UNIVERSIDAD REY JUAN CARLOS

Escuela de Ingeniería de Fuenlabrada

Biomedical Engineering

Course 2023-2024

# Lab II: Medical Image Segmentation

MEDICAL IMAGE ANALYSIS

Group 11:

Enrique Almazán Sánchez

Guillermo Ots

Javier Alfonso Villoldo Fernández

# OUTLINE

# 1. Objectives

The goal of this Laboratory is to become familiar with the different basic methods of segmentation for medical images that have been covered in class and to gain a deeper understanding of their underlying operation. In particular, thresholding, region growing, and watershed algorithms will be implemented.

Moreover, more advanced segmentation algorithms such as contour tracking as well as others in the available in the state-of-the-art of the field will be explored, thus, increasing our understanding of the available techniques and their applicability to medical image processing.

We will apply these algorithms to real clinical images to assess their effectiveness, evaluating the results on how well the algorithms achieve their intended goals, which are segmenting the desired Regions of Interest (ROIs) of the proposed images.

Additionally, not only will the results of the implemented techniques be compared with each other but also with a supplementary image with different characteristics.

# 2. Contributions

- Enrique Almazán Sánchez: Thresholding, Region Growing and Active Contours (brain MRI) and "advanced segmentation method (brain MRI and heart MRI)".
- Guillermo Ots Rodríguez: Watershed algorithm implementation (brain MRI and heart MRI).
- Javier Villoldo Fernández: Thresholding, Region Growing (heart MRI) and Active Contours (heart MRI and brain MRI).

# 3. Previous issues

Two images were selected to perform this lab practice, a brain MRI with a tumor as Region of Interest (ROI) to be segmented and a heart MRI, being the ventricles, the aorta and the left atrium the ROI.

The histogram of each of the images is computed, as it will be an important aspect in some of the following segmentation techniques. In addition, the Cumulative Distribution Function (CDF), will also be presented, as we will perform later the histogram equalization. The CDF shows how pixel values are distributed throughout the image.

In order to compute both, the histogram and CDF, the personalized function '*img_hist_cdf*' is used, which will also display them with their image, having the following results:

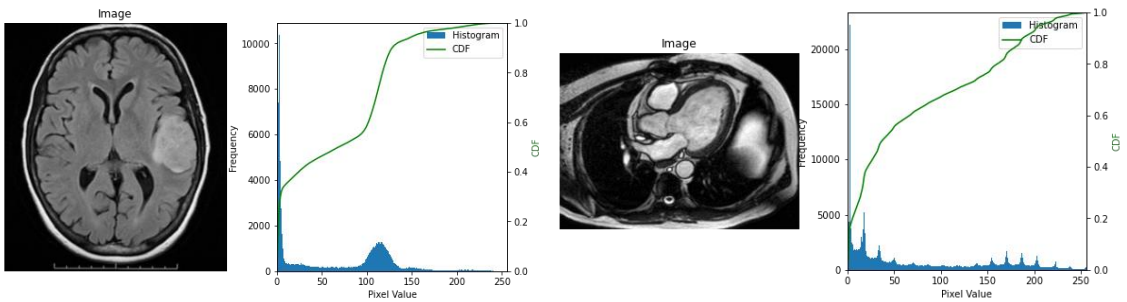


*Figure 1: Brain MRI and Heart MRI with their respective histogram and CDF.*

### Histogram Equalization

Histogram equalization is an image processing technique used to enhance the contrast and improve the visibility of details in digital images, by redistributing its intensity levels (or gray values) so that they are spread over a wider range. This results in a more balanced distribution of pixel intensities.

It works by computing the cumulative distribution function (CDF) of pixel intensities in the image's histogram and then mapping the pixel values according to this CDF. In simple terms, it stretches the intensity values in such a way that the full range of intensities is utilized.

The primary benefit of histogram equalization is the enhancement of image contrast. It can make details and features in an image more visible and distinguishable. This is particularly useful when dealing with images that have poor contrast or lighting conditions.

Thus, when applying segmentation techniques to images, the results can be influenced by the original image's intensity distribution. If an image has poor contrast or uneven lighting, segmentation algorithms might struggle to distinguish objects or regions effectively, so a standardization Intensity Distribution is needed.

However, the images used in this lab practice are already high contrast as seen above (the heart MRI more than the brain MRI). Then, why apply it? It has been seen that the background of the implemented images is entirely black, being a high majority of the pixel intensities on the left of the histogram. Then, it will be useful to distribute the gray levels more homogeneously among the entire histogram, by decreasing the peaks and increasing the rest. This would help to see if the results obtained following the different segmentation techniques are better with a preprocessed image than with the original.

Also, by applying histogram equalization to images before subjecting them to segmentation techniques, we standardize the intensity distribution across all images. This ensures that segmentation methods are tested on a level playing field. This is important for fair and meaningful comparisons between different segmentation algorithms. The results can be seen in the following figure.

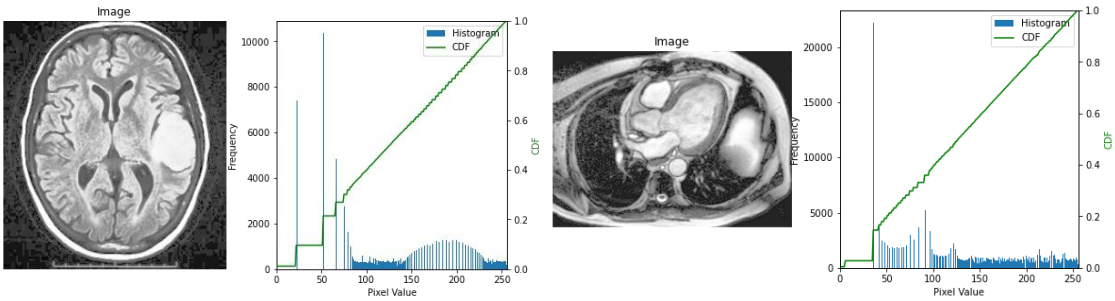


*Figure 2: Brain MRI and Heart MRI with their respective equalized histogram and CDF*

# 4. Segmentation Techniques

In this section Thresholding, Region Growing and Watershed are the chosen segmentation algorithms to be implemented through a python code. The respectively and corresponding explanations of each technique and their results are shown below.

## 4.1. Thresholding

**Thresholding** is a straightforward image segmentation technique that separates objects or regions in an image based on the pixel intensity. It works by classifying each pixel as part of the foreground (object or region of interest) giving the greater intensity (value 1) or the background (everything else) giving the lower intensity (value 0) using a predefined intensity threshold, having a binary image as a result.

**Advantages of Thresholding:**

- **Simplicity:** Thresholding is the simplest method of segmenting an image, as it relies only on information from the histogram and binarize based on a threshold. It is easy to implement and understand, making it suitable for quick, simple segmentation tasks.
- **Computationally Efficient:** It is a very fast procedure and requires minimal computational resources, making it suitable for real-time applications.
- **Effective for Simple Scenes:** It is useful when there are distinct intensity differences between objects and the background, being interactive, since the user is the one who imposes the thresholds depending on the application.

**Disadvantages of Thresholding:**

- **Ineffective for Complex Scenes:** It struggles when intensity differences aren't clear or when lighting varies significantly, as it assumes that the intensity level is constant in the image.
- **Sensitivity to Threshold Selection and Difficult automatic thresholding:** Choosing the right threshold value is often challenging and critical for effective segmentation, as well as when automating the thresholding process.
- **Point-Base Segmentation**: It is based only on the isolated information of each point, discarding any type of spatial information.
- **Limited to Binary Output:** Thresholding produces a binary image, which may not be suitable for multi-region segmentation.
- **Loss of Information:** Fine details and intensity variations can be lost.

### 4.1.1. Simple Thresholding

The OpenCV dependency has different types to apply the thresholding segmentation technique, which follows different methods.

| | |
|---|---|
| THRESH_BINARY<br>Python: cv.THRESH_BINARY | $\texttt{dst}(x,y) = \begin{cases} \texttt{maxval} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$ |
| THRESH_BINARY_INV<br>Python: cv.THRESH_BINARY_INV | $\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{maxval} & \text{otherwise} \end{cases}$ |
| THRESH_TRUNC<br>Python: cv.THRESH_TRUNC | $\texttt{dst}(x,y) = \begin{cases} \texttt{threshold} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$ |
| THRESH_TOZERO<br>Python: cv.THRESH_TOZERO | $\texttt{dst}(x,y) = \begin{cases} \texttt{src}(x,y) & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$ |
| THRESH_TOZERO_INV<br>Python: cv.THRESH_TOZERO_INV | $\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$ |

*Figure 3: Different types of thresholding applied by the OpenCV (cv2) dependency.*

All of them are compared in the personalized function *'simple_thres*' explained in the *Table 1* of *Annex 1*. Its goal is to compare the thresholding types given by the OpenCV dependency and then obtain the best thresholding type for each of our images. Thus, with the results, also shown in *Annex 1* (for both images, *Figure …* and *Figure …*), it has been seen that the **binary thresholding** suits better for both of them rather than the rest.

The truncated thresholding followed our hypothesis. It does not fit neither the brain MRI, nor the heart MRI due to its procedure seen in *Figure 3*. The '*tozero'* and negative '*tozero'* thresholding show better results for the brain MRI than for the heart MRI. However, the point is to obtain a binary image from the thresholding, reason why these types are discarded. Finally, the negative binary thresholding was not chosen as it was wanted to emphasize the region of interest (ROI) as white and the rest, which was consider as background, as black.

## Comparisons between the original image and its transformations

When the image's histogram lacks a clearly visible bimodal distribution or is heavily contaminated by noise, preprocessing techniques become essential to enhance the histogram's shape and facilitate the thresholding process. Several methods can be employed for this purpose, including histogram equalization, gamma corrections, and the application of filters.

For instance, the **median filter** is a noteworthy enhancement technique for thresholding. In contrast to the mean filter, which might blur the boundaries of larger regions due to its smoothing effect, the median filter excels at reducing small variations in texture without significantly affecting the edges of larger areas. This makes it particularly useful in scenarios where preserving fine details while reducing noise is crucial for accurate thresholding.

Additionally, when it comes to segmentation tasks, **applying histogram equalization before thresholding** is often beneficial. The enhanced contrast and reduced sensitivity to lighting variations provided by histogram equalization can lead to more reliable and accurate thresholding results, especially in scenarios where the objects of interest have varying or subtle intensity differences with the background.

Therefore, in practice, the combination of histogram equalization, filter, or other transformations with the segmentation techniques such as thresholding is a useful preprocessing step in image segmentation, helping to address some of the limitations of thresholding, particularly in scenarios with complex scenes or challenging lighting conditions.

After analysing the results previously obtained and understanding the importance of pre-processing before segmentation, with the thresholding technique chosen, the **binary thresholding** of the OpenCV dependency, a comparison of the results obtained thresholding the original image or its transformations (median blur and histogram equalization) is done. In order to do so the personalized function '*thresh_comp*' was created, explained in *Table 2* of *Annex 2*.

As in the previous function the type of thresholding was the parameter of the OpenCV method to be changed, in this function was the threshold value. Then, it is needed to take into account that the thresholds used for the original brain MRI image or its transformations, median filter and histogram equalization, were obtained following a trial an error procedure, having the threshold values of 127, 133, and 227 respectively. Each of them is the optimal value that gives the better segmentation (following thresholding) for each of the images. For the heart MRI the same procedure was followed, and the obtained values are given in *Annex 2*.

Even though, the images used were free of noise, it is seen in *Figure 4* below for the brain MRI and in *Annex 2* for the heart MRI, that the best result obtained when thresholding them is after applying a median filter.

In the case of the brain MRI, even though the image was free of noise, it is clearly seen that the ROI (the tumor) is better segmented if a median filter is applied first, as it separates better the background from the object to be segmented.

However, this filter has also the ability of preserving edges while smoothing the rest of the image as it softens small variations, in this case inside the tumor, while keeping sharp transitions between regions intact, in this case the tumor edges. This makes the thresholding more robust when distinguishing the tumor boundaries from the surrounding brain tissue. Then, false regions marked as part of the ROI are minimized compared to what happens when thresholding the original image or the one with an equalized histogram, where additional structures are included in the ROI.

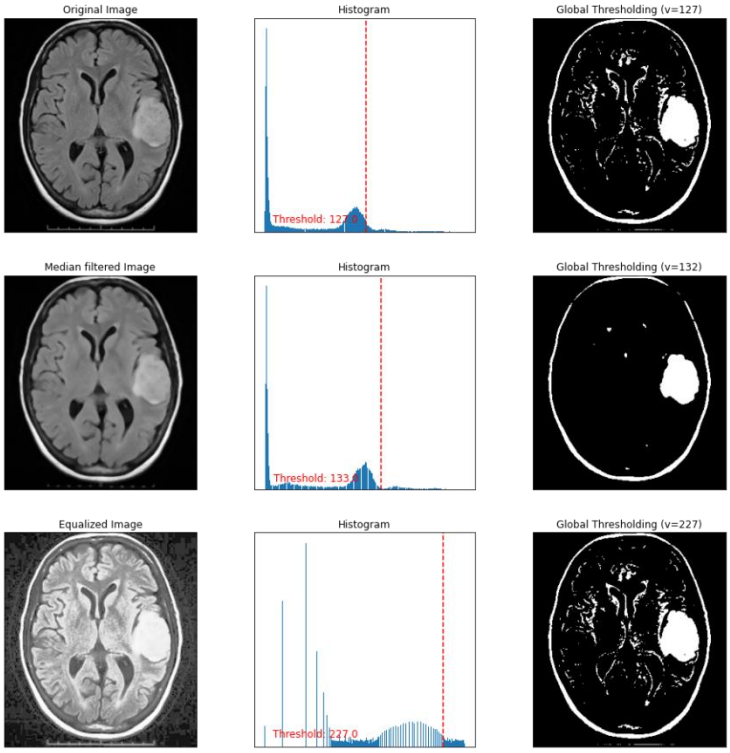


*Figure 4: The original brain MRI image and its transformations, median filter, and histogram equalization, with their respective thresholded result and the histogram with the threshold use.*

The bad results given by the original image are due to the boundaries not being well defined, as well as having similar gray values in the tumor than in other brain areas. Thus, as the histogram equalization standardizes the gray levels, distributing them more homogenously through the histograms, leads to similar results. However, the median filter, is a suitable preprocessing step for improving thresholding-based segmentation for the brain MRI image, due to its ability previously explained, making the false added parts to the ROI disappear.

Nonetheless, the effectiveness of the median filter with respect to applying it after thresholding depends on the image. For the brain MRI it was a good step to follow, but for the heart MRI, even though it still improves the outcome of the other two images (original and histogram equalization), it does not give meaningfully better results, to outperform the other results as it does for the former image.

This is due to the amount of similar gray levels of the ROI and the rest of the image, and their position. In the brain MRI, they are small in quantity representing the gray matter, being near to the outer surface, which is black, or located near the ventricles, also black. Thus, the

median filter is capable of smoothing these parts. However, in the heart MRI, the similar gray levels are bigger in quantity, representing other complete bodies, being impossible to smooth by the median filter.

Consequently, the thresholding technique is not a good approach for segmenting the proposed ROI for the heart MRI. Other reason can be its histogram, which it can be seen in *Figure 1* that is not bimodal, but it has several peaks, making very challenging or even impossible the selection of a threshold for this task, despite if it is done manually.

Other aspect to keep into account is the threshold implemented in each of the images. It is seen that this value increments very little from thresholding the original image to thresholding the median filter imaged. This small increase is due to the smoothing that takes place also in the histogram of the image, reducing the sharp peaks. But the considerable change in the increase in the threshold value when applying this segmentation method to the histogram equalized image. It has sense, as what was done when applying this technique was shifting the gray levels to the right, ergo, taking more whiter values as most of the original ones were black (due to the background), having as a result a shifted threshold to the right (easily seen in the figure above or in the one presented in *Annex 2*).

Thus, regarding the threshold it is easy to see that when the value is decrease it will add more objects to our ROI, as every value that it is above the threshold will be considered as part of it. On the other hand, when increasing the threshold, the ROI will decrease, as more objects will be considered as background.

The easy conclusion can be that thresholding it is a good approach for segmenting the brain MRI, due to the results obtained and the image having a bimodal histogram. However, as it is seen in *Figure 4* two key points stand out.

- It is a bimodal histogram due to the background, as this part includes most of the black and gray values of the image. Thus, thresholding would be a perfect approach to divide the image between the black parts, the ventricles and outer part of the brain with the background (peak in the right); and the gray and white parts (peak in the middle) which are the brain tissue, white and gray matter, as well as the skull.
- The threshold used to segment the brain is located in the right most part of the mid-histogram peak for the three images, not between both peaks as supposed. Thus, even doing this procedure manually, it will be challenging to find a valid threshold, not mentioning that it will be almost impossible to do it through an automated process such as Otsu´s thresholding (explained below).

These aspects show that the thresholding segmentation technique is good for segmenting the brain MRI, having good results for the implemented ROI. However, this process should be done manually, by an expert with anatomical knowledge, being a tedious process with lower repeatability as the results depend on factors such as previous experience, fatigue...

### 4.1.2. Otsu's Thresholding

The **Otsu's method**, also known as **Otsu's thresholding** or **maximum variance method**, is an automatic image thresholding technique used to separate an image into two classes: foreground and background. It does this by determining an optimal threshold that minimizes the intra-class variance and maximizes the inter-class variance of pixel intensities.

1. **Histogram Computation**: Otsu's method starts by computing the histogram of pixel intensities in the image. This histogram represents the frequency of each intensity level.
2. **Threshold Search**: It then systematically considers all possible threshold values between the minimum and maximum intensity levels present in the image. For each threshold, it divides the image into two classes: pixels with intensities below the threshold (background) and pixels with intensities above the threshold (foreground).
3. **Mean and Variance Computation**: The background ($m_b$, $v_b$), object ($m_f$, $v_f$) and complete image (m, v) means and variances are computed respectively.
4. **Two Classes Variance Computation**: For each threshold, Otsu's method calculates the weighted sum of variances of the two classes:
   - *Intra-Class Variance ($v_{within}$)*: The variance within each class, reflecting the spread of pixel intensities in that class.
   $$v_{within} = P_b(t)v_b + P_f(t)v_f$$
   - *Inter-Class Variance ($v_{inbetween}$)*: The variance between the two classes, indicating how well they are separated.
   $$v_{inbetween} = v - v_{within} = P_b(t)P_f(t)\left(m_b - m_f\right)^2$$
5. **Optimal Threshold Selection**: The threshold that minimizes the intra-class variance while maximizing the inter-class variance is chosen as the optimal threshold. This threshold value effectively separates the image into foreground and background, maximizing the contrast. Basically, it is the same as maximizing Q(t), meaning to effectively separate the image into two classes, typically foreground and background.
   $$Q(t) = \frac{v_{inbetween}}{v_{within}}$$
6. **Thresholding**: Finally, the image is thresholded using the optimal threshold. Pixels with intensities greater than or equal to the threshold are assigned to the foreground, while pixels with intensities lower than the threshold are assigned to the background.

**Advantages**:

- Otsu's method is an automatic thresholding technique, which means it doesn't require manual threshold selection.
- It works well for images with bimodal or near-bimodal intensity distributions.
- The resulting threshold provides a clear separation between objects and background.

**Disadvantages**:

- Otsu's method may not perform optimally for images with complex intensity distributions, such as those with multi-modal distributions.
- It assumes that the image can be separated into two classes, which might not be suitable for all segmentation tasks.

The Otsu method will be applied with two different dependencies, OpenCV (cv2) and skimage, seeing that it will have the same results. Each of them will be implemented with a personalized function, "*otsu_thresh*" explained in *Annex 3* which depending on the value of the parameter 'dependency' will use one dependency or another.

As two different dependencies are used, their similarities and differences are analysed. Both OpenCV and skimage use the same underlying Otsu's thresholding algorithm, explained above, applying the algorithm with the flag cv2.THRESH_OTSU for the former and using skimage.filters.threshold_otsu for the latter.

The primary difference is in how these functions are implemented in the code. In OpenCV, cv2.threshold must be used as Otsu´s thresholding is applied with a flag, expecting an input image, a threshold value (which is ignored as an automatic procedure is being followed), a maximum value, and a thresholding type (as explained above in the Simple Thresholding section). In contrast, skimage's function expects only the input image.

Then, they also differ in the return value. OpenCV's function returns two values: the threshold value that was computed (using Otsu's method) and the thresholded image. In contrast, skimage's function returns only the computed threshold value. To threshold the image, the threshold should be applied to the image separately.

However, even though they present these differences, the result obtained for the threshold is exactly the same using any dependency for both images, bran MRI and heart MRI.

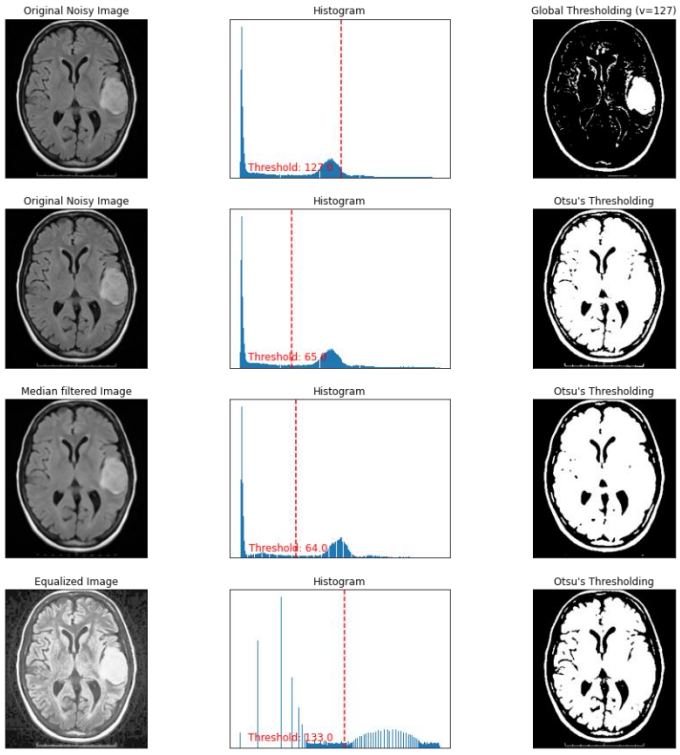


*Figure 5: Otsu´s thresholding applied to the original brain MRI image and its transformations, median filter, and histogram equalization, with their respective thresholded result and the histogram with the threshold use compared to the global thresholding.*

The statement made in the previous section is now confirmed, not having good results for the image nor for its transformations (median filter and histogram equalization).

The reasons were also explained above, thus, Otsu´s thresholding (automatic) is not a good approach for the images having histograms as the one presented by the heart MRI.

In the case of the brain MRI, it is worth noting that the hypothesis made are again confirmed with the practical result. The Otsu´s thresholding may not be a suitable approach for segmenting the ROI chosen, but it could be appropriate if we wanted to separate the brain tissue (white and gray) from the ventricles.

## 4.2. Region Growing

**Region Growing** is a segmentation technique that involves selecting a seed pixel (or a group of pixels) within an area of interest in an image, and then iteratively comparing and adding contiguous pixels to the region of interest (ROI) based on a homogeneity condition. This process is repeated across the entire image to segment objects or regions with similar characteristics. The key aspects of region growing are as follows:

- **Seed Selection**: The process begins by selecting a seed pixel or group of pixels, which can be chosen interactively by a user or automatically by algorithms.
- **Contiguity Comparison**: In each step, the algorithm compares the characteristics (e.g., intensity values) of contiguous pixels to the seed pixel. If these pixels meet a specified homogeneity condition, they are added to the ROI. The homogeneity condition is often defined based on intensity similarity or other relevant image properties.
- **Connectivity Consideration**: To facilitate region growing, it is essential to define the connectivity or neighbourhood relationships between pixels. This defines how pixels are considered in relation to one another during the comparison process.
  - **Connectivity of 4**: When considering only the vertical and horizontal directions with respect to one pixel, it is referred to as a neighbourhood of 4. This means that a pixel is compared to its top, bottom, left, and right neighbours.
  - **Connectivity of 8**: If the initial pixel is compared with all its adjacent pixels, it is known as a neighbourhood of 8. In this case, a pixel is compared to its top, bottom, left, right, and diagonal neighbours.
- **Impact of Connectivity**: The choice of connectivity (4-connectivity or 8-connectivity) can significantly influence the final segmentation result. A neighbourhood of 4 is more restrictive because it only considers horizontal and vertical relationships, while a neighbourhood of 8 is less restrictive as it includes diagonal comparisons.
- **Spatial Restriction**: Smaller neighbourhood definitions lead to more spatially restrictive growth of regions. Using a neighbourhood of 4 limits the region's expansion to adjacent pixels in the vertical and horizontal directions, while a neighbourhood of 8 allows for broader region growth, including diagonal directions.

First of all, the seed for each image is selected following a trial-and-error procedure. For the brain MRI the seed will be (155, 220), and for the heart MRI (150, 290). An important note is that, for representing it in the x-y plane of the image, the seed should be inverted, having the brain seed on the point (220, 155) and the heart seed on (290, 150). This is because of the difference in calling pixel coordinates and cardinal coordinates.

Then, in order to apply this segmentation technique, the optimal tolerance for the images used should be found. Thus, the personalized function '*find_tol*', which is explained in *Annex 4*, can be implemented.

The tolerance specifies the allowed difference in intensity or color between the starting pixel (seed) and the neighboring pixels for them to be included in the filled region. The results of this function, also shown in *Annex 4*, show that this parameter, when increasing include a wider range of similar colors or intensities, while decreasing, the range of similar colors or intensities will also diminish.

It is also worth mentioning that for the brain MRI there is a tolerance where all the brain tissue is selected as ROI, while for the heart MRI that same value will only cover the ROI selected (see Figure… in *Annex 4*). This is due to the difference in intensities between the ROI of the brain

MRI, which is the tumor, with the boundary regions, and the ventricles in the heart MRI with the surrounding, being much greater the difference between intensities in the latter image, and much smaller in the former. Thus, the region will easily grow in the brain MRI, while it will be much more difficult to overpass the sharp edges of the heart MRI.

Therefore, it can be said that for the heart MRI, region growing is an adequate segmentation technique, outperforming the results obtained with the thresholding process for this image. The results shown in *Annex 4* and *Annex 5* support this statement.

After finding an optimal range for the tolerance for each image, the ROI is computed, with the personalized function *'region_growing'* explained in *Annex 5*. Showing the results obtained for the brain MRI. In the case of the brain, the optimal range for the tolerance was between 25 and 35, taking at the end a tolerance of 30.

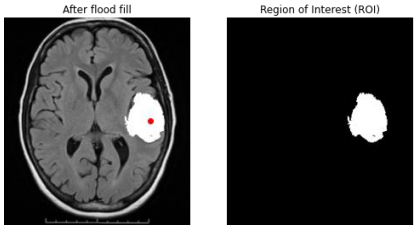


*Figure 6: Brain MRI after flood fill with its respective ROI computed.*

Comparing the previous results to those obtained after applying the same function to one of its previous transformations, the median filter image, with a lower tolerance, 25.

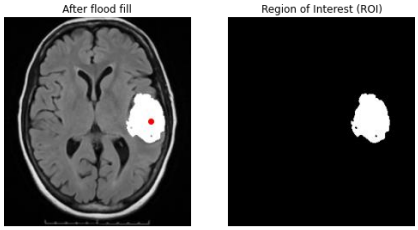


*Figure 7: Median filtered brain MRI after flood fill with its respective ROI computed.*

A completely different thing happened when applying region growing than when applying thresholding. First, the algorithm applied to the original brain MRI image give impressive results, something that did not happen in the previous section. Second, results obtained from the original image and its median filtered one are very similar, almost equal.

Hence, it will be preferable to use the original image in order to preserve the original features, retaining the complete and unaltered information about the objects and structures, as region growing aims to identify regions based on this.

Additionally, there would not be any additional filtering or smoothing artifact, which disrupt the connectivity between pixels, making this technique less effective. As it does not have noise, the original image (not in a real case) will also preserve better the edges, even though the median filter has the ability to preserve them, make. Furthermore, it will simplify the workflow, as any preprocessing step (no in a real case) is needed.

The same happens for the heart MRI, having alike, almost identical results, when applying region growing original image and its median filtered image.

## 4.3. Watershed algorithm

Watershed segmentation is an image processing technique used for segmenting objects or regions of interest. It is particularly useful when the object of interest has well-defined boundaries or when there is a significant contrast between the object and the background. In concrete, watershed segmentation algorithm is predominantly used in several applications, such as medical image analysis, remote sensing, and object recognition. At this project, we are going to focus on using it at medical image analysis. The functioning of this algorithm can be explained in several steps:

- **Grayscale Conversion:** First, we want our input image to be in grayscale, so that it is simplified, and it is easier to work with it.
- **Image preprocessing:** Before applying watershed, we have to perform some preprocessing to our image, to highlight important features and reduce noise (to avoid over-segmentation). Some preprocessing techniques that can be carried out can be smoothing, edge detection or gradient computation. In our code, we worked with the module of the gradient image.
- **Markers selection:** To guide the algorithm, we need to define markers. Markers are essentially seeds or starting points that indicate the regions you want to segment. There are two types of markers:
  - Foreground markers: Placed on the objects or areas you want to segment.
  - Background markers: Placed on the areas that are not part of the objects of interest, generally the background.

In our case, we just used foreground markers since that was enough to clearly segment our areas of interest.

- **Watershed transform:** Here, local minima and maxima play a really important role. This algorithm assumes that the different objects in the image are represented as if they were a valley. The top part of the valley (which is considered the maximum) is the connection between two local minimums. Hence, the local minimums are used as starting points to reach the maximums. The edges of the object are considered to be the top part of the valleys, while the interior of the object corresponds to the bottom of the valley.
  Therefore, the flooding process starts from each of the local minimums (or seeds if you have used them) and gradually fills up the surrounding area until they merge with adjacent regions by simply connecting local minimums and hence acquiring the edges of the object you are segmenting.
  If you don't define any starting point (by using a seed), the process will be made with all the local minimums of the image, and therefore the output image will tend to be over-segmented.

Watershed segmentation algorithm is a powerful technique for image segmentation, but it might be sensitive to noise and may over-segment an image if it is not used carefully. Proper marker selection and preprocessing are crucial to achieving good results.

### 4.3.1. Watershed applied to a brain MRI image

We have used Watershed algorithm to segment a tumor that is present in a brain image. First of all, we computed the module of the gradient image, so that the edges of our object of interest were easier to identify. We made several attempts with different derivative filters, such as Sobel, Canny, and Laplacian. However, we finally decided to use Sobel filter since it was the one that we saw that it worked the best.

Moreover, we manually decided our starting point. By plotting a red dot using the image coordinates, we identified that (180,175) were the coordinates that worked the best. Therefore, we created a matrix full of 0's with the same shape of our gradient image, that contained a value of 1 just in the coordinate (180,175), which is a point inside the tumor we want to segment. This matrix was set as the markers variable.

Additionally, we used a binary mask with the tumor as white color. The purpose of using the mask is that the seed that is placed inside the tumor grows up until it detects that the region has ended. That is why the mask is binary, to easily detect the end of the region by recognizing an abrupt change of intensity.

By having our three main parameters (gradient image, markers and mask), we just apply the Watershed function of skimage.restoration module. The line of code used once we have the parameters is:

- ***labels_sobel = watershed (sobel, markers=markers_sobel, mask = th1)***

being labels_sobel the output image, containing the segmented object in white, and the rest as background (black).

After computing our output image, we plotted the gradient images (although we just used Sobel one), the mask with the seed and the output segmented image as shown in ***Figure 8***:

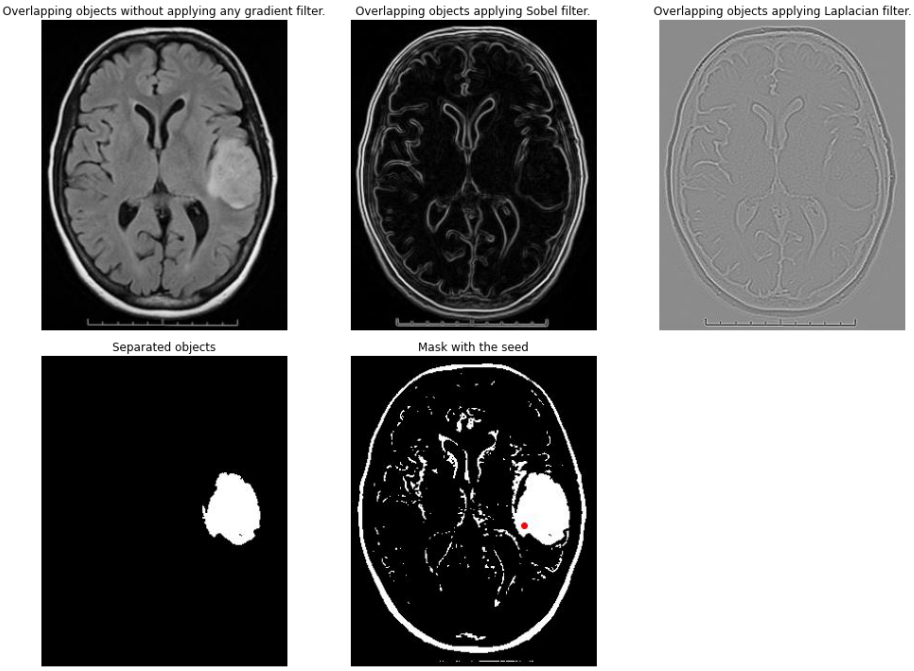


*Figure 8. Gradient images, segmented image and the mask containing the seed.*

## 4.3.2. Watershed applied to a heart MRI image

We have used Watershed algorithm to segment the left ventricle, the aorta, and the left atrium in a heart MRI image.

First of all, we computed the module of the gradient image, so that the edges of our object of interest were easier to identify. We made several attempts with different derivative filters, such as Sobel, Canny, and Laplacian. However, we finally decided to use Sobel filter since it was the one that we saw that it worked the best.

Moreover, we manually decided our starting point. By plotting a red dot using the image coordinates, we identified that (200,200) were the coordinates that worked the best and were inside of our object of interest. Therefore, we created a matrix full of 0's with the same shape of our gradient image, that contained a value of 1 just in the coordinate (200,200), which is a point inside our area of interest. This matrix was set as the "markers" variable.

Additionally, we used a binary mask with our segmentation objective as white color. The purpose of using the mask is that the seed that is placed inside the area of interest grows up until it detects that the region has ended. That is why the mask is binary, to easily detect the end of the region by recognizing an abrupt change of intensity.

By having our three main parameters (gradient image, markers and mask), we just apply the Watershed function of skimage.restoration module. The line of code used once we have the parameters is:

- ***labels_sobel = watershed (sobel, markers=markers_sobel, mask = th1)***

being labels_sobel the output image, containing the segmented object in white, and the rest as background (black).

After computing our output image, we plotted the gradient images (although we just used Sobel one), the mask with the seed and the output segmented image as shown in ***Figure 9***.

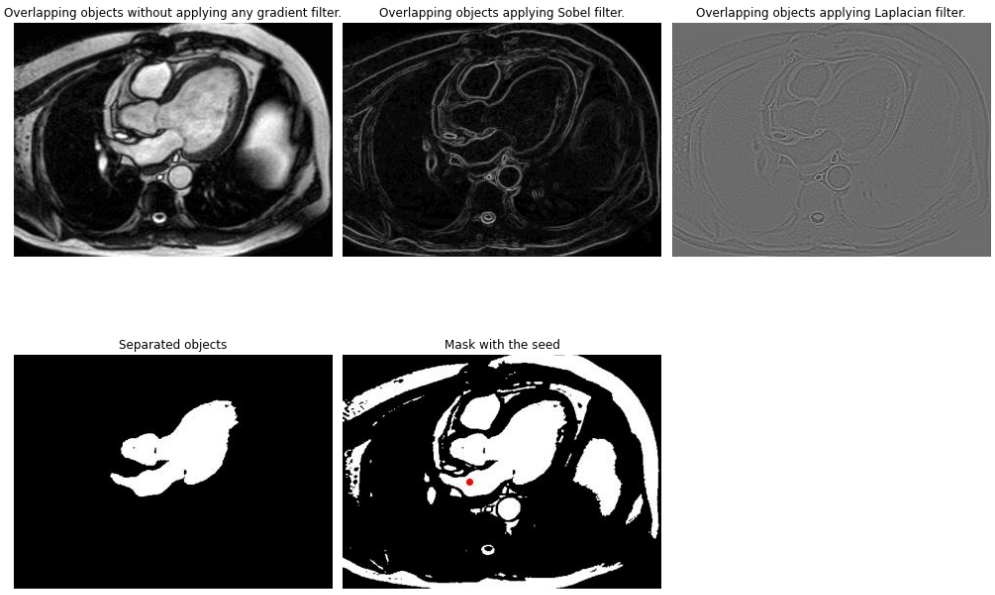


*Figure 9. Gradient images, segmented image and the mask containing the seed.*

# 5. Active Contours

In general, to solve situations in which discontinuous areas and contour points that do not belong appear on the object boundary (due to noise) by applying derivative masks for edge detection, contour tracking is used. The general implementation method will consist of:

a. First implementing a **smoothing filter**, for instance Gaussian, to increase the effectiveness of the segmentation. As read in the literature concerning this algorithm it is advised to smooth images before analyzing, solving for unwanted contour points that may appear due to noise.
b. Then applying **contour tracking**: it is an iterative process that consists of fitting an initial curve to the contour of the region of interest until a complete boundary is achieved.

In addition to this, if found necessary, a **background minimization** will be applied to the image, which consists of a series of morphological operations which intend to minimize the regions that are not of interest (background or non-tumorous regions).

Going into more detail regarding the operation of the snake's algorithm, its fundamental idea is to deform a curve over multiple iterations to fit the desired boundary that correctly defines the ROI within an image, by minimizing an energy function partly defined by the image and partly by the contour shape. Here are the steps:

1. First **initialize a curve/contour within the image** given an approximate boundary around the ROI. The initial contour may be user-defined (manually) or automatically generated.
2. **Minimize an energy function associated with the contour:** this energy is a combination of internal energy (smoothness) and external energy (attraction toward the object's boundaries). The energy function is mathematically defined to encourage the contour to fit the object's boundary.
3. **Deformation:** the contour evolves or deforms iteratively in the direction that minimizes the energy function, so that it is near pixels with high gradient (edges). Typically, gradient descent is used to update the contour. This means the contour is shifted in the direction that minimizes the energy, trying to fit the boundary better at each iteration.
4. **Convergence:** the algorithm iterates until the contour converges to the desired boundary or until a predefined stopping criterion is met.

## 5.1. Active Contour on Heart MRI image

For the heart MRI image, the **left ventricle**, the **left atrium,** and the **aorta** will be the ROI to segment. Due to the complexity of these regions (there are other anatomical structures within the splines of the desired contour to obtain and presents sharps turns) the implementation of the snake's algorithm will be done with a **manually defined initialization curve**. This consists of a ordered list of manually set 2D vertices connected by a line, that roughly describe the contour that is intended to be obtained after applying the snake's algorithm. By doing so a little assistance is provided to the algorithm, because by itself it does not solve the previously mentioned issues.

| Parameter Name | Parameter Value | Comment |
|---|---|---|
| alpha | 0.001 | Elasticity parameter (small for flexibility) |

| | | |
|---|---|---|
| beta | 10 | Rigidity parameter (large for edge following) |
| gamma | 0.001 | Step size for contour evolution |
| max_px_move | 0.5 | Maximum snake movement allowed per iteration |
| w_line | -10 | Weight of internal energy term (favors shorter contours) |
| w_edge | 2 | Weight of external energy term (favors edge following) |
| convergence | 5 | Convergence threshold |

The image is smoothed to reduce any noticeable noise and make edges more distinct with Gaussian filter. This step is repeated in every implementation of the snake's algorithm that will be shown in this report.

The 'alpha' parameter is set to a small value (0.001), making the snake relatively elastic. The 'beta' parameter is set to a relatively large value (10), making the snake favor smoothness, and closely following the edges in the image. The 'gamma' parameter is set to a small value (0.001), which controls the step size for contour evolution. Smaller values result in slower convergence. The 'max_px_move' at 0.5, restricts how much the snake can change between iterations. The 'w_line' parameter is set to -10, which means that the contour is attracted toward dark regions (we want to fit the dark contour of the ROI instead of jumping to other close bright objects). The 'w_edge' parameter is set to 2, indicating that the external energy term encourages the snake to adhere more closely to image edges.

With these parameter values the snake will be relatively elastic, preferring shorter contours, and it will closely follow the edges while making small, controlled movements. The algorithm will continue this process until convergence or until the maximum number of iterations is reached, producing a contour that outlines the object or region of interest in the image.

This final contour can be seen in *Figure 27*. The curve painted in red represents the initialized curve, and the curve painted in blue curve is the final deformation of the initial curve resulting from fitting it to the ROI (this will be the same for all pictures in the section):

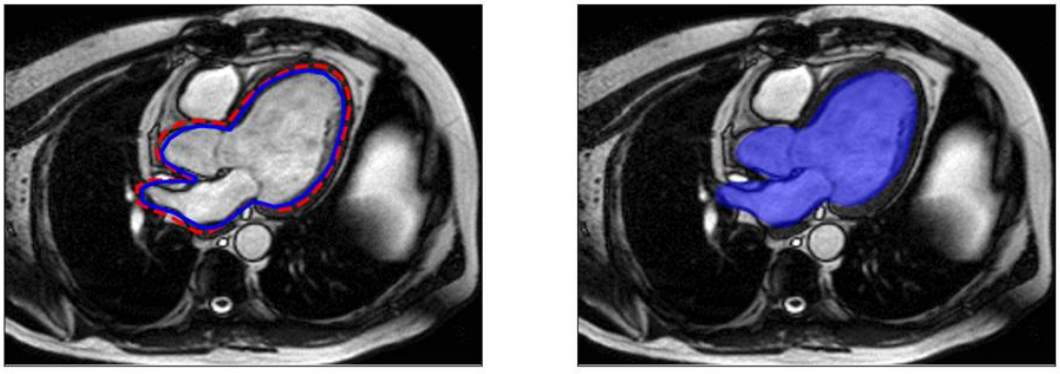

*Figure 10. Segmentation with Active Contour Snakes Algorithm (Manual Initialized Curve) on a Heart MRI image.*

The result obtained is quite decent but there can be observed parts of the final contour that are still attracted to other elements in the vicinities. Plus, in this case the left atria and the aorta are very close together but not joined, the algorithm is not able to differentiate this. In fact, different manually selected curves proved that the results strongly depend on the curve that is set, hence why the algorithm does not reach far into the spline between the aorta and the left atria.

Refer to *Annex 7, 8* and *9* to find a comparison of this method with other more elaborated methods, aimed to solve the deficiencies of applying a manual curve, as well as with a simpler initialization curve method for the sake of contrast.

## 5.2. Active contour on Brain MRI Image

For the brain MRI image, the goal is to segment the tumor on the right-hand side, which is a simpler and smoother region to segment and with brighter pixels than surrounding regions (this makes sense because tumors have increased cell density and hence higher proton density) which favors an effective performance of the algorithm.

From previous trials in the Heart MRI image, it has been concluded that with a simple initialization curve, such as the circle, placed around the tumorous region, the algorithm will do an acceptable segmentation (semi-automatic initialization). However, because the tumor is located on the edge of the brain, it is important to account for the bright edge corresponding to the bone tissue of the skull, since this will affect the fitting of the contour (fits to high gradient pixels). To do this, background minimization will be the solution.

As it can be seen the resulting background eliminated image will be one without the borders from the skull and with blurriness. This blurriness will not affect the algorithm, its fact it will be helpful for it since it gets rid of gradient differences due to noise. However, the algorithm still needs some tuning of the parameters to achieve the best results.

| Parameter Name | Parameter Value | Comment |
|---|---|---|
| alpha | 0.01 | Elasticity parameter (small for flexibility) |
| beta | 10 | Rigidity parameter (large for edge following) |
| gamma | 0.01 | Step size for contour evolution |
| max_px_move | *1* | Maximum snake movement allowed per iteration |
| w_line | 0.14 | Weight of internal energy term (favors shorter contours) |
| w_edge | *1* | Weight of external energy term (favors edge following) |

| | | |
|---|---|---|
| convergence | 0.13 | Convergence threshold |

After gaussian smoothing of the image the algorithm is implemented with the above parameters. These are not very aggressive. With an 'alpha' value of 0.01, which makes the snake more rigid that the other methods. Same with 'gamma' (0.01), 'max_px_move' (1) and 'w_edge' (1) which make the snake converge faster, with bigger steps and adhere to edges with a positive 'w_line' (0.14) that attracts the snake to bright areas.

For these parameters the following segmentation of a brain tumor is obtained:

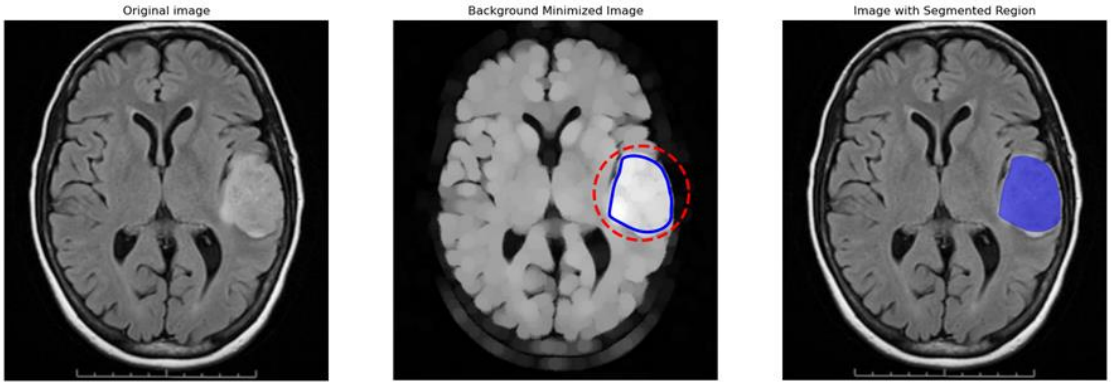


*Figure 11. Snakes Segmentation with Semi-Automatic Initialization Curve of a brain tumor in MRI image.*

As expected from the trials made in *Annex 7, 8, and 9* the segmented region of the brain tumor obtained with background minimization and a semi-automatic initialization curve is very satisfactory without the need of extremely complex initial curves or intensive parameter tuning.

# 6. Felzenszwalb Segmentation Technique

**Felzenszwalb Segmentation:** The Felzenszwalb segmentation algorithm aims to partition an image into meaningful regions by detecting abrupt changes in pixel intensity values. This technique is used to delineate objects or regions of interest within the image.

1. **Similarity-Based Segmentation:** The Felzenszwalb method operates on the principle that pixels with similar intensity values are likely to belong to the same region. It identifies these regions by considering intensity differences between neighbouring pixels.
2. **Scale Parameter (scale):** One of the central components of the algorithm is the scale parameter. This parameter governs the trade-off between the size and regularity of segments. Smaller values produce more irregularly shaped and smaller segments, while larger values yield smoother and larger regions. Choosing an appropriate scale is crucial for achieving the desired segmentation result.
3. **Gaussian Smoothing (sigma):** Before segmentation, the image is smoothed using Gaussian filtering with the sigma parameter. Smoothing helps reduce sensitivity to noise and minor intensity fluctuations, leading to more robust segmentation results.
4. **Minimum Segment Size (min_size):** The algorithm incorporates a min_size parameter, which defines the minimum number of pixels that a region must contain. Smaller regions are merged into neighbouring regions to maintain a meaningful segmentation. This parameter can influence the level of detail in the segmentation.

**Advantages:**

- **Scale Flexibility:** The Felzenszwalb algorithm can capture regions of various sizes and shapes, which makes it adaptable to diverse images with objects of different scales.
- **Automated Segmentation:** The algorithm does not rely on user-provided seeds or thresholds. It can operate autonomously, which is particularly advantageous for large datasets.
- **Broad Applicability:** The method is applicable to a wide range of image types, including medical images, where it can be used to segment anatomical structures or regions of interest.

**Disadvantages:**

- **Segmentation Sensitivity:** The choice of scale and sigma parameters significantly influences the segmentation result. Finding the optimal parameter values can be challenging and may require experimentation.
- **Shape Variability:** When using smaller scale values, the algorithm can produce irregularly shaped segments, which might not be suitable for all applications.
- **Sensitivity to Image Conditions:** The algorithm may still be sensitive to variations in image acquisition conditions, such as lighting and contrast.

First of all, the personalized function *'visualize_felzenszwalb'* is implemented for displaying the segmented image with different scale values, having the possibility to display the original segmentation or the boundaries computed by the *Felzenszwalb* algorithm. It is explained in *Annex 10*.

The scale controls the approximate size of the segments. A smaller scale leads to smaller segments and thus more in quantity, and a larger scale results in larger segments and thus less in quantity, as seen in the results obtained, shown in *Annex 10*.

However, as seen in the results shown in *Annex 10*, this segmentation technique is not able to segment the ROI established at the beginning of the project for each image (tumor for the brain MRI and the ventricles for the heart MRI).

Nonetheless, for the brain MRI, even though it seems that for *k = 150* and *option = 1* the result appears to be decent, when computing the boundaries with *option = 2*, it is seen that there exist some discontinuities for the computed edges of the corresponding ROI, the tumor.

Thus, Felzenszwalb segmentation is a popular image segmentation technique that is primarily designed for general-purpose image segmentation. Even though it is based on a bottom-up approach that groups similar pixels into segments, this segmentation technique is not typically used for detecting brain tumors directly or for medical images. It is more commonly used for tasks like object recognition, scene analysis, and image parsing.

However, in order to improve the implementation of the Felzenszwalb algorithm, and making this algorithm suitable for this specific task, the importance of preprocessing the images before the segmentation is recalled. Hence two methods can be applied (both explained above).

- Apply a mask similar to the one implemented in the active contours section, enabling a better segmentation, but following the Felzenszwalb technique. This will enable this algorithm to focus more on the region to be segmented.
- Apply a median filter, in order to smooth the image and thus make the difference between regions inside the edges increase. This will make the algorithm work better by making easier for him to differentiate regions.

The results, for the brain MRI, extremely improved as shown below.

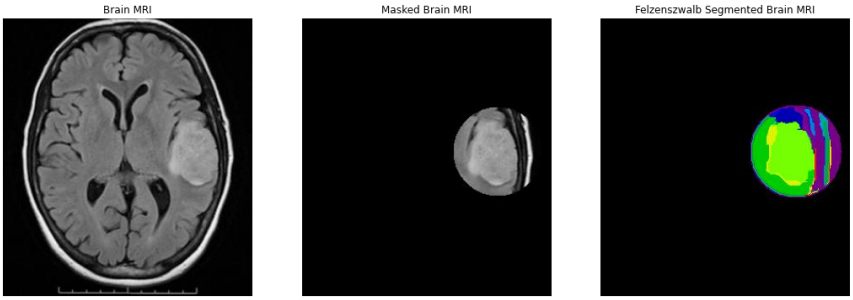


*Figure 12: Felzenszwalb segmentation after applying a mask to the brain MRI.*

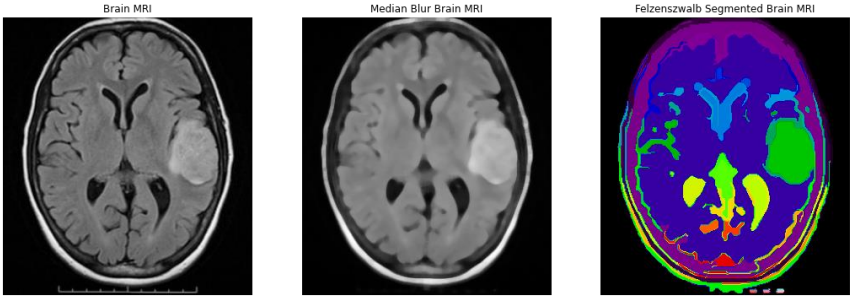


*Figure 13: Felzenszwalb segmentation after applying a median filter to the brain MRI.*

However, for the heart MRI, the results still to be mediocre applying both preprocessing steps separately (shown in *Annex 10*). Nonetheless, if they are applied together, an improved result is also obtained.

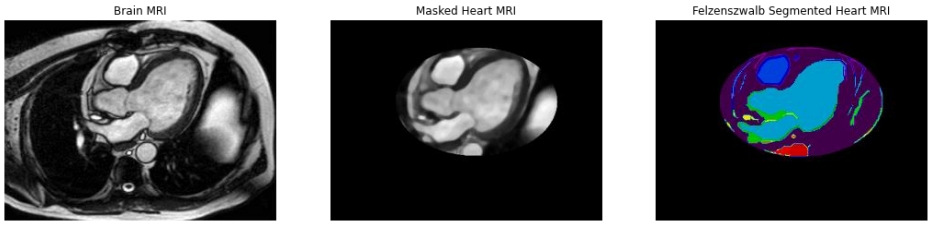


*Figure 14: Felzenszwalb segmentation after applying a mask and a median filter to the heart MRI.*

# 7. Conclusions

There is not one segmentation technique that segments every possible region because different anatomical regions have vastly different shapes, edges, characteristics, and overall morphology.

We have arrived at the conclusion that a great part of segmentation is feeding a properly transformed image to the algorithm depending on the goal of said segmentation. This will increase the effectiveness of the segmentation method applied. Also, applying several techniques (complementing each other) to the same image would generally provide better results, which consequently improves the quality and interpretability of medical image structures for accurate diagnosis, treatment planning, and medical research.

# 8. References

[1] https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html

[2] https://scikit-image.org/docs/stable/api/skimage.segmentation.html

[3] https://www.sciencedirect.com/science/article/pii/S1319157818313120

# 9. Annexes

## Annex 1: Table explaining the Personalized Function 'simple_thres' and its results for both images.

Take into account that the arguments presented in bold, are not only the ones corresponding to the predefined function of, in this case, the cv2 dependency, but also to the personalized one that was created in this practice. The parameters that do not appear in bold belong only to the predefined function of the corresponding dependency. This holds for every table presented in the report.

*Table 1: Personalized function 'simple_thres'.*

| **def simple_thresh(img, th):** | | |
|---|---|---|
| cv2.threshold | **img** | Source image, which should be a grayscale image. |
| | **th** | Threshold value which is used to classify the pixel values. |
| | 255 | Maximum value which is assigned to pixel values exceeding the threshold. |
| | Type of thresholding | See *Figure 3* for understanding. |
| Loop through the images to display them. | Subplot 1 | Original Image. |
| | Subplot 2 | Binary Thresholding. |
| | Subplot 3 | Negative Binary Thresholding. |
| | Subplot 4 | Truncated Thresholding. |
| | Subplot 5 | Tozero Thresholding. |
| | Subplot 6 | Negative Tozero Thresholding. |
| Output: Displays 6 images in a 2x3 grid corresponding to all the types of thresholding seen in *Figure 3* and the original image. | | |

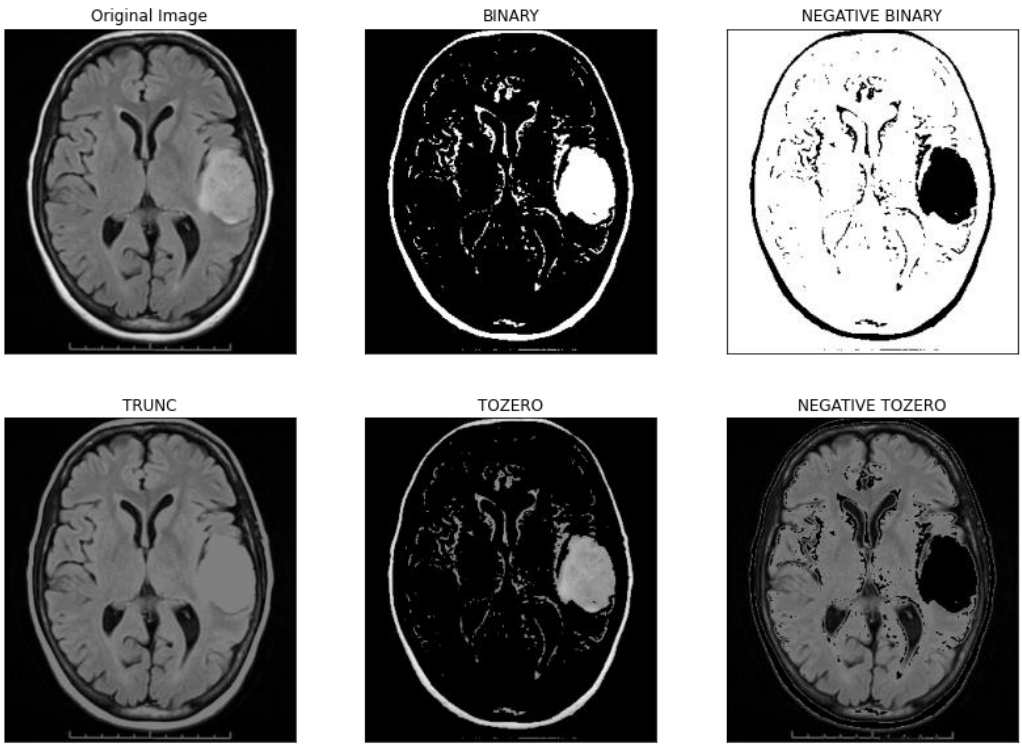


*Figure 15: Brain MRI image and its thresholding with the different types given by the OpenCV (cv2) dependency.*

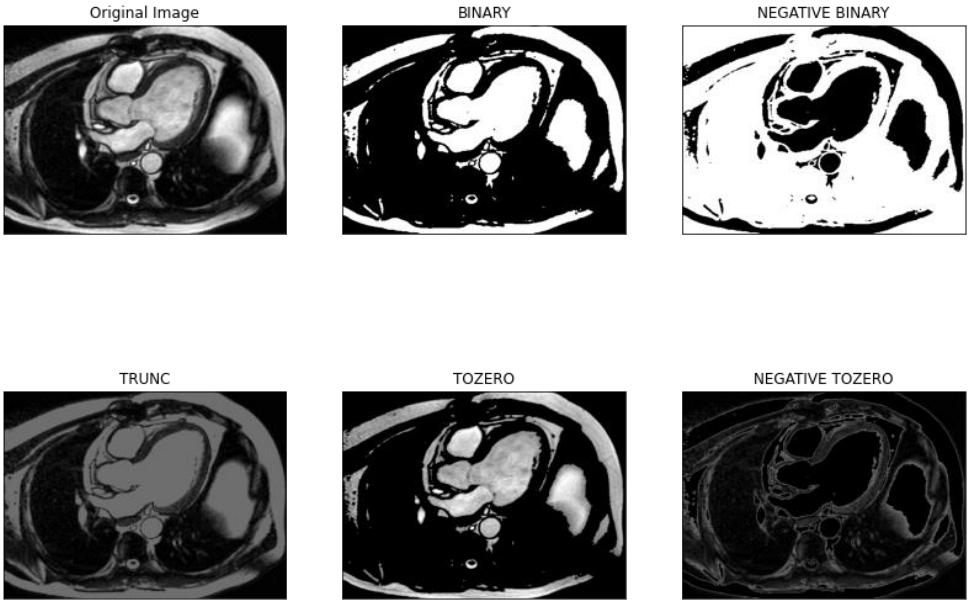


*Figure 16: Heart MRI image and its thresholding with the different types given by the OpenCV (cv2) dependency.*

## Annex 2: Table explaining the Personalized Function 'thresh_comp' and its results for Heart MRI and its Transformations.

*Table 2: Personalized function 'thresh_comp'.*

<table>
<tr><td colspan="3">def thresh_comp(img, img_eq, th_img, th_blur, th_eq):</td></tr>
<tr><td rowspan="4">cv2.threshold:<br>- Original Image.<br>- Median Blur Image.<br>- Equalized Image.</td><td>img<br>blur<br>img_eq</td><td rowspan="3">Previously explained in Table 1.<br>Three images, original and equalized given as parameters and blur computed inside, with their respective thresholds (as parameters).</td></tr>
<tr><td>th_img<br>th_blur<br>th_eq</td></tr>
<tr><td>255</td></tr>
<tr><td>Type of thresholding</td><td>See Figure 1 for understanding.</td></tr>
<tr><td rowspan="3">Loop through the images to display them.</td><td>Subplot 1</td><td>Image to be thresholded.</td></tr>
<tr><td>Subplot 2</td><td>Histogram with the threshold used.</td></tr>
<tr><td>Subplot 3</td><td>Thresholded image.</td></tr>
<tr><td colspan="3">Output: Displays the original image and the threshold image as well as the histogram with the threshold used for each of the transformation in a 3x3 grid.</td></tr>
</table>

It can be seen in the figure below that for the original image a threshold of 110, for the median filtered image a threshold of 127, and for the histogram equalized image a threshold of 175.

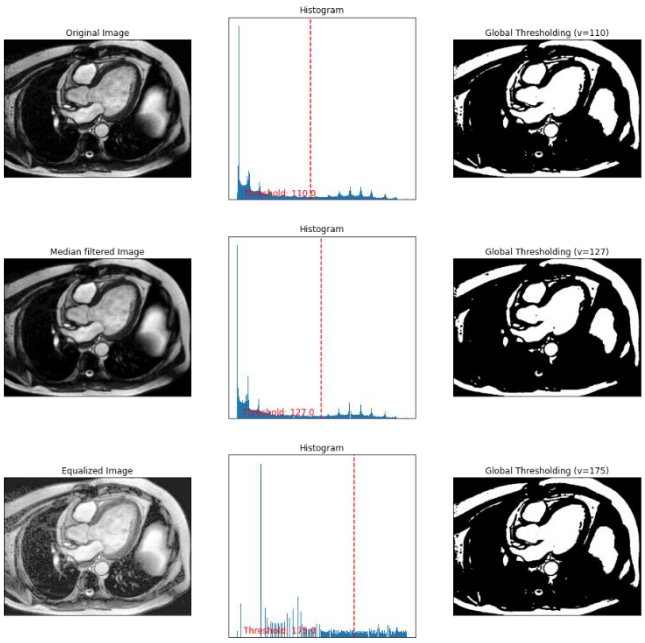


*Figure 17: The original heart MRI image and its transformations, median filter, and histogram equalization, with their respective thresholded result and the histogram with the threshold use.*

## Annex 3: Table explaining the Personalized Function 'thresh_comp' and its results for Heart MRI and its Transformations.

*Table 3: Personalized function 'otsu_thresh'.*

| **def otsu_thresh(img, img_eq, th, dependency) :** | | |
|---|---|---|
| cv2.threshold | **Img**<br>**img_eq** | Source or equalized image, which should be a grayscale image. |
| | **th_img** | Threshold value which is used to classify the pixel values. |
| | 255 | Maximum value which is assigned to pixel values exceeding the threshold. |
| | Type of thresholding | See *Figure 3* for understanding. |
| **dependency** == '**cv**' | cv2.threshold | The type used will be:<br>cv.THRESH_BINARY + cv.THRESH_OTSU |
| **dependency** == '**sk**' | threshold_otsu | Computes directly the threshold |
| Loop through the images to display them. | Subplot 1 | Image to be thresholded. |
| | Subplot 2 | Histogram with the threshold used. |
| | Subplot 3 | Thresholded image. |
| Output: Displays the original image and the threshold image as well as the histogram with the threshold used for each of the transformation in a 4x3 grid. | | |

It can be seen in the figure below that the Otsu's thresholding computed a threshold of 97, 97, and 131 for the original image, the median filtered image, and the histogram equalized image respectively.

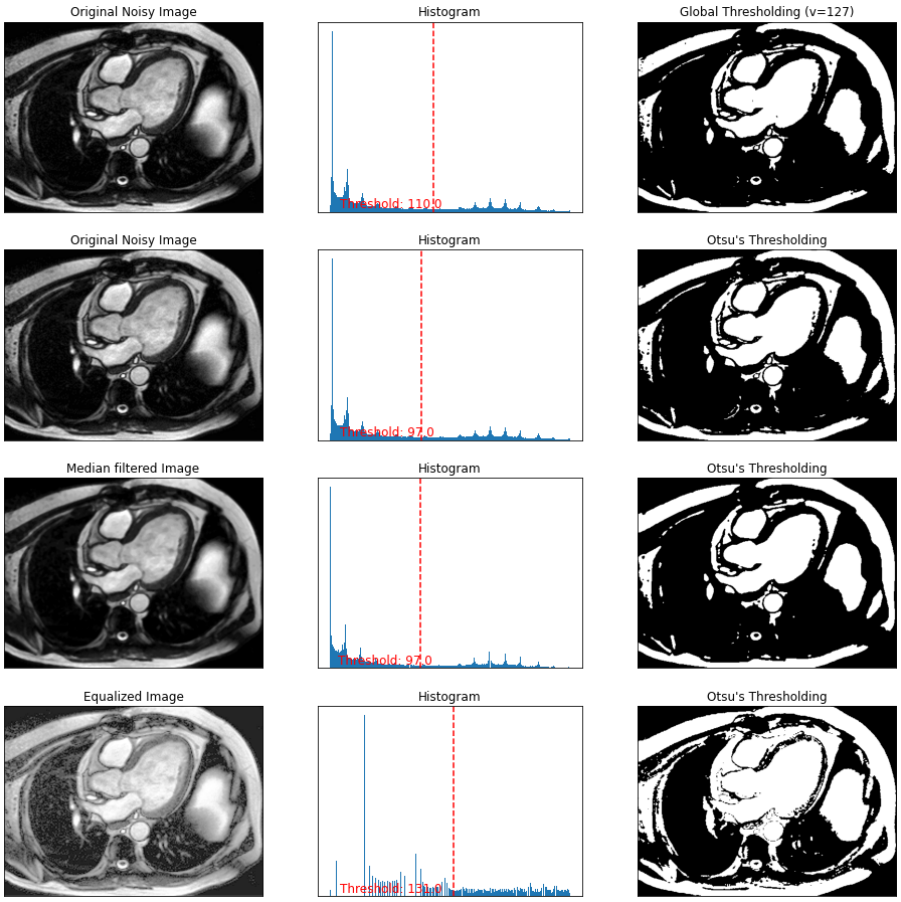


*Figure 18: Otsu´s thresholding applied to the original heart MRI image and its transformations, median filter, and histogram equalization, with their respective thresholded result and the histogram with the threshold use compared to the global thresholding.*

## Annex 4: Table explaining the Personalized Function 'find_tol' and its results for both images.

*Table 4: Personalized function 'find_tol'.*

<table>
<tr><td colspan="4">def find_tol(img, seed):</td></tr>
<tr><td rowspan="4">flood_fill()</td><td>img</td><td colspan="2">The input image as an n-dimensional array.</td></tr>
<tr><td>seed</td><td colspan="2">The point in 'img' used as the starting point for the flood fill. If the image is 1D, this point may be given as an integer.</td></tr>
<tr><td>connectivity=8</td><td colspan="2">A number used to determine the neighborhood of each evaluated pixel. Adjacent pixels whose squared distance from the center is less than or equal to connectivity are considered neighbors.<br>This parameter was not well understood, as it does not have the same meaning as in the theory seen in class. It has been tested using several different values and no change have been seen in any of the images. Thus, it will remain as 8.</td></tr>
<tr><td>tolerance</td><td colspan="2">Specifies the allowed difference in intensity or color between the starting pixel (seed) and the neighboring pixels for them to be included in the filled region. Higher tolerance values include a wider range of similar colors or intensities.<br>Value through which is iterated in order to find the optimal one.</td></tr>
<tr><td colspan="2">1. First loop: Iterate through different tolerance values and applying flood fill.</td><td>2. Initialize a plot with a grid of subplots and place the original image in the top-left subplot</td><td>3. Second Loop: Plot all eight different tolerances for comparison in the remaining subplots.</td></tr>
<tr><td colspan="4">Output: Displays the original image and the threshold image as well as the histogram with the threshold used for each of the transformation in a 4x3 grid.</td></tr>
</table>

Results for the brain MRI.

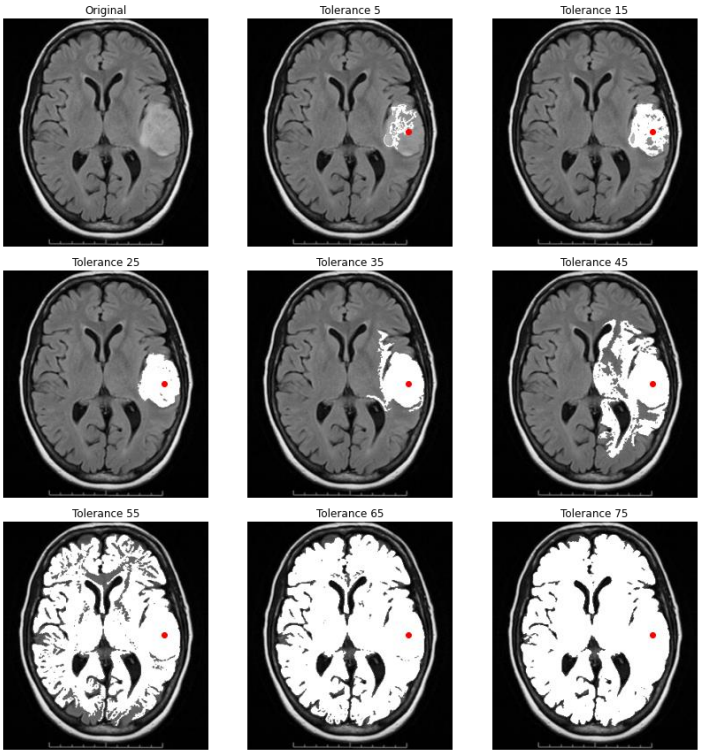


*Figure 19: Region Growing of brain MRI with seed = (155, 220) incrementing the tolerance.*

Results for the heart MRI.

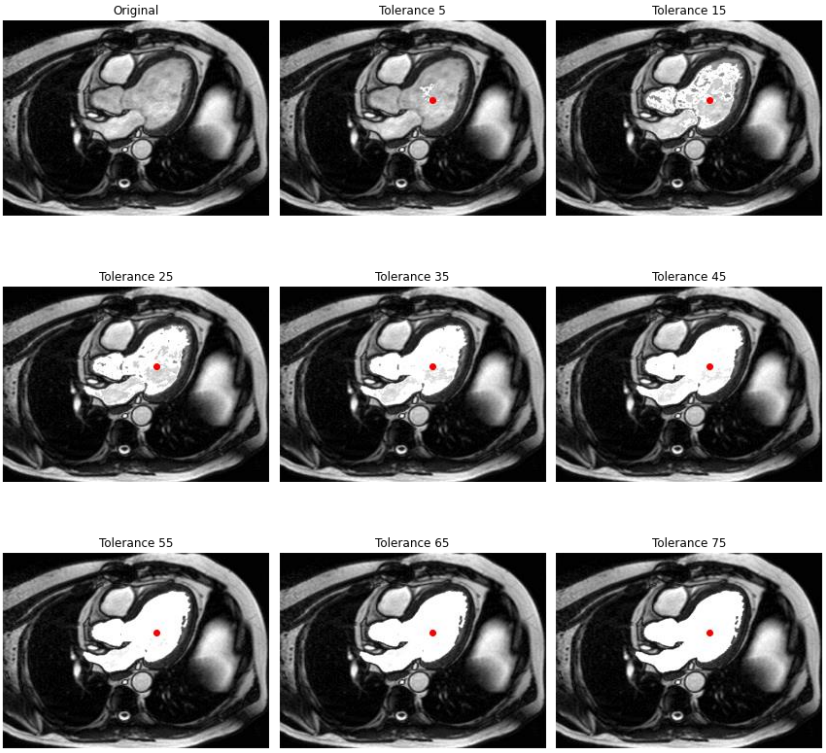


*Figure 20: Region Growing of heart MRI with seed = (150, 290) incrementing the tolerance.*

## Annex 5: Table explaining the Personalized Function 'region_growing' and its results for both images.

*Table 5: Personalized function 'region_growing'.*

| **def region_growing(img, seed, tol, con):** | | |
|---|---|---|
| flood_fill() | **img** | All these parameters were explained above, for the previous personalized function '*find_tol()*'. |
| | **seed** | |
| | **con** | Parameter connectivity for the *flood_fill()* predefined function. |
| | **tol** | Parameter tolerance for the *flood_fill()* predefined function. |
| Plots | Subplot 1 | Original image to be segmented. |
| | Subplot 2 | Image after being segmented. |
| | Subplot 3 | Region of Interest (ROI). |
| | Subplot 4 | Histogram of the image after being segmented. |
| Output: Displays the original image and the threshold image as well as the histogram with the threshold used for each of the transformation in a 4x3 grid. | | |

Results for the brain MRI and its median filtered image:

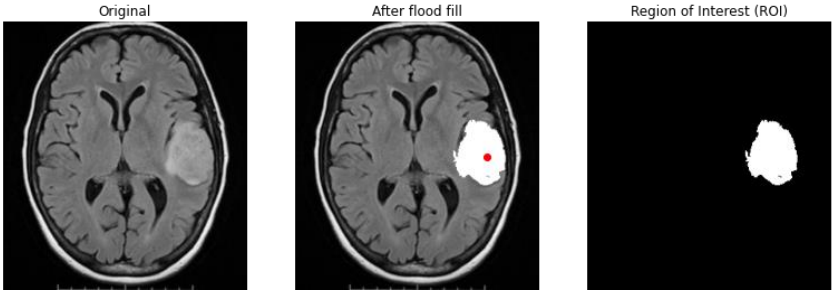


*Figure 21: Original brain MRI image, image after applying flood fill and the ROI computed.*

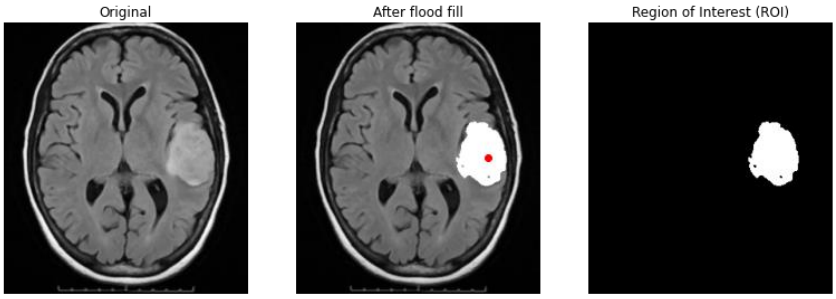


*Figure 22: Median filtered brain MRI image, image after applying flood fill and the ROI computed.*

Results for the heart MRI and its median filtered image:

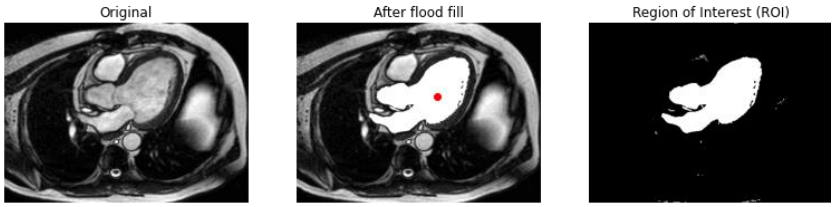


*Figure 23: Original heart MRI image, image after applying flood fill and the ROI computed.*

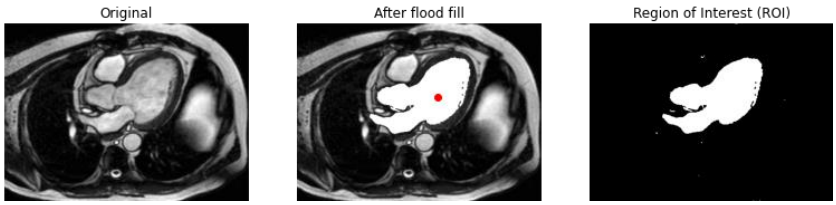


*Figure 24: Original heart MRI image, image after applying flood fill and the ROI computed.*

## Annex 6: Table explaining the predefined function 'active_contour'

A predefined function is used for the implementation of the snake's algorithm. Parameters shown in italics are default parameters of the '*active_contour*' function from the '*skimage.segmentation'* library.

<table>
<tr><td colspan="4">active_contour(gaussian(img, 3, preserve_range=False), init_points, max_num_iter='2500', boundary_condition='periodic')</td></tr>
<tr><td rowspan="11">active contour ()</td><td rowspan="3">gaussian ()</td><td>image</td><td>Input image on which to perform the active contour segmentation</td></tr>
<tr><td>sigma = 3</td><td>Standard deviation of the Gaussian kernel. A larger value of sigma results in a broader kernel, which leads to a smoother, more heavily blurred image. Smaller values of sigma preserve finer details while reducing noise</td></tr>
<tr><td>preserve_range = False</td><td>If True, it preserves the intensity range of the input image. which maintains the original image's contrast.</td></tr>
<tr><td>init_points</td><td colspan="2">Initial contour or snake. It is a NumPy array of shape (N, 2), where N is the number of points in the contour. The initial contour serves as the starting point for the active contour to evolve and fit the object's boundaries.</td></tr>
<tr><td>alpha</td><td colspan="2">Elasticity of the snake. It controls the trade-off between stretching and bending energy. A smaller value makes the snake more flexible, while a larger value makes it more rigid.</td></tr>
<tr><td>beta</td><td colspan="2">Rigidity of the snake. It controls how closely the snake follows the edges of objects in the image. A larger value makes the snake follow edges more closely.</td></tr>
<tr><td>gamma</td><td colspan="2">Step size for the contour evolution. It affects how quickly the snake moves toward its equilibrium position. Smaller values result in slower evolution, while larger values make the snake converge faster.</td></tr>
<tr><td>max_px_move</td><td colspan="2">Sets the maximum pixel movement allowed for the snake at each iteration. It limits the extent of movement to prevent the snake from changing too</td></tr>
</table>

| | | |
|---|---|---|
| | | much between iterations, or too little and not achieving convergence. |
| | max_num_iter = *'2500'* | Defines the maximum number of iterations over which the snake shape will be optimized. The default value will be used. |
| | w_line | Weight of the internal energy term. It influences the snake's preference for shorter or longer contours. If 'w_line' is set to a non-zero value, the snake tends to shrink. |
| | w_edge | Weight of the external (image gradient) energy term. It determines the snake's attraction to image edges. A higher 'w_edge' value makes the snake adhere more closely to object boundaries. |
| | convergence | It is the stopping criteria for the active contour algorithm. The algorithm will stop if the change in the snake's position between iterations falls below this threshold. |
| | boundary_condition = *'periodic'* | Can be set to 'periodic' or 'free'. 'periodic' means that the snake is treated as a closed contour, while 'free' means that it's treated as an open contour. The default value 'periodic' is used. |
| Output: snake contour resulting from fitting the initial curve to the ROI | | |

## Annex 7: Active Contour with Background Optimization & Manual Curve Initialization

To improve the performance of the active contour snake algorithm a technique called **Background Minimization will be applied**. It aims to minimize the information from regions that are not of interest while maintaining the features of the ROI. Background minimization works in the following way. The background is eliminated by using two levels of morphological operations. The first level morphological reconstruction is by erosion while the second level morphological reconstruction is by dilation. Below a more detailed explanation is provided.

**Opening** is primarily used to remove noise, small objects, and fine details while preserving the larger structures or objects in an image.

- **Erosion** is the first step of the opening operation. It involves moving a small structuring element (a small binary kernel, in our case a disk will be used) across the image. At each position of the structuring element, the center of the element is aligned with the corresponding pixel in the image. If all the pixels in the structuring element coincide with foreground pixels (non-zero) in the image, the center pixel of the structuring element is set to the foreground value. If any of the pixels in the structuring element overlap with background pixels (zero) in the image, the center pixel of the structuring element is set to the background value. The result of erosion is an image where foreground regions are shrunk or eroded away, and noise or small features are removed.
- **Dilation** is the second step of the opening operation. It involves moving the same structuring element across the eroded image. Similar to erosion, the structuring element is aligned with the image's pixels. If any part of the structuring element overlaps with a foreground pixel in the image, the center pixel of the structuring element is set to the foreground value. If all parts of the structuring element overlap with background pixels, the center pixel of the structuring element is set to the background value. The result of dilation is an image where the remaining foreground regions expand, and small gaps or holes within the objects are filled.

The combination of erosion followed by dilation in morphological opening results in the **preservation of the larger connected structures and softens the contours of objects** hence why it is very useful for segmentation.

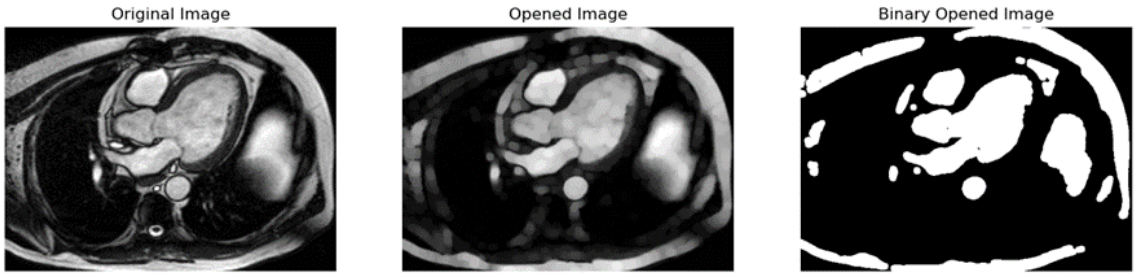


*Figure 25. Background Elimination by Opening Morphological Operation of the Heart MRI.*

On the 'Opened Image' image seen in the subplot from *Figure 28* it can be seen how information from the background is minimized while maintaining the features of the ROI. Then, the binary image is plotted to show what the results of applying the morphological operation are. Now the regions that are close to our ROI appear further away from it.

| Parameter Name | Parameter Value | Comment |
|---|---|---|
| alpha | 0.001 | Elasticity parameter (small for flexibility) |

| | | |
|---|---|---|
| beta | 10 | Rigidity parameter (large for edge following) |
| gamma | 0.001 | Step size for contour evolution |
| max_px_move | 0.5 | Maximum snake movement allowed per iteration |
| w_line | -5 | Weight of internal energy term (favors shorter contours) |
| w_edge | 10 | Weight of external energy term (favors edge following) |
| Convergence | 2 | Convergence threshold |

The image is first smoothed with a gaussian filter.

A small 'alpha' value (0.001) means the snake is relatively elastic, allowing it to deform to the image's features more flexibly. The 'beta' parameter, with a value of 10, heavily enforces smoothness and edge-following behavior. A small 'gamma' parameter (0.001) results in smaller steps and slower convergence. The 'max_px_move' is limited to 0.5 pixels, ensuring that the snake's movement is controlled. The 'w_line', with a value of -5, encourages the snake to expand slightly, and attract to dark regions. The 'w_edge' (10), strongly encourages the snake to adhere closely to image edges. The algorithm stops when the change in the snake's position between iterations falls below the 'convergence' threshold, which is set to 2 in this case.

The balance between flexibility (controlled by 'alpha'), smoothness (controlled by 'beta'), edge following (controlled by 'w_edge' and 'w_line'), and the step size (controlled by 'gamma') influences how the snake adapts to the object's boundaries while minimizing the energy function. The result can be observed in *Figure 29*:

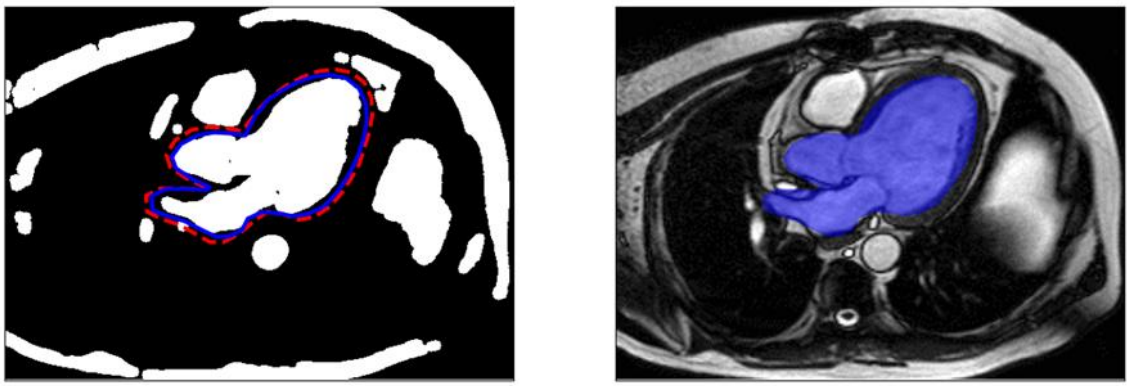

*Figure 26. Snakes Algorithm Applied to a Background Minimized Image of the Heart MRI.*

Similar results are obtained compared to the manually initialized curve without background minimization. This is probably because the initial curve is already very fit to the actual region that is to be segmented, but for the results explained results should be improved.

## Annex 8: Active Contour with Background Optimization & Semi-Automatic Curve Initialization

In this section, the results from previous active contour methods will be compared with the ones obtained from applying background minimization and a **semi-automatic** method of initializing the curve for the algorithm. This method is faster than manually selecting points and it consists of encapsulating the ROI with a pre-defined function, like a circle or in this case a rotated **ellipse**. After some parameter tuning these would be the best results of the segmentation obtained.

| Parameter Name | Parameter Value | Comment |
|---|---|---|
| alpha | 0.1 | Elasticity parameter (small for flexibility) |
| beta | 10 | Rigidity parameter (large for edge following) |
| gamma | 0.001 | Step size for contour evolution |
| max_px_move | 0.4 | Maximum snake movement allowed per iteration |
| w_line | -5 | Weight of internal energy term (favors shorter contours) |
| w_edge | 8 | Weight of external energy term (favors edge following) |
| convergence | 0.15 | Convergence threshold |

Gaussian smoothing prior to implementing the algorithm is done. Then regarding the parameter values: with a value of 0.1 on the 'alpha' parameter, the snake has moderate rigidity. The 'beta' with a value of 10 strongly favors smoothness and edge-following behavior. The w_line, with a value of -5, encourages the snake adherence to dark areas. The 'w_edge' with a value of 8, provides a strong edge-following incentive. The algorithm will stop when the change in the snake's position between iterations falls below this threshold, set at 0.15 in this case. With a small value (0.001), the 'gamma' helps ensure a stable and accurate convergence of the snake.

The combination of the parameters is clearly different than the one from other active contour methods, this because now, the intention is to have more aggressive fitting steps to hopefully reduce the effect of closely surrounding elements and increase adherence to complex splines.

For these parameters the following segmentation is obtained:

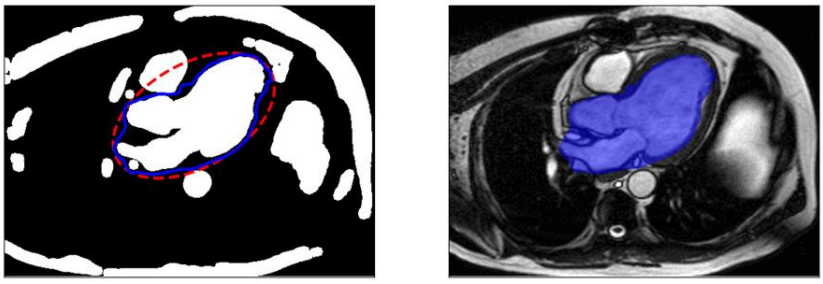

*Figure 27. . Snakes Algorithm with a Semi-Automatic Initialization Curved on a Background Minimized Heart MRI image.*

By using an angled ellipse as initialization curve two things can be observed:

1. The algorithm does a much better job at fitting the initial contour to the edges of the ROI when they are smoother, probably due to the smooth nature of the ellipse.
2. The algorithm performs worse than other methods when it has to fit the initialized curved to regions with complex splines and other structures in the middle of said splines. Plus, the curve also tends to fit other bright regions on the periphery of the ROI. In this respect, providing a manually selected region is a better alternative.

Now it can be properly appreciated the increase in effectiveness thanks to the background minimization.

Let's recap on the different snakes' algorithm applications, first, applying a manual initialization curve provided the best result yet (although this could be improved), but it strongly depended on the region set by the operator, and it is very time consuming and inefficient when segmenting several images. Then it has been shown how applying background minimization got rid of elements inside splines and separates other bright elements surrounding the ROI. It also assisted the algorithm in the minimization of the energy, since the pixel gradient is greater at the ROI, and it was easier for it to adjust to its edges and shape. Finally, by applying a semi-automatic method for the initialization curve, even though faster process, the algorithm was affected by surrounding structures, and it was not able to deform the initial curve to fit deep and sharp splines.

## Annex 9: Region Growing and Previous Active Contour Methods

With the conclusions from all the different applications in the Annexes a new method was reasoned which consists in implementing previous methods all together, in conjunction with region growing, to obtain the benefits that each provide and compensate for their individual weaknesses.

This process will basically consist of using the segmentation obtained from the region growing algorithm as the contour for the initialization of the active contour algorithm. To the segmentation obtained with region growing the closing morphological operation (dilation + erosion) will be applied and also an additional dilation to have a margin for fitting the actual ROI.

From that resulting image the edge of the region to segment will be retrieved and applied to the active contour algorithm as the initialization curve. This solution provides an automatic and precise way of defining the initialization curve, that will later be deformed to fit the ROI.

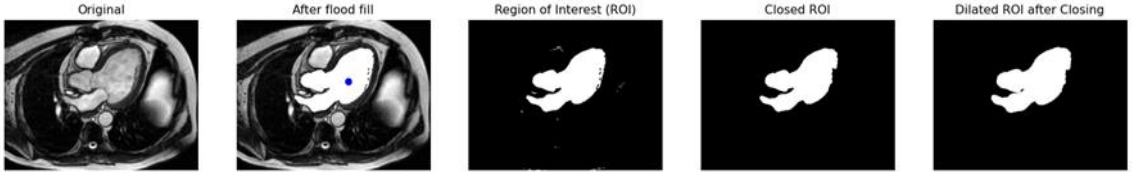


*Figure 28. Obtention of Dilated ROI After Region Growing and Closing Operation.*

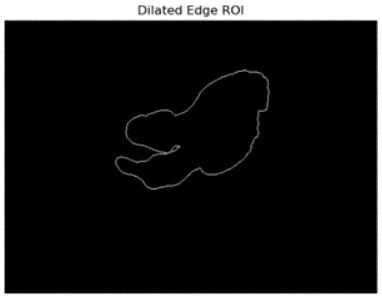


*Figure 29. Edge of the Dilated ROI.*

No parameter tuning was required to obtain the segmentation seen below. With this new method, the initial curve is already very well fitted to the region to be segmented and the default parameters of the active contour function are good enough. This makes the implementation of the algorithm more time efficient and precise than other methods.

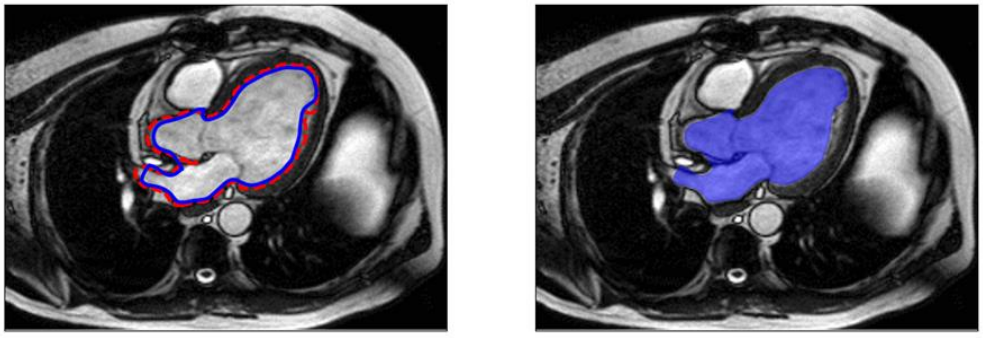

*Figure 30. Snakes Segmentation with Automatically Defined Initialization Curve.*

By implementing the two segmentation techniques together we are able to obtain the best segmentation result in this section, better than the results we would obtain by applying the techniques individually.

## Annex 10: Table explaining the Personalized Function 'visualize_felzenszwalb' and its results for both images.

*Table 6: Personalized function 'visualize_felzenszwalb'.*

<table>
<tr><td colspan="4">def visualize_felzenszwalb(img, option):</td></tr>
<tr><td rowspan="3">felzenszwalb()</td><td>img</td><td colspan="2">The input image as an n-dimensional array.</td></tr>
<tr><td>scale</td><td colspan="2">Controls the approximate size of the segments. A smaller scale leads to smaller segments and thus more in quantity, and a larger scale results in larger segments and thus less.<br>Value through which is iterated in order to find the optimal one.</td></tr>
<tr><td>sigma = 0.8</td><td colspan="2">Affects the threshold for segmenting regions. A higher sigma value makes the algorithm more permissive, while a lower sigma results in stricter segmentation.<br>It has been seen that the optimal value for this parameter is the default.</td></tr>
<tr><td colspan="3">ax.imshow(segmented_image)</td><td>If option = 1 display the original Felzenszwalb segmentation.</td></tr>
<tr><td colspan="3">ax.imshow(mark_boundaries(img, segmented_image)</td><td>If option = 2 display boundaries computed by the Felzenszwalb algorithm.</td></tr>
<tr><td colspan="4">Output: Display of segmented images with varying 'k' values, allowing users to observe the effects of different 'k' values on image segmentation results. It gives two possible options, seeing the original Felzenszwalb segmentation or the boundaries computed by the technique.</td></tr>
</table>

Results for the Brain MRI

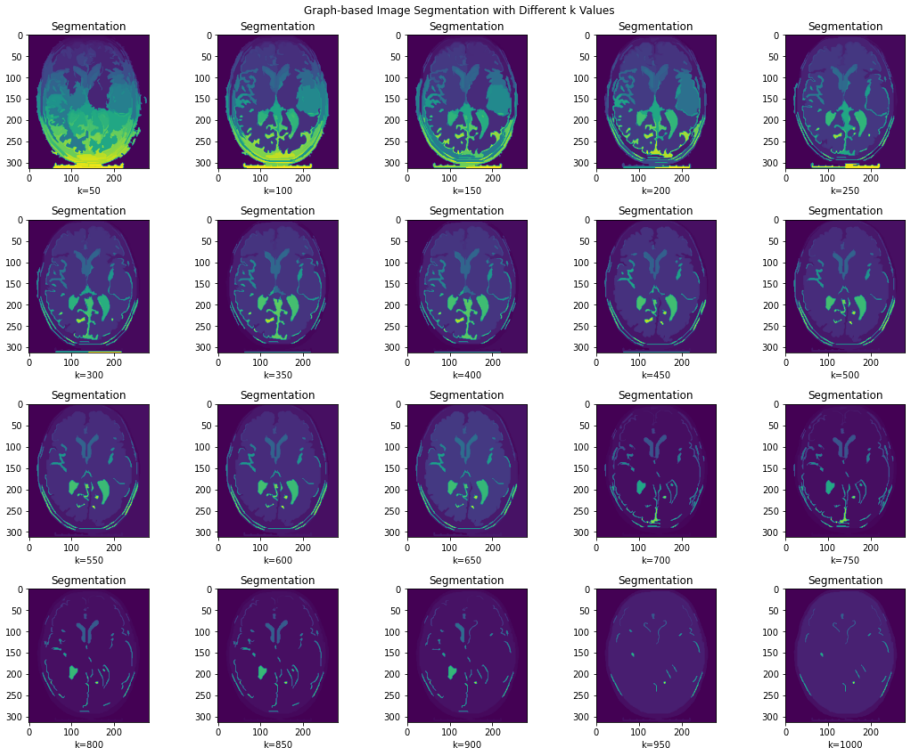


*Figure 31: Felzenszwalb graph-based image segmentation with different scale values for the brain MRI, when applying option = 1.*

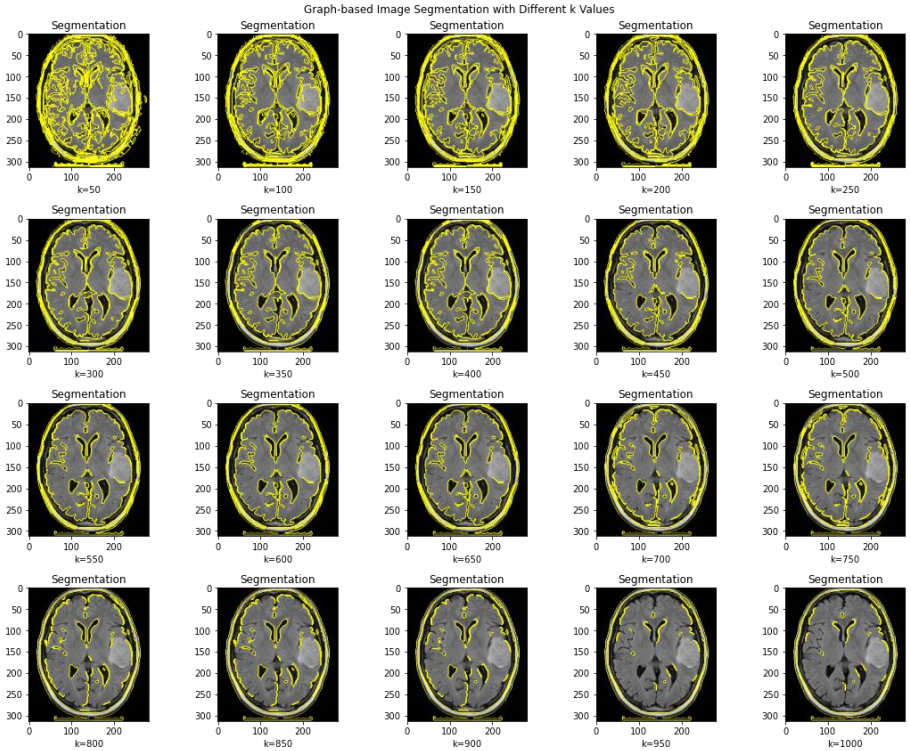


*Figure 32: Felzenszwalb graph-based image segmentation boundaries with different scale values for the brain MRI, when applying option = 2.*

Results for the heart MRI.

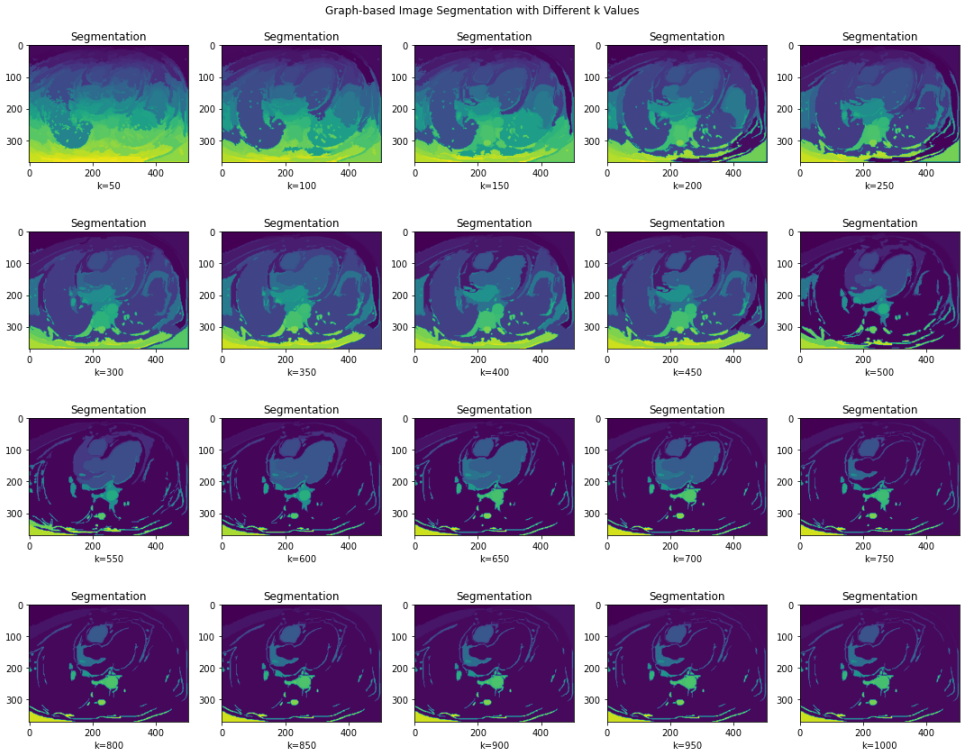


*Figure 33: Felzenszwalb graph-based image segmentation with different scale values for the heart MRI, when applying option = 1.*

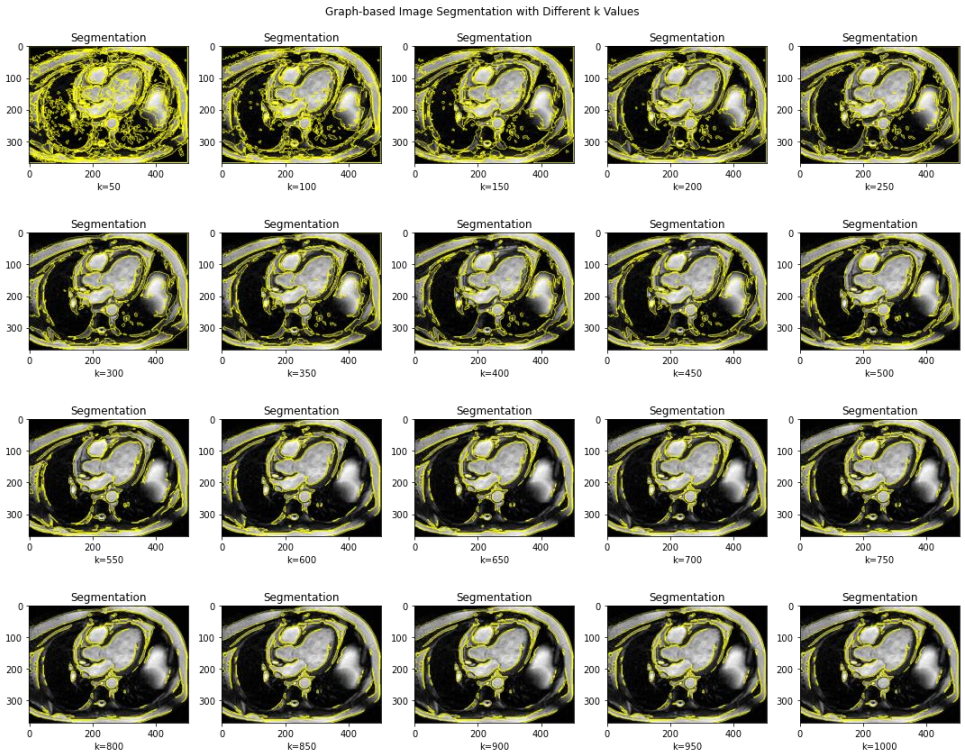


*Figure 34: Felzenszwalb graph-based image segmentation boundaries with different scale values for the brain MRI, when applying option = 2.*

The results obtained applying Felzenszwalb segmentation technique to the heart MRI after applying a mask and a median filter in *Figure 25* and *Figure 26* respectively.

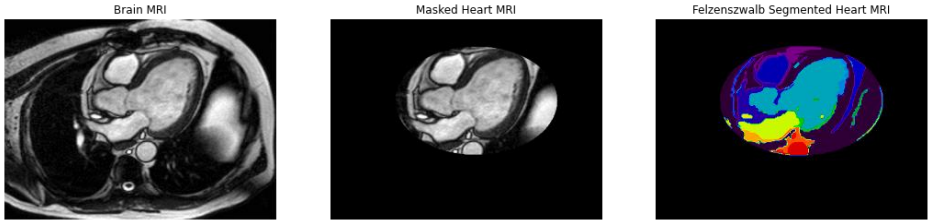


*Figure 35: Felzenszwalb segmentation after applying a mask filter to the heart MRI.*

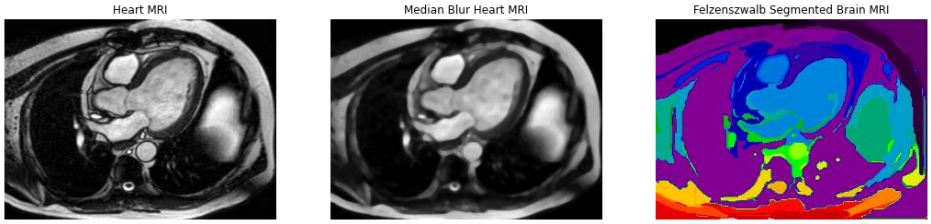


*Figure 36 Felzenszwalb segmentation after applying a median filter to the heart MRI.*

# 10. Table of figures