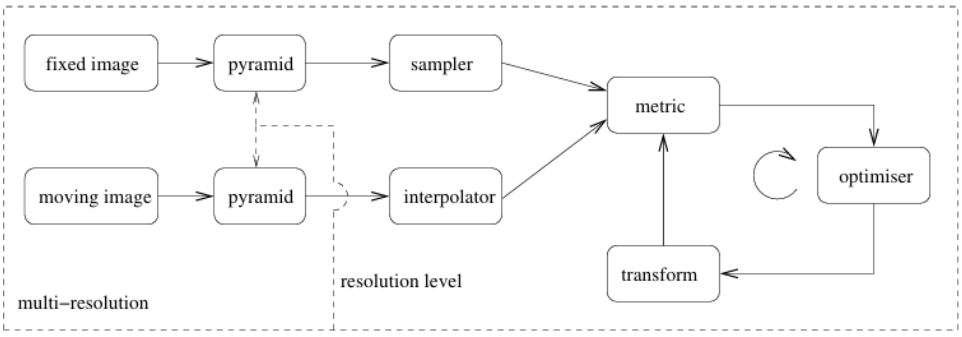


Figure 2.11: *The basic registration components.*

istration algorithm in a block scheme. First of all, we have the images. Then we have the cost function $\mathcal{C}$, or "metric", which defines the quality of alignment. As mentioned earlier, the cost function consists of a similarity measure $\mathcal{S}$ and a regularisation term $\mathcal{P}$.

Elastix takes as input:

(a) a fixed image,

(b) a moving image,

(c) optional masks (to restrict the registration to a specific region of interest, ignoring the rest),

(d) a parameter file that defines the registration settings (including the type of transformation, similarity metric, and interpolation method).

It applies a transformation, rigid, affine, or non-rigid, to align the moving image to the fixed one. A similarity metric (such as mutual information, mean squared difference, or normalized correlation) is used and optimized to find the transformation that maximizes similarity between the fixed image and the transformed moving image. Optimization methods include gradient descent, adaptive stochastic gradient descent, and others. The output consists of the transformed moving image and a file with the transformation parameters `TransformParameters.0.txt`.

In our case, the input includes:

(a) im1 = `immagine.crop1.nii.gz` (moving image),

(b) im2 = `immagine.crop2.nii.gz` (fixed image),

(c) the mask `mask.nii.gz`,

(d) the segmentation `segmentazione.crop.nii.gz`, from which the mesh will be extracted.

The process involves three main steps:

1. *Rigid Registration*: elastix performs a rigid alignment of im1 (moving) to im2 (fixed) using the mask. The output is stored in `res_elastix_1_2/rigid`.

2. *Non-Rigid Registration (B-spline)*: The rigid transformation is used as initialization for a deformable registration using a B-spline model.

3. *Inverse Transformation Calculation*: The inverse of the transformation (from im2 to im1) is computed. This inverse is necessary for deforming the mesh in step (iii).

(ii) **STL Mesh Creation from Segmentation**: The goal is to convert a 3D segmentation (in .nii format) into an STL mesh with coordinates consistent with the Elastix reference system, making it suitable for subsequent processing. The segmented mask is loaded and binarized, and a tetrahedral volumetric mesh is generated from the binary mask. The mesh points are then transformed into the Elastix coordinate system and saved as an .stl file.

(iii) **Transformix (deformation)** is a tool used to apply transformations to images or point sets. Once a transformation between two images has been estimated using elastix, transformix can be used to apply that transformation to an entire image, a mask (segmentation), or a set of points (e.g., a mesh).

After registration with elastix, a file called `TransformParameters.0.txt` is generated, which contains all the transformation parameters (rigid, affine, or B-spline). With transformix, this transformation can be applied to new data such as images, segmentations or masks, and 3D meshes (STL format). The output is `outputpoints.txt`, which contains the list of transformed points.

In our case, the mesh nodes are first written to a file named `InputPoints_1.txt`, formatted for compatibility with transformix. Then, the inverse transformation (obtained from the inverse registration) is applied to the mesh using the corresponding `TransformParameters.0.txt` file. Finally, the transformed nodes are imported, and the deformed mesh is visualized over the image im2 using an interactive 3D viewer.

This procedure is repeated to cover all 46 time frames. An iterative 3D image registration is performed using Elastix, followed by point transformation (mesh deformation) using Transformix, over a sequence of volumetric images. The sequence is processed first from frame 2 to 24 (`main_mesh_forward`), and then from frame 46 to 25 (`main_mesh_backward`).

The goal is to align each volumetric image to the next one in time (e.g., $2 \rightarrow 3$, $3 \rightarrow 4$, etc.). Afterward, the inverse transformation is applied to a set of 3D points (the anatomical mesh) to deform it over time. The transformed points (`Outputpoints`) are then saved frame by frame.

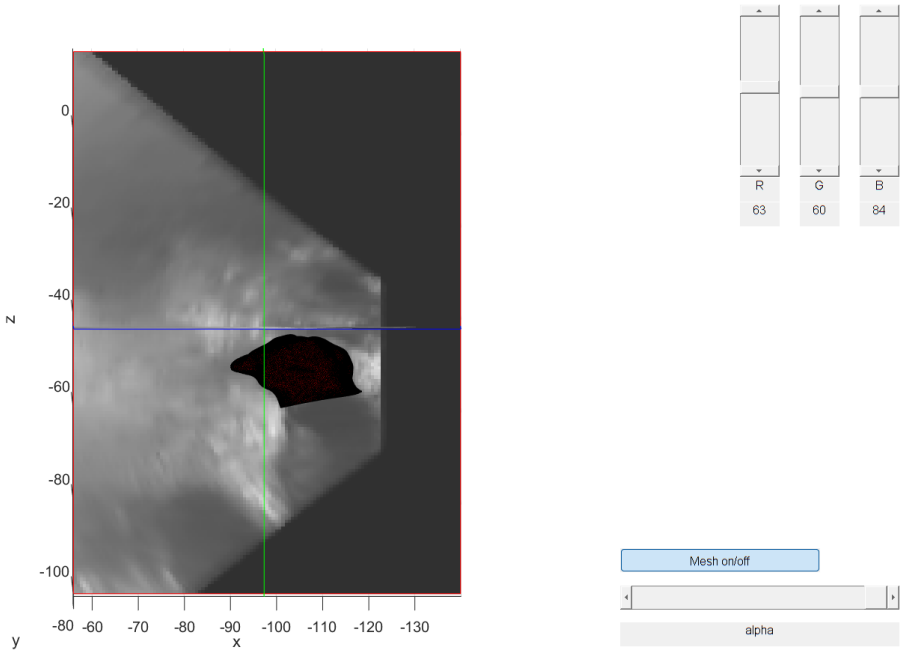


Figure 2.12: *Deformed mesh*

Chapter 3

Variational model for 2D segmentation

We chose to focus on the 2D segmentation stage and to explore alternative approaches employing different mathematical models for anatomical segmentation, in contrast to those implemented in ITK-SNAP.

Image segmentation is the process of dividing an image into regions that are homogeneous with respect to certain features, such as intensity or texture, in order to extract higher-level, more meaningful information from the image. This task is essential in numerous applications, particularly in computer vision, including object detection, recognition, measurement, and tracking. Many effective segmentation approaches rely on variational models, in which the desired regions or their boundaries are obtained by minimizing appropriately defined energy functionals.

3.1 Convex non-convex image segmentation

Starting from the paper "Convex non-convex image segmentation", [11], we present a variational model, combining convex and non-convex components, for multiphase image segmentation. Better control over segmentation boundaries and efficient management of intensity inhomogeneities are made possible by the model's newly created non-convex regularization term, which locally adjusts to image structures. The Alternating Direction Method of Multipliers (ADMM) is an effective method for solving the resulting nonlinear optimization problem.

This model is given by the sum of a smooth convex quadratic fidelity term, which measures how much the segmentation deviates from the original image, and a non-smooth non-convex regularization term, which quantifies the deviation of the segmentation from the original image:

$$\min_{u\in\mathbb{R}^n} \mathcal{J}(u;\lambda,T,a), \quad \mathcal{J}(u;\lambda,T,a) := \frac{\lambda}{2}||u-b||_2^2 + \sum_{i=1}^{n} \phi(||(\nabla u)_i||_2;T,a) \tag{3.1}$$

where $\lambda > 0$ is the regularization parameter, $b \in \mathbb{R}^n$ is the (vectorized) observed image, $(\nabla u)_i \in \mathbb{R}^2$ represents the discrete gradient of the image $u \in \mathbb{R}^n$ at pixels i, $||\cdot||_2$ denotes the l_2-norm and $\phi(\cdot; T, a) : [0, +\infty) \to \mathbb{R}$ is a parameterized, piecewise-defined non-convex penalty function with parameters $T > 0,\ a > 0$ with the following properties. The parameter a allows to tune the degree of non-convexity of the regularizer, i.e. it controls the extent to which the function ϕ is non-convex, while the parameter T is devised to represent a given gradient magnitude threshold above which the boundaries surrounding the features in the image are considered salient in a given context. This parameter is essential for determining which pixels in the image do not need to be regarded as the borders of the segmented areas. The penalty function ϕ plays two roles in the regularization term of functional $\mathcal{J}$. Since the corresponding pixels belong to the inner parts of the regions to be segmented, ϕ smooths the image values when the gradient magnitudes fall within the first interval $[0, T)$. In the interval $[T, +\infty)$, ϕ is non-convex and then flat, penalizing all possible gradient magnitudes in a manner that is roughly equivalent. The parameters λ and a determine whether the non-smooth functional $\mathcal{J}$ is convex or non-convex. The quadratic fidelity term is really highly convex, and its positive second-order derivatives have the ability to offset the regularization term's negative second-order derivatives.

The idea of optimizing convex functionals that include non-convex, sparsity-promoting terms is known as the Convex Non-Convex (CNC) strategy. This approach constructs a globally convex functional that still incorporates locally non-convex components, such as sparsity-inducing terms that encourage a limited number of boundaries and sharp, well-defined segmentations. The CNC paradigm is designed to combine the best of both worlds: the modeling flexibility and expressiveness of non-convex terms, together with the stability and uniqueness of solutions ensured by global convexity.

In this framework, promoting sparsity in the image gradient consists in encouraging the gradient magnitude to be zero almost everywhere, producing large homogeneous regions, while permitting only a small number of locations where the gradient is significant, corresponding to clear discontinuities between regions. This behavior is a direct consequence of non-convex regularization: sparsity induces piecewise-constant solutions in which boundaries are sharp and localized. Large variations are penalized less severely, as the penalty function ϕ becomes flat beyond a prescribed threshold T while small variations incur a stronger penalty.

Sparsity is typically enforced through non-convex penalty functions (such as ϕ in the proposed model), which are designed to heavily penalize small gradient magnitudes, driving them to zero, while imposing only mild penalties on large gradients, thus preserving true edges. This selective treatment enables the model to suppress spurious variations

likely attributable to noise, while maintaining significant discontinuities that correspond to genuine structural boundaries in the image.

The attractiveness of such CNC approach resides in its ability to promote sparsity more strongly than it is possible by using only convex terms while at the same time maintaining convexity of the total optimization problem, so that well-known reliable convex minimization approaches can be used to compute the (unique) solution.

The method used is a two-stage variational segmentation method: in the first stage an approximate solution u^* is computed, which represents a smoothed or regularized version of the original image, obtained by minimizing the variational functional $\mathcal{J}(u)$; once u^* is obtained, the second stage consists in segmenting the image by dividing the values into regions (phases) using thresholds. The thresholds can be selected manually, or determined automatically, for example, using K-means, a clustering algorithm that groups pixels into K classes.

In the variational model, a specially designed regularization function ϕ is introduced, serving two purposes simultaneously:

(i) Smoothing the interior regions of the areas to be segmented, thereby removing noise and unnecessary discontinuities,

(ii) Preserving important details along the boundaries (e.g., edges, contours, corners), thus avoiding the smoothing of significant features at the margins.

Achieving both objectives is challenging with a simple or purely convex regularization function, as "classical" convex regularization tends to smooth the entire image, including its edges. The preservation of boundary features in this model is made possible by the non-convex nature of the regularization function. Specifically: for small gradient magnitudes (inside homogeneous regions), the function imposes a high penalty, leading to strong smoothing; for large gradient magnitudes (at boundaries), the function applies a low or constant penalty, thereby preserving the edges.

The practical advantage of this approach lies in its relation to the original Mumford-Shah model, which aims to achieve sharp boundaries and smooth regions but is fully non-convex and thus computationally difficult to solve. The proposed method adopts the Convex-Non-Convex (CNC) strategy: the functional is constructed to retain overall (or partial) convexity while incorporating a non-convex term to ensure crisp, well-defined boundaries.

3.1.1 Construction of the penalty function

The penalty function $\phi : \mathbb{R}_+ \to \mathbb{R}$ is constructed such that it can fulfill the twofold aim: in the first interval $[0, T)$ it has to behave like a smoothing regularizer, namely a quadratic penalty, and in the second interval $[T, +\infty)$ it serves to control the length of the region boundaries, and is realized by a concave penalty function prolonged with a horizontal line. To fulfill the above requirements we used a piecewise polynomial function defined over three subdomains $[0, T), [T, T_2)$ and $[T2, \infty)$, with the following properties:

- ϕ continuously differentiable for $t \in \mathbb{R}_+$
- ϕ twice continuously differentiable for $t \in \mathbb{R}_+ \setminus \{T, T_2\}$
- ϕ convex and monotonically increasing for $t \in [0, T)$
- ϕ concave and monotonically non-decreasing for $t \in [T, T2)$
- ϕ constant for $t \in [T2, \infty)$
- $\inf_{t \in \mathbb{R}_+ \setminus \{T, T_2\}} \phi'' = -a$

The parameter T_2 is defined to allow for a good balancing between the two terms, a and T in the functional. In particular, the graph of the penalty function ϕ must be pushed down when a increases. Towards this aim, we set T_2 as $T_2(a)$ in such a way that the slope in T given by $\phi'(T; T, a) = (T_2 - T)a$ is a monotonically decreasing function of the parameter a. In this work we set $\phi'(T; T, a) = 1/a$ so that T_2 is set to be $T_2 = T + \frac{1}{a^2}$. Therefore, in the following we restrict the number of free parameters to a and T only.

The minimal degree polynomial function fulfilling the above requirements, turns out to be the following piecewise quadratic penalty function:

$$\phi(t; T, a) := \begin{cases} \phi_1(t; T, a) := \frac{a(T_2 - T)}{2T} t^2 & t \in [0, T) \\ \phi_2(t; T, a) := -\frac{a}{2} t^2 + a T_2 t - \frac{a T T_2}{2} & t \in [T, T_2) \\ \phi_3(t; T, a) := \frac{a T_2 (T_2 - T)}{2} & t \in [T_2, \infty) \end{cases} \tag{3.2}$$

which has been obtained by imposing the following constraints:

- $\phi_1(0; T, a) = \phi_1'(0; T, a) = 0$
- $\phi_1(T; T, a) = \phi_2(T; T, a)$
- $\phi_1'(T; T, a) = \phi_2'(T; T, a)$
- $\phi_2'(T_2; T, a) = 0$

- $\phi_2''(t;T,a) = -a \quad \forall t \in [T, T_2)$
- ϕ_3 constant $\quad \forall t \in [T_2, \infty)$

The use of a simple second-order piecewise polynomial as the penalty function is justified by considerations of computational efficiency.

3.1.2 Convexity analysis and ADMM algorithm

We now present sufficient conditions on the model parameters, λ, T, a, guaranteeing the strict convexity of $\mathcal{J}$. Before stating the theorem we introduce some lemmas that will be fundamental for the proof of the theorem.

First of all we rewrite $\mathcal{J}(\cdot;\lambda,T,a)$ in (3.1) in explicit double-indexed form:

$$\mathcal{J}(\cdot;\lambda,T,a) = \sum_{(i,j)\in\Omega} \frac{\lambda}{2}(u_{i,j} - b_{i,j})^2 + \sum_{(i,j)\in\Omega} \phi\left(\sqrt{(u_{i+1,j} - u_{i,j})^2 + (u_{i,j+1} - u_{i,j})^2}; T, a\right) \tag{3.3}$$

Lemma 3.1. *The function $\mathcal{J}(\cdot;\lambda,T,a) : \mathbb{R}^n \to \mathbb{R}$ defined in (3.3) is strictly convex if the function $f(\cdot;\lambda,T,a) : \mathbb{R}^3 \to \mathbb{R}$ defined by*

$$f(x_1, x_3, x_3; \lambda, T, a) = \frac{\lambda}{6}(x_1^2 + x_2^2 + x_3^2) + \phi(\sqrt{(x_2 - x_1)^2 + (x_3 - x_1)^2}; T, a) \tag{3.4}$$

is strictly convex.

Lemma 3.2. *The function $f(\cdot;\lambda,T,a) : \mathbb{R}^3 \to \mathbb{R}$ defined in (3.4) is strictly convex if the function $g(\cdot;\lambda,T,a) : \mathbb{R}^2 \to \mathbb{R}$ defined by*

$$g(y_1, y_2; \lambda, T, a) = \frac{\lambda}{18}(y_1^2 + y_2^2) + \phi\left(\sqrt{y_1^2 + y_2^2}; T, a\right) \tag{3.5}$$

is strictly convex.

Lemma 3.3. *Let $\psi : \mathbb{R}^2 \to \mathbb{R}$ be a radially symmetric function defined as*

$$\psi(x) := z(||x||_2), \quad z : \mathbb{R}_+ \to \mathbb{R}, \;\; z \in C^1(\mathbb{R}_+). \tag{3.6}$$

Then, ψ is strictly convex in x if and only if the function $\overline{z} : \mathbb{R} \to \mathbb{R}$ defined by

$$\overline{z}(t) := z(|t|) \tag{3.7}$$

is strictly convex in t.

Lemma 3.4. *The function $g(\cdot;\lambda,T,a) : \mathbb{R}^2 \to \mathbb{R}$ defined in (3.5) is strictly convex if and only if the function $h(\cdot;\lambda,T,a) : \mathbb{R} \to \mathbb{R}$ defined by*

$$h(t;\lambda,T,a) = \frac{\lambda}{18}t^2 + \phi(|t|;T,a) \tag{3.8}$$

is strictly convex.

Theorem 3.5. *Let $\phi(\cdot;T,a):\mathbb{R}_+\to\mathbb{R}$ be the penalty function defined in* (3.1)*. Then, a sufficient condition for the functional $\mathcal{J}(\cdot;\lambda,T,a)$ to be strictly convex is that the pair of parameters $(\lambda,a)\in\mathbb{R}_+^*\times\mathbb{R}_+^*$ satisfies:*

$$\lambda>9a\Leftrightarrow\lambda=\tau_c 9a,\quad \tau_c\in(1,+\infty) \tag{3.9}$$

Proof. It follows from Lemmas 3.1, 3.2 and 3.4 that a sufficient condition for the functional $\mathcal{J}(\cdot;\lambda,T,a)$ in (3.1) to be strictly convex is that the function $h(\cdot;\lambda,T,a)$ in (3.8) is strictly convex. Recalling the definition of the penalty function $\phi(\cdot;T,a)$ in (3.2), $h(\cdot;\lambda,T,a)$ in (3.8) can be rewritten in the following explicit form:

$$h(t;\lambda,T,a):=\begin{cases}h_1(t;\lambda,T,a):=(\frac{\lambda}{18}-\frac{a}{2}+a\frac{T_2}{2T})t^2 & |t|\in[0,T)\\ h_2(t;\lambda,T,a):=(\frac{\lambda}{18}-\frac{a}{2})t^2+aT_2|t|-\frac{aTT_2}{2} & |t|\in[T,T_2)\\ h_3(t;\lambda,T,a):=\frac{\lambda}{18}t^2+\frac{aT_2(T_2-T)}{2} & |t|\in[T_2,\infty)\end{cases} \tag{3.10}$$

Clearly, the function h above is even and piecewise quadratic; and, as far as regularity is concerned, it is immediate to verify that $h\in\mathbb{C}^\infty(\mathbb{R}\setminus\{\pm T,\pm T_2\})\cap C^1(\mathbb{R})$. In particular, the first-order derivative function $h':\mathbb{R}\to\mathbb{R}$ and the second-order derivative function $h'':\mathbb{R}\setminus\{\pm T,\pm T_2\}\to\mathbb{R}$ are as follows:

$$h'(t;\lambda,T,a):=\begin{cases}h_1'(t;\lambda,T,a):=(\frac{\lambda}{9}-a+a\frac{T_2}{T})t & |t|\in[0,T)\\ h_2'(t;\lambda,T,a):=(\frac{\lambda}{9}-a)t+aT_2\mathrm{sign}(t) & |t|\in[T,T_2)\\ h_3'(t;\lambda,T,a):=\frac{\lambda}{9}t & |t|\in[T_2,\infty)\end{cases} \tag{3.11}$$

$$h''(t;\lambda,T,a):=\begin{cases}h_1''(t;\lambda,T,a):=(\frac{\lambda}{9}-a+a\frac{T_2}{T}) & |t|\in[0,T)\\ h_2''(t;\lambda,T,a):=\frac{\lambda}{9}-a & |t|\in[T,T_2)\\ h_3''(t;\lambda,T,a):=\frac{\lambda}{9} & |t|\in[T_2,\infty)\end{cases} \tag{3.12}$$

We notice that the functions h in (3.10) and h' in (3.11)) are both continuous at points $t\in\{\pm T,\pm T_2\}$, whereas for the function h'' in (3.12) we have at points $t\in\{T,T_2\}$ (analogously at points $t\in\{-T,-T_2\}$):

$$\begin{aligned}T:\lim_{t\uparrow T}h_1''(t;\lambda,T,a)&=\frac{\lambda}{9}-a+a\frac{T_2}{T}\neq\frac{\lambda}{9}-a=\lim_{t\downarrow T}h_2''(t;\lambda,T,a)\\ T_2:\lim_{t\uparrow T_2}h_2''(t;\lambda,T,a)&=\frac{\lambda}{9}-a\neq\frac{\lambda}{9}=\lim_{t\downarrow T_2}h_3''(t;\lambda,T,a)\end{aligned} \tag{3.13}$$

After recalling that $\lambda,T,a>0$ and $T_2>T$, we notice that

$$h_1''(t;\lambda,T,a)=\frac{\lambda}{9}+\frac{a}{T}(T_2-T)>0,\quad h_3''(t;\lambda,T,a)=\frac{\lambda}{9}>0 \tag{3.14}$$

hence the function h is strictly convex for $|t|\in[0,T)$ and $|t|\in(T_2,\infty)$. A sufficient condition (it is also a necessary condition since the function is quadratic) for h to be

strictly convex in $|t| \in (T, T_2)$ is that the second-order derivative $h_2''(t; \lambda, T, a)$ defined in (3.12) is positive. This clearly leads to condition (3.9) in the theorem statement. We have thus demonstrated that if (3.9) is satisfied then h is strictly convex for $t \in \{\pm T, \pm T_2\}$. It remains to handle the points $\pm T, \pm T_2$ where the function h does not admit second-order derivatives. Since h is even and continuously differentiable, it is sufficient to demonstrate that if condition (3.9) is satisfied then the first-order derivative h' is monotonically increasing at points $t \in \{T, T_2\}$. In particular, we aim to prove:

$$\begin{aligned} T: &\begin{cases} h'(t_1; \lambda, T, a) < h'(T; \lambda, T, a) & \forall t_1 \in (0, T) \\ h'(t_2; \lambda, T, a) > h'(T; \lambda, T, a) & \forall t_2 \in (T, T_2) \end{cases} \\ T_2: &\begin{cases} h'(t_2; \lambda, T, a) < h'(T_2; \lambda, T, a) & \forall t_2 \in (T, T_2) \\ h'(t_3; \lambda, T, a) > h'(T_2; \lambda, T, a) & \forall t_3 \in (T_2, +\infty) \end{cases} \end{aligned} \tag{3.15}$$

Recalling the definition of h' in (3.11), we obtain:

$$\begin{aligned} T: &\begin{cases} (\frac{\lambda}{9} - a + a\frac{T_2}{T})t_1 < (\frac{\lambda}{9} - a)T + aT_2 & \forall t_1 \in (0, T) \\ (\frac{\lambda}{9} - a)t_2 + aT_2 > (\frac{\lambda}{9} - a)T + aT_2 & \forall t_2 \in (T, T_2) \end{cases} \\ T_2: &\begin{cases} (\frac{\lambda}{9} - a)t_2 + aT_2 < \frac{\lambda}{9}T_2 & \forall t_2 \in (T, T_2) \\ \frac{\lambda}{9}t^3 > \frac{\lambda}{9}T_2 & \forall t_3 \in (T_2, +\infty) \end{cases} \end{aligned} \tag{3.16}$$

and, after simple algebraic manipulations:

$$\begin{aligned} T: &\begin{cases} (\underbrace{\frac{\lambda}{9} - a}_{>0} + \underbrace{a\frac{T_2}{T}}_{>0})\underbrace{(t_1 - T)}_{<0} < 0 & \forall t_1 \in (0, T) \\ \underbrace{(\frac{\lambda}{9} - a)}_{>0}\underbrace{(t_2 - T)}_{>0} > 0 & \forall t_2 \in (T, T_2) \end{cases} \\ T_2: &\begin{cases} \underbrace{(\frac{\lambda}{9} - a)}_{>0}\underbrace{(t_2 - T)}_{<0} < 0 & \forall t_2 \in (T, T_2) \\ \underbrace{\frac{\lambda}{9}}_{>0}\underbrace{(t_3 - T_2)}_{>0} > 0 & \forall t_3 \in (T_2, +\infty) \end{cases} \end{aligned} \tag{3.17}$$

Since $\lambda, T, a > 0$ and we are assuming $\lambda > 9a$ and $0 < t_1 < T < t_2 < T_2 < t_3$, inequalities in (3.17) are satisfied, hence the proof is completed.

□

The algorithm used to numerically solve the model (3.1) under the parameters condition of the theorem 3.5 is the ADMM. First of all we resort to the variable splitting

technique and introduce the auxiliary variable $t \in \mathbb{R}^{2n}$, such that model (3.1) is rewritten in the following linearly constrained equivalent form:

$$\{u^*, t^*\} \leftarrow \arg\min_{u,t} \left\{ \frac{\lambda}{2}||u-b||_2^2 + \sum_{i=1}^{n} \phi(||t_i||_2; T, a) \right\} \tag{3.18}$$

subject to $t = Du$.

To solve problem (3.18) we define the augmented Lagrangian functional

$$\mathcal{L}(u, t; \rho) = \frac{\lambda}{2}||u-b||_2^2 + \sum_{i=1}^{n} \phi(||t_i||_2; T, a) - \langle \rho, t - Du \rangle + \frac{\beta}{2}||t - Du||_2^2 \tag{3.19}$$

where $\beta > 0$ is a scalar penalty parameter and $\rho \in \mathbb{R}^{2n}$ is the vector of Lagrange multipliers associated with the system of linear constraints. We then consider the following saddle-point problem:

$$\text{Find } (u^*, t^*; \rho^*) \in \mathbb{R}^n \times \mathbb{R}^{2n} \times \mathbb{R}^{2n} \text{ s.t. } \mathcal{L}(u^*, t^*; \rho) \leq \mathcal{L}(u^*, t^*; \rho^*) \leq \mathcal{L}(u, t; \rho^*) \tag{3.20}$$

$\forall (u, t; \rho) \in \mathbb{R}^n \times \mathbb{R}^{2n} \times \mathbb{R}^{2n}$.

Given the previously computed (or initialized for $k = 1$) vectors $t^{(k-1)}$ and $\rho^{(k)}$, the k-th iteration of the proposed ADMM-based scheme applied to the solution of the saddle-point problem (3.19)-(3.20) reads as follows

$$u^{(k)} \leftarrow \arg\min_{u \in \mathbb{R}^n} \mathcal{L}(u, t^{(k-1)}; \rho^{(k)}), \tag{3.21}$$

$$t^{(k)} \leftarrow \arg\min_{t \in \mathbb{R}^{2n}} \mathcal{L}(u^{(k)}, t; \rho^{(k)}), \tag{3.22}$$

$$\rho^{(k+1)} \leftarrow \rho^{(k)} - \beta(t^{(k)} - Du^{(k)}). \tag{3.23}$$

Our focus now is on addressing the subproblems for u, (3.21), and t, (3.22).

The minimization subproblem for u in (3.21) can be rewritten as follows

$$u^{(k)} \leftarrow \arg\min_{u \in \mathbb{R}^n} \left\{ \frac{\lambda}{2}||u-b||_2^2 - \langle \rho^{(k)}, Du \rangle + \frac{\beta}{2}||t^{(k-1)} - Du||_2^2 \right\} \tag{3.24}$$

where constant terms have been omitted. Equivalently the minimization subproblem for t in (3.22) can be rewritten in the following component-wise form

$$t^{(k)} \leftarrow \arg\min_{t \in \mathbb{R}^{2n}} \left\{ \sum_{i=1}^{n} \phi(||t_i||_2; T, a) + \frac{\beta}{2}||t_i - r_i^{(k)}||_2^2 \right\} \tag{3.25}$$

where the n vectors $r_i^{(k)} \in \mathbb{R}^2$, $i = 1, \ldots, n$ which are constant with respect to the optimization variable t, are defined by

$$r_i(k) = (Du^{(k)})_i + \frac{1}{\beta}(\rho^{(k)})_i \tag{3.26}$$

The quadratic minimization problem (3.24) has first-order optimality conditions which lead to the following linear system:

$$\left(I_n + \frac{\beta}{\lambda} D^T D\right) u = b + \frac{\beta}{\lambda} D^T \left(t^{(k-1)} - \frac{1}{\beta}\rho^{(k)}\right) \tag{3.27}$$

Since, $\frac{\beta}{\lambda}$, the coefficient matrix of the linear system (3.27) is symmetric, positive definite and highly sparse, therefore (3.27) can be solved very efficiently by the iterative (preconditioned) conjugate gradient method.

With regard to the subproblem for t, the minimization problem in (3.25) is thus equivalent to the following n independent 2-dimensional problems:

$$t_i^{(k)} \leftarrow \arg\min_{t_i \in \mathbb{R}^2} \left\{ \Theta_i(t_i) := \phi(||t_i||_2; T, a) + \frac{\beta}{2} ||t_i - r_i^{(k)}||_2^2 \right\}, \quad i = 1, \ldots, n \tag{3.28}$$

where we introduced the cost functions $\Theta_i : \mathbb{R}^2 \to \mathbb{R},\ i = 1, \ldots, n$.

Since we are under the condition of Theorem 3.5, such that the original functional $\mathcal{J}(u; \lambda, T, a)$ is strictly convex we aim at avoiding non convexity of the ADMM subproblems (3.28). In the first part of Proposition 3.6 below we give a necessary and sufficient condition for strict convexity of the cost functions in (3.28).

Proposition 3.6. *Let $T, a, \beta \in \mathbb{R}_+^*$ and $r \in \mathbb{R}^2$ be given constants, and let $\phi(\cdot; T, a) : \mathbb{R}_+ \to \mathbb{R}$ be the penalty function defined in* (3.2)*. Then:*

1. *The function*

$$\Theta(x) := \phi(||x||_2; T, a) + \frac{\beta}{2} ||x - r||_2^2, \quad x \in \mathbb{R}^2 \tag{3.29}$$

is strictly convex (convex) if and only if the following condition holds

$$\beta > a \quad (\beta \geq a) \tag{3.30}$$

2. *In case that* (3.30) *holds, the strictly convex minimization problem*

$$\arg\min_{x \in \mathbb{R}^2} \Theta(x) \tag{3.31}$$

admits the unique solution $x^* \in \mathbb{R}^2$ given by the following shrinkage operator

$$x^* = \xi^* r, \quad \text{with } \xi^* \in (0, 1] \tag{3.32}$$

equal to

$$a) \quad \xi^* = k_1 \qquad \text{if } ||r||_2 \in [0, k_0) \tag{3.33}$$

$$b) \quad \xi^* = k_2 - \frac{k_3}{||r||_2} \qquad \text{if } ||r||_2 \in [k_0, T_2) \tag{3.34}$$

$$c) \quad \xi^* = 1 \qquad \text{if } ||r||_2 \in [T_2, +\infty) \tag{3.35}$$

and with

$$k_0 = T + \frac{a}{\beta}(T_2 - T), \quad k_1 = \frac{T}{k_0}, \quad k_2 = \frac{\beta}{\beta - a}, \quad k_3 = \frac{aT_2}{\beta - a} \tag{3.36}$$

Based on (3.29) and (3.30) we can state that all the problems in (3.28) are strictly convex if and only if

$$\beta > a \tag{3.37}$$

In case that (3.37) is satisfied, the unique solutions of the strictly convex problems (3.28) can be obtained based on (3.32) in the above proposition, that is:

$$t_i^{(k)} = \xi_i^{(k)} r_i^{(k)}, \quad i = 1, \ldots, n, \tag{3.38}$$

where the shrinkage coefficients $\xi_i^{(k)} \in (0, 1]$ are computed according to 3.33, 3.20, 3.21.

The solutions of problems (3.28) can thus be determined very efficiently by the closed forms given in 3.33, 3.20, 3.21 with computational cost $O(n)$.

3.1.3 Convergence analysis

In this section, we analyze convergence of the proposed ADMM-based minimization approach, whose main computational steps are reported in Algorithm 1.

Algorithm 1 ADMM-based scheme for the solution of CNC problem (4.1)

input: observed image $b \in \mathbb{R}^n$

output: approximate solution $u^* \in \mathbb{R}^n$ of (4.1)

parameters: MODEL: $T > 0$ and $\lambda > 0$

1. initialization: $t^{(0)} = Db$, $\rho^{(1)} = 0$
   set a s.t. $\lambda > 9a$ according to (4.3),
   set $\beta > 0$ s.t. $\beta \geq \max\{2a, a\frac{\lambda}{\lambda - 8a}\}$ according to (4.30)
2. **for** $k = 1, 2, 3, \ldots$ until convergence **do**:
   update primal variables:
   · compute $u^{(k)}$ by solving (4.13)
   · compute $t^{(k)}$ by (4.12), (4.19), (4.20), (4.21) and (4.24)
   update dual variable:
   · compute $\rho^{(k+1)}$ by (4.9)
3. **end for**
4. $u^* = u^{(k)}$

In particular, we prove convergence of Algorithm 1 in case that conditions (3.9) and (3.44) are satisfied.

To simplify the notations in the subsequent discussion, we give the following definitions concerning the objective functional in the (u,t)-split problem (3.18):

$$G(u,t) := \underbrace{\frac{\lambda}{2}||u-b||_2^2}_{F(u)} + \underbrace{\sum_{i=1}^{n}\phi(||t_i||_2;T,a)}_{R(t)}, \tag{3.39}$$

where the parametric dependencies of the fidelity term $F(u)$ on λ and of the regularization term $R(t)$ on T and a are dropped for brevity. The augmented Lagrangian functional in (3.19) can thus be rewritten as

$$\mathcal{L}(u,t;\rho) = F(u) + R(t) - \langle \rho, t - Du\rangle + \frac{\beta}{2}||t-Du||_2^2, \tag{3.40}$$

and the regularization term in the original proposed model (3.1), referred to as $R(u)$, reads

$$\mathcal{R}(u) = R(Du). \tag{3.41}$$

To prove convergence, we rely on the optimality conditions of the augmented Lagrangian with respect to the primal variables (u,t) and on the construction of suitable Fejér-monotone sequences. In our model, the overall functional $\mathcal{J}$ is convex, but the regularization term is non-convex, which requires adapting the standard proof techniques. The proof is structured as follows:

1. derivation of the optimality conditions for problem (3.1);
2. analysis of convexity conditions for the augmented Lagrangian (3.40);
3. proof of the equivalence between the split problem (3.18) and the saddle-point problem (3.19)-(3.20);
4. proof of convergence of Algorithm 1 to a solution of (3.19)-(3.20), and hence to the unique solution of (3.1).

Regarding point 1, specifically the analysis of the optimality condition, since the regularization term is non-smooth and non-convex, tools from non-smooth nonconvex analysis are required, in particular the concept of Clarke's generalized gradient. In the following we will denote by $\partial_x[f](x^*)$ and by $\overline{\partial}_x[f](x^*)$ the subdifferential and the Clarke generalized gradient, respectively, with respect to x of the function f calculated at x^*.

Before giving the first-order optimality conditions for problem (3.1) we state a lemma providing some results on locally Lipschitz continuity which are necessary for the generalized gradients being defined.

Lemma 3.7. *For any pair of parameters* (λ, a) *satisfying condition* (3.9)*, the functional* $\mathcal{J}$ *in* (3.1) *and, separately, the regularization term* $\mathcal{R}$ *in* (3.41) *and the quadratic fidelity term, are locally Lipschitz continuous functions.*

Proposition 3.8. *For any pair of parameters* (λ, a) *satisfying condition* (3.9)*, the functional* $\mathcal{J} : \mathbb{R}^n \to \mathbb{R}$ *in* (3.1) *has a unique (global) minimizer* u^* *which satisfies*

$$0 \in \partial_u[\mathcal{J}](u^*) \tag{3.42}$$

where 0 *denotes the null vector in* $\mathbb{R}^n$ *and* $\partial_u[\mathcal{J}](u^*) \subset \mathbb{R}^n$ *represents the sub-differential (with respect to* u*, calculated at* u^**) of functional* $\mathcal{J}$*. Moreover, it follows that*

$$0 \in D^T \overline{\partial}_t[R](Du^*) + \lambda(u^* - b), \tag{3.43}$$

where $\overline{\partial}_t[R](Du^*) \subset \mathbb{R}^n$ *denotes the Clarke generalized gradient (with respect to* t*, calculated at* Du^**) of the non-convex non-smooth regularization function* R *defined in* (3.39)*.*

Subsequently, regarding point 2, namely the convexity conditions for the augmented Lagrangian we focus on showing the convexity with respect to the pair of primal variables (u, t).

In our case, where the regularization term is non-convex, the convexity conditions is given by the following Proposition:

Proposition 3.9. *For any given vector of Lagrange multipliers* $\rho \in \mathbb{R}^{2n}$*, the augmented Lagrangian functional* $\mathcal{L}(u, t; \rho)$ *in* (3.40) *is proper, continuous and coercive jointly in the pair of primal variables* (u, t)*. Moreover, in case that condition* (3.9) *is satisfied,* $\mathcal{L}(u, t; \rho)$ *is jointly convex in* (u, t) *if the penalty parameter* β *satisfies*

$$\beta \geq a\frac{\lambda}{\lambda - 8a}, \tag{3.44}$$

or, equivalently

$$\beta \geq 9a\frac{\tau_c}{9\tau_c - 8}, \quad \tau_c \in (1, +\infty). \tag{3.45}$$

Finally, we focus on point 4, that is, the demonstration of convergence of Algorithm 1 to a solution of (3.19)-(3.20), and hence to the unique solution of (3.1). We will not dwell on point 3, namely the demonstration of equivalence (in terms of solutions) between the split problem (3.18) and the saddle-point problem (3.19)-(3.20), since this aspect is not essential for the purposes of the present work. The only statement we will make regarding point 3 is the following theorem:

Theorem 3.10. *For any pair of parameters* (λ, a) *satisfying condition* (3.9) *and any parameter* β *fulfilling condition* (3.45)*, the saddle-point problem* (3.19)-(3.20) *admits at least one solution and all the solutions have the form* $(u^*, Du^*; \rho^*)$*, with* u^* *denoting the unique global minimizer of functional* $\mathcal{J}$ *in* (3.1)*.*

Given the existence and the good properties of the saddle points of the augmented Lagrangian functional in (3.40), highlighted in Theorem 3.10, it remains to demonstrate that the ADMM iterative scheme outlined in Algorithm 1 converges towards one of these saddle points, that is towards a solution of the saddle-point problem (3.19)-(3.20). This is the goal of Theorem 3.11 below.

Theorem 3.11. *Assume that* $(u^*, t^*; \rho^*)$ *is a solution of the saddle-point problem* (3.19)-(3.20). *Then, for any pair of parameters* (λ, a) *satisfying condition* (3.9) *and any parameter* β *fulfilling condition*

$$\beta > \overline{\beta} := \max\left\{2a, 9a\frac{\tau_c}{9\tau_c - 8}\right\}, \tag{3.46}$$

the sequence $\{(u^{(k)}, t^{(k)}; \rho^{(k)}\}_{k=1}^{+\infty}$ *generated by Algorithm* 1 *satisfies:*

$$\lim_{k\to+\infty} u^{(k)} = u^*, \tag{3.47}$$

$$\lim_{k\to+\infty} t^{(k)} = t^* = Du^*. \tag{3.48}$$

Proof. Let us define the following errors:

$$\overline{u}^{(k)} = u^{(k)} - u^*, \;\; \overline{t}^{(k)} = t^{(k)} - t^*, \;\; \overline{\rho}^{(k)} = \rho^{(k)} - \rho^*. \tag{3.49}$$

Since (u^*, t^*, ρ^*) is a saddle-point of the augmented Lagrangian functional in (3.19), it follows from Theorem 3.10 that $t^* = Du^*$. This relationship, together with the ADMM updating formula for the vector of Lagrange multipliers in (3.23), yields:

$$\overline{\rho}^{(k+1)} = \overline{\rho}^{(k)} - \beta(\overline{t}^{(k)} - D\overline{u}^{(k)}). \tag{3.50}$$

It then follows easily from (3.50) that

$$||\overline{\rho}^{(k)}||_2^2 - ||\overline{\rho}^{(k+1)}||_2^2 = 2\beta\langle\overline{\rho}^{(k)}, \overline{t}^{(k)} - D\overline{u}^{(k)}\rangle - \beta^2||\overline{t}^{(k)} - D\overline{u}^{(k)}||_2^2. \tag{3.51}$$

Now we focus on the computation of a lower bound for the right-hand side of (3.51). Since $(u^*, t^*; \rho^*)$) is a saddle-point of the augmented Lagrangian functional in (3.19), it satisfies the following optimality conditions:

$$\begin{aligned} &F(u) - \frac{\beta_1}{2}\|t^* - Du\|_2^2 - F(u^*) + \frac{\beta_1}{2}\|t^* - Du^*\|_2^2 + \\ &- \left\langle D^T\big((\beta + \beta_1)(t^* - Du^*) - \rho^*\big),\, u - u^*\right\rangle \geq 0 \quad \forall u \in \mathbb{R}^n, \end{aligned} \tag{3.52}$$

$$\begin{aligned} &R(t) + \frac{\beta_2}{2}||t - Du^*||_2^2 - R(t^*) - \frac{\beta_2}{2}||t^* - Du^*||_2^2 \\ &+ \langle(\beta - \beta_2)(t^* - Du^*) - \rho^*, t - t^*\rangle \geq 0 \;\; \forall t \in \mathbb{R}^{2n}. \end{aligned} \tag{3.53}$$

Similarly, by the construction of $(u^{(k)}, t^{(k)})$ in Algorithm 1, we have:

$$F(u) - \frac{\beta_1}{2}\|t^{(k-1)} - Du\|_2^2 - F(u^{(k)}) + \frac{\beta_1}{2}\|t^{(k-1)} - Du^{(k)}\|_2^2 + \\ - \left\langle D^T\big((\beta+\beta_1)(t^{(k-1)} - Du^{(k)}) - \rho^{(k)}\big),\, u - u^{(k)}\right\rangle \geq 0 \quad \forall u \in \mathbb{R}^n, \tag{3.54}$$

$$R(t) + \frac{\beta_2}{2}||t - Du^{(k)}||_2^2 - R(t^{(k)}) - \frac{\beta_2}{2}||t^{(k)} - Du^{(k)}||_2^2 \\ + \left\langle (\beta - \beta_2)(t^{(k)} - Du^{(k)}) - \rho^{(k)}, t - t^{(k)}\right\rangle \geq 0 \ \ \forall t \in \mathbb{R}^{2n}. \tag{3.55}$$

Taking $u = u^{(k)}$ in (3.52), $u = u^*$ in (3.54) and recalling that $\langle D^T w, z\rangle = \langle w, Dz\rangle$, by addition we obtain:

$$\underbrace{-\langle \overline{\rho}^{(k)}, D\overline{u}^{(k)}\rangle}_{A_1} + \underbrace{\beta\langle \overline{t}^{(k-1)}, D\overline{u}^{(k)}\rangle}_{B_1} - \underbrace{(\beta+\beta_1)||D\overline{u}^{(k)}||_2^2}_{C_1} \geq 0. \tag{3.56}$$

Similarly, taking $t = t^{(k)}$ in (3.53) and $t = t^*$ in (3.55), after addition we have:

$$\underbrace{-\langle \overline{\rho}^{(k)}, \overline{t}^{(k)}\rangle}_{A_2} + \underbrace{\beta\langle \overline{t}^{(k)}, D\overline{u}^{(k)}\rangle}_{B_2} - \underbrace{(\beta+\beta_2)||\overline{t}^{(k)}||_2^2}_{C_2} \geq 0, \tag{3.57}$$

where, we recall, the parameters β_1 and β_2 in (3.56)-(3.57) satisfy the constraints

$$-\beta < \beta_1 \leq \tau_c\frac{9}{8}a, \ a \leq \beta_2 < \beta. \tag{3.58}$$

By summing up (3.56)-(3.57), we obtain:

$$\langle \overline{\rho}^{(k)}, \overline{t}^{(k)} - D\overline{u}^{(k)}\rangle - \beta\langle \overline{t}^{(k)} - \overline{t}^{(k-1)}, D\overline{u}^{(k)}\rangle + \\ - \big((\beta-\beta_2)||\overline{t}^{(k)}||_2^2 - 2\beta\langle \overline{t}^{(k)}, D\overline{u}(k)\rangle + (\beta+\beta_1)||D\overline{u}^{(k)}||_2^2\big) \geq 0 \tag{3.59}$$

that is

$$\langle \overline{\rho}^{(k)}, \overline{t}^{(k)} - D\overline{u}^{(k)}\rangle - \beta\langle \overline{t}^{(k)} - \overline{t}^{(k-1)}, D\overline{u}^{(k)}\rangle - \frac{\beta+\beta_3}{2}||\overline{t}^{(k)} - D\overline{u}^{(k)}||_2^2 \\ \Big(\Big(-\beta_2 - \frac{\beta_3}{2} + \frac{\beta}{2}\Big)||\overline{t}^{(k)}||_2^2 - (\beta-\beta_3)\langle \overline{t}^{(k)}, D\overline{u}^{(k)}\rangle + \Big(\beta_1 - \frac{\beta_3}{2} + \frac{\beta}{2}\Big)||D\overline{u}^{(k)}||_2^2\Big) \geq 0 \tag{3.60}$$

where we introduced the positive coefficient $\beta_3 > 0$ (the reason will be clear later on). We want that the last term in (3.60) takes the form

$$-||c_1\overline{t}^{(k)} - c_2 D\overline{u}^{(k)}||_2^2$$

with $c_1, c_2 > 0$. Hence, first we impose that the coefficients of $||\overline{t}^{(k)}||_2^2$ and $||D\overline{u}^{(k)}||_2^2$ in (3.60) are strictly positive, which yields:

$$\beta_1 > \frac{\beta_3}{2} - \frac{\beta}{2}, \quad \beta_2 < -\frac{\beta_3}{2} + \frac{\beta}{2}. \tag{3.61}$$

Combining (3.61) with conditions (3.58), we obtain:

$$\frac{\beta_3}{2} - \frac{\beta}{2} < \beta_1 \leq \tau_c \frac{9}{8} a, \quad a \leq \beta_2 < -\frac{\beta_3}{2} + \frac{\beta}{2}, \quad 0 < \beta_3 < \beta - 2a. \tag{3.62}$$

From condition on β_3 in (3.62), the following constraint for β is derived:

$$\beta > 2a. \tag{3.63}$$

We notice that condition (3.63) can be more stringent than (3.45), depending on τ_c, hence it has been taken as an hypothesis of this theorem and will be considered, together with (3.45), in the rest of the proof. From condition on β_3 in (3.62) it also follows that the coefficient $\beta - \beta_3$ of the scalar product in (3.60) is positive.

Then, we have to impose that the coefficient of the term $-\langle \overline{t}^{(k)}, D\overline{u}^{(k)} \rangle$ in (3.60) is twice the product of the square roots of the (positive) coefficients of $||\overline{t}^{(k)}||_2^2$ and $||D\overline{u}^{(k)}||_2^2$, that is:

$$\beta - \beta_3 = 2\sqrt{\Big(-\beta_2 - \frac{\beta_3}{2} + \frac{\beta}{2} \Big)\Big(\beta_1 - \frac{\beta_3}{2} + \frac{\beta}{2} \Big)} \Longrightarrow \beta = \beta_3 + 2\frac{\beta_1 \beta_2}{\beta_1 - \beta_2}. \tag{3.64}$$

By imposing condition on β_3 in (3.62), namely $\beta - \beta_3 > 2a$, it is easy to verify that (3.64) admits acceptable solutions only in case that $\beta_1 > \beta_2$. By setting in (3.64) $\beta_1 = \tau_c \frac{9}{8} a$ and $\beta_2 = a$, which are acceptable values according to this last result (since $\tau_c > 1$, clearly $\beta_1 > \beta_2$) and also conditions (3.62), we obtain:

$$\beta = \beta_3 + 2a\frac{9\tau_c}{9\tau_c - 8}. \tag{3.65}$$

We now check if there exist acceptable values for the two remaining free parameters, namely β and β_3, such that (3.65) holds. We impose that β in (3.65) satisfies its constraint in (3.45), which guarantees convexity of the augmented Lagrangian functional, and the derived condition in (3.63):

$$\begin{cases} \beta_3 + 2a\frac{9\tau_c}{9\tau_c - 8} \geq a\frac{9\tau_c}{9\tau_c - 8} \\ \beta_3 + 2a\frac{9\tau_c}{9\tau_c - 8} > 2a \end{cases} \Longrightarrow \begin{cases} \beta_3 \geq -a\frac{9\tau_c}{9\tau_c - 8} \\ \beta_3 > -a\frac{16}{9\tau_c - 8} \end{cases} \tag{3.66}$$

Since $\tau_c > 1$ (and $a > 0$), bot conditions in (3.66) are satisfied for any $\beta_3 > 0$. Hence, for $\beta_1 = \tau_c \frac{9}{8} a$, $\beta_2 = a$ and any $0 < \beta_3 < \beta - 2a$, with $\beta > 2a$, the last term in (3.60) can be written in the form

$$-||c_1 \overline{t}^{(k)} - c_2 D\overline{u}^{(k)}||_2^2 \text{ with } \begin{cases} c_1 = \frac{\beta - \beta_3}{2} - a \\ c_2 = \frac{\beta - \beta_3}{2} + \tau_c \frac{9}{8} a \end{cases} \tag{3.67}$$

where $c_1, c_2 > 0, c_1 \neq c_2$. Replacing the expression in (3.67) for the last term in (3.60), we have:

$$\begin{aligned}
&\langle \overline{\rho}^{(k)}, \overline{t}^{(k)} - D\overline{u}^{(k)} \rangle - \frac{\beta + \beta_3}{2} ||\overline{t}^{(k)} - D\overline{u}^{(k)}||_2^2 - \beta \langle \overline{t}^{(k)} - \overline{t}^{(k-1)}, D\overline{u}^{(k)} \rangle + \\
&- ||c_1 \overline{t}^{(k)} - c_2 D\overline{u}^{(k)}||_2^2 \geq 0 \\
\Longleftrightarrow &2\beta \langle \overline{\rho}^{(k)}, \overline{t}^{(k)} - D\overline{u}^{(k)} \rangle - \beta^2 ||\overline{t}^{(k)} - D\overline{u}^{(k)}||_2^2 \geq \beta\beta_3 ||\overline{t}^{(k)} - D\overline{u}^{(k)}||_2^2 \\
&+ 2\beta^2 \langle \overline{t}^{(k)} - \overline{t}^{(k-1)}, D\overline{u}^{(k)} \rangle + 2\beta ||c_1 \overline{t}^{(k)} - c_2 D\overline{u}^{(k)}||_2^2, \qquad (3.68)
\end{aligned}$$

where in (3.68) we multiplied both sides by the positive coefficient 2β. We notice that the left-hand side of (3.68) coincides with the right-hand side of (3.51), hence it follows that:

$$\begin{aligned}
||\overline{\rho}^{(k)}||_2^2 - ||\overline{\rho}^{(k+1)}||_2^2 \geq \beta\beta_3 ||\overline{t}^{(k)} - D\overline{u}^{(k)}||_2^2 + 2\beta^2 \underbrace{\langle \overline{t}^{(k)} - \overline{t}^{(k-1)}, D\overline{u}^{(k)} \rangle}_{T} \\
+ 2\beta ||c_1 \overline{t}^{(k)} - c_2 D\overline{u}^{(k)}||_2^2. \qquad (3.69)
\end{aligned}$$

At this point, we concentrate on the computation of a lower bound for the term T in (3.69).

We can write:

$$\begin{aligned}
\langle \overline{t}^{(k)} - \overline{t}^{(k-1)}, D\overline{u}^{(k)} \rangle &= \langle \overline{t}^{(k)} - \overline{t}^{(k-1)}, D\overline{u}^{(k)} - D\overline{u}^{(k-1)} \rangle \\
&+ \langle \overline{t}^{(k)} - \overline{t}^{(k-1)}, D\overline{u}^{(k-1)} - \overline{t}^{(k-1)} \rangle \\
&+ \langle \overline{t}^{(k)} - \overline{t}^{(k-1)}, \overline{t}^{(k-1)} \rangle. \qquad (3.70)
\end{aligned}$$

First, we notice that:

$$\langle \overline{t}^{(k)} - \overline{t}^{(k-1)}, \overline{t}^{(k-1)} \rangle = \frac{1}{2} \Big(||\overline{t}^{(k)}||_2^2 - ||\overline{t}^{(k-1)}||_2^2 - ||\overline{t}^{(k)} - \overline{t}^{(k-1)}||_2^2 \Big). \qquad (3.71)$$

Then, from the construction of $t^{(k-1)}$ (from $u^{(k-1)}$), we have:

$$\begin{aligned}
&R(t) + \frac{\beta_2}{2} ||t - Du^{(k-1)}||_2^2 - R(t^{(k-1)}) - \frac{\beta_2}{2} ||t^{(k-1)} - Du^{(k-1)}||_2^2 \\
&+ \langle (\beta - \beta_2)(t^{(k-1)} - Du^{(k-1)}) - \rho^{(k-1)}, t - t^{(k-1)} \rangle \geq 0 \ \ \forall t \in \mathbb{R}^{2n}. \qquad (3.72)
\end{aligned}$$

Taking $t = t^{(k-1)}$ in (3.55) and $t = t^{(k)}$ in (3.72), we obtain:

$$\begin{aligned}
&R(t^{(k-1)} + \frac{\beta}{2} ||t^{(k-1)} - Du^{(k)}||_2^2 - R(t^{(k)}) - \frac{\beta_2}{2} ||t^{(k)} - Du^{(k)}||_2^2 \\
&+ \langle (\beta - \beta_2)(t^{(k)} - Du^{(k)}) - \rho^{(k)}, t^{(k-1)} - t^{(k)} \rangle \geq 0, \qquad (3.73) \\
&R(t^{(k)}) + \frac{\beta_2}{2} ||t^{(k)} - Du^{(k-1)}||_2^2 - R(t^{(k-1)}) - \frac{\beta_2}{2} ||t^{(k-1)} - Du^{(k-1)}||_2^2 \\
&+ \langle (\beta - \beta_2)(t^{(k-1)} - Du^{(k-1)}) - \rho^{(k-1)}, t^{(k)} - t^{(k-1)} \rangle \geq 0. \qquad (3.74)
\end{aligned}$$

By addition of (3.73) and (3.74), we have that

$$\beta\langle\langle\bar{t}^{(k)}-\bar{t}^{(k-1)},D\overline{u}^{(k)}-D\overline{u}^{(k-1)}\rangle+\langle\bar{t}^{(k)}-\bar{t}^{(k-1)},\overline{\rho}^{(k)}-\overline{\rho}^{(k-1)}\rangle\geq(\beta-\beta_2)||\bar{t}^{(k)}-\bar{t}^{(k-1)}||_2^2 \tag{3.75}$$

Recalling that

$$\overline{\rho}^{(k)}-\overline{\rho}^{(k-1)}=\rho^{(k)}-\rho^{(k-1)}=-\beta(\bar{t}^{(k-1)}-D\overline{u}^{(k-1)}), \tag{3.76}$$

replacing (3.76) into (3.75) and then dividing by β, we obtain:

$$\langle\bar{t}^{(k)}-\bar{t}^{(k-1)},D\overline{u}^{(k)}-D\overline{u}^{(k-1)}\rangle+\langle\bar{t}^{(k)}-\bar{t}^{(k-1)},D\overline{u}^{(k-1)}-\bar{t}^{(k-1)}\rangle\geq\frac{\beta-\beta_2}{\beta}||\bar{t}^{(k)}-\bar{t}^{(k-1)}||_2^2. \tag{3.77}$$

From (3.70), (3.71) and (3.77), we have:

$$\begin{aligned}\langle\bar{t}^{(k)}-\bar{t}^{(k-1)},D\overline{u}^{(k)}\rangle&\geq\frac{1}{2}\Big(||\bar{t}^{(k)}||_2^2-||\bar{t}^{(k-1)}||_2^2-||\bar{t}^{(k)}-\bar{t}^{(k-1)}||_2^2\Big)+\frac{\beta-\beta_2}{\beta}||\bar{t}^{(k)}-\bar{t}^{(k-1)}||_2^2\\&=\frac{1}{2}\Big(||\bar{t}^{(k)}||_2^2-||\bar{t}^{(k-1)}||_2^2+\frac{\beta-2\beta_2}{\beta}||\bar{t}^{(k)}-\bar{t}^{(k-1)}||_2^2\Big).\end{aligned} \tag{3.78}$$

We proceed to state the convergence results for sequences $t^{(k)}, Du^{(k)}, \rho^{(k)}$.

From (3.69) and (3.78), we obtain:

$$\begin{aligned}||\overline{\rho}^{(k)}||_2^2-||\overline{\rho}^{(k+1)}||_2^2\geq\beta^2||\bar{t}^{(k)}||_2^2-\beta^2||\bar{t}^{(k-1)}||_2^2+\beta(\beta-2\beta)||\bar{t}^{(k)}-\bar{t}^{(k-1)}||_2^2\\+\beta\beta_3||\bar{t}^{(k)}-D\overline{u}^{(k)}||_2^2+2\beta||c_1\bar{t}^{(k)}-c_2D\overline{u}^{(k)}||_2^2,\end{aligned} \tag{3.79}$$

that is

$$\begin{aligned}&\underbrace{\Big(||\overline{\rho}^{(k)}||_2^2+\beta^2||\bar{t}^{(k-1)}||_2^2\Big)}_{s^{(k)}}-\underbrace{\Big(||\overline{\rho}^{(k+1)}||_2^2+\beta^2||\bar{t}^{(k)}||_2^2\Big)}_{s^{(k+1)}}\\&\geq\beta(\beta-2\beta_2)||\bar{t}^{(k)}-\bar{t}^{(k-1)}||_2^2+\beta\beta_3||\bar{t}^{(k)}-D\overline{u}^{(k)}||_2^2\\&+2\beta||c_1\bar{t}^{(k)}-c_2D\overline{u}^{(k)}||_2^2\geq0.\end{aligned} \tag{3.80}$$

where we have introduced the scalar sequence $\{s^{(k)}\}$, which is clearly bounded from below by zero. We notice that the coefficient $\beta-2\beta_2$ in (3.80) is positive due to the constraint $\beta>2a$. Since the right-hand side of the first inequality in (3.80) is nonnegative, $\{s^{(k)}\}$ is monotonically non-increasing, hence convergent. This implies that the right-hand side of (3.80) tend to zero as $k\to\infty$. From these considerations and (3.80) it follows that:

$$\{\overline{\rho}^{(k)}\},\{\bar{t}^{(k)}\},\{D\overline{u}^{(k)}\}\text{ are bounded }\Longrightarrow\{\rho^{(k)}\},\{t^{(k)}\},\{Du^{(k)}\}\text{ bounded}, \tag{3.81}$$

$$\lim_{k\to\infty}||\bar{t}^{(k)}-\bar{t}^{(k-1)}||_2=\lim_{k\to\infty}||t^{(k)}-t^{(k-1)}||_2=0, \tag{3.82}$$

$$\lim_{k\to\infty}||\bar{t}^{(k)}-D\overline{u}^{(k)}||_2=\lim_{k\to\infty}||t^{(k)}-Du^{(k)}||_2=0, \tag{3.83}$$

$$\lim_{k\to\infty}||c_1\bar{t}^{(k)}-c_2D\overline{u}^{(k)}||_2=0. \tag{3.84}$$

Since the two coefficients c_1, c_2 in (3.84) satisfy $c_1, c_2 \neq 0, c_1 \neq c_2$, then it follows from (3.83)-(3.84) that both the sequences $\{\bar{t}^{(k)}\}$ and $\{D\bar{u}^{(k)}\}$ tend to zero as $k \to \infty$. Results in (3.81)-(3.84) can thus be rewritten in the following more concise and informative form:

$$\{\rho^{(k)}\} \text{ is bounded,} \tag{3.85}$$

$$\lim_{k\to\infty} \bar{t}^{(k)} = 0 \Longleftrightarrow \lim_{k\to\infty} t^{(k)} = t^* = Du^*, \tag{3.86}$$

$$\lim_{k\to\infty} D\bar{u}^{(k)} = 0 \Longleftrightarrow \lim_{k\to\infty} Du^{(k)} = Du^*, \tag{3.87}$$

where the last equality in (3.86) comes from the saddle-point properties stated in Theorem 3.10. Since it will be useful later on, we note that it follows from (3.86) that

$$\lim_{k\to\infty} R(t^{(k)}) = R(t^*). \tag{3.88}$$

Finally we give the convergence results for sequence $u^{(k)}$. We now prove that $\lim_{k\to\infty} u^{(k)} = u^*$. Since $(u^*, t^*; \rho^*)$ is a saddle point of the augmented Lagrangian functional $\mathcal{L}(u, t; \rho)$, we have

$$\mathcal{L}(u^*, t^*; \rho^*) \leq \mathcal{L}(u, t; \rho^*) \ \ \forall (u, t) \in \mathbb{R}^n \times \mathbb{R}^{2n}. \tag{3.89}$$

By taking $u = u^{(k)}, t = t^{(k)}$ in (3.89) and recalling the definition of $\mathcal{L}(u, t; \rho)$ in (3.40) we have:

$$\begin{aligned}
&F(u^*) + R(t^*) - \langle \rho^*, \underbrace{t^* - Du^*}_{0} \rangle + \frac{\beta}{2} || \underbrace{t^* - Du^*}_{0} ||_2^2 \\
&\leq F(u^{(k)}) + R(t^{(k)}) - \langle \rho^*, t^{(k)} - Du^{(k)} \rangle + \frac{\beta}{2} ||t^{(k)} - Du^{(k)}||_2^2 \\
&\Longleftrightarrow F(u^*) \leq F(u^{(k)}) + R(t^{(k)}) - R(t^*) \\
&- \langle \rho^*, t^{(k)} - Du^{(k)} \rangle + \frac{\beta}{2} ||t^{(k)} - Du^{(k)}||_2^2.
\end{aligned} \tag{3.90}$$

Taking $u = u^*$ in (3.54) and $t = t^*$ in (3.55), we obtain:

$$\begin{aligned}
&F(u^*) - \frac{\beta_1}{2} \|t^{(k-1)} - Du^*\|_2^2 - F(u^{(k)}) + \frac{\beta_1}{2} \|t^{(k-1)} - Du^{(k)}\|_2^2 + \\
&- \left\langle D^T \big((\beta + \beta_1)(t^{(k-1)} - Du^{(k)}) - \rho^{(k)} \big),\ u^* - u^{(k)} \right\rangle \geq 0,
\end{aligned} \tag{3.91}$$

$$\begin{aligned}
&R(t^*) + \frac{\beta_2}{2} ||t^* - Du^{(k)}||_2^2 - R(t^{(k)}) - \frac{\beta_2}{2} ||t^{(k)} - Du^{(k)}||_2^2 \\
&+ \left\langle (\beta - \beta_2)(t^{(k)} - Du^{(k)}) - \rho^{(k)}, t^* - t^{(k)} \right\rangle \geq 0.
\end{aligned} \tag{3.92}$$

By summing up (3.91) and (3.92), we have:

$$\begin{aligned}
&F(u^*) \geq F(u^{(k)}) + R(t^{(k)}) - R(t^*) + \frac{\beta_1}{2} ||Du^{(k)}||_2^2 - \frac{\beta_1}{2} ||Du^{(k)}||_2^2 \\
&- \beta_1 \langle t^{(k-1)}, Du^* - Du^{(k)} \rangle - \frac{\beta_2}{2} ||t^* - Du^{(k)}||_2^2 + \frac{\beta_2}{2} ||t^{(k)} - Du^{(k)}||_2^2 \\
&+ \langle (\beta + \beta_1)(t^{(k-1)} - Du^{(k)}) - \rho^{(k)}, Du^* - Du^{(k)} \rangle \\
&- \langle (\beta - \beta_2)(t^{(k)} - Du^{(k)}) - \rho^{(k)}, t^* - t^{(k)} \rangle.
\end{aligned} \tag{3.93}$$

Taking lim inf of (3.90) and lim sup of (3.93), and using the result in (3.85)-(3.88), we have

$$\liminf F(u^{(k)}) \geq F(u^*) \geq \limsup F(u^{(k)}) \tag{3.94}$$

It follows from (3.94) that

$$\lim_{k\to\infty} F(u^{(k)}) = F(u^*). \tag{3.95}$$

We now manipulate $F(u^{(k)})$ as follows:

$$\begin{aligned}
F(u^{(k)}) &= \frac{\lambda}{2}||u^{(k)} - b||_2^2 = \frac{\lambda}{2}\langle u^{(k)} - b, u^{(k)} - b\rangle \\
&= \frac{\lambda}{2}\Big\langle \frac{u^{(k)} + u^*}{2} - b, u^{(k)} - b\Big\rangle + \frac{\lambda}{2}\Big\langle \frac{u^{(k)} - u^*}{2}, u^{(k)} - b\Big\rangle \\
&= \frac{\lambda}{2}\Big\langle \frac{u^{(k)} + u^*}{2} - b, \frac{u^{(k)} + u^*}{2} - b\Big\rangle + \frac{\lambda}{2}\Big\langle \frac{u^{(k)} + u^*}{2} - b, \frac{u^{(k)} - u^*}{2}\Big\rangle \\
&+ \frac{\lambda}{2}\Big\langle \frac{u^{(k)} - u^*}{2}, u^{(k)} - b\Big\rangle \\
&= \frac{\lambda}{2}\left\| \frac{u^{(k)} + u^*}{2} - b\right\|_2^2 + \frac{\lambda}{2}\Big\langle \frac{u^{(k)} - u^*}{2}, \frac{u^{(k)} + u^*}{2} - b + u^{(k)} - b\Big\rangle \\
&= \frac{\lambda}{2}\left\| \frac{u^{(k)} + u^*}{2} - b\right\|_2^2 + \frac{\lambda}{2}\Big\langle \frac{u^{(k)} - u^*}{2}, \frac{u^{(k)} - u^*}{2} + u^{(k)} + u^* - 2b\Big\rangle \\
&= \frac{\lambda}{2}\left\| \frac{u^{(k)} + u^*}{2} - b\right\|_2^2 + \frac{\lambda}{2}\left\| \frac{u^{(k)} - u^*}{2}\right\|_2^2 + \lambda\Big\langle \frac{u^{(k)} - u^*}{2}, \frac{u^{(k)} + u^*}{2} - b\Big\rangle \\
&\geq \frac{\lambda}{2}\left\| \frac{u^{(k)} + u^*}{2} - b\right\|_2^2 + \lambda\Big\langle \frac{u^{(k)} - u^*}{2}, \frac{u^{(k)} + u^*}{2} - b\Big\rangle.
\end{aligned} \tag{3.96}$$

On the other hand, we have that

$$\begin{aligned}
\langle \rho^*, Du^{(k)} - Du^*\rangle &= \langle \rho^*, D(u^{(k)} - u^*)\rangle = \langle D^T\rho^*, u^{(k)} - u^*\rangle \\
&= \lambda\langle u^* - b, u^* - u^{(k)}\rangle,
\end{aligned} \tag{3.97}$$

where in (3.97) we have used the (optimality) condition

$$\rho^* \in \overline{\partial}_t[R](Du^*) \text{ such that } D^T\rho^* + \lambda(u^* - b). \tag{3.98}$$

From (3.96) and (3.97) it follows that

$$\begin{aligned}
&F(u^{(k)}) + \langle \rho^*, Du^{(k)} - Du^*\rangle \\
&\geq \frac{\lambda}{2}\left\| \frac{u^{(k)} + u^*}{2} - b\right\|_2^2 + \lambda\Big\langle \frac{u^{(k)} - u^*}{2}, \frac{u^{(k)} + u^*}{2} - b\Big\rangle + \lambda\langle u^* - b, u^* - u^{(k)}\rangle \\
&= \underbrace{\frac{\lambda}{2}||u^* - b||_2^2}_{F(u^*)} + \frac{3}{8}\lambda||u^{(k)} - u^*||_2^2,
\end{aligned} \tag{3.99}$$

that is

$$F(u^{(k)}) - F(u^*) + \langle \rho^*, Du^{(k)} - Du^* \rangle \geq \frac{3}{8}\lambda ||u^{(k)} - u^*||_2^2. \tag{3.100}$$

Taking the limit for $k \to \infty$ of both sides of (3.100) and recalling (3.87) and (3.95), we obtain:

$$0 \geq \lim_{k\to\infty} \frac{3}{8}\lambda ||u^{(k)} - u^*||_2^2 \Longrightarrow \lim_{k\to\infty} u^{(k)} = u^*, \tag{3.101}$$

thus completing the proof. □

We conclude this analysis with the following final theorem,

Theorem 3.12. *Let the pair of parameters* (λ, a) *satisfy condition* (3.9) *and let the parameter* β *satisfy condition* (3.46)*. Then, for any initial guess, the sequence* $\{u^{(k)}\}_{k=1}^{+\infty}$ *generated by Algorithm* 1 *satisfies:*

$$\lim_{k\to+\infty} u^{(k)} = u^*, \tag{3.102}$$

with u^* *denoting the unique solution of problem* (3.1)*.*

3.1.4 Results

We recall that the proposed CNC method consists of two steps. In the first step an approximate solution of the minimization problem (3.1) is computed by means of the ADMM-based iterative procedure illustrated in Algorithm 1; iterations are stopped as soon as two successive iterates satisfy $||u^{(k)} - u^{(k-1)}||_2 / ||u^{(k-1)}||_2 < 10^{-4}$. In the second step the segmented regions are obtained by means of a thresholding procedure.

In order to satisfy all the conditions mentioned above, such as (3.9), (3.30), (3.44) or (3.45) we set $\tau_c = 10$ and $\lambda = 30$, then the parameter a and, then, the lower bound $\overline{\beta}$ of the ADMM penalty parameter β are automatically obtained based on (3.9) and (3.45): $a = \frac{\lambda}{9\tau_c} = \frac{1}{3}$ and $\beta > \overline{\beta} = \max\left\{2a, 9a\frac{\tau_c}{9\tau_c - 8}\right\} = \max\left\{\frac{2}{3}, \frac{30}{82}\right\} = \frac{2}{3}$. So we choose to impose $\beta = 10$.

Once we have choose the parameters, in the first part of the code, we address the resolution of `OUR_Seg_SPL` (Phase 1), which consists in solving a CNC variant of the Mumford-Shah/Chan-Vese segmentation model formulated in a gradient-regularized framework.

The objective is to recover a function u, representing a smoothed and segmented image, that remains faithful to the observed data *Img* (fidelity term weighted by the parameter λ) while promoting piecewise regular structures with sharp discontinuities (nonconvex regularization on the gradient magnitude $|\nabla u|$, controlled by the parameters a, T, α).

To this end, we employ the Alternating Direction Method of Multipliers (ADMM), introducing an auxiliary variable $t = (t_x, t_y)$ to approximate the image gradient $|\nabla u|$.

This reformulation enables the treatment of the nonconvex regularizer through an associated proximal mapping, which is implemented as a nonlinear shrinkage operator with a double-threshold structure. The ADMM iterations consist of two main steps:

(i) Update in u: the variable u is obtained as the solution of the elliptic linear system

$$(\lambda I - \beta\Delta)u = \lambda Img + \beta \mathrm{div}(t - \rho/\beta),$$

which is solved efficiently in the Fourier domain (FFT) under periodic boundary conditions.

(ii) Update in t: the variable t is updated by applying a nonconvex proximal map (depending on a, T, α) to the "perturbed" gradient vectors, which incorporate the ADMM multipliers ρ.

The input arguments of the function are

- the image Img,
- λ: weight of the data fidelity term (the larger it is the closer u remains to Img),
- β: ADMM penalty parameter that influences convergence and the subproblems,
- τ_c: scaling factor for a,
- T: "shape threshold" of the nonconvex regularizer,
- α: shape parameter of the penalty (handled in a special way in the code when $\alpha == 1/3$),
- its_max: maximum number of iterations,
- err_th: stopping threshold on the relative variation of u.

whereas the outputs are the reconstructed image u and the number of iterations i.

We now provide a detailed description of the algorithmic steps implemented in the code, specifying the operations performed at each stage.

1. **Initialization and constants for the penalty** where a must satisfy $a < \frac{\lambda}{9}$ for convexity of the CNC functional, $\beta > a$ to guarantee convexity of the t-subproblem, $K1, TH1, TH2, \gamma1, \gamma2, \gamma3$ define the proximal operator of the SPL penalty (piecewise shrinkage thresholds). Here it is also called the function `phi_spl2` that has as inputs an array of points (ts), a threshold that divides the behaviors of the function (T), parameters that shape the shape of the penalty (a, α), a constant $K1$

```
a = tau_c * lambda/9;
beta = max(beta,1.2*a);
K1 = 1/(a^(8));
TH1 = T + K1/beta;
TH2 = T + K1/a;
gamma1 = beta/(beta-a);
gamma2 = (K1+a*T)/(beta-a);
gamma3 = (4*K1)/(beta*T^2);
ts = linspace(0,1,2000);
[ys,V] = phi_spl2(ts,T,a,alpha,K1,0);
plot_y_max = 1.05*max(ys);
```

Figure 3.1

(depending on a) and $flag\_norm$ which normalizes the function so that it gets to 1 at most. The outputs are ys whose values are the values of the function $\phi(ts)$, i.e. the penalty applied to the gradient and V that is the upper threshold value. The function defines a piecewise non-convex penalty. In case of small gradients ($ts < T$) the penalty grows slowly so as not to over-penalize small variations; in case of medium gradients ($T \leq ts < V$) the penalty grows less than linearly reducing bias on strong edges; in case of big gradients ($ts \geq V$) the penalty is constant preserving edges.

2. **ADMM variables:** We start from $u = \text{Img}$ and we initialize the auxiliary t and

```
u = Img;
tx = 0*Img;
ty = tx;
rhox = tx;
rhoy = tx;
```

Figure 3.2

multipliers ρ to zero.

3. **Kernel for u subproblem:** It builds the 5-point Laplacian stencil with periodic boundary conditions; it transforms it to Fourier space: $\hat{A} = \lambda + \beta\hat{\Delta}$. It enables solving linear systems with a simple division in Fourier domain.

4. **Main loop:** The first part is for the u-step: it is computed the right hand side

```
uker = 0*Img;
uker(1,1)=4; uker(1,2)=-1; uker(2,1)=-1; uker(end,1)=-1; uker(1,end)=-1;
uker = lambda + beta * fft2(uker);
```

Figure 3.3

```
for i = 1:its_max
rhs = lambda * Img + beta * ( Dxt(tx-rhox/beta) + Dyt(ty-rhoy/beta) );
u0 = u;
u = ifft2(fft2(rhs)./uker);
err=norm(u-u0,'fro')/norm(u,'fro');
if mod(i,10)==0
    disp(['iterations: ' num2str(i) '!  ' 'error is:   ' num2str(err)]);
end
if err<err_th
    break;
end
rx   = Dx(u) + rhox / beta;
ry   = Dy(u) + rhoy / beta;
if ( a == 0 )
    [tx,ty] = shrink2(temp1,temp2,1./beta);
else
    r_norm = sqrt(rx.^2 + ry.^2);
    xi = ones( size(rx) );
    if ( alpha == 1/3 )
        xi(r_norm<TH1)                      = 2 ./ ( 1 + sqrt( 1 + gamma3 * r_norm(r_norm<TH1) ) ) ;
        xi((r_norm>=TH1) & (r_norm<TH2))    = gamma1 - gamma2 ./ r_norm((r_norm>=TH1) & (r_norm<TH2));
    end
    tx  = rx .* xi;
    ty  = ry .* xi;
end
rhox = rhox - beta * ( tx - Dx(u) );
rhoy = rhoy - beta * ( ty - Dy(u) );
```

Figure 3.4

of the linear system $\lambda Img + \beta \text{div}(t - \rho/\beta)$ where `Dxt, Dyt` are adjoint difference operators and the linear system $(\lambda I - \beta\Delta)u = \lambda Img + \beta \text{div}(t - \rho/\beta)$ is solved using the inverse fast Fourier transform. It is measured then the relative change in u for stopping. In the second part the subproblem for t is solved: first of all, $r = \nabla u + \rho/\beta$ is computed, it is the argument fed into the proximal operator; it represents the current gradient of the image corrected by the ADMM dual variable serving as the input to the shrinkage step, where the algorithm decides whether to shrink (suppress noise/small gradients) or keep (preserve edges/large gradients). Then there are two cases: the one in which $a = 0$ and the one in which $a > 0$, that is the nonconvex case. When $a = 0$, the non convexity of the CNC regularizer disappears and the penalty reduces to the Total Variation (TV) norm of the gradient, i.e.

$$\sum \phi(|\nabla u|) \Rightarrow \sum |\nabla u|.$$

The segmentation model, thus becomes a Chan-Vese type model with TV regularization. In this situation is applied an isotropic soft-thresholding shrinkage. When $a > 0$, the shrinkage factor ξ_i is introduced and if $\alpha = 1/3$, i.e. our case, it is defined piecewise: to small $||r||$ corresponds heavy shrinkage (denoising), to medium $||r||$ corresponds partial shrinkage and to large $||r||$ corresponds no shrinkage (preserve strong edges). Then tx and ty are computed. Finally the Lagrange multiply ρ is updated enforcing the $t = \nabla u$.

Subsequently, we analyze the Phase 2 of the code, the one that focuses on the segmentation. Once the image u has been regularized, phase 2 consists in two steps: calculating segmentation thresholds automatically with `ThdKmeans` and visualization and construction of final segmentation with `SegResultShow`.

```
function th=ThdKmeans(uu,k)

[xLen,yLen]=size(uu);
idx=kmeans(uu(:),k);
idx=reshape(idx,[xLen,yLen]);
mean_u(k)=0;
for i=1:k
    temp=(i==idx).*uu;
    mean_u(i)=sum(temp(:))/(sum(temp(:)>0));
end
mean_u=sort(mean_u);

th(k-1)=0;
for i=1:k-1
    th(i)=mean(mean_u(i:i+1));
end
end
```

Figure 3.5

1. This function is intended to find the $(k-1)$ thresholds that divide the image into k classes. The input arguments are the denoised image u and the k classes. A K-means pixel clustering is performed. K-means is applied to u image intensity values (not to coordinates) and each pixel receives a label from 1 to k. The result is `idx`, that is an image of cluster labels. Then the average values for each cluster are calculated. For each cluster, it takes the pixels of u in that cluster and it calculate the average intensity of the pixels. Next it reorders these averages in ascending order. Last the construction of the thresholds is carried out computing the averages between adjacent average values. The output will be `th`, a vector of $(k-1)$ thresholds dividing the image into k regions.

```
function seg=SegResultShow(Img,uu,th,k)
for i=1:k-1
    figure,imshow(Img,'border','tight');
    title('contour')
    hold on;
    if 1==i
        temp=uu<th(i);
    elseif k-1==i
        temp=(uu>=th(i-1)).*(uu<th(i));
        temp1=uu>=th(i);
    else
        temp=(uu>=th(i-1)).*(uu<th(i));
    end
    contour(temp,'y');
end
if k>2
    figure,imshow(Img,'border','tight');
    hold on;
    contour(temp1,'y');
end
[xLen,yLen]=size(Img);
seg=zeros(xLen,yLen);
```

```
for i=1:k-1
    if 1==i
        temp=uu<th(i);
        seg=seg+temp*sum(sum(temp.*uu))/sum(temp(:));
        if 2==k
            temp=uu>=th(i);
            seg=seg+temp*sum(sum(temp.*uu))/sum(temp(:));
        end
    elseif k-1==i
        temp=(uu>=th(i-1)).*(uu<th(i));
        seg=seg+temp*sum(sum(temp.*uu))/sum(temp(:));
        temp=uu>=th(i);
        seg=seg+temp*sum(sum(temp.*uu))/sum(temp(:));
    else
        temp=(uu>=th(i-1)).*(uu<th(i));
        seg=seg+temp*sum(sum(temp.*uu))/sum(temp(:));
    end
end
figure,imshow(seg,'border','tight');
end
```

Figure 3.6

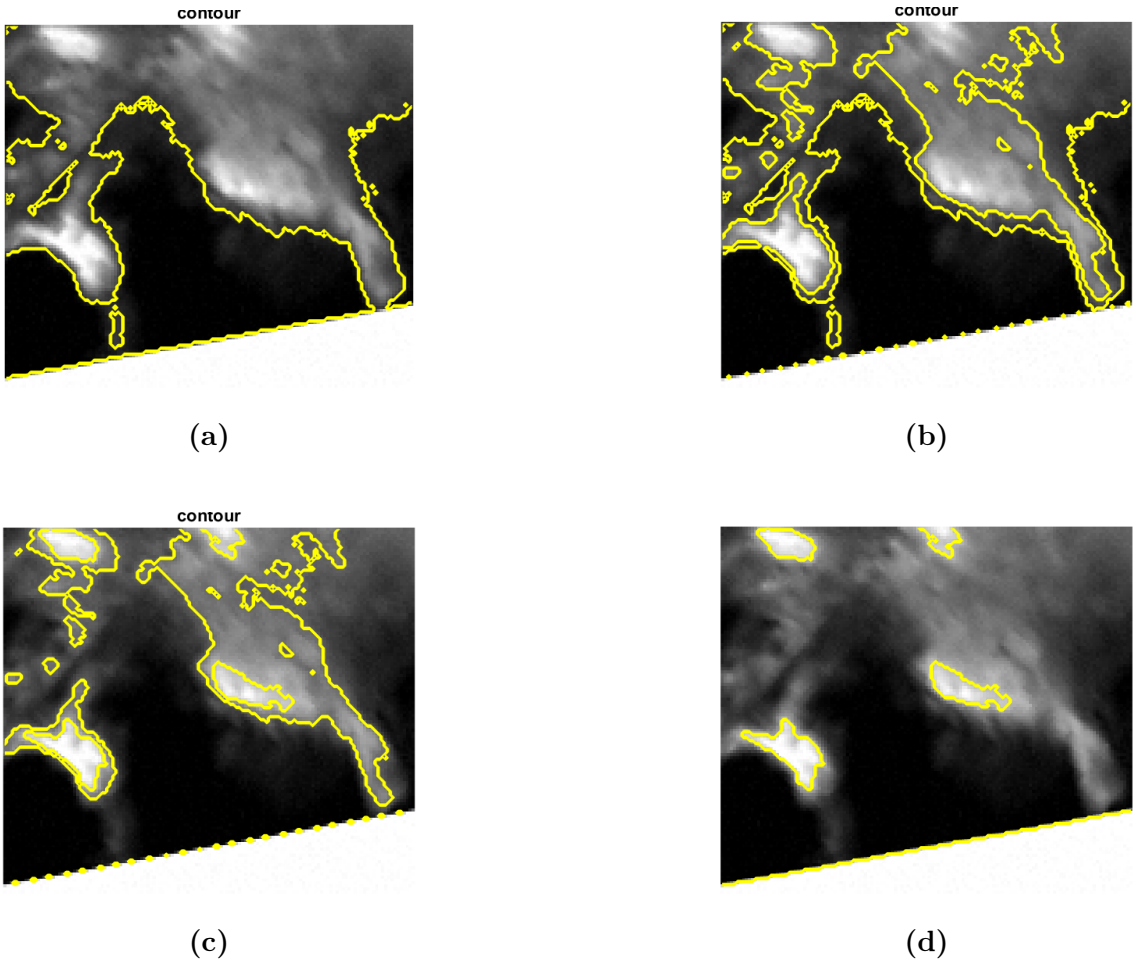
2. This function has the objective to display contours and generate segmented image

`seg`.

In the first part, the outlines are visualized by constructing a Boolean mask for each threshold. The second part concerns the construction of the final segmentation: for each region, defined by thresholds, all pixels are assigned the average value of u in the region and then it is added to `seg`. The output `seg` is a piecewise-constant image with k regions, that represents the final segmentation.

The algorithm was applied to three transesophageal ultrasound images, each corresponding to a different individual with atrial fibrillation. For each image, a region of interest (ROI) was selected to make the segmentation more targeted. The parameter $k = 4$ was chosen for the thresholds employed later in the segmentation process. Subsequently, the parameters λ, a and T were selected to obtain a clean and accurate segmentation. After the execution of the two phases of the algorithm, the ADMM phase and the actual segmentation phase, the largest connected component within the segmentation was extracted. Since this component consisted of two parts, namely the left atrial appendage and an adjacent region, a manual separation was performed between the two, and only the left atrial appendage was retained as the final mask.

1. **Patient** 1: $\lambda = 30,\ T = 0.001,\ k = 4,\ \alpha = 1/3,\ \tau_c = 10,\ \beta = 10.$



(a) (b)

(c) (d)

Figure 3.7

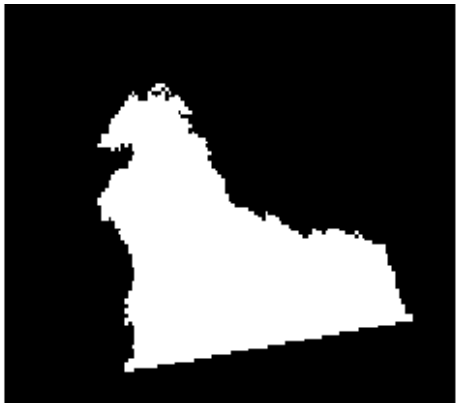
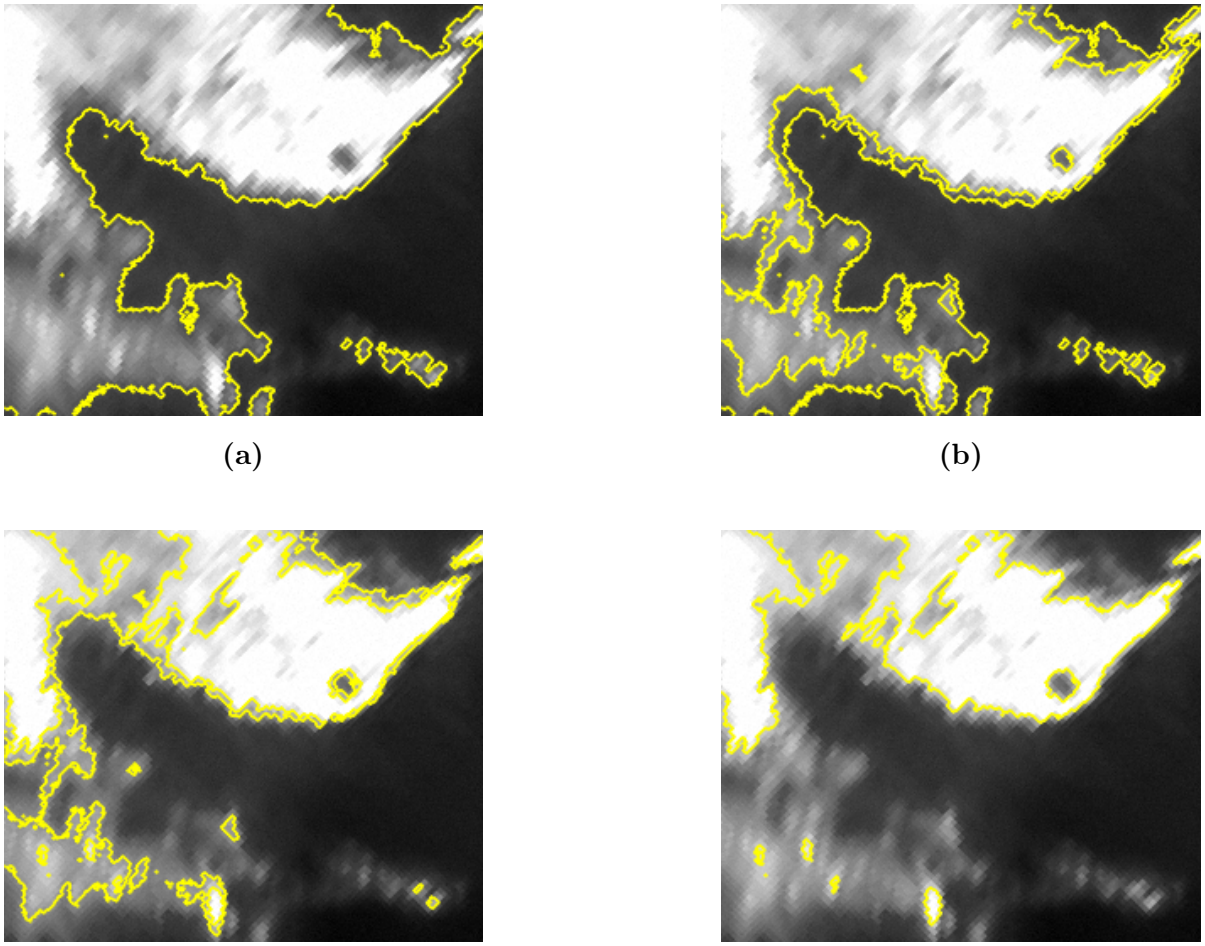


Figure 3.8: *Final segmentation*

2. **Patient** 2: $\lambda = 30,\ T = 0.001,\ k = 4,\ \alpha = 1/3,\ \tau_c = 10,\ \beta = 10.$



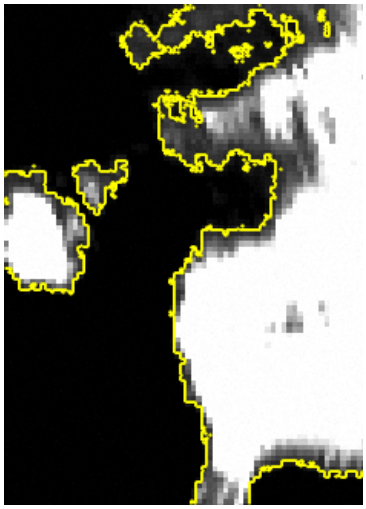
(a) **(b)**

(c) **(d)**

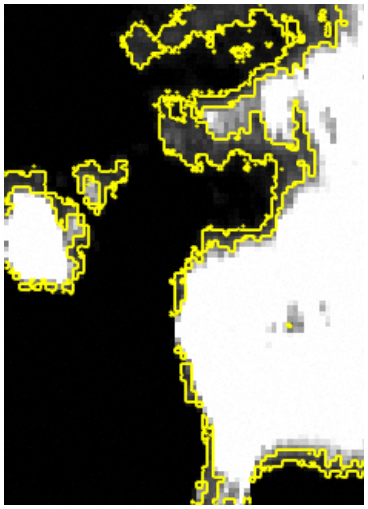
Figure 3.9

Figure 3.10: *Final segmentation*

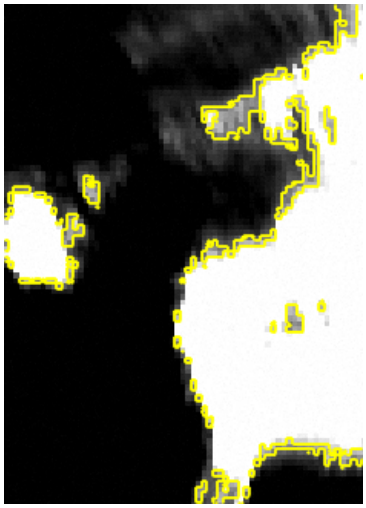
3. **Patient** 3: $\lambda = 30,\ T = 0.001,\ k = 4,\ \alpha = 1/3,\ \tau_c = 10,\ \beta = 10.$



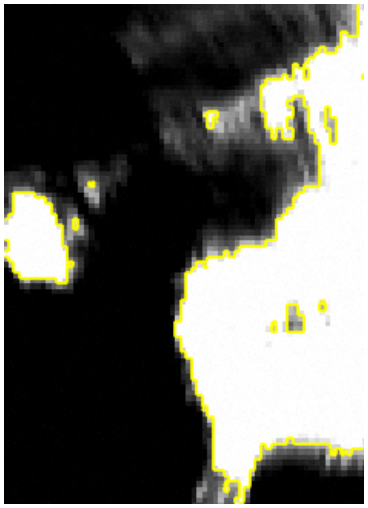
(a)



(b)



(c)



(d)

Figure 3.11

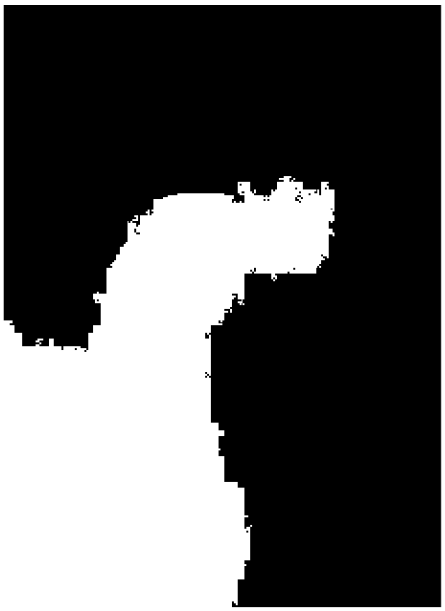


Figure 3.12: *Final segmentation*

3.2 Unsupervised Multiphase Segmentation

Starting from the paper "Unsupervised Multiphase Segmentation: a phase balancing model", [12] we introduce a variational model investigating multiphase segmentation with a new regularization term that yields an unsupervised segmentation model.

In this work, we introduce a method for multiphase image segmentation that incorporates a new regularization term, leading to an unsupervised segmentation model. In other words, this model does not require the number of phases to be fixed a priori, but instead decides it automatically during the segmentation process.

The proposed functional is designed not only to segment the image but also to automatically determine a suitable number of phases. This means that the model simultaneously identifies the regions and decides how many distinct phases are necessary, avoiding both under- and over-segmentation.

The new regularization term relies on the scale measure of the phases, which can be seen as a weight depending on the size of each region. Unlike classical models such as Chan-Vese, where the regularization typically penalizes the length of the contours to enforce smooth boundaries, here the scale measure introduces a preference toward larger and more stable regions. In practice, bigger objects are "rewarded", while very small or noisy phases are penalized, reducing the risk of spurious segmentations.

As a consequence, larger objects are preferred to be identified, while small or insignificant regions tend to merge into bigger ones unless they exhibit a strong intensity difference. At the same time, the segmentation itself is still driven by the intensity fitting term, which ensures that pixels are grouped into phases whose mean intensity closely

matches their own gray-level values. Thus, the intensity fitting term enforces the separation of regions based on their grayscale distributions, while the scale-based regularization determines how many phases should remain and favors the elimination of unstable small regions.

Our focus is on multiphase segmentation, meaning the task of distinguishing more than two distinct regions within a given image. This is fundamentally more challenging than simple object/background separation, since the model must simultaneously locate the regions and determine their number in an unsupervised fashion.

Image segmentation divides an image into different regions in order to simplify its analysis and facilitate object identification. Over time, numerous approaches have been developed, ranging from mixed random field models and the variational model of Mumford and Shah to more complex methods such as Monte Carlo Markov chains, graph-cut techniques and spectral methods, and variational models for texture segmentation. A fundamental contribution came from the Mumford-Shah model, which inspired many extensions.

This work specifically addresses multiphase segmentation, namely the identification of more than two regions within an image. Several region-based multiphase models have been proposed in the literature, including those of Vese and Chan, Chung and Vese, Brox and Weickert, Tai and Chan, Lie, Lysaker and Tai, Jung, Kang and Shen, and Bae and Tai. These approaches include: the generalization of the two-phase model using multiple level sets, multilayer methods inspired by island dynamics in epitaxial growth, energy minimization strategies within the level set framework, piecewise level set methods with constant values for each phase, the use of graph-cut algorithms in the multiphase Mumford-Shah model, and relaxed models based on Γ-convergence analysis.

Let Ω be a bounded Lipschitz domain, and $u_0 : \Omega \to \mathbb{R}_+ \cup \{0\}$ be a given image. Recall that the classical Mumford-Shah segmentation is to minimize

$$\mathcal{E}_{ms}[u, \Gamma | u_0] = \alpha \int_{\Omega \setminus \Gamma} |\nabla u|^2 dx + \beta \mathcal{H}^1(\Gamma) + \int_{\Omega} (u - u_0)^2 dx, \tag{3.103}$$

where $\Gamma \in \Omega$ denotes the edge set of the image u, and $\mathcal{H}^1$ represents the 1-dimensional Hausdorff measure. The energy functional is made of three main terms:

- $\alpha \int_{\Omega \setminus \Gamma} |\nabla u|^2 dx$ penalizes intensity variations inside the regions. In practice, it enforces smoothness of the function u within each phase, away from the edges. This means that, within a single region, the intensity is encouraged to remain regular and not oscillate too much.

- $\beta\mathcal{H}^1(\Gamma)$ penalizes the length of the boundaries Γ. Every additional contour increases the energy, so the model favors segmentations with fewer, simpler, and smoother boundaries. This prevents over-segmentation and promotes more compact regions.

- $\int_\Omega (u-u_0)^2 dx$ is the data fitting term. It requires the segmented image u to remain close to the original image u_0. In other words, it prevents the segmentation from deviating too far from the actual data and ensures that the result still reflects the observed image.

Multiphase segmentation identifies different phases by the intensity discontinuities. For identifying piecewise constant objects, the reduced Mumford-Shah functional can be written as, minimizing

$$\mathcal{E}_{cv}[u, \Gamma | u_0] = \beta\mathcal{H}^1(\Gamma) + \sum_{i=1}^{N} \int_{\Omega_i} |u_0 - c_i|^2, \tag{3.104}$$

where Ω_is are the connected components of $\Omega \setminus \Gamma$ and c_i is the intensity average of u in each Ω_i.

The simplified Mumford-Shah model, when successfully implemented with level-sets for a two-region segmentation, is commonly known as the Chan-Vese (CV) model. One key disadvantage of these multiphase segmentation approaches is that the number of separate regions (phases) must usually be fixed in advance, that is, chosen before running the algorithm. Many classical methods, such as simplified Mumford-Shah or Chan-Vese, require the user to explicitly set a parameter K, telling the algorithm to divide the image into K regions. This is a limitation, since in practice we do not always know beforehand how many objects or classes are present in an image.

One common strategy to avoid making this choice is to start with a large number of initial contours, in the hope that, during the optimization, the unnecessary contours will vanish and only the meaningful ones will remain. The idea is that the evolution of the level sets or contours will naturally suppress redundant phases. However, this approach is heuristic and does not guarantee an optimal result.

In practice, if more phases are supplied than truly needed, the outcome is often over-segmentation: the image is divided into too many regions, more than what is semantically meaningful. This results in fragmented outputs where small, insignificant areas are treated as separate phases, even though they do not correspond to real objects or relevant structures. Over-segmentation is therefore a direct consequence of initializing the model with an excessive number of phases or setting the phase parameter too high. In addition, for the functional models, such as (3.103) and (3.104), there are at least one

free parameter to choose, like α, β, μ and, often for multiphase segmentation, the result becomes very sensitive to the choice of these parameters.

3.2.1 Description of the model

We propose a new model for unsupervised segmentation, which automatically chooses the number of phases without any user input. In other words, the model decides on its own how many distinct regions are required. Unlike classical approaches, the user does not need to specify in advance whether the image should be divided into, for example, three or five classes, the number of phases emerges automatically during the process.

This model not only detects the boundaries of the regions but also simultaneously determines a "reasonable" number of phases, neither too few nor too many. As a result, the method avoids common issues such as under-segmentation, where relevant structures are missed, and over-segmentation, where the image is fragmented into excessively many small regions.

The contribution of this work is twofold. First, we propose a new variational functional for unsupervised multiphase segmentation. This functional incorporates a regularization mechanism that naturally penalizes unnecessary phases, allowing the appropriate number of regions to emerge from the optimization. Second, we provide a fast numerical algorithm for solving the problem, making the method not only theoretically well-founded but also computationally practical.

In particular, instead of deriving and solving the Euler-Lagrange equations typically used for such nonlinear variational models, which would lead to a system of PDEs, we introduce a brute-force algorithm. This approach, while simple, is efficient and easy to implement, often working by updating pixel labels in a discrete fashion. Such a strategy makes the method more accessible while still maintaining strong performance.

The segmented image is represented as a linear combination of different phases, each of which is defined by a characteristic functions χ_i representing that phase. The $\sum_i \chi_i$ covers the entire image domain Ω and $\chi_i \cap \chi_j = \emptyset$. Each χ_i is defined by one intensity average value c_i, and each phase may be disconnected, i.e. it may consist of many separate regions. The average value is computed by

$$c_i = \frac{\int_{\chi_i} u_0(x)dx}{\int_{\chi_i} 1dx},$$

which corresponds to the ratio of the sum of the pixel intensities in phase i to the number of pixels in that phase. Then, the final segmented result is obtained combining all the

phases where each pixel takes the average value of the phase to which it belongs:

$$u = \sum_{i=1}^{K} c_i * \chi_i \tag{3.105}$$

In the CV model (3.104) the main two terms represent the intensity fitting term and the regularization term. In order to prevent oscillatory boundary identification, the length of the boundaries is kept to a minimum while the intensity drives the segmentation.

We design our new functional based on this CV model (3.104) and retain these two terms for our segmentation model. In order to create an unsupervised model that automatically determines the number of phases and image segmentation, we add two more goals. The objectives are as follows:

1. *Phase*: find the objects with significant sizes. We prefer not to have small partitions of the image, but want to identify relatively big objects which can be understood as a feature of the image.

2. *Balance*: we assume each identified phases are all equally important, i.e. no a priory information is given on which phase is more important than the other. Therefore, this functional should partition the image uniformly among different phases

A. Phase

We focus on the scale term defined by the formula $scale := \frac{area}{length}$. A larger object results in a greater scale value, whereas a smaller object results in a lower scale value. This concept is examined in relation to total variation (TV) denoising. To achieve our goal of detecting significant sized objects while reducing a functional, we utilize the inverse of that quantity, that is the *inverse scale term*,

$$\mathcal{S}_i := \frac{P(\chi_i)}{|\chi_i|} \tag{3.106}$$

where $P(\chi_i)$ denotes the perimeter of a phase χ_i and $|\chi_i|$ denotes the 2-dimensional area of a phase χ_i, i.e. the number of pixels in the discrete implementation. If an object is large, its area increases faster than its perimeter: when scaling linearly by a factor t, the area grows proportionally to t^2, while the perimeter grows only proportionally to t. As a result, for larger objects the ratio $\frac{P(\chi_i)}{|\chi_i|}$ tends to decrease. On the other hand, if an object is small or very irregular or thin, the value of $\frac{P(\chi_i)}{|\chi_i|}$ is relatively high. Therefore, minimizing the term (3.106) naturally favors phases with a low perimeter-to-area ratio, that is, objects that are larger and more compact. This directly addresses the stated objective of the model: to prioritize the identification of meaningful, significant-size objects while discouraging excessively small partitions of the image, thus reducing the risk of over-segmentation.

We can remark that the inverse scale $\mathcal{S}_i$ is defined on each phase χ_i, thus it is feasible to have disconnected regions within a phase. Regardless of the number of connected components in this phase χ_i, we calculate the total length of the edges in the phase and divide it by the total area of objects in the phase χ_i. A number of objects with comparable intensities, for instance, will be in the same phase χ_i and contribute to $\mathcal{S}_i$ collectively.

We can also state two properties: the first one is that for a phase χ_i with a single object, if the perimeter is fixed, convex object have smaller $\mathcal{S}_i$ compared to the concave objects. Therefore, by minimizing $\mathcal{S}_i$ the shape of object prefers to be closer to a circle rather than an ellipse.

The second one is that objects with different shape can have the same inverse scale value. For example, any regular (equilateral) convex polygon, B, which incircles a circle with radius r has $\frac{P(B)}{|B|} = \frac{2}{r}$, which is the same as the inverse scale value of a circle with radius r.

B. Balance

To maintain equilibrium between the phases, we do not assign any specific weight to any phase and we do not consider the summation with any particular weight,

$$\sum_{i=1}^{K} \mathcal{S}_i = \sum_{i=1}^{K} \frac{P(\chi_i)}{|\chi_i|} \tag{3.107}$$

so the functional considers every phase in the same way.

Then, for a given discrete bounded image with $|\Omega| < \infty$ and $\sum_{i=1}^{K} P(\chi_i) < \infty$, for a fixed number of phases K, the minimum of the summation is achieved when $\mathcal{S}_i$ are all equal across phases, for $\forall i = 1, \ldots, K$, i.e. $\mathcal{S}_1 = \mathcal{S}_2 = \ldots = \mathcal{S}_K$. This is basically a "balancing" condition: the functional pushes the values of each phase's scale ratio toward equality.

Therefore, by minimizing this term, the objects of various sizes in the image will uniformly be distributed among all different phases and $\mathcal{S}_i$ value will be similar to each other. That means no single phase will dominate while others collapse into negligible regions. We refer to this term as the *phase balancing term*, since it prefers to find balance among the scales of each phases.

Proposition 3.13. *For a fixed K, given a piecewise constant image with multiple objects B_j with the same ratio, $\frac{P(B_j)}{|B_j|} := p_1$ (except for the background), any distribution of these objects B_j to different phases χ_i (no empty phases, no partial objects) gives minimum of the phase balancing term, and*

$$\sum_{i=1}^{K} \frac{P(\chi_i)}{|\chi_i|} = (K-1)p_1 + \frac{P(\chi_b)}{|\chi_b|} \tag{3.108}$$

where χ_p represents the background.

It implies that if an image contains only one type of object (that is, all objects share the same ratio $P(B)/|B|$), then the segmentation may distribute them across the different clusters χ_i in very different ways. For instance, one phase might contain many objects, another only a single one, and yet another just a few. Nevertheless, the value of the balancing term would still remain minimal. In contrast to the balancing effect, the Proposition 3.13 states that if an image contains only one type of item, the quantity of objects in each phase may differ significantly. This is because of how we calculate the $\mathcal{S}_i$, since it depends only on the ratio $P/|\chi|$. As a consequence, the balancing mechanism fails to distinguish between more balanced and less balanced distributions whenever the geometries are homogeneous.

C. The proposed model

By taking these extra goals into consideration, we propose the following functional for automatic multiphase segmentation, a *phase balancing model*,

$$E[K, \chi_i, c_i|u_0] = \hat{\mu}\left(\sum_{i=1}^{K} \mathcal{S}_i\right)\mathcal{H}^1(\Gamma) + \sum_{i=1}^{K}\int_{\chi_i} |u_0 - c_i|^2, \tag{3.109}$$

where Γ is set of all the boundaries of χ_i for $i = 1, \ldots, K$, i.e. $\Gamma = \bigcup_{i=1}^{K}\{\partial_{\chi_i}\}$, $\mathcal{H}^1$ represents the 1-dimensional Hausdorff measure. that is the 1-dimensional measure of the boundaries, as in (3.103) and (3.104) and the average value c_is are defined as in (3.104), $c_i = \frac{\int_{\chi_i} u_0(x)dx}{\int_{\chi_i} 1dx}$. Notice that K, χ_is and c_is in $E[K, \chi_i, c_i|u_0]$ are unknown variables while only the original image u_0 is given. Compared to other multiphase models, which do not minimize the functional with regard to the number of phases K, this is one of the primary differences. The suggested functional can also be expressed as follows, using $P(A)$ to represent the finite perimeter of the set A:

$$E[K, \chi_i, c_i|u_0] = \mu\left(\sum_{i=1}^{K}\frac{P(\chi_i)}{|\chi_i|}\right)\sum_{i=1}^{K} P(\chi_i) + \sum_{i=1}^{K}|u_0 - c_i|^2\chi_i. \tag{3.110}$$

Here $\mu = \hat{\mu}/2$ from $\hat{\mu}$ in (3.109), since When summing the perimeters of all the phases, each internal boundary between two adjacent phases is counted twice, once for each region sharing that boundary, whereas $\mathcal{H}^1(\Gamma)$ counts each boundary segment only once. Therefore, in the absence of image-domain boundaries, we approximately have $\sum_{i=1}^{K} P(\chi_i) = 2\mathcal{H}^1(\Gamma)$.

The only free parameter in this functional is μ. Its role is to balance the relative importance of the scale regularization term (the first term) against the data fitting term (the second term). This μ is a parameter that represents the area of the segmentation

because the first term of (3.110) is virtually unit-free, while the second term $\sum_{i=1}^{K} |u_0 - c_i|^2 \chi_i$ corresponds to the area of the phase. When μ is large, the model gives more weight to regularization and phase balancing, thus the segmentation favors phases with larger regions; when μ is small, more emphasis is placed on data fitting. This allows the model to create smaller phases in order to capture finer intensity variations, but at the cost of a higher risk of over-segmentation. We maintained this $\mu = 1$ for unsupervised segmentation, and the implications of these variations of μ are examined in the next section.

The energy functional is highly non-convex, particularly because both the number of phases K and the topology of the regions χ_i are free variables. This non-convexity implies the presence of many local minima, a strong dependence on the initialization, and the necessity of using greedy or heuristic algorithms in order to obtain a good solution. We remark that the perimeter $P(\chi_i)$ appears twice in the model (3.110): once in the total length term and once in the phase balancing term. Since a convex polygon and a circle can have the same inverse scale value, the smoothness of the border is independent of minimizing $\mathcal{S}_i$. For its smoothness quality (and well-fitted borders), the total length term $\mathcal{H}^1(\Gamma)$ is therefore required.

A second observation is that intensity is the primary factor driving segmentation. Although the bigger continuous zone is preferred by the model, smaller sections of comparable intensity will also enter the same phase. Small items with a large inverse scale value, like noise or tiny stars, will be recognized as features in the image rather than being totally obscured. Denoising happens within a specific intensity range. The intensity will be included in the phase that is closest to it.

Finally we can notice that after proposing to add two additional objectives in the form of the phase balancing term, $\sum_{i=1}^{K} \mathcal{S}_i$, we had different options to modify the new functional. For example, Case I, adding (not multiplying) the phase balancing term to CV model (3.104): this looses the unsupervised properties and the results become heavily depended on the choice of two parameters, β and μ. This inherits limitations of CV model with additional parameter to choose. Case II, multiplying $\sum \frac{1}{\mathcal{S}_i}$ to the total length term, i.e. two terms in the segmentation function both represents the area, and Case III, multiplying the phase term to the fitting term, i.e. both terms represents length of the segmentation. In both Case II and Case III, we found that the number of phases K grows to reduce energy, in addition to being sensitive to the parameter selection. Both situations usually result in a large number of phases since the fitting term might become zero by continuously increasing the number of phases, but the total length term is never zero.

3.2.2 Fast algorithm for Multiphase Segmentation

The proposed segmentation method is based on representing each phase of the image by a characteristic function χ_i, where $\chi_i(x, y) = 1$ if pixel (x, y) belongs to phase i and $\chi_i(x, y) = 0$ otherwise. The segmented image is then expressed as

$$u = \sum_{i=1}^{K} c_i \chi_i \tag{3.111}$$

where c_i denotes the average intensity value of phase i, and K is the current number of phases. This representation has two advantages: it simplifies the addition of new phases and avoids bias in the transition between phases since each phase is independent.

The energy functional to be minimized consists of two main components:

1. A regularization term, weighted by μ, which combines a phase balancing measure (perimeter-to-area ratio of each phase) and the total length of the interfaces. This favors larger and more coherent regions.

2. An intensity fitting term, which measures the squared difference between the observed image u_0 and the mean intensity c_i within each phase:

$$\sum_{i=1}^{K} \sum_{(x,y)} |u_0(x, y) - c_i|^2 \chi_i(x, y).$$

The regularization term mixes a measure of the phase's scale (such as the perimeter-to-area ratio, $\frac{P(\chi_i)}{n_i}$) with the total boundary length (the sum of all perimeters). This interplay causes the model to favor larger objects, since small regions have a high perimeter-to-area ratio and are consequently penalized.

The proposed model (3.110)

$$E[K, \chi_i, c_i | u_0] = \mu \left(\sum_{i=1}^{K} \frac{P(\chi_i)}{|\chi_i|} \right) \sum_{i=1}^{K} P(\chi_i) + \sum_{i=1}^{K} |u_0 - c_i|^2 \chi_i. \tag{3.112}$$

had some analytical difficulties: it is non linear with 3 different unknown, K, χ_i and c_i and deriving the Euler-Lagrange equations (analytical solutions) is complicated. For this reason, we adopt a direct and efficient minimization strategy based on a greedy algorithm. At each iteration, the algorithm evaluates, for every pixel, the change in energy that would occur if the pixel were reassigned from its current phase to another. The pixel is then moved to the phase that produces the largest energy decrease. The greedy algorithm operates by evaluating the energy change (ΔE) for each possible assignment of a pixel to a phase. The pixel is then assigned to the phase that results in the greatest reduction in energy. This process is repeated for all pixels, leading to an efficient and

effective segmentation. This idea follows the approach of monitoring the variation of the functional when a pixel changes its membership (inside/outside) and applying a greedy decision rule. While such methods have been successfully used in two-phase segmentation, here we extend them to the multiphase setting. In our case, the algorithm not only considers moving a pixel among all existing phases but also includes the possibility of creating a new phase whenever this further reduces the energy.

For $(x, y) \in \Omega$ the change in energy when (x, y) moves from one phase l to another phase j is computed by,

$$\Delta E_{lj} = \mu \Delta ST + (u - c_j)^2 \frac{n_j}{n_j + 1} - (u - c_l)^2 \frac{n_l}{n_l - 1} \tag{3.113}$$

where $u = u(x, y)$ is the intensity value at the pixel (x, y), c_i is the average of each phase i, and n_i is the number of pixels in phase i, i.e. area $|\chi_i| = n_i$. The first term ΔST is the change of the phase balancing and total length term in (3.110), and other two terms are the change of the intensity fitting term.

The expression above can be interpreted term by term.

- The first term, $\mu \Delta ST$, reflects the effect of the regularization. This includes both the phase-balancing contribution and the change in the total perimeter of the segmentation. In other words, it measures how the smoothness and balance of the partition are affected when the pixel moves from phase l to phase j.

- The second term, $(u - c_j)^2 \frac{n_j}{n_j+1}$, represents the increase in the squared error when the intensity value u is added to phase j. This comes from the fact that, when a point is included in a cluster with mean c_j, the sum of squared errors increases by exactly this amount.

- The third term, $-(u - c_l)^2 \frac{n_l}{n_l-1}$, accounts for the decrease in the squared error of phase l when the pixel u is removed from it. The derivation is symmetric to the previous case, but now the removal reduces the total fitting error for phase l.

Taken together, ΔE_{lj} expresses the total change in energy resulting from moving the pixel from phase l to phase j. It combines the effect on the regularization term with the variation in data-fitting error, providing the greedy criterion that guides the algorithm's decision at each pixel.

Then, this ΔST is

$$\Delta ST = S_j T_j - S_l T_l = S_j(T_l + \Delta T) - S_l T_l = (S_j - S_l)T_l + S_j \Delta T = T_l \Delta S + S_j \Delta T \tag{3.114}$$

where S_l presents the phase balancing energy ($\sum \mathcal{S}_i$) and T_l the total length energy ($\sum P(\chi_i)$) when (x, y) is in phase l. So the effect of regularization can be assessed if we

know how the perimeters of the two phases change and their corresponding perimeter-to-area ratios. To compute the total length T_l, since each phase is represented by a characteristic function χ_i, we simply add all the edges in the phase to get the length,

$$P(\chi_i) = \sum_{(x,y)\in\Omega} \{|\chi_i(x+1,y) - \chi_i(x,y)| + |\chi_i(x,y+1) - \chi_i(x,y)|\} \tag{3.115}$$

then, when the pixel is in phase l, $T_l = \sum_{i=1}^{K} P(\chi_i)$ and $S_l = \sum_{i=1}^{K} \frac{P(\chi_i)}{n_i}$.

The difference in total length energy, ΔT becomes an addition of the change of perimeter in phase j and the change in phase l, $\Delta T = \Delta P(\chi_j) + \Delta P(\chi_l)$. In phase j, if pixel (x, y) changes from 0 to 1, the change in the length can be computed from the values of the neighboring points, i.e. $\Delta P(\chi_j) = 4 - 2\sum_{(a,b)\in\mathcal{N}} \chi_j(a,b)$, where $\mathcal{N}$ refers to four neighboring points (N,S,E,W of (x,y)). When there were no edges ($\forall(a,b) \in \mathcal{N}, \chi_j(a,b) = 0$), by changing this pixel from 0 to 1, it creates four new edges. If there is one edge, by flipping $\chi_j(x,y) = 0 \to 1$, it creates two additional edges. If there were two edges, the change creates no new edges, but if there were four edges, it will remove those four edges (-4). Similarly, change in the perimeter of phase l becomes $\Delta P(\chi_l) = -4 + 2\sum_{(i,j)\in\mathcal{N}} \chi_l(i,j)$. Then, the difference in the total length becomes

$$\Delta T = -2\left(\sum_{(i,j)\in\mathcal{N}} \chi_j(i,j) - \sum_{(i,j)\in\mathcal{N}} \chi_l(i,j)\right) \tag{3.116}$$

The difference ΔE_{lj} in (3.113) can be computed by gathering all these terms,

$$\Delta E_{l,j} = \mu(T_l\Delta S + S_j\Delta T) + (u - c_j)^2\frac{n_j}{n_j+1} - (u - c_l)^2\frac{n_l}{n_l-1}. \tag{3.117}$$

This is an explicit difference of the energy when the pixel changes from one phase l to another phase j, which is used in the algorithm, Table 3.1.

If $\Delta E_{lj} > 0$, the pixel will not change to phase j since that will increase the energy. While, if this value ΔE_{lj} is negative, it is better to move (x, y) to phase j. By iterating this process over the entire image domain, the segmentation gradually converges to a configuration that minimizes the functional.

At the beginning of the computation, the whole image is assigned to a single phase ($K = 1$). During the iterations, if moving a pixel into a newly created phase decreases the energy, then a new phase is introduced automatically. This new phase is represented as phase $k + 1$ and the difference in the energy in the algorithm 3.1 is calculated using $n_{k+1} = 0$. This mechanism allows the algorithm to estimate the number of phases directly from the data, without requiring prior specification.

The computational complexity of the algorithm is relatively simple to analyze. Let m denote the total number of pixels in the image. At the very first pixel, there are only

Algorithm

- Set an initial phase: $|\chi_1| = |\Omega|$ with $k_o = 1$, where k_o is the number of phase.
- Iterate
  1. At each pixel $(x, y) \in \Omega$ which belongs to phase l ($\chi_l(x, y) = 1$ and $\chi_i(x, y) = 0\ \forall i \neq l$), compute

$$value = \min_j \{\Delta E_{lj} \mid j \neq l,\ j = 1, \ldots, k+1\},$$

  and let $h = \arg\min_j \{\Delta E_{lj} \mid j \neq l,\ j = 1, \ldots, k+1\}$. Here $k+1$ refers to the new empty phase. Then,

$$\begin{cases} \text{if } value < 0, & \chi_h(x, y) = 1 \text{ and } \chi_l(x, y) = 0, \\ \text{if } value > 0, & \text{do nothing.} \end{cases}$$

  2. Update $k = h$, calculate $n_i = |\chi_i|$ and c_i for each phase $i = 1, \ldots, k$.

Table 3.1: A Pixelwise Brute-force Algorithm

two possible choices: remaining in the current phase χ_1 or creating a new phase χ_2. As the algorithm proceeds through the domain, each pixel has $k + 1$ possible assignments, where k is the current number of phases and the additional option corresponds to the creation of a new phase. For a brute-force approach, if the number of phases is fixed to r, the complexity is $O(rm)$ When the number of phases grows with each iteration, the complexity becomes $O(m + 2m + \ldots + sm)$ where s is the maximum number of phases considered, which leads to an overall complexity of $O(s^2 m)$. In contrast, the proposed algorithm typically identifies the correct number of phases after only one sweep over the image. As a result, its complexity reduces to $O(km) = O(m)$, where k is the number of phases found. This matches the performance of the fast algorithm in [24], which also achieves linear complexity $O(m)$.

3.2.3 Results

Now we will analyze the code used for the segmentation of the ultrasound data.

The code implements a greedy, discrete, pixel-wise algorithm to minimize the functional of a model of unsupervised multiphase segmentation with phase balancing. The objective is to assign each pixel an `Index`$(x, y) = i$ phase label so as to reduce energy

$$E = \mu\Big(\sum_i \frac{P(\chi_i)}{|\chi_i|}\Big)\Big(\sum_i P(\chi_i)\Big) + \sum_i \sum_{x \in \chi_i} (u(x) - c_i)^2$$

where $P(\chi_i)$ defines the perimeter of the phase i, $|\chi_i| = n_i$ is the number of pixels in the phase i and μ is the parameter that balances regularization and fitting: it can be seen as the weight that decides how important geometric regularization (edge/perimeter, phase balancing) is compared to data fitting. If μ is large the algorithm favors solutions with short edges and regular and compact regions, even at the cost of getting it a little wrong the intensity. Regions become more compact, smooth and less fragmented, but you risk losing fine details. On the other hand if μ is small data fitting dominates. Pixels are labeled almost only by looking at the similarity of intensity to phase averages in fact each pixel looks for the phase with average closest to its intensity. Very precise segmentation on the value plane is achieved, but often with jagged edges and scattered tiny phases.

The code updates label by label deciding whether to move the pixel to another phase, even creating $k + 1$ new phase, only if the move decreases energy ($\Delta E < 0$).

Now we report the nomenclature of the variables used in the code:

- `Im0` is the image;
- `Index` is the map of labels (phase index) for each pixel;
- `n(i)` is the number of pixels in the phase i, ($|\chi_i|$);
- `c(i)` is the phase intensity average i;
- `length(i)` is the perimeter ($P(\chi_i)$);
- `T` is the total sum of perimeters ($\sum_i P(\chi_i)$);
- `F` is the sum of $\mathcal{S}_i$ ($\sum_i \frac{P(\chi_i)}{n_i}$);
- μ is a balancing parameter;
- `k` is the current number of actual phases (initializes to 1 and can increase to the allocated maximum).

The "local" energy that the code evaluates when it tries to move the pixel (x, y) from phase l to phase j is:

$$\Delta E_{lj} = \mu((F + dF)(T + dT) - FT) + \Delta(\text{fitting})$$

that in the code is:

$$\mu(dF \cdot T + F \cdot dT + dF \cdot dT) + \frac{(u - c_j)^2 n_j}{n_j + 1} - \frac{(u - c_l)^2 n_l}{n_l - 1},$$

where the last two terms are the variation of the fitting term (pixels to j are added, pixels to l are removed) and `dF`, `dT` are the local variations of `F` and `T`.

The decision to move the pixel from phase l to phase j combines several criterion: fitting, pixel prefers the phase with average `c(j)` similar to its intensity; perimeters, pixel prefers moves that shorten total contours; scale balancing, it penalizes small and jagged phases and favors larger/compact regions.

Below I will analyze the code block by block.

1. The image is loaded and normalized, μ is set, space is pre-allocated for up to 20 phases (they could be also 10, the number of phases depends on the parameter μ). `Index` assigns all pixels to stage 1 initially; `n(1)` is given by $N_x \times N_y$ and `c(1)` is given by the global average. `T` and `F` are initialized to 0.

```
[Im0,map] = imread('TEE2.png'); Im0=rgb2gray(Im0);
[Nx,Ny] = size(Im0);
Im0 = double(Im0);
mu = 0.7;
n=zeros(1,20); c=n; length=c;
Index = ones(size(Im0));
k = 1;
n(1) = sum(sum((Index==1)));
c(1) = sum(sum(Im0.*(Index==1)))/n(1);
T=0; F=0;
```

Figure 3.13

2. For every pixel the best energy decrease found is initialized (`dE`$= 0$), note that 0 means "no improvement" and it will only move the pixel if it finds `dE_temp` < 0.
   `l` is given as the current phase of the pixel (x, y). We consider all stages $1, \ldots, k$ and also $k + 1$ (create a new stage) as possible destinations, skipping the case $j = l$.
   `tj,tl` are initialized to 0 and then they will accumulate the perimeter variation of the target phase j and the source phase l. For each neighbor, add a contribution: for j, `tj`$+ = 1$ if the neighbor is not j, $= -1$ if it is j but it is scaled as $-2*(== j)+1$. At the end `tj`$= 4-2*s_j$ with s_j is the number of neighbors in j. For l, `tl`$= 2*s_l-4$ with s_l the number of neighbors in l. If many neighbors are already j, adding the pixel to j reduces the perimeter by j, so `tj` becomes negative. If many neighbors are l, removing the pixel from l increases the perimeter by l, so `tl` grows.
   The change in the sum of the perimeters, `dT` is then calculated `dT=tj+tl`. If the pixel is more surrounded by j than by l, `dT` is negative, thus the total contours are shortened. It is also computed the variation of the balancing term, `dF`: only phases j and l change, $P_j \to P_j+tj$, $n_j \to n_j+1$, $P_l \to P_l+tl$, $n_l \to n_l-1$. The code expands algebraically $\Delta(\frac{P}{n})$ for the two phases. Then, local variation in energy, `dE_temp` is computed. The term $\mu(F \cdot T)$ is derived as $\Delta(FT) \sim dF \cot T + F \cdot dT + dF \cdot dT$. The last two terms are fitting: cost of adding the pixel to j (with average `c(j)` and new area `n(j)`$+1$) minus the cost of leaving it in l (but recalculated after removal:

`n(l)`-1). The fractions with `n(j)`$/$(`n(j)`$+1$) and `n(l)`$/$(`n(l)`-1) derive from the incremental calculation of the sum of the quadratic deviations with respect to the average.

If the move is the best yet and improves energy, so if `dE_temp` is less than `dE`, `dE` is replaced with `dE_temp`, and a record is kept of the destination $h = j$ and what variations to apply if you then decide to move. So if we have achieved a real improvement, that is, if `dE`< 0, the value h is assigned to `Index`(x, y), i.e. the pixel is moved to phase h. Furthermore, if h is the new phase $(k + 1)$, it increases the number of phases k. At last the averages, areas and geometric terms are updated incrementally.

```
for iter=1:5
    num_fasi = max(Index(:));
    colors = lines(num_fasi);
    rgbImage = zeros(Nx, Ny, 3);
    for x=1:Nx
        for y=1:Ny
            dE = 0;
            l=Index(x,y);
             rgbImage(x,y,:) = colors(Index(x,y), :);
            for j=1:k+1
                if j~=l
                    tj=0;tl=0;
                    if x ~= 1
                        tj=tj-2*(Index(x-1,y)==j)+1;
                        tl=tl+2*(Index(x-1,y)==l)-1;
                    end
                    if y ~= 1
                        tj=tj-2*(Index(x,y-1)==j)+1;
                        tl=tl+2*(Index(x,y-1)==l)-1;
if x ~= Nx
    tj=tj-2*(Index(x+1,y)==j)+1;
    tl=tl+2*(Index(x+1,y)==l)-1;
end
if y ~= Ny
    tj=tj-2*(Index(x,y+1)==j)+1;
    tl=tl+2*(Index(x,y+1)==l)-1;
end
dT = tj+tl;
dF = (tj)/(n(j)+1)+(tl)/(n(l)-1) +length(l)/((n(l)-1)*n(l));
if n(j)~=0
    dF = dF - length(j)/((n(j)+1)*n(j));
end
dE_temp = mu*(dF*T+F*dT+dF*dT)+(Im0(x,y)-c(j))^2*n(j)/(n(j)+1)-(Im0(x,y)-c(l))^2*n(l)/(n(l)-1);
if dE_temp < dE
    dE=dE_temp;
    dF_c=dF;
    h=j;
                        dT_h=tj;
                        dT_l=tl;
                    end
                end
            end
            if dE<0
                Index(x,y)=h;
                if (h==(k+1))
                    k=k+1;
                end
                c(l) = c(l) + (c(l)-Im0(x,y))/(n(l)-1); n(l) = n(l)-1;
                c(h) = c(h) - (c(h)-Im0(x,y))/(n(h)+1); n(h) = n(h)+1;
                F=F+dF_c;
                T=T+dT_h+dT_l;
                length(l) = length(l) + dT_l;
                length(h) = length(h) + dT_h;
            end
        end
    end
end
```

Figure 3.14: *Main sequence*

The algorithm was applied to three transesophageal ultrasound images, each corresponding to a different individual with atrial fibrillation. For each image, a region of interest (ROI) focusing on the left atrial appendage was selected, and the parameter μ was set accordingly. After running the algorithm, the resulting segmentations were visualized for iterations 1 to 5 and for phases 1 to 20, depending on the selected μ value. This visualization allowed the identification of the most suitable segmentation by selecting the corresponding iteration and phase. Once the optimal segmentation was chosen, the largest connected component was extracted, and, where necessary, a manual cropping was performed to isolate the left atrial appendage.

1. **Patient** 1: $\mu = 0.7$

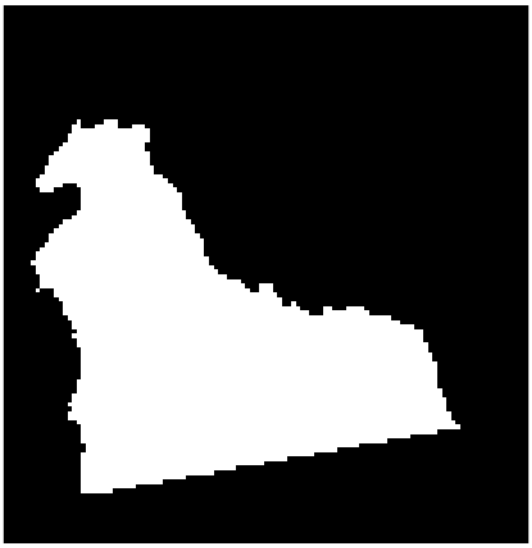


Figure 3.15: *Final segmentation*

2. **Patient** 2: $\mu = 4$

Figure 3.16: *Final segmentation*

3. **Patient** 3: $\mu = 0.7$

Figure 3.17: *Final segmentation*

3.3 Frame based segmentation for medical images

Starting from the paper "Frame based segmentation for medical images", [13] we present a model that combines ideas of the frame based image restoration model with ideas of the total variation based segmentation model.

A frame-based model is a mathematical approach used to restore degraded images by leveraging concepts from frame theory. In mathematics and signal processing, a frame can be seen as a redundant but flexible representation, similar to a basis. Unlike orthogonal bases, such as Fourier coefficients, frames are not required to be independent of each other, and they may even overlap. This redundancy makes them more robust against noise and information loss, which is particularly useful in image restoration tasks. The process of frame-based image restoration generally involves three main steps:

- **Analysis**: The degraded image is transformed into a frame domain (for example, using wavelet frames, curvelet frames, or other types of redundant representations). In this domain, structural features such as edges and textures become easier to analyze.

- **Thresholding / Denoising**: The coefficients obtained from the frame transformation are filtered. This step removes noise, blur, or other artifacts while preserving important image details.

- **Synthesis**: Finally, the filtered coefficients are transformed back into the original image domain, reconstructing a clean and restored version of the image.

In practice, frame-based models are powerful because they combine mathematical rigor with practical robustness. By working in a redundant representation, these models can effectively separate meaningful image structures from unwanted degradations, making them a cornerstone in modern image processing and restoration.

Within this framework, we propose a segmentation model based on tight frames. Tight frames are a special class of mathematical frames, conceptually similar to wavelets, that provide a redundant yet stable representation of signals and images. Unlike orthogonal bases, they do not require independence among components, allowing for overlap and redundancy. This property enables simple and accurate reconstruction while maintaining robustness to noise and data loss.

A key advantage of tight frames is that their redundancy typically leads to sparse approximations of images: most of the image information can be captured by only a few significant coefficients. This sparsity is highly desirable in many restoration problems, such as denoising, inpainting, and deblurring, since it makes it easier to separate essential image structures from noise or artifacts.

Moreover, tight frames are particularly effective because real-world images are often piecewise smooth: they consist of large homogeneous regions, like the sky, interspersed with sharp transitions, such as object edges. Tight frames can efficiently represent both aspects capturing smooth regions compactly while preserving edges with high precision. This dual ability to encode uniform areas and sharp discontinuities makes them especially suitable for segmentation tasks, where distinguishing between homogeneous regions and boundaries is crucial for accuracy.

3.3.1 Frames and Framelets

In this subsection, we briefly introduce the concept of tight frames and framelets.

A countable set $X \subset L_2(\mathbb{R})$ is called a tight frame of $L_2(\mathbb{R})$ if

$$f = \sum_{h \in X} \langle f, h \rangle h \quad \forall f \in L_2(\mathbb{R}), \tag{3.118}$$

where $\langle \cdot, \cdot \rangle$ is the inner product of $L_2(\mathbb{R})$. For given $\Psi := \{\psi_1, \ldots, \psi_r\} \subset L_2(\mathbb{R})$, the affine (or wavelet) system is defined by the collection of dilations and the shifts of Ψ as

$$X(\Psi) := \{\psi_{l,j,k} : 1 \leq l \leq r; j, k \in \mathbb{Z}\} \quad \text{with} \quad \psi_{l,j,k} := 2^{j/2} \psi_l(2^j \cdot -k). \tag{3.119}$$

When $X(\Psi)$ forms a tight frame of $L_2(\mathbb{R})$, it is called a tight wavelet frame, and ψ_l, $l = 1, \ldots, r$ are called the (tight) framelets.

To construct a set of framelets, usually, one starts from a compactly supported refinable function $\phi \in L_2(\mathbb{R})$ (a scaling function) with a refinement mask h_0 satisfying

$$\hat{\phi}(2\cdot) = \hat{h}_0\hat{\phi}. \tag{3.120}$$

Here $\hat{\phi}$ is the Fourier transform of ϕ, and $\hat{h}_0$ is the Fourier series of h_0 with $h_0(0) = 1$ which means that a refinement mask of a refinable function must be a lowpass filter. A lowpass filter allows low frequencies to pass through while attenuating or eliminating high frequencies. The purpose of a lowpass filter is to smooth or blur the image by removing high-frequency details such as edges and textures. In doing so, it suppresses the fine variations in the signal and emphasizes its broader components. In the context of signal processing, this means that the lowpass filter is designed to capture the coarse structure or the global shape of the signal, providing a simplified representation that retains the essential large-scale features while discarding the small-scale details. On the other hand, a highpass filter does the opposite: it allows high frequencies to pass through while attenuating or eliminating low frequencies. This makes it particularly effective at highlighting edges and fine details, which are represented by high-frequency components. In signal processing, the highpass filter is therefore used to capture local details and rapid changes in the signal, such as edges or textures, complementing the coarse representation obtained with the lowpass filter.

For a given compactly supported refinable function, the construction of a tight framelet system is to find a finite set Ψ that can be represented in the Fourier domain as

$$\hat{\psi}_l(2\cdot) = \hat{h}_l\hat{\phi} \tag{3.121}$$

for some 2π-periodic $\hat{h}_l$. The unitary extension principle (UEP) of [19] says that $X(\Psi)$ in (3.119) generated by Ψ forms a tight frame in $L_2(\mathbb{R})$ provided that the masks $\hat{h}_l$ for $l = 0, 1, \ldots, r$ satisfy

$$\sum_{l=0}^{r} \hat{h}_l(\xi)\overline{\hat{h}_l(\xi + \gamma\pi)} = \delta_{\gamma,0}, \quad \gamma = 0, 1 \tag{3.122}$$

for almost all ξ in $\mathbb{R}$. While h_0 corresponds to a lowpass filter, $\{h_l; l = 1, 2, \ldots, r\}$ must correspond to highpass filters by the UEP. The sequences of Fourier coefficients $\{h_l; l = 1, 2, \ldots, r\}$ are called *framelet masks*. In our implementation, we adopt the piecewise linear B-spline framelet. The refinement mask is $\hat{h}_0(\xi) = \cos^2(\frac{\xi}{2})$, whose corresponding lowpass filter is $h_0 = \frac{1}{4}[1, 2, 1]$. Two framelets are $\hat{h}_1 = -\frac{\sqrt{2}i}{2}\sin(\xi)$ and $\hat{h}_2 = \sin^2(\frac{\xi}{2})$, whose corresponding highpass filters are

$$h_1 = \frac{\sqrt{2}}{4}[1, 0, -1], \quad h_2 = \frac{1}{4}[-1, 2, -1]$$

The associated refinable function and framelets are given in Figure 3.18.

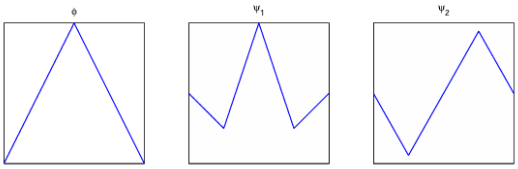


Figure 3.18: *Piecewise linear refinable spline and framelets*

With a one-dimensional framelet system for $L_2(\mathbb{R})$, the s-dimensional framelet system for $L_2(\mathbb{R}^s)$ can be easily constructed by tensor products of one-dimensional framelets. If we have one scaling function and r tight framelets in 1D, then after tensor product, we obtain a tight frame system generated by one scaling function and $(r+1)^s - 1$ tight framelets. In the discrete setting, a discrete image f is considered as the coefficients $\{f_i = \langle f_c, \phi(\cdot - i)\rangle\}$ up to a dilation, where f_c is the continuous version of f, ϕ is the refinable function associated with the framelet system, and $\langle \cdot, \cdot \rangle$ is the inner product in $L_2(\mathbb{R}^s)$. The L-level discrete framelet decomposition of f is then the coefficients $\{\langle f, 2^{-L/2}\phi(2^{-L} \cdot -j)\rangle\}$ at a prescribed coarsest level L, and the framelet coefficients

$$\{\langle f, 2^{-l/2}\psi_i(2^{-l} \cdot -j)\rangle\},\ 1 \le i \le (r+1)^s - 1 \tag{3.123}$$

for $0 \le l \le L$.

A discrete s-dimensional image, which is an s-dimensional array, can be un derstood as a vector living in $\mathbb{R}^n$, with n the total number of pixels in the image. For simplicity of notations, we represent the framelet decomposition and reconstruction as matrix multiplications Wu and $W^T v$ respectively. Here $W \in \mathbb{R}^{k \times n}$ satisfies $W^T W = I$, i.e. $u = W^T W u$, $\forall u \in \mathbb{R}^n$, by the unitary extension principle.

Now we introduce some notations: let W_0 be the submatrix of W that corresponds to the decomposition with respect to the refinable function; and denote $W_{l,i}$ with $1 \le l \le L$ and $1 \le i \le (r+1)^s - 1$, the submatrix of W that corresponds to the decomposition at the l-th level with respect to the i-th framelet. Under this notation, W can be written as

$$W = \begin{pmatrix} W_0 \\ W_{l,i} \end{pmatrix} = \begin{pmatrix} W_0 \\ W_{1,1} \\ W_{1,2} \\ \vdots \\ W_{L,(r+1)^s-1} \end{pmatrix}.$$

All $W_{l,i}$ and W_0 have the same number of rows, and we denote that number as m.

3.3.2 Segmentation model

The model is inspired by TV-based methods (such as Mumford-Shah or Chan-Vese), but instead of using total variation regularization, it penalizes the l_1-norm of tight frame coefficients Wu instead of the l_1-norm of the gradient ($|\nabla u|$). This has two main advantages:

- **Sparsity and Smoothness**: Using tight frames allows for a sparser representation of not only piecewise constant functions (like standard TV) but also piecewise smooth functions. This leads to better results in image restoration and segmentation, especially for images that aren't perfectly piecewise constant.
- **Richer Geometric Information**: The framelet transform contains more detailed geometric information. Specifically, for a certain choice of parameters ($s = 2$), the model uses not only first-order difference operators (like gradients) but also second and even fourth-order difference operators. This provides a much richer description of the image geometry.

For an observed image $f \in \mathbb{R}^n$ the goal is to find $u \in [0,1]^n$, a "soft" indicator of the segmented region, and two intensity values c_1, c_2, minimizing:

$$\min_{0\leq u\leq 1, c_1, c_2} ||g_W \cdot Wu||_1 + \mu r(c_1, c_2)^T u, \tag{3.124}$$

where $||\cdot||_1$ denotes the l_1-norm, W a framelet transform, μ a parameter that balances the two terms of the equation and $r(c_1, c_2) = (c_1 - f)^2 - (c_2 - f)^2$ where c_1 and c_2 are real constants and they represent the average intensity values inside and outside the segmented region. Here, g_W is a diagonal weight matrix depending on the image and it is defined as,

$$g_W = \text{diag}\{0^T, v_{1,1}^T, v_{1,2}^T, \ldots, v_{1,(r+1)^s-1}^T, \ldots, v_{L,(r+1)^s-1}\}$$

where $v_{l,i} \in \mathbb{R}^m$ and $0 \in \mathbb{R}^m$. Then

$$g_W \cdot Wu = \sum_{l,i} v_{l,i}(W_{l,i}u).$$

It acts as an edge indicator, designed to stop the evolution of the segmentation at the boundaries of objects. Its weights are calculated based on the gradient of the image under the framelet transform: near edges (large values of Wf), the weights are smaller, so the penalty is weaker, preventing oversmoothing at discontinuities; in flat regions, the penalty is stronger. A simple choice is to use the same weighting function across scales and directions. The weight function $v_{l,i}$ can be defined in several different ways. In this

work, however, we set it to be the same function v for every l and i, where

$$v(j) = \frac{1}{1 + \sigma \sum_{i=1}^{(r+1)^s - 1} |(W_{1,i} f)(j)|^2} \quad \text{for } j = 1, 2, \ldots, m.$$

To solve this complex problem,it is used an iterative optimization technique called the split Bregman iteration. This method breaks the main problem down into a series of simpler sub-problems that can be solved more easily. Following a similar derivation and using the fact that $W^T W = I$, we obtain the following algorithm for (3.124):

$$\begin{aligned}
u^{k+\frac{1}{2}} &= W^T(d^k - b^k) - \frac{\mu}{\lambda} r(c_1^k, c_2^k) \\
u^{k+1} &= \max\{\min\{u^{k+\frac{1}{2}}, 1\}, 0\} \\
d^{k+1} &= \mathcal{T}_{g_W/\lambda}(W u^{k+1} + b^k) \\
b^{k+1} &= b^k + (W u^{k+1} - d^{k+1}) \\
c_1^{k+1} &= M(f, \Omega^{k+1}), \;\; c_2^{k+1} = M(f, (\Omega^{k+1})^c), \;\; \Omega^{(k+1)} = \{u^{(k+1)} > \alpha\}.
\end{aligned} \tag{3.125}$$

where μ is a parameter from equation (3.124) that controls the trade-off between the regularization term and the data fidelity term; λ is another parameter that comes from Bregman iteration; $\alpha \in [0, 1]$ is a parameter used to define the level set of the solution u^*, which determines the final segmentation boundary; $\mathcal{T}_\delta$ is a soft-thresholding operator, a key component in l_1-minimization algorithms like the one used here, defined as

$$(\mathcal{T}_\delta(x))(j) := \begin{cases} x(j) - \delta(j) & \text{if } x(j) > \delta(j) \\ 0 & \text{if } -\delta(j) \leq x(j) \leq \delta(j) \\ x(j) + \delta(j) & \text{if } x(j) < -\delta(j) \end{cases} \tag{3.126}$$

and $M(f, \Omega)$ returns the mean value of f within domain Ω.

After we obtain a solution u^* from the algorithm (3.125), the segmentation of image f is given by the α level set of u^* . It is proven for TV-based model that any α level set of u^* , for almost all $\alpha \in [0, 1]$, gives a meaningful segmentation of image f.

We can note that (3.125) is a very efficient algorithm. For each iteration k, the most time consuming operation is performing fast framelet decomposition and reconstruction, i.e. W and W^T, which are of the same complexity as fast Fourier transform. In practice, the method remains efficient and converges within a few hundred iterations.

3.3.3 Results

Let's analyze the code block by block.

1. First, the ultrasound image is loaded and converted from integer pixel matrix to double type: this is important to avoid truncations in numerical operations. A copy of the original image is created to display it and normalized to the range $[0, 1]$. A control variable, used later to handle the order of c_1/c_2 averages or other modes internal to the algorithm, `Switch` is set to 1.

```
f = double(rgb2gray(imread('TEE12.png')));
f0=f;
f0=f/255;
Switch=0;
```

Figure 3.19

2. The main parameters are set: `tol` is the convergence tolerance for the stopping criterion; `mu` is the weight of the data term, a term that links u to the averages c_1 and c_2 in the Chan-Relaxed Vessel model and higher values imply greater attachment to data and so less smoothing; `lambda` is the splitting parameter, used in the Split-Bregman scheme. Its choice balances the convergence speed and shrinkage threshold.

```
tol=1e-3;
mu=5;
lambda=mu*0.05;
```

Figure 3.20

3. Preparation begins for calculating the edge indicator function $g(x)$.

   `frame` and `frame_edge` choose the type of framelet ($0 \rightarrow$ Haar, $1 \rightarrow$piecewise linear, $3 \rightarrow$piecewise cubic). They are used to regularization in segmentation and to estimate edges, respectively.

   `Level` and `Level_edge` specify the number of multiscale decomposition levels.

   `GenerateFrameletFilter(frame_edge)` returns the 1D filters for analysis (`De`) and synthesis (`R`) associated with the chosen framelet. Then, by applying a 2D Gaussian mask, a smoothed version of the image is obtained, where noise and fine details are attenuated. The blurring is used to reduce noise and to compute more reliable gradients, and therefore stronger edges in the image, which will later be used to guide the segmentation.

```
maxit = 200;
frame = 3;
Level = 2;
frame_edge = 3;
Level_edge = 2;
[De,R] = GenerateFrameletFilter(frame_edge);
nDe = length(De);
[m,n] = size(f);
std_Gb = 1;
beta = 50/255^2;
Ng = ceil(6*std_Gb)+1;
Gaussian = fspecial('gaussian',[Ng Ng],std_Gb);
fs = conv2(f,Gaussian,'same');
```

Figure 3.21

4. A multiscale framelet decomposition and detail energy calculation occurs. Function `FraDecMultilevel` is called to decompose the smoothed image `fs` into multi-level framelet coefficients.

   `C_data`$\{ki\}\{ji, jj\}$ is the coefficient matrix at the level ki and for the horizon/ vertical filter combination indicated by (ji, jj). `NormGrad` is constructed by adding the squares of the coefficients of the detail bands, excluding the $ji = 1$ and $jj = 1$ band which is the low-low component, i.e. the low frequency approximation. The sum of the squares is a measure of high-frequency energy: points with high values in `NormGrad` correspond to strong edges or details. The equation

   $$\texttt{NormGrad}(x) = \sum_{k=1}^{L} \sum_{(i,j)\neq(1,1)} C_{k,i,j}(x)^2$$

   provides a spatial map of the "edge strength" that will be used to construct an edge indicator.

```
C_data=FraDecMultiLevel(fs,De,Level_edge);
NormGrad = 0;
for ki=1:Level_edge
    for ji=1:nDe-1
        for jj=1:nDe-1
            if ji~=1 || jj~=1
                NormGrad=NormGrad+C_data{ki}{ji,jj}.^2;
            end
        end
    end
end
```

Figure 3.22

5. The edge indicator, `d_data_s` is constructed there. It is initialized with all 1s; then in the internal positions (excluding the edges of the image with $3 : \text{end} - 2$) it is defined as

   $$d(x) = \frac{1}{1 + \beta\ \text{NormGrad}(x)}.$$

   When `NormGrad` is large (i.e. strong edge) then $d(x)$ it is small (near 0). When `NormGrad` is small (i.e. flat area) then $d(x)$ it is close to 1. This function is used

as an edge indicator: regularization must be weaker and allow variations near the edges and stronger in flat regions.

The use of $3 : \text{end}-2$ serves to avoid edge problems due to convolutions with filters (border zoning). In practice, we avoid applying the formula to the few marginal pixels where the convolutions can be less reliable.

```
d_data_s=ones(size(NormGrad));
d_data_s(3:end-2,3:end-2) = 1./ (1 + beta* NormGrad(3:end-2,3:end-2));
f=f/255;
```

Figure 3.23

6. `GenerateFrameletFilter(frame)` calls the function that constructs the analysis filters `D` and synthesis `R` corresponding to the chosen type of framelet (`frame`= 0:Haar, = 1:piecewise linear, = 3:cubic). These filters will be used by `FraDecMultiLevel` and `FraRecMultiLevel` inside the function `SplitBregFrameSeg`. The three loops for ki, ji and jj construct the `d_data` structure with the same organization used for multiscale framelet coefficients: ki is the decomposition level $(1, \ldots, \text{Level})$; (ji, jj) is the horizontal and vertical filter index $(1, \ldots, nD-1)$ and `d_data`$\{ki\}\{ji, jj\}$ is assigned equal to `d_data_s`, the map of the edge indicator calculated before. The reason for this assignment is that `d_data` must have the form `d_data`$\{ki\}\{ji, jj\}$ because later, in `SplitBregFrameSeg`, we will call a function that builds thresholds/shrink spaces for each layer and each band. Having `d_data`$\{ki\}\{ji, jj\}$ defined, the regularization (or thresholds) can vary in scale and direction.

```
[D,R]=GenerateFrameletFilter(frame);nD=length(D);
for ki=1:Level
    for ji=1:nD-1
        for jj=1:nD-1
            d_data{ki}{ji,jj}=d_data_s;
        end
    end
end
```

Figure 3.24

7. `SplitBregFrameSeg` is the main routine that implements the Split-Bregman scheme to solve the segmentation model with regularization in the framelet domain. The main inputs are the normalized image f, the edge-indicator weight maps for each layer/band `d_data`, the numerical parameters (data term weight, split parameter, tolerance) μ, λ and `tol`, `frame` and `Level` that specify the framelet filters and how many layers to use, the maximum iterations `maxit`, the flag to handle the order of c_1/c_2 `Switch` and the type of shrink `'s'` .

The algorithm uses thresholds to carry out a thresholding weighted on the framelet coefficients, alternating the updates of u (WT reconstruction), d (shrinkage) and b (Bregman). The output u is a continuous map to which it belongs; the final mask is obtained by binarizing u (typically with threshold 0.5) and applying morphological operations for finishing.

```
[u c1 c2]=SplitBregFrameSeg(f,d_data,mu,lambda,tol,frame,Level,maxit,Switch,'s');
figure(1000);imshow(f0,[]);hold on;
save u u;
save f f0;
[cc hh]=contour(u,[0.5 0.5],'r');set(hh,'linewidth',2.0);
```

Figure 3.25

Let's now focus on functions called within the main code.

- The function `GenerateFrameletFilter` constructs the analysis filters `D` and synthesis filters `R` needed for the framelet transform. `D` filters are used to decompose

```
function [D,R]=GenerateFrameletFilter(frame)
if frame==0
    D{1}=[0 1 1]/2;
    D{2}=[0 1 -1]/2;
    D{3}='cc';
    R{1}=[1 1 0]/2;
    R{2}=[-1 1 0]/2;
    R{3}='cc';
elseif frame==1
    D{1}=[1 2 1]/4;
    D{2}=[1 0 -1]/4*sqrt(2);
    D{3}=[-1 2 -1]/4;
    D{4}='ccc';
    R{1}=[1 2 1]/4;
    R{2}=[-1 0 1]/4*sqrt(2);
    R{3}=[-1 2 -1]/4;
    R{4}='ccc';
elseif frame==3
    D{1}=[1 4 6 4 1]/16;
    D{2}=[1 2 0 -2 -1]/8;
    D{3}=[-1 0 2 0 -1]/16*sqrt(6);
    D{4}=[-1 2 0 -2 1]/8;
    D{5}=[1 -4 6 -4 1]/16;
    D{6}='ccccc';
    R{1}=[1 4 6 4 1]/16;
    R{2}=[-1 -2 0 2 1]/8;
    R{3}=[-1 0 2 0 -1]/16*sqrt(6);
    R{4}=[1 -2 0 2 -1]/8;
    R{5}=[1 -4 6 -4 1]/16;
    R{6}='ccccc';
end
end
```

Figure 3.26: *GenerateFrameletFilter*

the image into different components (lowpass and highpass) during the analysis phase; while filters `R` are used to reconstruct the image (or segmentation function u) from the calculated coefficients. These filters depend on the choice of the type of framelet: Haar, piecewise linear, piecewise cubic. In the Haar wavelet, `D{1}` is the lowpass filter (calculate average), `D{2}` is the highpass filter (calculate difference so the edges) and `D{3}` represents a string specifying symmetry conditions. `R{1}`, `R{2}` are the corresponding reconstruction filters. In the Piecewise Linear wavelet filters are derived from the linear B-spline (a linear piecewise function). `D{1}` is always a lowpass filter, but we have multiple highpass filters that capture different details: `D{2}` is used for the detection of horizontal/vertical variations, `D{3}` for secondary finer variations. `D{4}` indicates that all filters are symmetrical

and the `R` filters correspond to those of `D` but for reconstruction. Lastly, in the Piecewise Cubic framelet filters are derived from the cubic B-spline (more regular and smoother). There are multiple highpass filters (`D{2}`,...,`D{5}`), each capturing different details (first, second,... derivatives) with sensitivity to edges, curvatures, and complex transitions. `D{6}` specifies that the 5 filters are symmetrical and `R` are the corresponding reconstruction filters.

This function is called twice in the main script: for edges (frame_edge), since it builds filters used in `FraDecMultiLevel` to calculate the edge map, `NormGrad`; for segmentation (frame), since it builds filters that are used within `SplitBregFrameSeg` to decompose and regularize the u segmentation function.

- Functions `FraDecMultiLevel` and `FraDec` implement the multiscale 2-D framelet decomposition (analysis) used by your segmentation algorithm. `FraDec` calculates the 2-D decomposition of a single layer by applying 1-D `D{`i`}` filters separably on rows and columns using tensor product; `FraDecMultilevel` applies `FraDec` recursively to obtain a decomposition on L levels: at each step it takes the low-low component (approximation) and uses it as input to the next level.

```
function Dec=FraDecMultiLevel(A,D,L)

nD=length(D);
kDec=A;
for k=1:L
    Dec{k}=FraDec(kDec,D,k);
    kDec=Dec{k}{1,1};
end


function Dec=FraDec(A,D,L)

nD=length(D);
SorAS=D{nD};
for i=1:nD-1
    M1=D{i};
    tempi=ConvSymAsym(A,M1,SorAS(i),L);
    for j=1:nD-1
        M2=D{j};
        tempj=ConvSymAsym(tempi',M2,SorAS(j),L);
        Dec{i,j}=tempj';

    end
end
```

Figure 3.27: *FraDecMultilevel, FraDec*

Regarding the first function, we define `kDec` as the variable that contains the signal to decompose for the current layer, initially it is the original image. Then a cycle is done on all levels 1 to L: `Dec{`k`}` =`FraDec(kDec,D,k)` computes the decomposition of a single level of the `kDec` image. The output is a 2-D cell (size $(nD-1)\times(nD-1)$) containing the coefficient matrices for all horizontal/vertical filter combinations. In particular `Dec{`k`}{`$1,1$`}` is the low-low coefficient (approximation) of that level.

`kDec` = `Dec`$\{k\}\{1,1\}$ takes the low-low component and uses it as the input image for the next layer. Since no downsampling occurs, `Dec`$\{k\}\{1,1\}$ has the same spatial dimension as te original image and so the process is multilevel redundant (each level has coefficient matrices of the same dimensions).

In the second function, `FraDec`, a loop is made on the horizontal filters where `M1` = `D`$\{i\}$, 1-d filter for horizontal direction (rows),is used in `tempi`=`ConvSymAsym` in the convolution of the original image line by line with `M1`. The auxiliary function ConvSymAsym performs convolution taking into account the condition at the edges and probably using the correct orientation. The use of `ConvSymAsym`, rather than `conv2`, to perform convolution allows edges (symmetric or antisymmetric padding) and filter alignment (filter center) to be managed in a controlled manner. Consequently a loop is performed on the vertical filters where `M2` = `D`$\{j\}$, 1-d filter for vertical direction (columns), is used in `tempj`=`ConvSymAsym` with `tempi'` as argument. This means that the convolution with `M2` is carried out on the rows of the transposed matrix, equivalent to the convolution on the columns of the original. Then, it re-transposes to put the matrix back in the original direction: the result corresponds to the separable 2-D convolution of the original image with the 2-D filter = `M1` $\otimes$ `M2` (tensor product).

- Function `SplitBregFrame` implements a Split-Bregman version of an Active Contours Without Edges (Chan-Vese) type energy in which spatial regularization is carried out in the framelet domain with penalties l_1. Optimization is accomplished by alternating closed (or semi-explicit) updates: updating u (relaxed membership function, values in $[0,1]$), updating d (auxiliary variables = framelet coefficients subject to shrink/threshold) and updating the term Bregman b. Regularization is edge-aware because shrink thresholds depend on `d_data`, that is a map inversely proportional to edge strength.

```
function [u c1 c2]=SplitBregFrame(f,d_data,mu,lambda,tol,frame,Level,maxit,Switch,shrinktype)

[D,R]=GenerateFrameletFilter(frame);
W  = @(x) FraDecMultiLevel(x,D,Level);
WT = @(x) FraRecMultiLevel(x,R,Level);

[m,n]=size(f);
muLevel=getwThresh(d_data,lambda,Level,D);
normg=norm(f,'fro');
u=zeros(m,n);
b=W(u);
d=b;
c1=0;
c2=max(f(:));
r=(c1-f).^2-(c2-f).^2;

for nstep=1:maxit
    for innerstep=1:1
        c=CoeffOper('-',d,b);
        u=WT(C)-mu*r/lambda;
        u(u>=1)=1;u(u<0)=0;
        C=W(u);
        Cpb=CoeffOper('+',C,b);
        d=CoeffOper(shrinktype,Cpb,muLevel);
    end
    if mod(nstep,1)==0 && nstep<=100
        alpha =0.5;
        if isempty(f(u>=alpha))~=1
            c1=mean(mean(f(u>=alpha)));
        end
        if isempty(f(u<alpha))~=1
            c2=mean(mean(f(u<alpha)));
        end
        if Switch==1
            if c1<c2
                temp=c1;c1=c2;c2=temp;
            end
        end
        r=(c1-f).^2-(c2-f).^2;
    end
    deltab=CoeffOper('-',C,d);
    error=sqrt(cellnorm(deltab))/normg;
    b=CoeffOper('+',b,deltab);
    if mod(nstep,10)==0
        disp(['error on step ' num2str(nstep) ' is ' num2str(error) '.']);
    end
    if error<tol
        break;
    end
end
```

Figure 3.28: *SplitBregFrame*

First of all, framelet operators (W, WT) are constructed using

`GenerateFrameletFilter`, `FraDecMultiLevel` and `FraRecMultiLevel`. This function performs the inverse framelet transform (reconstruction) of an image (or signal) from its multi-level framelet coefficients. Starting from the deepest decomposition level, it iteratively applies the reconstruction filters to combine the approximation and detail coefficients until the original image is recovered at level 1. `FraRec` (single-level) reconstructs an image from its `C`$\{k\}$ bands using `R` synthesis filters (generally via separable convolutions and component summation). Apply `R`$\{i\}$ synthesis filters to rows and columns (separable) and add the contributions to get the reconstruction for that layer. It has the reverse behavior of `FraDec`.

```
function Rec=FraRecMultiLevel(C,R,L)

nR=length(R);
for k=L:-1:2
    C{k-1}{1,1}=FraRec(C{k},R,k);
end
Rec=FraRec(C{1},R,1);


function Rec=FraRec(C,R,L)

nR=length(R);
SorAS=R{nR};
ImSize=size(C{1,1});
Rec=zeros(ImSize);
for i=1:nR-1
    temp=zeros(ImSize);
    for j=1:nR-1
        M2=R{j};
        temp=temp+(ConvSymAsym((C{i,j})',M2,SorAS(j),L))';
    end
    M1=R{i};
    Rec=Rec+ConvSymAsym(temp,M1,SorAS(i),L);
end
```

Figure 3.29: *FraRecMultileve,FraRec*

Then the weighted thresholds for the shrink are calculated utilizing `getwThresh` where for each ki-level and each band (ji, jj) builds `muLevel`$\{ki\}\{ji, jj\}$: if $ji == 1$ and $jj == 1$ (low-low band) then there is no thresholding and it is equal to 0; otherwise `muLevel`$\{ki\}\{ji, jj\}$=`d_data` $\{ki\}\{ji, jj\}/\lambda$. This means that shrink thresholds are proportional to the `d_data` map (edge indicator) and scaled from $1/\lambda$. Where `d_data` large (flat zone) the threshold is large and so we have plus suppression of details; where `d_data` is small (edge) the threshold is small and so the details are retained.

u is initialized with 0, b (Bregman terml) is initialized as $W(u)$ (initial coefficients) and d is initially equal to b, so the constraint $d = Wu$ starts satisfied; c_1 is initialized with 0 and c_2 as the maximum of the values of the image and $r(x) = (c_1 - f(x))^2 - (c_2 - f(x))^2$ is the data field.

```
function muLevel=getwThresh(d_data,lambda,Level,D)
nD=length(D);
for ki=1:Level
    for ji=1:nD-1
        for jj=1:nD-1
            if ji==1 && jj==1
                muLevel{ki}{ji,jj}=0;
            else
                muLevel{ki}{ji,jj}=d_data{ki}{ji,jj}/lambda;
            end
        end
    end
end
```

Figure 3.30: *getwThresh*

Then the main loop starts by updating u. The subproblem for u after split is

$$\min_{u\in[0,1]} \frac{\lambda}{2}||Wu-(d-b)||_2^2+\mu\langle r,u\rangle$$

First order condition (derivative $= 0$) gives:

$$\lambda W^T(Wu-(d-b))+\mu r=0.$$

If W is a tight frame, or we assume $W^TW = I$,:

$$\lambda(u-W^T(d-b))+\mu r=0\Rightarrow u=W^T(d-b)-\frac{\mu}{\lambda}r$$

. Next d is updated by solving the subproblem

$$\min_d \frac{\lambda}{2}||d-(Wu+b)||_2^2+\sum_{k,i,j} w_{kij}(x)|d_{kij}(x)|,$$

where w_{kij} are the weights derived from `d_data` (hence `muLevel`). The solution is weighted soft-thresholding:

$$d_{kij}(x)=\text{shrink}((Wu+b)_{kij}(x),w_{kij}(x)/\lambda).$$

Function `CoeffOper` applies the specified shrink type. The internal/external averages are recalculated using the threshold of u at $alpha = 0.5$. This is analogous to the Chan-Vese algorithm which alternates between updating the curve and updating the inner/outer means. r is updated with new averages. Finally Bregman b is updated.

The algorithm was applied to three transesophageal ultrasound images, each corresponding to a patient with atrial fibrillation. For each case, a region of interest (ROI) was selected, and after the segmentation was completed, cropping and additional processing were performed on the mask to better isolate the left atrial appendage. For this second processing, a different ROI was chosen compared to the one used previously, as

the algorithm requires higher contrast and sharper edges to achieve effective segmentation. The parameters were set as follows: μ and $\lambda = \mu \times 0.05$, in order to produce sufficiently accurate contours, avoiding both excessive sensitivity to noise and overly smoothed segmentations. Additionally, the following settings were used: `frame` = 3, `Level`= 2, `frame_edge` = 3, and `Level_edge` = 2.

After running the algorithm, as in the previous cases, the largest connected component was extracted and cropped to isolate the left atrial appendage.

1. **Patient** 1: $\mu = 5,\ \lambda = \mu * 0.05$.

Figure 3.31: *Final segmentation*

2. **Patient** 2: $\mu = 100,\ \lambda = \mu * 0.05$.

Figure 3.32: *Final segmentation*

3. **Patient** 3: $\mu = 100,\ \lambda = \mu * 0.05$.

Figure 3.33: *Final segmentation*

3.4 Comparative Evaluation of Different Methods

In this section, a comparative analysis of the previously described methods is presented. The goal is to highlight the main strengths and limitations of each approach, focusing in particular on their practical applicability. Moreover, computational costs and processing times are evaluated to provide a clear overview of the efficiency of the different techniques.

3.4.1 Analysis of the computational costs

Let us conduct an analysis of the computational costs of the three codes.

Convex non-convex image segmentation

The total computational cost of convex non convex algorithm is given by the sum of the costs of data loading, stage-one (`OUR_Seg_SPL`), stage-two (`ThdKmeans`), and the visualization of results (`SegResultShow`). We first introduce the notation that will be used:

- N: number of pixels, given by $xLen \times yLen$;
- I: number of ADMM iterations executed by `OUR_Seg_SPL`;
- I_k: number of iterations of the internal K-means algorithm;
- k: number of phases (clusters), usually small $(2-7)$.

We also note that elementwise operations on matrices with N elements cost $O(N)$, while the 2D Fast Fourier Transform (FFT2) on a matrix with N elements costs $O(N \cdot \log N)$.

We now proceed with a detailed cost analysis. First, the image loading and the initialization of the values T and k have a cost of $O(N)$. The dominant cost comes from stage-one, which corresponds to the execution of `OUR_Seg_SPL`. For each ADMM iteration ($i = 1 : \texttt{its\_max}$), the following contributions must be considered: $O(N)$ for building `rhs` using `Dx, Dy, Dxt, Dyt`, which are discrete difference operators implemented through elementwise operations and shifts; $O(N \cdot \log N)$ for solving the linear system for u using the Fast Fourier Transform; $O(N)$ for the computation of the error; $O(N)$ for calculating `rx, ry` and the norm `r_norm`; $O(N)$ for calculating `tx, ty`; and $O(N)$ for updating the

multipliers `rhox, rhoy`. Summing these terms yields

$$T_{iter} = O(N \cdot \log N) + O(N) = O(N \cdot \log N),$$

and therefore, considering I iterations, we obtain

$$T_{stage1} = O(I \cdot N \cdot \log N).$$

As for stage-two, which consists of the execution of `ThdKmeans`, the costs are as follows. The K-means procedure has complexity $O(I_k \cdot k \cdot N)$, which can be approximated as $O(I_k \cdot N)$ since k is small. In addition, there is a cost of $O(k \cdot N)$ for computing the means and $O(k)$ for computing the thresholds. Therefore, the total complexity of the second stage is

$$T_{stage2} = O(I_k \cdot N),$$

which is significantly lower than stage-one.

Finally, we analyze the cost of `SegResultShow`. Constructing the masks `temp` costs $O(k \cdot N)$, the `contour` function has cost $O(N)$, and the construction of the final segmented image `seg` costs $O(k \cdot N)$. Thus, we have

$$T_{SegResultShow} = O(k \cdot N).$$

In conclusion, the total cost of the CNC algorithm is

$$T_{tot} = T_{stage1} + T_{stage2} + T_{SegResultShow} = O(I \cdot N \cdot \log N).$$

Unsupervised Multiphase Segmentation

We now introduce the notation used:

- N_x and N_y: the image dimensions, with $N = N_x \cdot N_y$;
- I: the number of outer iterations (loop `iter`$= 1:5$), in our case $I = 5$;
- K: the average number of active phases (variable k), and K_{max} the maximum allowed value (set to 20 in the code).

The dominant contribution comes from the triple loop `for` $x, y, j = 1 : k + 1$, which leads to an asymptotic complexity of $O(I \cdot N \cdot K)$. Since in the code K is bounded by 20, in practice this reduces to $O(I \cdot N)$.

For each pair (x, y) and each j, up to four comparisons with the neighboring pixels are performed, together with about eight arithmetic operations for `tj/tl`, some calculations for `dF`, the evaluation of `dE_temp` (a few multiplications and subtractions), and finally one comparison `dE_temp` $<$ `dE`. Thus, each evaluation in j requires only a small constant number of arithmetic operations (on the order of a few dozen FLOPs). Therefore, the computational work per pixel is $O(K)$ FLOPs.

In conclusion, the overall computational cost is

$$T_{tot} = O(I \cdot K \cdot N).$$

Frame based segmentation for medical images

We begin by introducing the notation:

- N: number of pixels in the image, given by $xLen \times yLen$;
- nD: number of filters in the decomposition filter D;
- nR: number of filters in the reconstruction filter R;
- L: number of levels in the framelet decomposition;
- L_{edge}: number of decomposition levels used exclusively in the preprocessing stage to compute the edge indicator `d_data`.

The image loading and normalization steps have a cost of $O(N)$. We now analyze the functions used within the code. The function `GenerateFrameletFilter` has complexity $O(1)$, as it only returns fixed coefficients and does not depend on the image size.

The functions `FraDecMultiLevel` and `FraDec` have a complexity of $O(L \cdot N \cdot (nD - 1)^2)$. Indeed, each level L applies 2D convolutions via `ConvSymAsym` on progressively smaller images. A 2D convolution has cost $O(N \cdot k^2)$ (with k the filter size). Each decomposition level performs approximately $(nD-1)^2$ convolutions, where nD typically ranges from 3 to 6 depending on the framelet.

An analogous reasoning applies to `FraRecMultiLevel` and `FraRec`, which yield a complexity of $O(L \cdot N \cdot (nR - 1)^2)$. The function `CoeffOper` has complexity $O(L \cdot N)$.

The segmentation algorithm `SplitBregFrame`, which represents the core of the method, has complexity $O(L \cdot N \cdot (nD - 1)^2)$ per iteration and thus $O(I \cdot L \cdot N \cdot (nD - 1)^2)$ in

total. Finally, the computation of the edge indicator functions `NormGrad` and `d_data_s` costs $O(L_{edge} \cdot N \cdot (nD-1)^2)$.

Therefore, the overall complexity of the algorithm is dominated by the segmentation iterations:

$$T_{tot} = O(I \cdot L \cdot N \cdot (nD-1)^2).$$

In terms of computational complexity, the three segmentation algorithms under consideration exhibit distinct behaviors. The unsupervised multi-segmentation model has a complexity of $O(I \cdot K \cdot N)$, where N is the number of pixels, I is the number of iterations, and K is the number of clusters or regions. Since K is typically small, this method scales linearly with image size, making it the most computationally efficient among the three. The convex/non-convex segmentation algorithm has a complexity of $O(I \cdot N \cdot \log N)$. Although the logarithmic factor introduces a slight overhead compared to purely linear scaling, the method remains relatively efficient for moderate image sizes. In contrast, the frame-based segmentation method has a complexity of $O(I \cdot L \cdot N \cdot (n_D-1)^2)$, where I is the maximum number of iterations, L is the framelet decomposition level, and n_D is the number of framelet filters. Even for moderate values of these parameters, the multiplicative effect makes this approach computationally more expensive. In practice, therefore, the unsupervised multi-segmentation model is the least demanding in terms of computation, infact we measured 0.39 seconds, the convex/non-convex segmentation algorithm is intermediate, with 4.88 seconds, and the frame-based segmentation is the most computationally intensive, with 8.86 seconds.

3.4.2 Limitations of the three Segmentation Algorithms

In order to assess the strengths and weaknesses of the three proposed segmentation algorithms, it is essential to identify their limitations. Understanding these constraints helps both in selecting the appropriate method for a given dataset and in guiding future improvements. Below, I discuss for each algorithm its main limitations.

- Convex Non-Convex image segmentation

  The convex non-convex image segmentation method provides many benefits: strong theoretical guarantees, well-posed energy minimization, and often sharp boundaries. However, it suffers from a significant sensitivity to parameter choice. Parameters such as weighting coefficients (e.g. balancing data fidelity vs regularization), thresholds, or stopping tolerances dramatically affect outcome: small changes can

lead to either over-segmentation (too many small regions) or under-segmentation (missing structures). Furthermore, in certain imaging modalities (such as ultrasound), image noise or artifacts can amplify instabilities, because the model assumes somewhat clean or strongly defined contrast at edges. Finally, convergence speed may degrade if parameter values force the optimization to balance conflicting terms (data vs regularity), making practical tuning both time-consuming and dataset-specific.

- Frame Based Segmentation

  Frame based segmentation, which leverages multiscale framelet decompositions, excels in capturing image structure at multiple scales and enforcing regularization in transform domains. Yet, it also has limitations. Firstly, it requires relatively high contrast in the image, especially at the boundaries of the structures to be segmented; if edges are weak or blurred, the framelet coefficients used to detect boundaries may not be reliable. Secondly, it can be computationally heavy: the multilevel decomposition and reconstruction over several scales for every iteration make it less suitable for real-time processing or for devices with constrained computing power. Thirdly, it typically has more internal parameters (number of filters, decomposition levels, thresholds), which also require careful tuning. In addition, noise in the image tends to produce spurious high-frequency components, which may lead to false edges unless appropriately suppressed, potentially reducing segmentation robustness.

- Unsupervised Multiphase Segmentation

  Unsupervised multiphase segmentation algorithms have the advantage of not requiring manual labels and being more flexible in adapting to different images. However, they are not without their own limitations:

  (i) **Dependence on the number of phases (clusters)**: While being unsupervised means fewer ground truth labels, the algorithm still typically requires a parameter specifying how many phases/clusters to segment. If the number of clusters is chosen too high, many small spurious segments may appear (over-segmentation); if too low, distinct structures may be merged (under-segmentation).

  (ii) **Ambiguous intensity overlap and noise sensitivity**: In many images, different regions (phases) have overlapping intensity distributions, or intensity transitions are gradual (rather than sharp). In such cases, clustering or

threshold-based assignments struggle, since pixels in overlap zones may be misclassified, and the algorithm can be sensitive to noise or outliers.

(iii) **Lack of strong structural priors**: Because unsupervised methods often rely on low-level features (intensity, texture, simple statistics), they may fail to capture more complex anatomical or contextual structure, such as shape constraints, spatial continuity, or known tissue boundaries, resulting in anatomically implausible segmentation.

(iv) **Evaluation difficulties**: Without labels, validating the correctness of segmentation is harder. Metrics may be unreliable or based on proxies; and applicability in clinical settings may be limited unless manual review is available.

To summarize, Convex Non-Convex Segmentation method is limited mainly by parameter sensitivity and requirements for strong contrast; it can be less robust in noisy images, especially ultrasound. Frame-Based Segmentation method needs high boundary contrast, is computationally heavier, and has more internal parameters; it may perform poorly if edges are ill-defined or image noise is significant. Unsupervised Multiphase Segmentation method, while more flexible, has limitations tied to the choice of number of clusters, overlapping intensities, noise, lack of strong priors, and difficulties in objective evaluation.

Chapter 4

Comparison Between 3D Segmentations: ITK-SNAP and Variational Methods

In this chapter, a direct comparison is presented between the 3D segmentation obtained using the variational methods described in the previous chapters and the 3D segmentation obtained with the ITK-SNAP software, which performs automatic segmentation based on intensity and region-growing techniques. The goal of this comparison is to evaluate whether the algorithms implemented in MATLAB can serve as a valid alternative to commercial or open-source software for left atrial appendage segmentation from transesophageal echocardiographic images.

4.1 3D Reconstruction from Variational Segmentation

For the MATLAB-based segmentation, a single variational method was selected to generate the three dimensional model. Specifically, the *Frame based segmentation* method was chosen due to its flexibility and its ability to handle noisy ultrasound images. The starting point was a TEE image sequence that had been preprocessed with suitable denoising filters to reduce speckle noise and to enhance the contrast between dark and bright regions. From each filtered frame, a region of interest (ROI) was extracted, focusing on the area containing the left atrial appendage.

The selected segmentation algorithm was applied slice by slice to the preprocessed ROI. The resulting binary masks were then stacked to reconstruct the 3D volume of

the LAA. However, this raw 3D segmentation was not directly used as the final model. Post-processing was required to improve its quality. To this end, external software tools such as MeshLab and ParaView were employed to refine the geometry. In particular, these tools were used to:

- Remove extraneous regions not belonging to the appendage;
- Fill potential holes in the segmentation;
- Apply smoothing filters to obtain a more regular and anatomically plausible surface.

The resulting 3D model of the LAA, obtained entirely through MATLAB-based segmentation followed by mesh post-processing, is shown in the figure below.

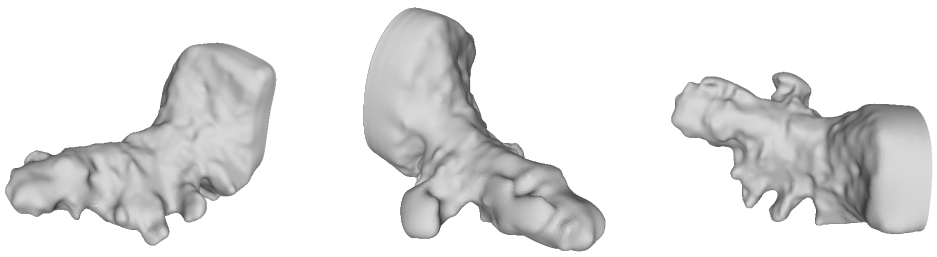


Figure 4.1: *3D segmentation with TV method*

4.2 3D Segmentation with ITK-SNAP

For comparison, the same TEE dataset was segmented using ITK-SNAP, an open-source software widely used in medical image analysis. ITK-SNAP provides semi-automatic and automatic segmentation tools, including active contour and region-growing methods that rely on intensity thresholds. In this work, the automatic segmentation mode was applied to the filtered TEE volumes to extract the LAA region.

As with the MATLAB-based approach, the ITK-SNAP segmentation was subsequently refined using MeshLab and ParaView to remove noise and apply light smoothing, ensuring a fair comparison between the two methods.

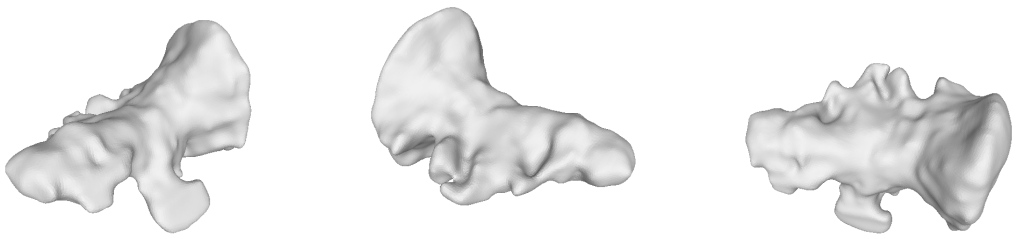


Figure 4.2: *3D segmentation with ITK-SNAP*

With a carefully executed post-processing workflow and the use of advanced visualization tools such as MeshLab and ParaView, highly accurate and realistic 3D reconstructions can be achieved. In the hands of skilled operators, this procedure allows for the generation of smooth and well-defined anatomical boundaries that closely replicate the patient's actual anatomy, as we can see in the following figures.

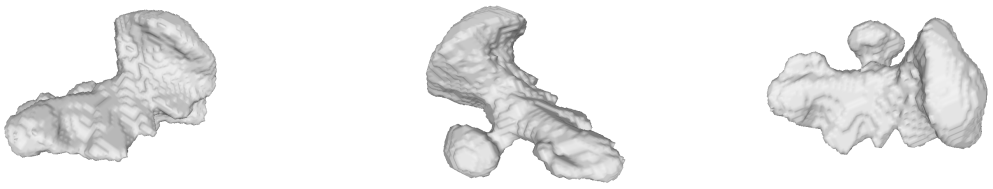


Figure 4.3: *3D segmentation with post production*

4.3 Qualitative Comparison and Discussion

Before comparing the 3D reconstructions, it is useful to first evaluate the segmentation results on individual 2D slices. This step allows for a more direct and detailed comparison between the MATLAB-based segmentation and the one obtained with ITK-SNAP. By examining the two contours slice by slice, it is possible to assess how each method responds to image noise and anatomical boundaries, and to identify local differences that might not be immediately visible in the 3D models. This analysis provides valuable insight into the behavior and sensitivity of the two algorithms at the slice level, serving as a foundation for the subsequent 3D comparison.

Figure 4.4: *2D segmentation comparison*

The visual comparison between the two 3D segmentations shows that the results obtained with ITK-SNAP and the variational MATLAB methods are overall very similar in terms of accuracy. Both approaches successfully capture the global structure of the left atrial appendage, and the reconstructed volumes are largely consistent with each other. The main differences between the two methods are not related to their accuracy but rather to the level of detail and the way each algorithm interacts with the intrinsic noise of the echocardiographic images. In other words, each segmentation method responds differently to image features depending on its sensitivity, leading to slightly different surface appearances.

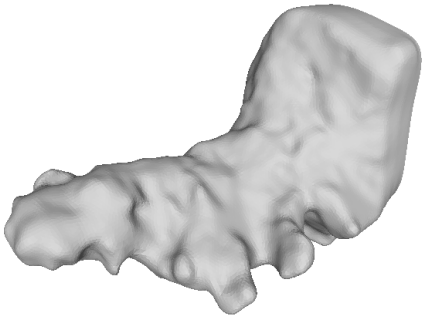
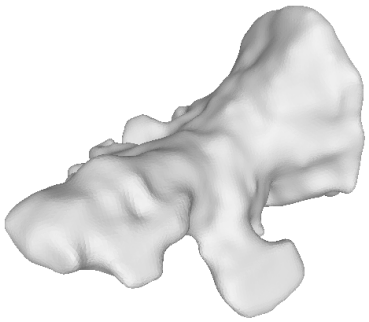
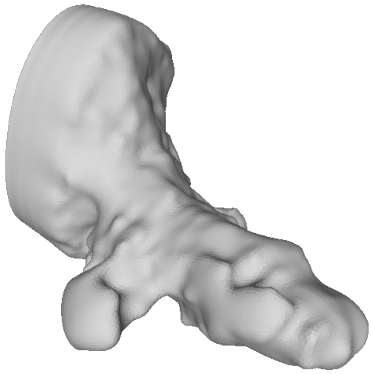
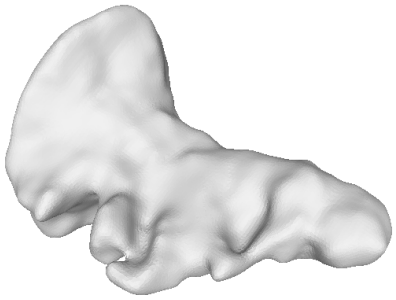
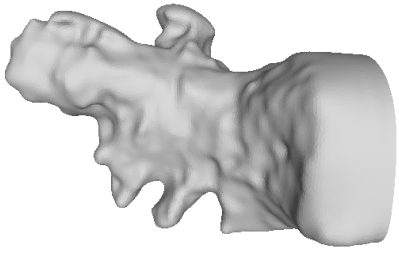
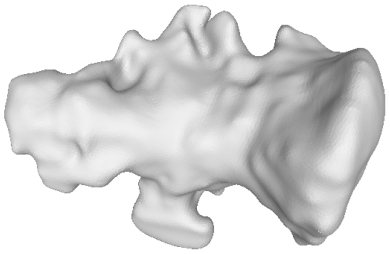
The ITK-SNAP segmentation tends to produce smoother and more anatomically detailed borders, which is partly a result of the algorithm's sensitivity and partly due to the ability to manually refine the segmentation slice by slice. This manual correction allows experienced users to remove artifacts, adjust contours, and achieve a higher level of precision, especially along complex anatomical structures such as the borders of the appendage. This flexibility is one of the advantages of ITK-SNAP: after the automatic segmentation, the user can directly modify the segmentation in 2D, leading to more controlled and refined 3D results.

On the other hand, the MATLAB-based segmentation relies on variational models

and is fully automated. Once the method, the echocardiographic dataset and the region of interest (ROI) are selected, the entire segmentation is performed automatically, without the possibility to manually intervene on individual slices. As a result, the final surface strictly reflects the behavior of the algorithm with respect to the image noise and resolution, without any manual correction. Despite this limitation, the results obtained with the variational MATLAB methods are remarkably good, showing a clear and accurate representation of the appendage even without manual editing. This highlights the robustness and potential of variational approaches for automated medical image segmentation.

It is important to note that both methods share a common limitation related to the quality of the TEE data. The presence of speckle noise, artifacts, and the relatively low spatial resolution make it difficult to achieve a perfectly smooth and accurate segmentation in a fully automatic way. These issues affect both ITK-SNAP and MATLAB methods, although ITK-SNAP can mitigate them more effectively through manual refinement.

In summary, both ITK-SNAP and the variational MATLAB approach produce accurate and anatomically meaningful segmentations of the left atrial appendage. The main differences lie in the level of detail and the workflow: ITK-SNAP allows for slice-by-slice manual refinement, resulting in potentially smoother and more detailed surfaces, while MATLAB provides a fast and fully automatic segmentation pipeline that still achieves a very good level of accuracy. These characteristics suggest that ITK-SNAP may be preferable when manual editing is feasible and high anatomical detail is required, whereas the MATLAB-based approach is well suited for automated processing, research pipelines, or situations where manual intervention is not possible.

Bibliography

[1] A. Masci, M. Alessandrini, D. Forti, F. Menghini, L. Dede, C. Tomasi, A. Quarteroni, C. Corsi: *A Proof of Concept for Computational Fluid Dynamic Analysis of the Left Atrium in Atrial Fibrillation on a Patient-Specific Basis*, 2020.

[2] M. Falanga, C. Cortesi, A. Chiaravalloti, A. Dal Monte, C. Tomasi, C. Corsi: *A digital twin approach for stroke risk assessment in Atrial Fibrillation Patients*, 2024.

[3] S.W. Smith, H.G. Pavy, O.T. von Ramm: *High-speed Ultrasound Volumetric Imaging System, Part I: Transducer Design and Beam Steering*, "IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control" (Volume: 38, Number: 2, 1991), 1999, pp. 100-108.

[4] S.W. Smith, H.G. Pavy, O.T. von Ramm: *High-speed Ultrasound Volumetric Imaging System, Part II: Parallel Processing and Image Display* , "IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control", (Volume 38, Number 2, 1991), 1999, pp. 108-115.

[5] J. S. Mattoon, T. G. Nyland: *Principi fondamentali di ecografia diagnostica*, 2016, pp. 1-26.

[6] R. Beigel, N. C. Wunderlich, S. Yen Ho, R. Arsanjani, R. J. Siegel: *The Left Atrial Appendage: Anatomy, Function, and Noninvasive Evaluation*, " JACC: Cardiovascular Imaging" (Volume 7, Number 12, 2014), 2014.

[7] S. Klein, M. Staring: *Elastix, The manual.*, 2024.

[8] M. Kass, A. Witkin, D. Terzopoulos: *Snakes: Active contour models*, " INT Journal of Computer Vision" (Volume 1, 1988), 1988, pp. 321-331.

[9] T. F. Chan, L. A. Vese: *Active Contours Without Edges*, " IEEE Transactions on Image Processing" (Volume 10, Number 2, 2001), 2001, pp. 266-277.

[10] R. Malladi, J. A. Sethian, B. C. Vemuri: *A fast level set based algorithm for topology-independent shape modeling*, " Journal of Mathematical Imaging and Vision" (Volume 6, 1996), 1996, pp. 269-289.

[11] R. Chan, A. Lanza, S. Morigi, F. Sgallari: *Convex Non-Convex Image Segmentation*, " Numerische Mathematik" (Volume 138, 2018), 2017, pp. 635-680.

[12] B.Sandberg, S. Ha Kang, T. F. Chan: *Unsupervised Multiphase Segmentation: a phase balancing model*, "IEEE Transactions on Image Processing", (Volume 19, Number 1, 2010), 2009, pp. 119-130.

[13] B. Dong, A. Chien, Z. Shen, T. F. Chan: *Frame based segmentation for medical images*, "Communications in Mathematical Sciences", (Volume 9, Number 2, 2010), 2010, pp. 1724-1739.

[14] R. Beigel, N. C. Wunderlich, S. Yen Ho, R. Arsanjani, R. J. Siegel: *The Left Atrial Appendage: Anatomy, Function, and Noninvasive Evaluation*, "JACC: Cardiovascular Imaging", (Volume 7, Number 12, 2014), 2014, pp. 1251 - 1265.

[15] A. Buades, B. Coll, J. -M. Morel: *A non-local algorithm for image denoising*, "JACC: IEEE Computer Society Conference on Computer Vision and Pattern Recognition", (Volume 2, 2005), 2005, pp. 60-65.

[16] J. Immerkær: *Fast Noise Variance Estimation*, "Computer Vision and Image Understanding", (Volume 64, Number 2, 1996), 1996, pp. 300-302.

[17] P. Perona, J. Malik: *Scale-Space and Edge Detection Using Anisotropic Diffusion*, "IEEE Transactions on Pattern Analysis and Machine Intelligence", (Volume 12, Number 7, 1990), 1990, pp. 629-639.

[18] C. A. Bouman: *Foundations of Computational Imaging: A Model-Based Approach*, 2022.

[19] S. Marchesini, A. Trivedi, P. Enfedaque, T. Perciano, D. Parkinson: *Sparse Matrix-Based HPC Tomography*, "Computational Science-ICCS", 2020.

[20] R. C. Gonzalez, R. E. Woods: *Digital Image Processing*, Pearson, 4th edition, 2018.

[21] T. F. Chan, J. Shen: *Image Processing and Analysis: Variational, PDE, Wavelet, and Stochastic Methods*, "BioMed Eng OnLine", (Volume 5, Number 38, 2006), 2006.

Web References

- Fibrillazione atriale, Policlinico Gemelli: https://privato.policlinicogemelli.it/approfondimenti/fibrillazione-atriale/.
- Fibrillazione atriale, Humanitas: https://www.humanitas.it/malattie/fibrillazione-atriale/
- Left atrial appendage, Cleveland clinic: https://my.clevelandclinic.org/health/body/left-atrial-appendage.
- SNR: https://wraycastle.com/it/blogs/knowledge-base/what-does-snr-stand-for?srsltid=AfmBOoqCvp1fqWonDC6yUTzeyINVEbv_h_BJmCSgKnSN9NHrAGh8_BoW.
- Tomografia computerizzata, MSD manuale: https://www.msdmanuals.com/it/casa/argomenti-speciali/esami-comuni-di-diagnostica-per-immagini/tomografia-computerizzata-tc.
- Quantum noise, Radiopaedia: https://radiopaedia.org/articles/quantum-noise?lang=us&utm_source=chatgpt.com