Reimagining Image Segmentation using Active Contour: From Chan Vese Algorithm into a Proposal Novel Functional Loss Framework

Gianluca Guzzetta
Politecnico di Torino
s308449@studenti.polito.it

Abstract

In this paper, we present a comprehensive study and analysis of the Chan Vese algorithm for image segmentation, using a discretized scheme obtained from the empirical study of Chan-Vese model's functional energy and its partial differential equation based on its level set function. The proof to such result is shown, and an implementate using MATLAB. Taking advantange of nowadays methodologies in computer vision, we also propose a functional segmentation loss based on active contour based on `pytorch.nn.ModuleLoss` *and level set based on Chan-Vese algorithm proposed, and comparing such results to common computer vision segmentation datasets and we compare classical loss performances with the proposed one. All code and and used material are available at cv_functional_loss repository.*

1. Problem overview

Image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels). Hence, image segmentation consists into portioning a digital image into valuable and reliable pieces of information, easier to analyze. The whole picture does not contain only relevant data, in most computer vision scenarios, we are interested in a portion of area of pixel of the image, which could be represented as people or some certain objects we are interested in, as our cat who crossed the perimeter in the garden. Nowadays the aims are immeasurable.

Although it is a common problem, image segmentation has a wide set of resolution methods which aim at collecting set of pixels or a set of relevant pixels called objects.

These methods are in general (i.e. not limited to):

- **Thresholding**: This method involves partitioning an image based on pixel intensity. Pixels with intensity above a certain threshold are classified differently from those below, allowing for straightforward segmentation of images into foreground and background.
- **Clustering**: This technique partitions image pixels into clusters based on attributes such as color and intensity. K-means clustering is a notable example, categorizing pixels into clusters with similar values.
- **Contour-based methods**: These methods aim to find contours within an image, identifying boundaries between different image regions. The active contour model, or snakes, dynamically adjusts contours to outline object edges.
- **Region-based methods**: These methods segment an image based on the homogeneity of pixel regions. Techniques like Region Growing start from seed points and aggregate neighboring pixels with similar properties to form a segment.

In this proposed work, we used an active contour modeling using the Chan-Vese method, an active contour modeling method renowned for its efficacy in capturing complex object boundaries and independent on image intensities.

2. Proposed approach

2.1. Preprocessing

In our experiment, an image goes through a well-defined process of image preprocessing, moreover the experiments are run onto different scenarios to prove its effectiveness, hence the starting image is segmented using the proposed method and compared to the noisy images processed through the next proposed approeaches. Indeed noise is introduced in order to add a non linear factor to the problem, as in real world applications. Each image has been treated, introducing noise and removing those factors later, using the following techniques:

2.2. Data Augmentation: noise introduction to the dataset

Data augmentation plays a critical role in the development of machine learning models, particularly in the field

of computer vision. By introducing variations in the training data, models can learn to generalize better and become more robust to unseen data. In this work, we focus on the incorporation of Gaussian noise and Salt-and-Pepper (S&P) noise as augmentation strategies to simulate real-world imperfections in images.

Gaussian Noise Addition: Gaussian noise, characterized by its normal distribution, is added to images to mimic the effect of random variations in pixel values. This is mathematically represented as:

$$I_{\text{noisy}} = I + \mathcal{N}(\mu, \sigma^2), \tag{1}$$

where I is the original image, $\mathcal{N}(\mu, \sigma^2)$ denotes the Gaussian distribution with mean μ and variance σ^2, and I_{noisy} is the resultant image. This technique aids in training models that are less sensitive to slight pixel intensity variations.

Salt-and-Pepper Noise Injection: S&P noise, also known as impulse noise, introduces sharp, sudden disturbances in the image, represented by randomly scattering white and black pixels. This type of noise challenges the model's ability to maintain performance despite the presence of significant pixel-level anomalies.

All these techniques aim at improving the model's robustness, meaning the efficiency of the segmentation to low quality datasets.

2.3. Feature Engineering: Advanced Noise Removal Techniques

Effective feature engineering is paramount in preparing data to increase its data quality for our model, especially when dealing with noisy datasets. To counteract the noise introduced during data augmentation, we employ sophisticated noise removal techniques as part of our preprocessing pipeline.

- **Gaussian Noise Elimination via Gaussian Filtering**: Gaussian filtering is a smoothing technique that reduces Gaussian noise. It utilizes a Gaussian kernel to blur the image, effectively diminishing the impact of noise:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{2}$$

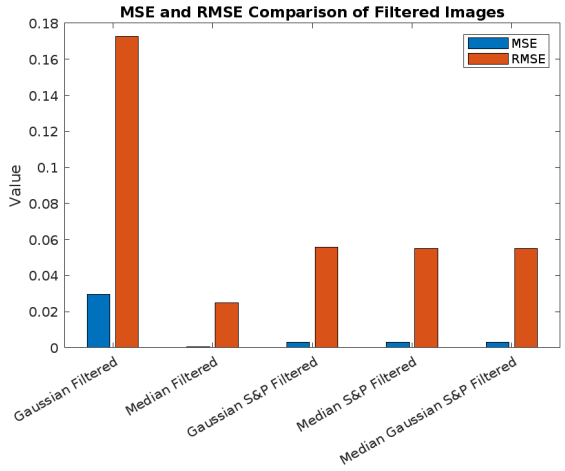
  where $G(x, y)$ is the Gaussian kernel, and σ is the standard deviation of the distribution. This method preserves edge properties while reducing noise, making it ideal for preprocessing in image segmentation tasks.

- **Salt-and-Pepper Noise Reduction via Median Filtering**: Median filtering is a non-linear process used to remove S&P noise. It replaces each pixel's value with the median value of the intensities in its neighborhood. This approach effectively preserves edges while removing noise, essential for maintaining the structural integrity of objects in the image.

Table 1. Filtered images: MSE and RMSE

Method	MSE	RMSE
Gaussian Filtered	0.029759	0.17251
Median Filtered	0.00061407	0.02478
Gaussian S&P Filtered	0.0030896	0.055584
Median S&P Filtered	0.0030237	0.054988
Median Gaussian S&P Filtered	0.0030237	0.054988

As evidenced in our final benchmark tests in Figure 18, even with the introduction of random noise, the models maintain high performance levels. This underscores the effectiveness of our proposed augmentation and preprocessing strategies, demonstrating the model's robustness against various noise types.



From Mumford-Shah to Chan-Vese

Introduced by Mumford and Shah in their seminal 1989 paper [4], the Mumford-Shah model has been a cornerstone in the field of image segmentation. The model seeks to partition an image into regions of approximately uniform intensity by minimizing an energy functional. For a given image I defined over a domain $\Omega \in \mathbb{R}^2$, the Mumford-Shah functional can be expressed as:

The Mumford-Shah model

$$\begin{aligned} E(u, C) = &\int_{\Omega \setminus C} |\nabla u|^2 \, dx \\ &+ \mu \int_{\Omega \setminus C} (u - I)^2 \, dx \\ &+ \nu \, \text{Length}(C), \end{aligned} \tag{3}$$

where:

- u is an approximation of the image
- I within the domain $\Omega \setminus C$
- C denotes the set of edges in the image
- μ, ν are regularization parameters, chosen arbitrarily by the user.

In the Mumford-Shah functional, Ω represents the entire image domain, while $\Omega \setminus C$ denotes the image domain excluding the set of edges C, essentially covering the regions within the image that are not part of the segmentation boundaries. The parameter μ controls the trade-off between the image's fidelity to its approximation u and smoothness, whereas ν regulates the importance of the contour's length in the overall segmentation, aiming to minimize unnecessary segmentation complexity. The functional aims to balance the smoothness of u and the fidelity to the original image I, along with the complexity of the segmentation represented by Length(C).

The model aims to find piecewise smooth representations of images by minimizing an energy function that penalizes the image's deviation from the model and the length of the contours.

Expanding upon the Mumford-Shah model, Chan and Vese introduced a simplified version in 2001 [3], targeting images with two distinct intensity levels. This model, known as the Chan-Vese model, is designed for binary segmentation tasks, optimizing a similar energy functional to directly distinguish between the two regions [3].

While the original Mumford-Shah model does not inherently rely on level set functions, its adaptations, including the Chan-Vese model, often utilize level set methods to represent the evolving curves in the segmentation process. This approach allows for a more flexible representation of contours and simplifies the numerical implementation of the segmentation task.

The Chan-Vese model

The Chan-Vese model is a specialization of the Mumford-Shah model for segmenting images into two distinct regions. It uses an energy function defined as:

$$\begin{aligned} F(c_1, c_2, C) = {} & \mu \,\text{Length}(C) \\ & + \nu \,\text{Area}(\text{int}(C)) \\ & + \lambda_1 \int_{\text{int}(C)} |u_0 - c_1|^2 \, dx \\ & + \lambda_2 \int_{\text{ext}(C)} |u_0 - c_2|^2 \, dx \end{aligned} \tag{4}$$

where c_1 and c_2 represent the mean intensities inside and outside the contour C. The terms λ_1, λ_2, and μ are weighting parameters that control the segmentation's adherence to the model's assumptions.

This model effectively segments images by evolving a contour C to minimize the above energy, adapting well to images with homogenous regions.

The Chan-Vese model simplifies this approach by assuming images with two intensity levels, optimizing a similar energy function to distinguish between these regions directly.

Example: The Chan-Vese algorithm excels in segmenting images with clear object boundaries against a uniform background, efficiently handling variations in intensity and texture without prior knowledge of object location.

2.4. Level Set Methods

Level set methods play a pivotal role in the domain of computational image segmentation, offering a robust framework for dynamically evolving contours. Within the Chan-Vese model, these methods facilitate the optimization of an energy functional that guides the segmentation process. The level set function, $\phi(x, y, t)$, encapsulates the contour C implicitly as its zero level set, evolving under a carefully designed partial differential equation (PDE). This evolution is meticulously driven to minimize the energy functional, as detailed in Equation 4.

where c_1 and c_2 denote the mean intensities within and outside the evolving contour C, respectively. The weighting parameters λ_1, λ_2, and μ are calibrated to balance the trade-offs inherent in the segmentation task, ensuring an adherence to the image's inherent structures while promoting contour smoothness and minimal complexity.

2.4.1 The Lipschitz Function

A critical aspect of the level set formulation is the initialization of the level set function $\phi_0(x, y)$, which should ideally be a Lipschitz function. This property is crucial for ensuring the numerical stability and convergence of the evolving contour. A Lipschitz function adheres to the condition:

$$|f(x) - f(y)| \leq L\|x - y\|, \tag{5}$$

for every pair x, y within the domain, where L represents the Lipschitz constant. This constraint ensures a controlled evolution pace, preventing excessive, abrupt changes in the contour's geometry.

Initialization with a Circle or Ellipse: For initializing the level set function in the Chan-Vese framework, employing geometric shapes such as circles or ellipses offers a pragmatic and theoretically sound approach. Both configurations qualify as Lipschitz functions, attributable to their

smooth, continuous boundaries that can be mathematically articulated with bounded derivatives.

A circular initial contour centered at (x_0, y_0) with radius r is defined by:

$$\phi_0(x, y) = \sqrt{(x - x_0)^2 + (y - y_0)^2} - r, \tag{6}$$

demonstrating Lipschitz continuity as the magnitude of its gradient is uniformly bounded [2]. Similarly, an elliptical contour, characterized by its semi-major and semi-minor axes a and b, respectively, the subtraction by 1 adjusts the level set to position the zero level set contour exactly at the boundary of the ellipse. This is the correct formulation for representing an elliptical contour in terms of a level set function; it is represented as:

$$\phi_0(x, y) = \sqrt{\left(\frac{x - x_0}{a}\right)^2 + \left(\frac{y - y_0}{b}\right)^2} - 1, \tag{7}$$

which also exhibits bounded gradients, affirming its suitability as an initial Lipschitz condition for the level set evolution.

The employment of these geometric forms as initial conditions not only simplifies the numerical implementation but also aligns with the theoretical underpinnings required for a stable and efficient segmentation process. This approach underscores the Chan-Vese model's versatility in adapting to images with diverse intensity profiles, facilitating precise segmentation outcomes.

Level Set Formulation

Instead of searching for the solution in terms of C, we can redefine the problem in the level set formalism. In the level set method, $C \subseteq \Omega$ is represented by the zero level set of some Lipschitz function $\Phi : \Omega \to \mathbb{R}$, such that:

$$\begin{cases} C = \{(x, y) \in \Omega : \Phi(x, y) = 0\} \\ \text{inside}(C) = \{(x, y) \in \Omega : \Phi(x, y) > 0\} \\ \text{outside}(C) = \Omega \backslash \text{inside}(C) = \{(x, y) \in \Omega : \Phi(x, y) < 0\} \end{cases} \tag{8}$$

Energy Functional

The energy functional in terms of Φ is given by:

$$\begin{aligned} F(c_1, c_2, \Phi) = &\mu \int_\Omega \delta(\Phi(x, y)) \|\nabla \Phi(x, y)\| \, dx \, dy \\ &+ \nu \int_\Omega H(\Phi(x, y)) \, dx \, dy \\ &+ \lambda_1 \int_\Omega |u_0(x, y) - c_1|^2 H(\Phi(x, y)) \, dx \, dy \\ &+ \lambda_2 \int_\Omega |u_0(x, y) - c_2|^2 (1 - H(\Phi(x, y))) \, dx \, dy \end{aligned} \tag{9}$$

where:

- μ is the parameter that weights the length of the contour, controlling the smoothness of the segmentation boundary;
- ν weights the area inside the contour, affecting how the area influences the segmentation;
- λ_1 and λ_2 weight the intensity differences within and outside the contour, respectively, aiding in fitting the model to the image data;
- H is the Heaviside function, used to differentiate the regions inside and outside the contour;
- x and y are the spatial coordinates in the image domain Ω;
- $\nabla \Phi$ is the gradient of Φ, with its norm representing the rate of change of Φ, related to the curvature of the contour C in the level set method.
- The domain Ω represents the area over which the image is defined and the segmentation is performed.

Partial Differential Equation for Φ

The function Φ which minimizes the energy functional satisfies the PDE:

$$\begin{aligned} \frac{\partial \Phi}{\partial t} = \delta(\Phi) \Bigg(&\mu \text{div} \left(\frac{\nabla \Phi}{\|\nabla \Phi\|} \right) - \nu \\ &- \lambda_1 (u_0 - c_1)^2 + \lambda_2 (u_0 - c_2)^2 \Bigg) \end{aligned} \tag{10}$$

The curvature $\kappa(\Phi)$ [1] of the evolving contour is given by the spatial derivatives of Φ up to the second order:

$$\kappa(\Phi) = \frac{\Phi_{xx} \Phi_y^2 - 2 \Phi_{xy} \Phi_x \Phi_y + \Phi_{yy} \Phi_x^2}{(\Phi_x^2 + \Phi_y^2)^{3/2}} \tag{11}$$

Numerical Scheme

First, we introduce the regularization of the Heaviside and Dirac delta functions for some constant $\epsilon > 0$ as follows:

$$\begin{aligned} H_\epsilon(x) &= \frac{1}{2} \left(1 + \frac{2}{\pi} \arctan \left(\frac{x}{\epsilon} \right) \right), \\ \delta_\epsilon(x) &= \frac{1}{\pi} \frac{\epsilon}{\epsilon^2 + x^2}. \end{aligned}$$

These regularizations are used in simulations to lead to the global minimum of the energy efficiently. Choosing $\epsilon = h$, where h is the space step, is reasonable since h is the smallest space step in the problem. In the available case we used $\epsilon = h = 1$.

Discretization of the PDE

Let $\Phi_{i,j}^n = \Phi(n\Delta t, x_i, y_j)$, where Δt is the time step. The PDE can be discretized using spatial finite differences as follows:

$$\Delta_x \Phi_{i,j}^n = \frac{\Phi_{i+1,j}^n - \Phi_{i,j}^n}{h}, \quad \Delta_x^- \Phi_{i,j}^n = \frac{\Phi_{i,j}^n - \Phi_{i-1,j}^n}{h},$$
$$\Delta_y \Phi_{i,j}^n = \frac{\Phi_{i,j+1}^n - \Phi_{i,j}^n}{h}, \quad \Delta_y^- \Phi_{i,j}^n = \frac{\Phi_{i,j}^n - \Phi_{i,j-1}^n}{h}.$$

The linearized discretized PDE becomes:

$$\begin{aligned}\frac{\Phi_{i,j}^{n+1} - \Phi_{i,j}^n}{\Delta t} = \delta_h(\Phi_{i,j}^n) \Bigg[\mu \Bigg(& \frac{\Delta_x^+ \Phi_{i,j}^n + \Delta_x^- \Phi_{i,j}^n}{2h^2} \\ & + \frac{\Delta_y^+ \Phi_{i,j}^n + \Delta_y^- \Phi_{i,j}^n}{2h^2} \Bigg) \\ & - \delta_h(\Phi_{i,j}^n) \Big(v\lambda + \frac{1}{u_c^{i,j}} - \frac{1}{2\Phi_{i,j}^n - \lambda_{u_c^{i,j}}} \Big) \Bigg].\end{aligned}$$

Constants Definition

Defining constants for a given Φ^n as:

$$C_1 = \frac{1}{4} \left(\Phi_{i,j+1}^n - \Phi_{i,j-1}^n \right)^2,$$
$$C_2 = \frac{1}{4} \left(\Phi_{i+1,j}^n - \Phi_{i-1,j}^n \right)^2,$$
$$C_3 = \frac{1}{4} \left(\Phi_{i,j+1}^n - \Phi_{i,j-1}^n \right)^2,$$
$$C_4 = \frac{1}{4} \left(\Phi_{i+1,j}^n - \Phi_{i-1,j}^n \right)^2.$$

The simplified equation then is:

$$\Phi_{i,j}^{n+1} = \Phi_{i,j}^n + \Delta t \delta_h(\Phi_{i,j}^n) \mu \left(C_1 + C_2 + C_3 + C_4 \right).$$

Summary of the Algorithm

1. Initialize $\Phi_{i,j}^0$ to some Lipschitz function Φ_0.
2. Compute c_1 and c_2 using the regularized Heaviside function.
3. Solve the PDE using the discretized equation.
4. Reinitialize $\Phi_{i,j}^{n+1}$ to be the signed distance function to its zero level set using the reinitialization equation.
5. Check whether the solution is stationary. If not, continue; otherwise, stop.

The process of evolving the level set function Φ during image segmentation aims to minimize an energy functional that accurately represents the object's boundary within the image. This iterative update of Φ continues until it converges to a stationary state, which signifies that the contour of the object has been precisely delineated.

The update mechanism for Φ at each grid point (i, j) from iteration n to $n+1$ relies on the image data and the curvature of Φ itself. This iterative update is terminated when the changes in Φ between successive iterations become insignificant across the entire image domain, implying that Φ has reached a stationary state and further iterations would not meaningfully alter its configuration. The mathematical expression for this stopping criteria is as follows:

The iterative process ceases when the discrepancy between $\Phi_{i,j}^{n+1}$ and $\Phi_{i,j}^n$ is lesser than $\Delta t \cdot h^2$, indicating the attainment of a stationary state. This condition is verified through the computation of a quantity Q, which assesses the stationarity of Φ across iterations:

$$Q = \sum \left(\left| \Phi_{i,j}^{n+1} - \Phi_{i,j}^n \right| < \Delta t \cdot h^2 \right)$$

In this context, Δt symbolizes the timestep size, and h signifies the spatial resolution of the grid. The summation traverses all grid points (i, j), with the inner comparison producing a Boolean value that denotes whether the change at each point is beneath the predefined threshold. The algorithm concludes when a significant majority of the points meet this stipulation, suggesting that Φ has become stationary.

For the purpose of reinitializing Φ, the ensuing equation is utilized to guarantee the maintenance of the signed distance function property, which is pivotal for the numerical stability and precision of the level set method:

$$\Phi_{i,j}^{n+1} = \Phi_{i,j}^n - \Delta t \cdot \text{sign}(\Phi(x, y, t)) \cdot G(\Phi_{i,j}^n)$$

Herein, $\text{sign}(\Phi(x, y, t))$ ensures the directionality of the update respects the gradient of the level set function, aiming to minimize deviations from a signed distance function. Concurrently, $G(\Phi_{i,j}^n)$ is a function derived from the gradient of Φ, adjusted to realign its values towards sustaining the signed distance property.

2.5. Chan-Vese Loss Implementation for RGB Images in PyTorch

In the pursuit of enhancing image segmentation models, we introduce the `ChanVeseLossPyTorchRGB`, a PyTorch module designed to adapt the classical Chan-Vese loss for RGB images. This module is particularly tailored for segmentation tasks, leveraging the characteristics of RGB images to compute the loss more effectively.

2.5.1 Implementation Details

The PyTorch module `ChanVeseLossPyTorchRGB` implements this adapted loss function. It requires the predicted segmentation masks, the ground truth masks, and the original RGB images as inputs. The intensity difference loss, $\mathcal{L}_{\text{intensity},c}$, is computed for each RGB channel and averaged. The smoothness term, $\mathcal{L}_{\text{smoothness}}$, utilizes the gradient of the predicted mask, emphasizing the regularization of the segmentation boundary.

The key components of this module are:

- The `heaviside` function, approximated using the hyperbolic tangent to ensure a smooth transition, facilitating the differentiation process required for gradient descent optimization.
- The `mean_intensities` function, which calculates the average intensities inside and outside the segmented region for each channel, crucial for the intensity difference computation.
- The computation of the smoothness loss, employing the gradient of the sigmoid-activated predictions, to enforce boundary smoothness.

The incorporation of this loss function into segmentation models promises enhanced performance by effectively leveraging the color information inherent in RGB images and enforcing smooth, coherent segmentations.

`Python` code snippet for the `ChanVeseLossPyTorchRGB` module:

See code here.

3. Results

A solution in `MATLAB` code is provided at the end for initialization of level set functions, the evolution of the curvature using the above discretized method, able to cope the dinamic solution of numerical systems.

Consider some assumptions have been done, as for instance using a circle as initial level set method for final test, since it is common use in the literature (look for reference and citation), moreovfer $dy = dx = 1$ has been consider to 1 supposing that images are equally spaced, $epsilon = h = 1$ since we are dealing with square images, each step should be of length 1 (moving of one step at the time). At last, the area is set to 0 since it doesnt efficiently help with the image segmentation resolution.

Here below there are few experimental results from the aforementioned resolution method:

3.1. Grey images

3.2. RGB images

When processing RGB images, as in **??** with the segmentation algorithm, it is imperative to consider the distinct

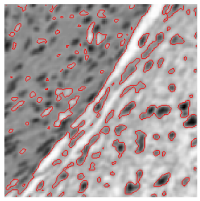


Figure 1. Original segmented image in grey scale

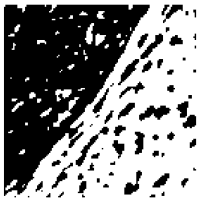


Figure 2. Original segmented image in grey scale

nature of each RGB channel. Consequently, the algorithm is executed separately on each of the RGB channels, resulting in three preliminary segmentation maps. To synthesize these into a final, comprehensive segmentation result, an aggregation step is employed, utilizing logical operations such as AND, OR, and Majority Vote. The aggregation process is delineated as follows:

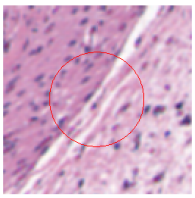


Figure 3. Original image with initial level set function to circle

- **Logical AND**: This operation aggregates the segmentation maps by considering a pixel in the final segmentation to be part of the object if and only if it is identified as part of the object in all three RGB channel segmentations. Mathematically, for a given pixel position (i, j), the final segmentation $S_{i,j}$ is defined as:

$$S_{i,j} = R_{i,j} \wedge G_{i,j} \wedge B_{i,j}$$

  where $R_{i,j}$, $G_{i,j}$, and $B_{i,j}$ represent the segmentation decisions at pixel (i, j) for the Red, Green, and Blue channels, respectively.

- **Logical OR**: In contrast, the OR operation aggregates the maps by marking a pixel as part of the object if it

is identified as such in any one of the RGB channel segmentations. Thus, for each pixel (i, j):

$$S_{i,j} = R_{i,j} \vee G_{i,j} \vee B_{i,j}$$

- **Majority Vote**: This approach determines a pixel's inclusion in the final segmentation based on a majority rule among the three channels. A pixel is considered part of the object if at least two out of the three channel segmentations agree on its inclusion. Formally:

$$S_{i,j} = \text{Majority}(R_{i,j}, G_{i,j}, B_{i,j})$$

where $\text{Majority}(R_{i,j}, G_{i,j}, B_{i,j})$ returns the majority value among $R_{i,j}$, $G_{i,j}$, and $B_{i,j}$.

Following the segmentation of each RGB image channel and the subsequent aggregation via these logical operations, the resulting segmentation maps are illustrated below each corresponding RGB image, showcasing the impact of the chosen aggregation method on the final segmentation outcome.

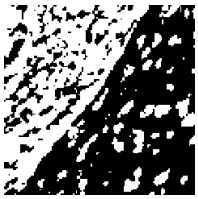


Figure 4. Generated mask after segmentation on original image: channel R

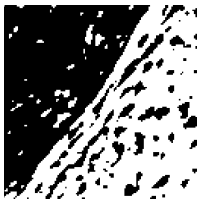


Figure 5. Generated mask after segmentation on original image: channel G

Here below an example of generated mask.

4. Appendix A: Project results visualization

4.1. Preprocessing and data preparation

As we can establish from Section 2. 'Proposed Approach', within the Preprocessing and Data Augmentation part, we can see how the image data is treated. Recall that

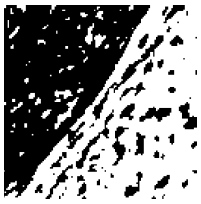


Figure 6. Generated mask after segmentation on original image: channel B

Figure 7. RGB with GN applied using Gaussian filter, generated Mask aggregated using logical OR.

Figure 8. RGB with GN applied using Gaussian filter, generated Mask aggregated using logical AND.

Figure 9. RGB with GN applied using Gaussian filter, generated Mask aggregated using logical Majority Vote.

a digital image is stored as $I \in \mathbf{R}^{r \times c \times n_{channels}}$, where r represents the image length, whereas c (# columns), and $N_{channels}$ is the number of channels of colored images, in this case since we dealt with RGB images, it is indeed $I \in \mathbf{R}^{128 \times 128 \times 3}$. Recall that for grey scaled images, the task is a bit easier, since the level method evolution works only on a unique channel, instead of RGB images where multiple tasks has to run for each channel, and its result is made up by the aggregation of those 3 channels. Various techniques are exploited, as shown in subsection 3.2, here

Figure 10. RGB with GN and S&P applied using Median filter, generated Mask aggregated using logical OR.

Figure 11. RGB with GN and S&P applied using Median filter, generated Mask aggregated using logical AND.

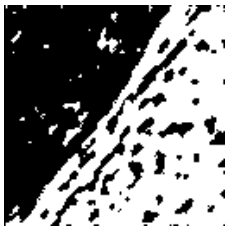


Figure 12. RGB with GN and S&P applied using Median filter, generated Mask aggregated using logical Majority Vote.

Figure 13. RGB with GN and S&P applied using only Gaussian filter on Gaussian Noise and Salt & Pepper, generated Mask aggregated using logical OR.

below you can find experimental results on the dataset.

5. Appendix B: Additional run experiments examples

Here below in figure 20 we can also see an additional result run using MRI (Magnetic resonance imaging) as initial input as shown in figure 21.

As we can see from the segmented mask, the area in black filled with unsual white patterns shows us a tumor

Figure 14. RGB with GN and S&P applied using only Gaussian filter on Gaussian Noise and Salt & Pepper, generated Mask aggregated using logical AND.

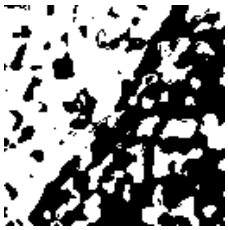


Figure 15. RGB with GN and S&P applied using only Gaussian filter on Gaussian Noise and Salt & Pepper, generated Mask aggregated using logical Majority Vote.

Figure 16. RGB with GN and S&P applied using both Gaussian and Median filter, generated Mask aggregated using logical OR.

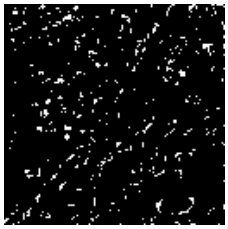


Figure 17. RGB with GN and S&P applied using both Gaussian and Median filter, generated Mask aggregated using logical AND.

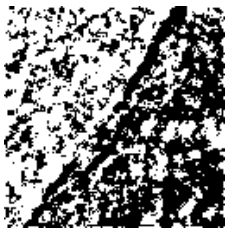


Figure 18. RGB with GN and S&P applied using both Gaussian and Median filter, generated Mask aggregated using logical Majority Vote.

alike pattern.

6. Discussion

In this paper we saw how the Chan-Vese method can be applied to a wide range of images accomplishing overall good results, even with images subject to various types of noise. The project could be expanded by exploiting the loss implemented both in `MATLAB` and in `python` is possible to train on a custom dataset to further explore better optimization techniques for increased performances in the loss module, in contrast to new SoAT methodologies.

References

[1] G. Aubert, M. Barlaud, O. Faugeras, and S. Jehan-Besson. Image segmentation using active contours: Calculus of variations or shape gradients? *SIAM Journal on Applied Mathematics*, 63(6):2128–2154, 2003. 4

[2] L. Bregman. The relaxation method of finding the common points of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7:200–217, 1967. 4

[3] Tony F Chan and Luminita A Vese. Active contours without edges. *IEEE Transactions on image processing*, 10(2):266–277, 2001. 3

[4] David Mumford and Jayant Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Mathematics*, 42(4):577–685, 1989. 2

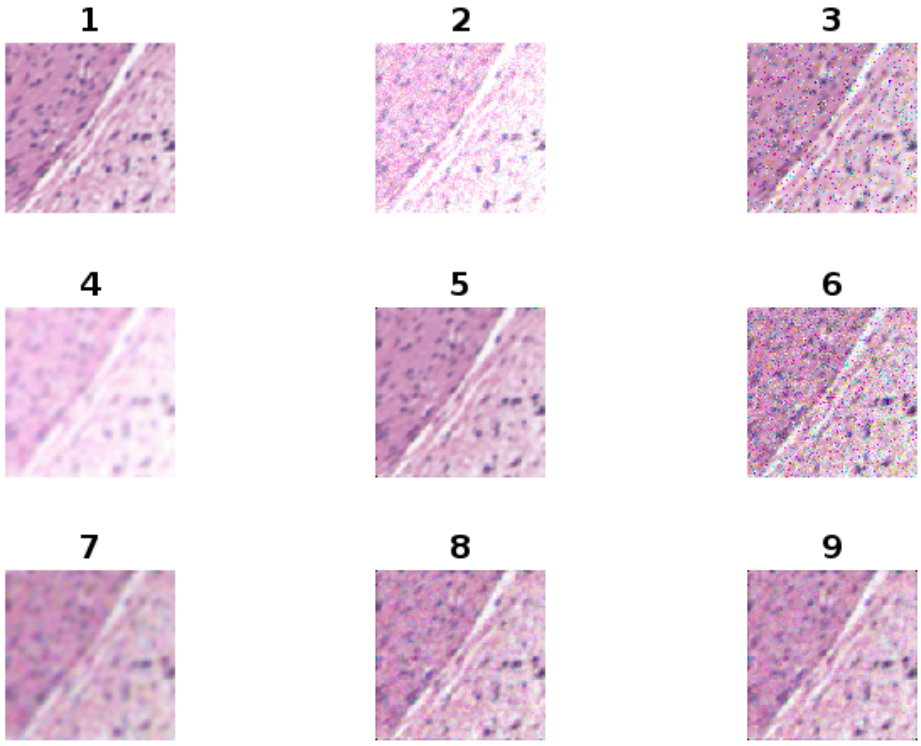


Figure 19. Legend of the preprocessed images and filtered images

Label	Description
(1)	Original Image
(2)	Gaussian Noisy Image
(3)	Salt and Pepper Noisy Image
(4)	Gaussian Filtered
(5)	Median Filtered
(6)	Gaussian Salt and Pepper Image
(7)	Gaussian Salt and Pepper Filtered
(8)	Median Salt and Pepper Filtered
(9)	Median Gaussian Salt and Pepper Filtered

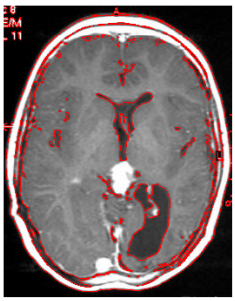


Figure 20. Segmented image MRI brain with tumor

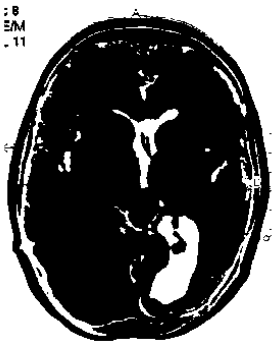


Figure 21. Generated segmented mask MRI brain with tumor