BIOM 5202 / SYSC 5202 - Fall 2024: Assignment #2: Image Segmentation
Name: Amirhossein Ebrahimi   (300396640)

For the sake of space, I included only a small part of the code in the answers. The full code is accessible through the Colab. Each Section is hyperlinked to its corresponding code snippet in the google Colab.

Instructions

- This assignment must be completed individually. It is okay to discuss things with your colleagues, but do not copy any of their answers, codes or notes. Plagiarism is a serious instructional offence that will not be tolerated. Please cite all sources when referring to external resources (e.g. course notes, books, papers, etc).
- For this assignment, it is recommended that you work with Matlab (or equivalent software). Carleton students can install Matlab on their personally owned computers for learning purposes (see http://carleton.ca/ccs/matlab/). Python is also acceptable provided the students submit codes with clear instructions on how to run their codes. All codes must be appropriately commented. Do not simply copy codes (or parts of it) from the internet! All sources must be cited.
- Questions require 1–2 sentences to answer. Long answers are not required.
- The deliverable for this assignment is a short report that must include: i) the answers to each question; ii) the images and plots generated in each step, carefully captioned; iii) discussion of all information requested in each section; iv) the Matlab codes, properly commented (.m file) and the image(s) you used to generate the results presented (yes! I will run your script and check the results).
- This assignment requires you to use ChatGPT (or a similar LLM) to generate a code for you. Think carefully about which prompt question you will use for this. Then, once the code is generated, do not modify it! Simply apply it to a medical or microscopy image and include snapshots in the report.
- Images used in this assignment

Axial_thoraxCT.jpg	MRI_brain2.jpg	MRI_brain.jpg	MRI_heart_SAX.jpg

- Due date and time: Nov 6th, 2024 at 23:59.

SECTION 1 - Grey-level histogram-based segmentation

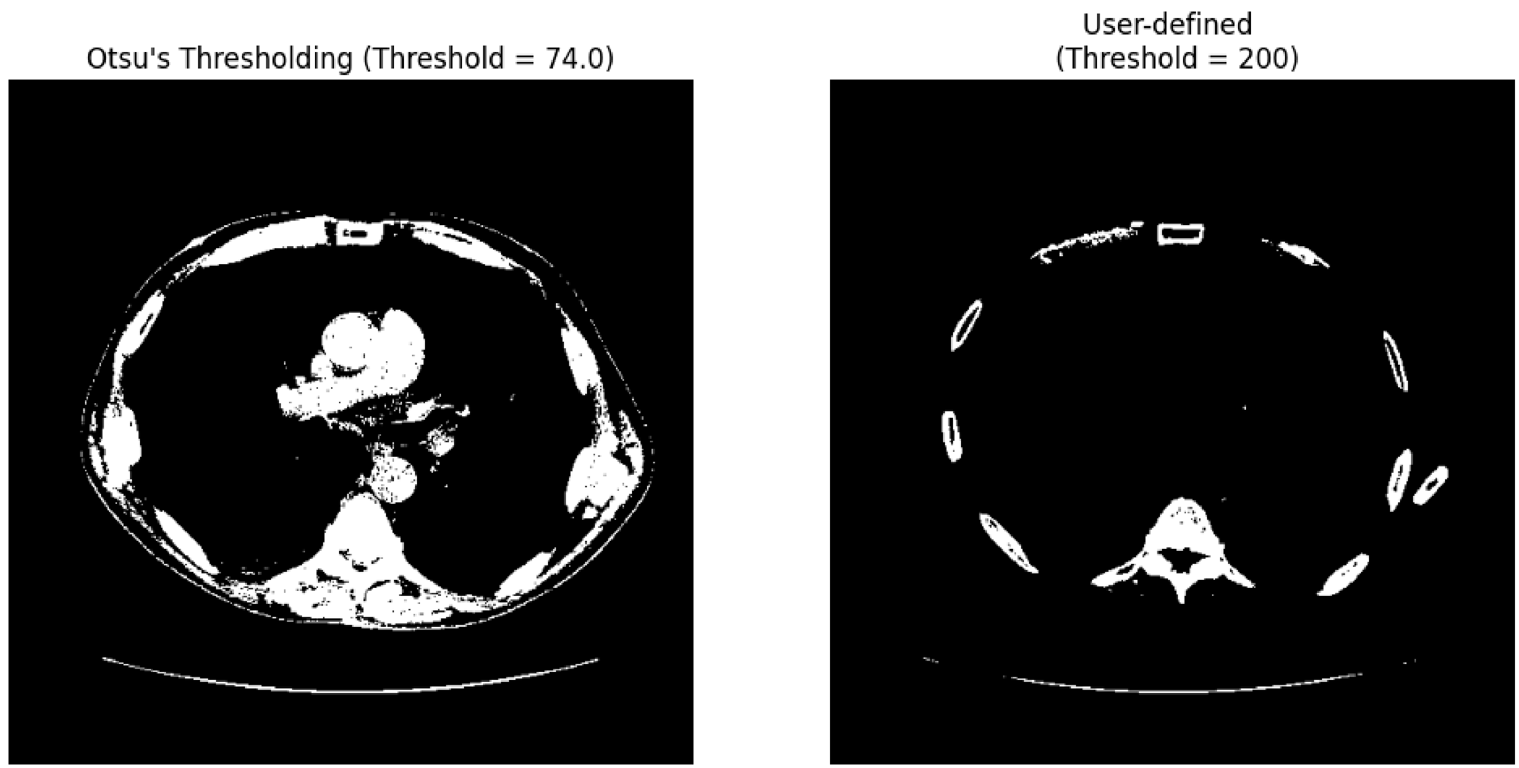
For this part, please use the image named "Axial_thoraxCT.jpg".
Based on the image histogram, your goal is to segment all the structures related to bones present in the image. Be careful with the bone marrow / spinal cord areas (darker areas present within bones). Use two different threshold-based segmentation methods and compare the results (e.g. one mathematically defined, one based on user entry, one empirical/qualitative or even one you created).

Question:	**Answer / Code / Image**

<table>
<tr>
<td>1A: Show the software developed for the two segmentation methods</td>
<td>

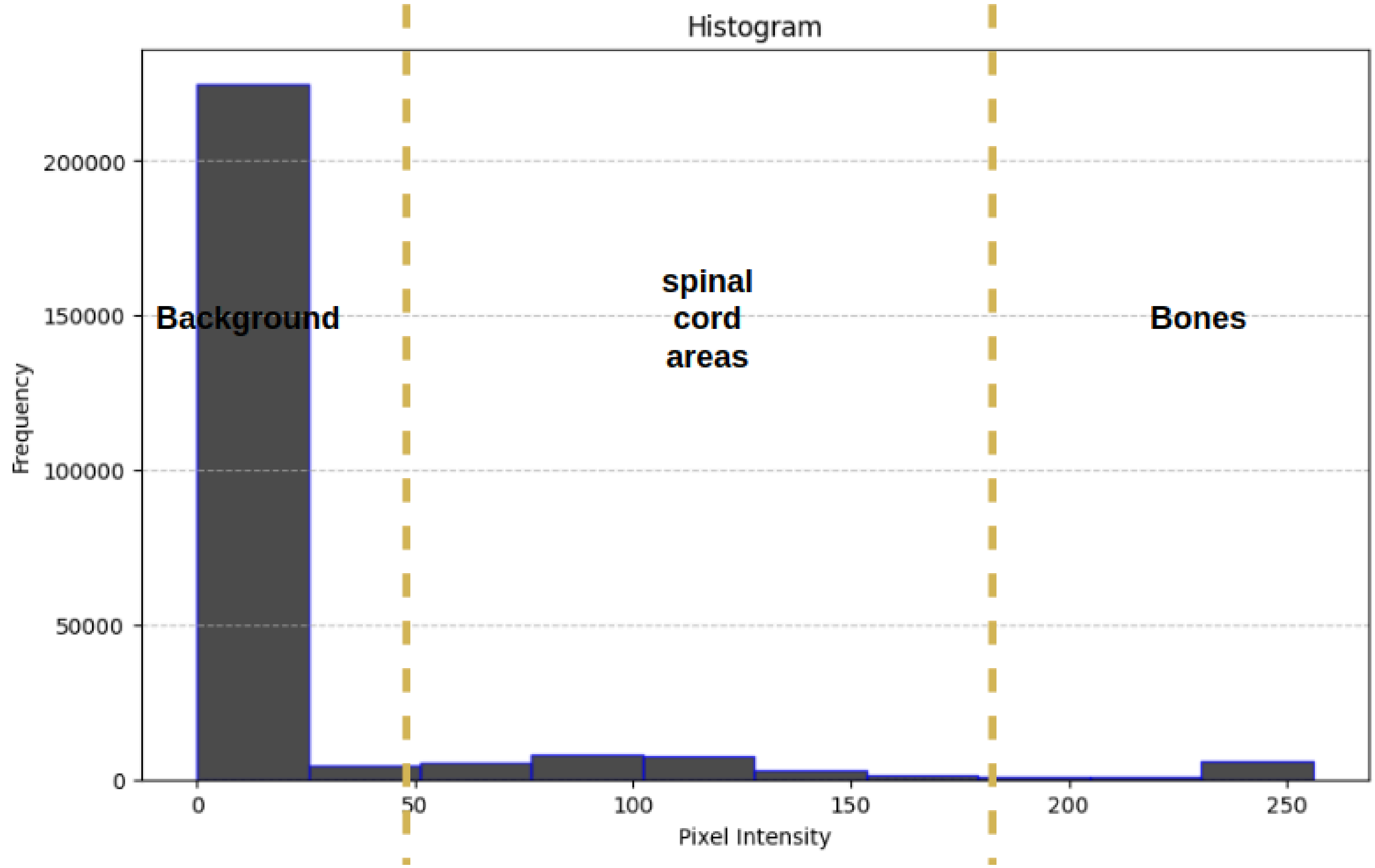
```
# 1: Otsu's Thresholding
otsu_value, otsu_thresh = cv2.threshold(image, 0, 255, cv2.THRESH_BINARY +
cv2.THRESH_OTSU)

# 3: User-defined Thresholding after seeing histogram
user_threshold = 200
_, user_thresh = cv2.threshold(image, user_threshold, 255, cv2.THRESH_BINARY)
```

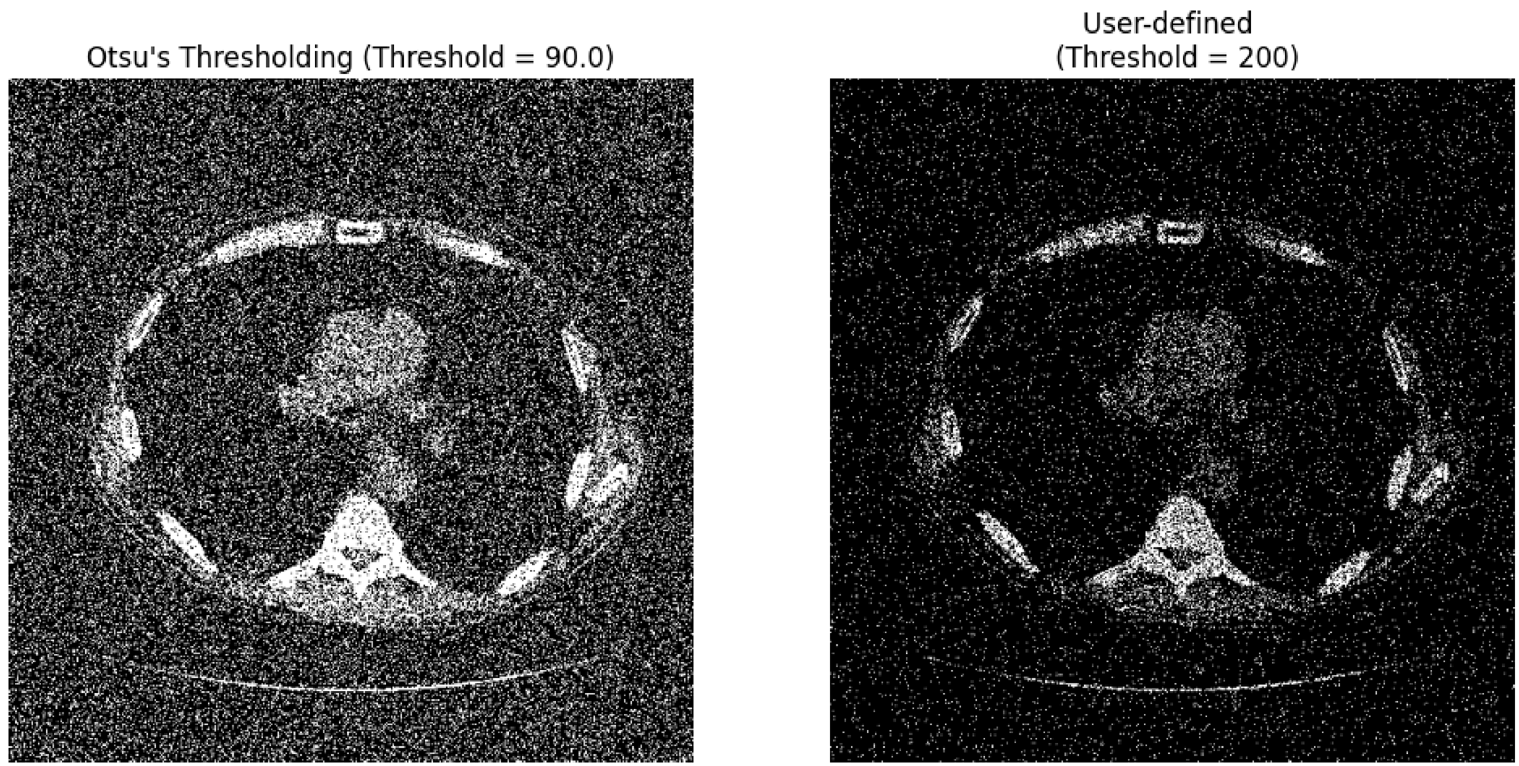
</td>
</tr>
<tr>
<td>1B: Use your developed approaches on the image. Show the results</td>
<td>



</td>
</tr>
<tr>
<td>1C: How have you defined the best threshold value? Briefly explain the rationale for choosing certain parameters. Did one of the methods performed better</td>
<td>Otsu's Method: Automatically finds the best threshold by maximizing the between-class variance.
This method is effective when there is a distinct separation between foreground and background. Where in our case is not a great method as there is no clear distinction between foreground and background.

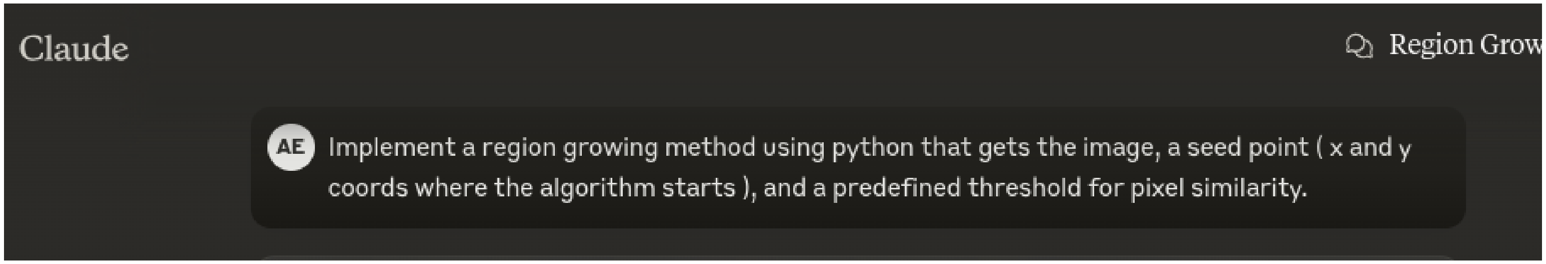
Manual Threshold (200): Chosen by looking at the image and selecting a value that clearly separates the bones from other parts. By looking at the histogram, it can be seen that there are three separate parts in the image</td>
</tr>
</table>

than the other. Discuss the results.	 .

1D: Now explore how noise could affect the results. Artificially add a significant amount of Gaussian noise to the image. Re-trace your steps and attempt to segment the structures using the same parameters as before. Show the resulting images	 Gaussian noise with var 0.2: `gaussian_noise_axial_thorax = util.random_noise(image_axial_thorax, mode='gaussian', var=0.2)`
1E: Has noise affected the outcome of both methods to the same extend? Which method (if any) was more robust? Briefly explain [<100 words]	The presence of noise leads to poor results. Because we're working with histograms, Gaussian noise significantly affects them. With the previous thresholding methods, we ended up with poor outcomes, detecting almost everything—including bones, spinal cord areas, and background. Using Otsu's method, the threshold changed from 74 to 90, but this did not make any significant impact. Regarding robustness, I don't think any of the methods are robust. Otsu's method is consistent but consistently yields poor results.

SECTION 2 – Segmentation with AI

Implement a region growing method by using ChatGPT (or similar LLM) to generate a code for you. It should accept a seed point and then perform the region growing segmentation based on a pre-defined threshold. Think carefully about which prompt question you will use for this. Then, once the code is generated, do not modify it!

Question:	**Answer / Code / Image**
2A: Include a screenshot of your question to the LLM and generated answer.	

Region Growing Algorithm Implementation

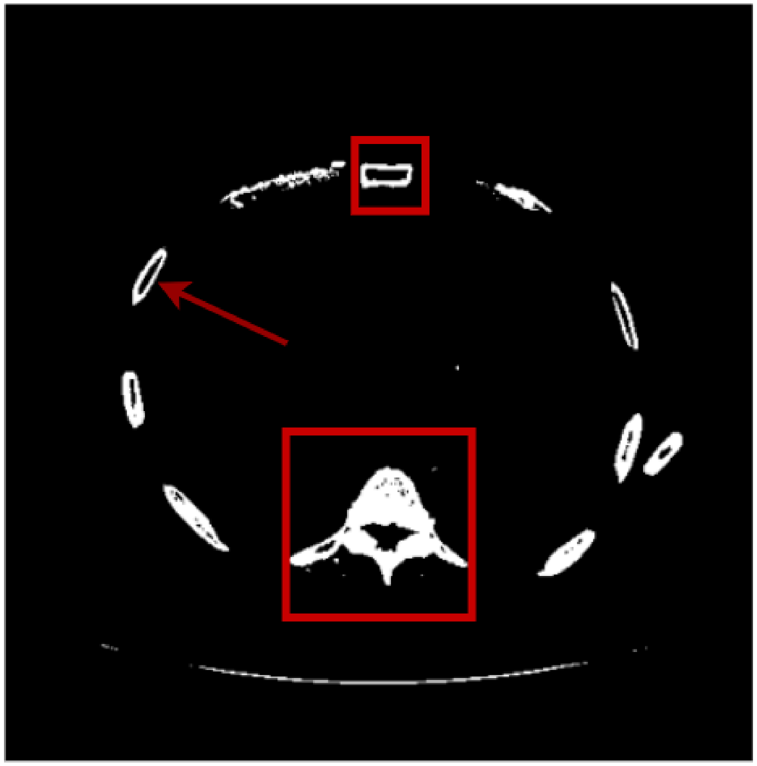
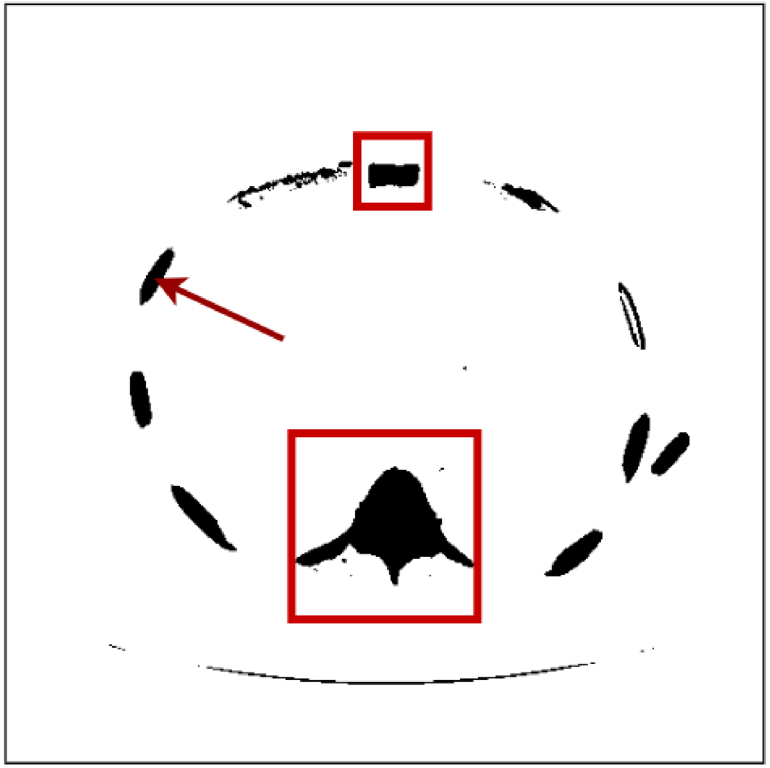
```python
import numpy as np
from scipy.ndimage import generate_binary_structure
from collections import deque

def region_growing(image, seed_point, threshold):
    """
    Implements region growing algorithm on a grayscale image.

    Parameters:
    image: numpy.ndarray
        Input image (grayscale)
    seed_point: tuple
        Starting point coordinates (x, y)
    threshold: float
        Maximum absolute difference in pixel values to be considered similar

    Returns:
    numpy.ndarray
        Binary mask of the segmented region
    """

    # Ensure the image is in the correct format
    if len(image.shape) > 2:
        raise ValueError("Input image must be grayscale")

    # Create a mask to store the segmented region
    mask = np.zeros_like(image, dtype=np.bool_)

    # Get image dimensions
    height, width = image.shape

    # Validate seed point
    x, y = seed_point
```

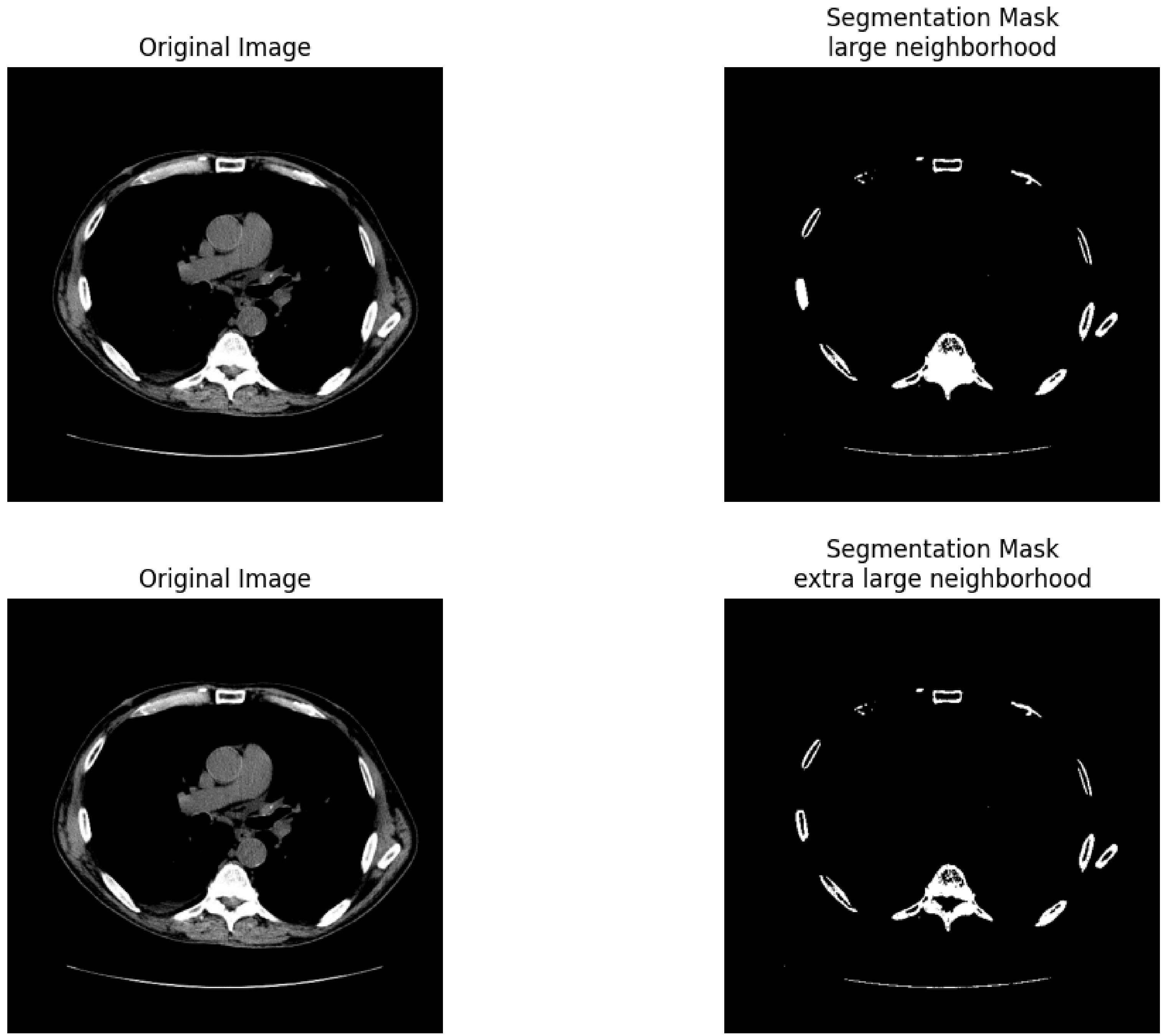
Region Growing Algorithm Implementation

```python
    # Validate seed point
    x, y = seed_point
    if not (0 <= x < width and 0 <= y < height):
        raise ValueError("Seed point outside image boundaries")

    # Get the seed point's intensity value
    seed_value = image[y, x]

    # Initialize queue with seed point
    queue = deque([(x, y)])
    mask[y, x] = True

    # Define 8-connectivity neighborhood
    neighbors = [(-1, -1), (-1, 0), (-1, 1),
                 (0, -1),           (0, 1),
                 (1, -1),  (1, 0),  (1, 1)]

    # Region growing
    while queue:
        current_x, current_y = queue.popleft()

        # Check all neighboring pixels
        for dx, dy in neighbors:
            next_x, next_y = current_x + dx, current_y + dy

            # Check if the neighbor is within image boundaries
            if not (0 <= next_x < width and 0 <= next_y < height):
                continue

            # Skip if pixel has already been processed
            if mask[next_y, next_x]:
                continue
```

```
            if mask[next_y, next_x]:
                continue

            # Check if neighbor's intensity is within threshold
            if abs(image[next_y, next_x] - seed_value) <= threshold:
                mask[next_y, next_x] = True
                queue.append((next_x, next_y))

    return mask
```

2B: Apply the LLM-derived code to the "Axial_thoraxCT.jpg" image.	
2C: Discuss the quality of the result. What's good and bad about it?	Well, as you can see below, the LLM-generated algorithm is performing surprisingly well, except for some cases shown in the image below where there are closed boundaries.

	 
2D: Based on the code suggested by the LLM, modify it aiming to improve the output.	


Original Image
Segmentation Mask
large neighborhood
Original Image
Segmentation Mask
extra large neighborhood

<table>
<tr><td></td><td></td></tr>
<tr><td>

2E: What changes did you make, and why? (3–4 sentences)

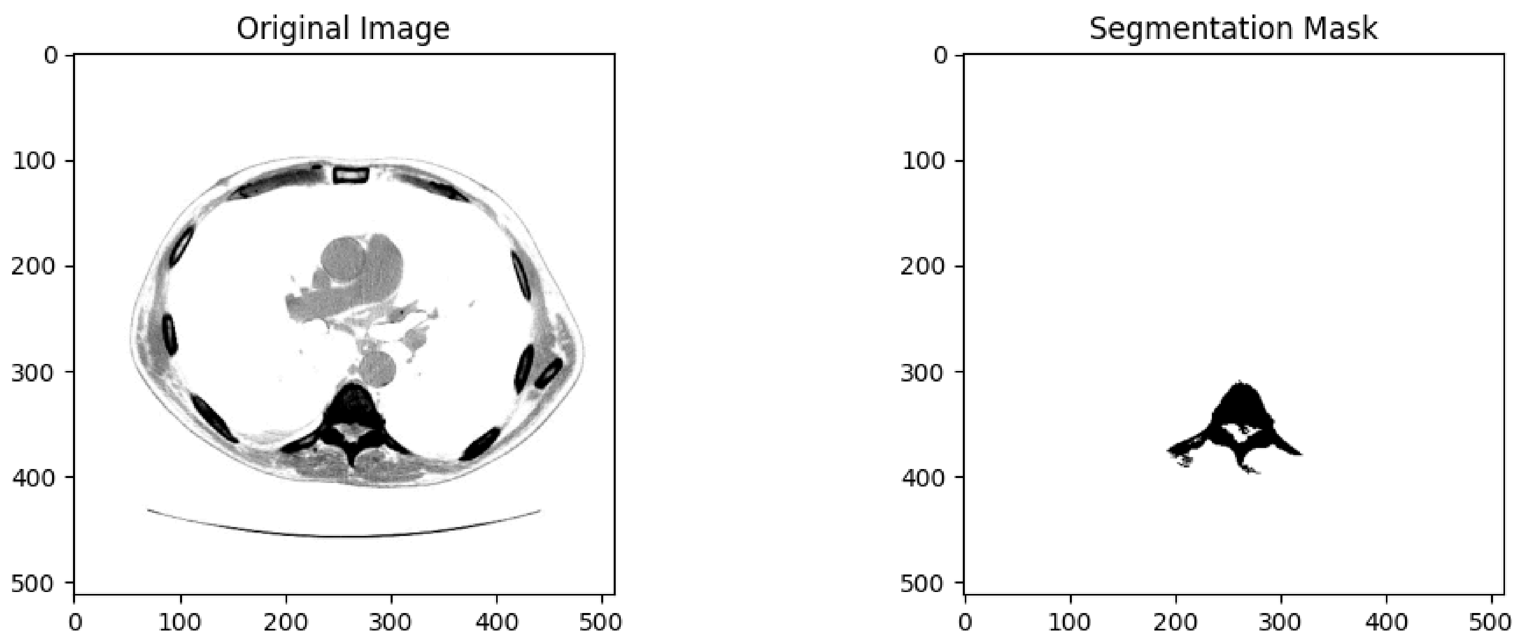
</td><td>

Attempt 1:

I tried to find the hole mentioned above, but the only way to achieve it was by setting the parameters to:

```
seed_point = (250,330)  # Center of the image
threshold = 120
```

To the **Reverted** image:

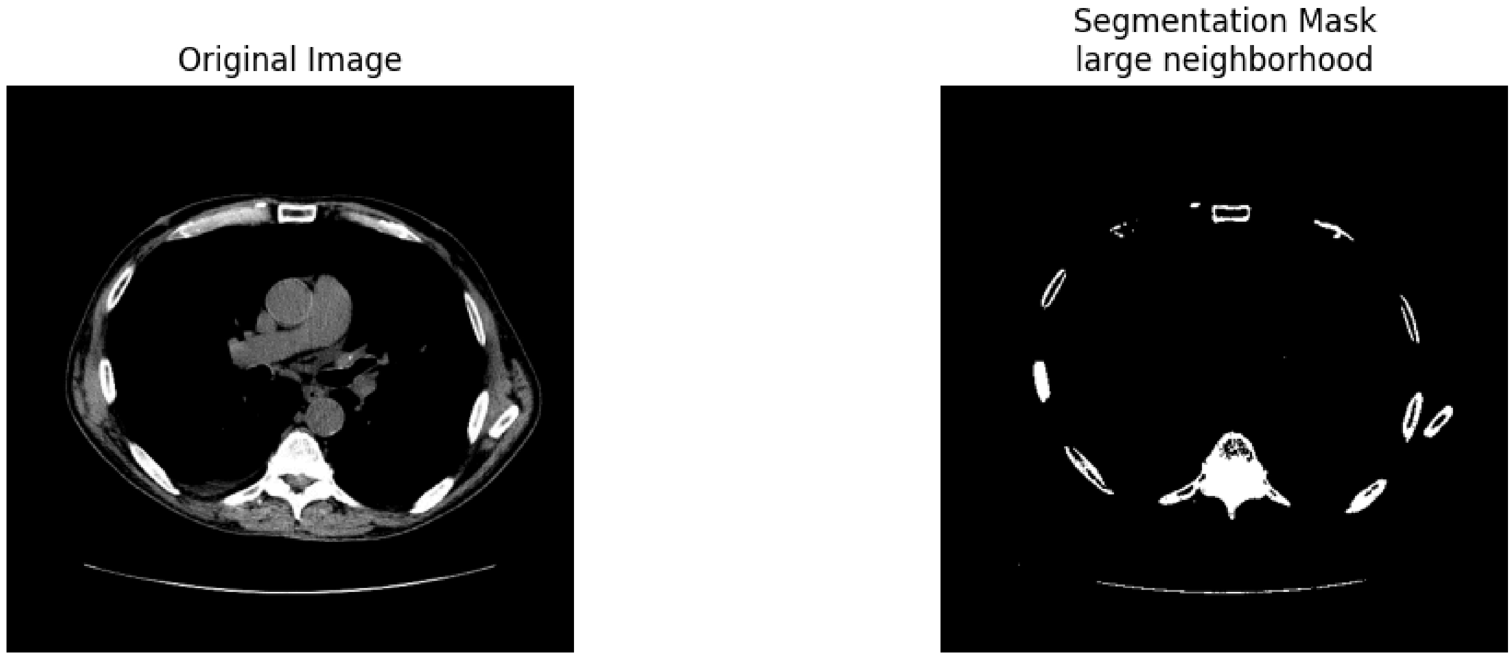


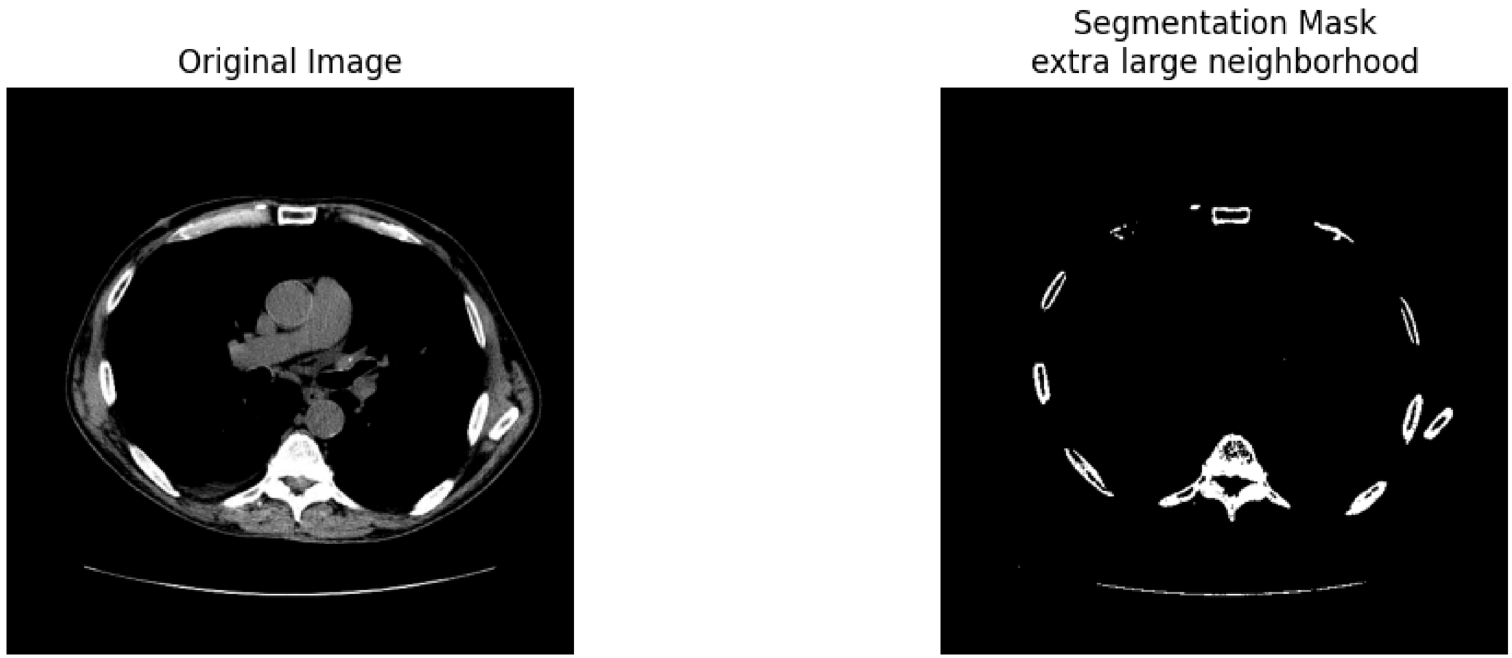
However, that was not satisfying. I tried adaptive thresholding, meaning that instead of a fixed threshold, I used an average of the neighboring intensities. But it was highly sensitive to the initial seed. I also tried a random jump, allowing the algorithm to jump to other coordinates with a small probability, but all failed.

Nevertheless, I was able to improve the results by changing the **neighborhood size**. I adjusted the neighborhood

</td></tr>
</table>

connectivity to include variant members with 4-connectivity (up, down, left, right), 8-connectivity, 24-connectivity, and 48-connectivity. I began to see explicit segmentation starting with 24-connectivity and found a perfect match with 48-connectivity. However, increasing this parameter leads to a loss of details where we have narrow edges:

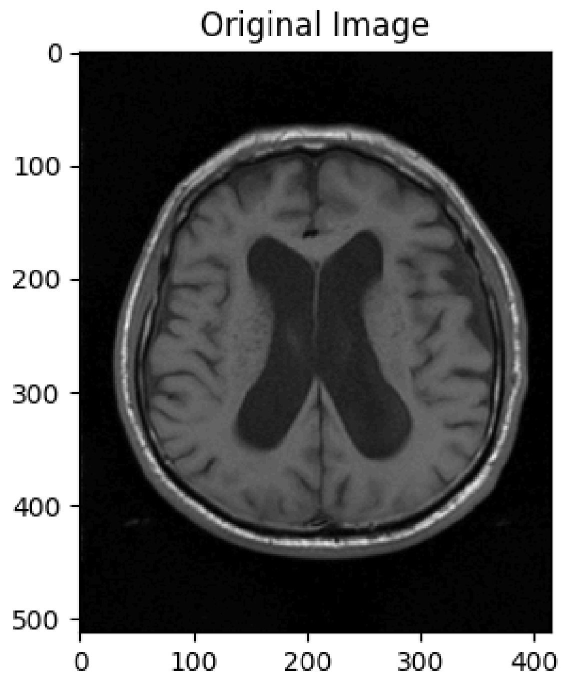
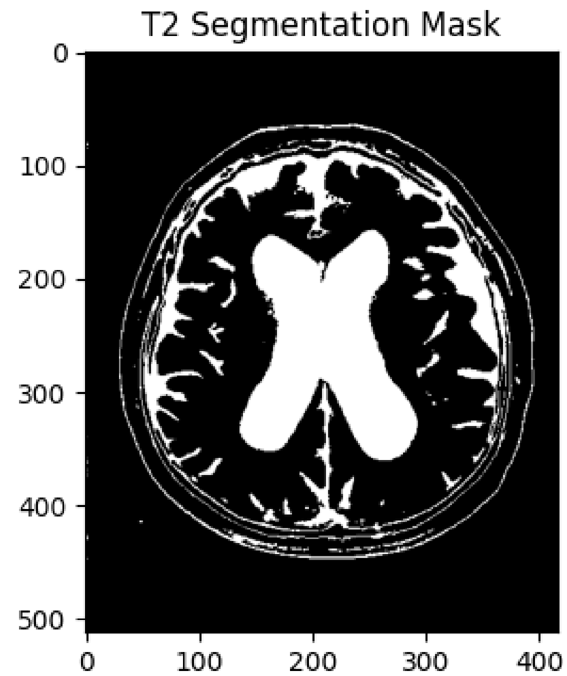


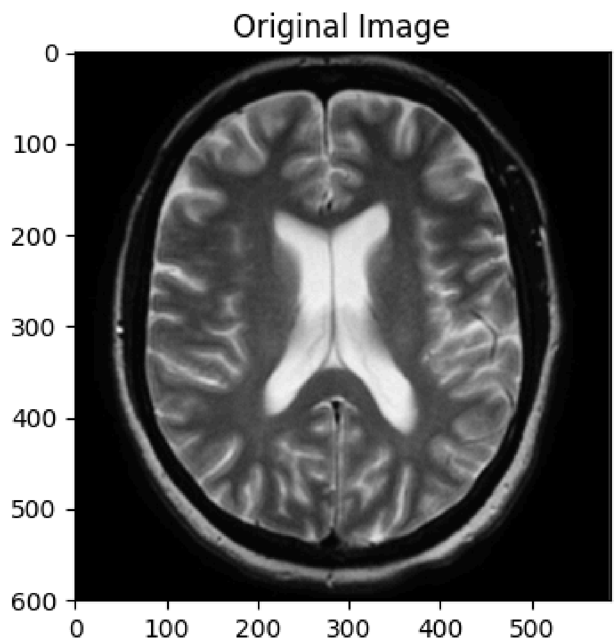
	
2F: Apply it again to the "[Axial_thoraxCT.jpg]" image and include snapshots in the report of the prompts used to improve it and iterations.	The output is already presented in the previous section.
2G: Compare and discuss the algorithm generated by the AI and the improvements you proposed. Comment on how your initial prompt question could have been improved and what would be your strategy to integrate LLMs into your workflow – being critical in regards to assess	Well, the comparison is done in the previous sections. In summary, the LLM's generated code was perfect, making it very hard to improve. However, by changing the neighbor size, I could get better results (**holes in the bones**). In my prompt, I could've asked, *"I need highly detailed output (even the holes in the bones),"* but I don't think it would have ended up with a large neighbor because it would require many trials and tests. Overall, LLMs are not great for detailed and professional tasks; they're more practical for general code generation, providing the base code to start. To use AI tools effectively in my workflow, I would use them as initial code drafts. However, I don't like debugging others' code; I prefer starting my own code and asking ChatGPT for debugging or detecting any small mistakes that may be neglected by me. That's the best way to use AI: ***to help you catch the fish, not get you the fish itself.***

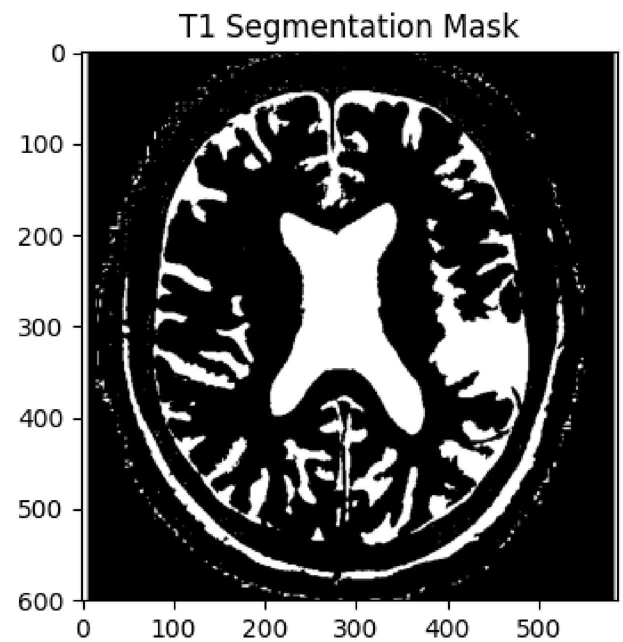
the output provided	

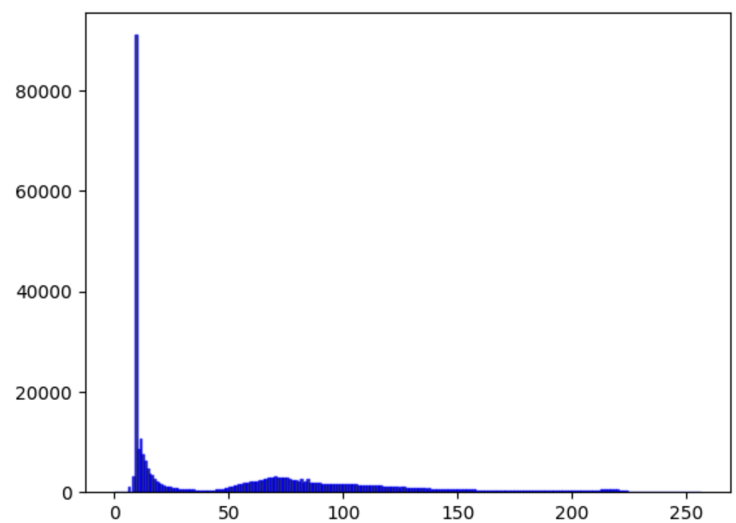
SECTION 3 - 2D Segmentation

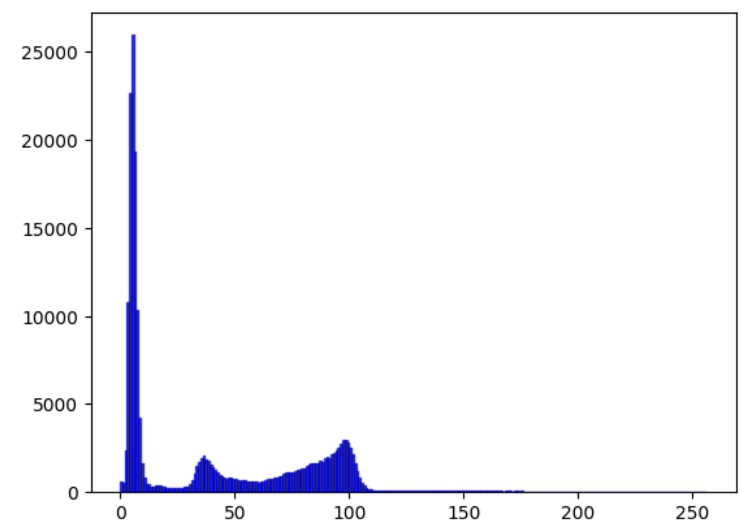
Use the images “[MRI_brain.jpg]” and “[MRI_brain2.jpg]”. (one is a T1- and the other one is a T2-weighted image) Image source [[embodi3d.com/files/category/37-medical-scans]]. Your goal is to find the best segmentation strategy you know of to segment the brain lateral ventricles for each image

Question:	**Answer / Code / Image**
3A: Use the developed segmentation approaches from section #2 on the provided MRI images.	 

<table>
<tr>
<td></td>
<td>




</td>
</tr>
<tr>
<td>3B: From your observations, which version of MRI images is more influenced by the initialization parameters of your chosen method (e.g. region growing seed, threshold value, gradient, etc)?</td>
<td>
<table>
<tr><th>T1</th><th>T2</th></tr>
<tr>
<td>


</td>
<td>


</td>
</tr>
</table>
</td>
</tr>
</table>

	From my observation, T2 seems to be more influenced as it has three peaks, meaning that color intensity in T2 is more so the chance of RG to get stuck in local minima is higher. **T1-weighted images:** ● Less sensitive to threshold changes and seed placement due to clearer tissue boundaries and higher contrast between tissue types. **T2-weighted images:** ● More affected by threshold and seed placement due to softer, gradual changes in brightness, making it harder to separate certain areas accurately.

SECTION 4 - 2D Segmentation Using Level Sets

Open the 2D short axis cardiac MRI image (named [‘MRI_heart_SAX.jpg’]). Your objective is to segment the left ventricular blood pool (please see the diagram shown in Fig.1).

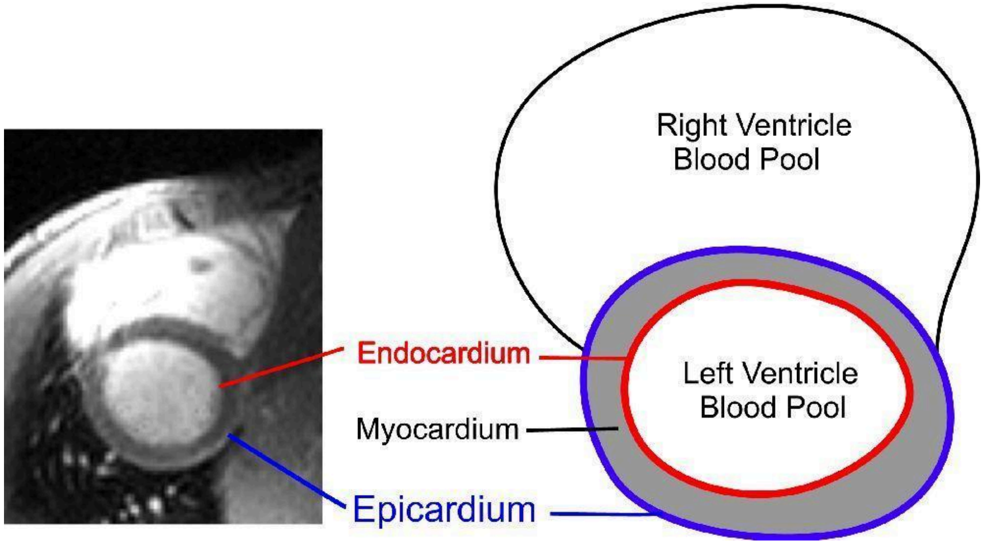


Figure 1 - MRI_heart_SAX.jpg image and its schematic representation, illustrating the main anatomical features one can identify through segmentation. Image source: O'Brien, S. (2011). Integrating Contour-Coupling with Spatio-Temporal Models in Multi-Dimensional Cardiac Image Segmentation (Doctoral dissertation, Dublin City University). Hu, Huaifei, et al. "Automatic segmentation of the left ventricle in cardiac MRI using local binary fitting model and dynamic programming techniques." PloS one 9.12 (2014): e114760.

In this question, apply the level set method based on Chan-Vase method (aka 'activecontour' in Matlab) to the image and obtain the final segmentation for the following conditions:

A. Initial contour within the object (i.e. segmenting the exact boundary of the left ventricle blood pool)

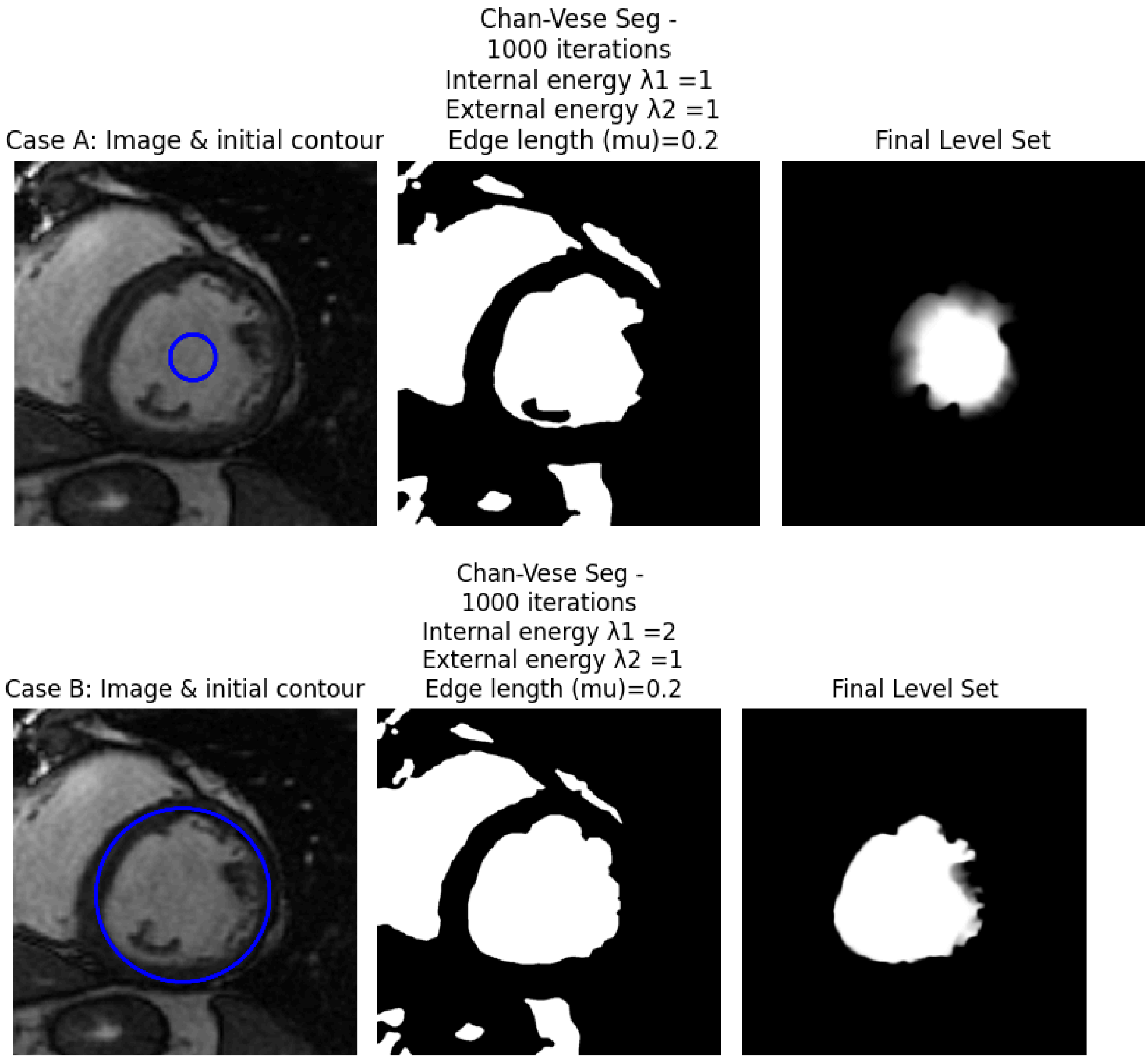
B. Initial contour enclosing the object (i.e. segmenting the epicardium boundary, including the boundary of the left ventricle blood pool within its limits)

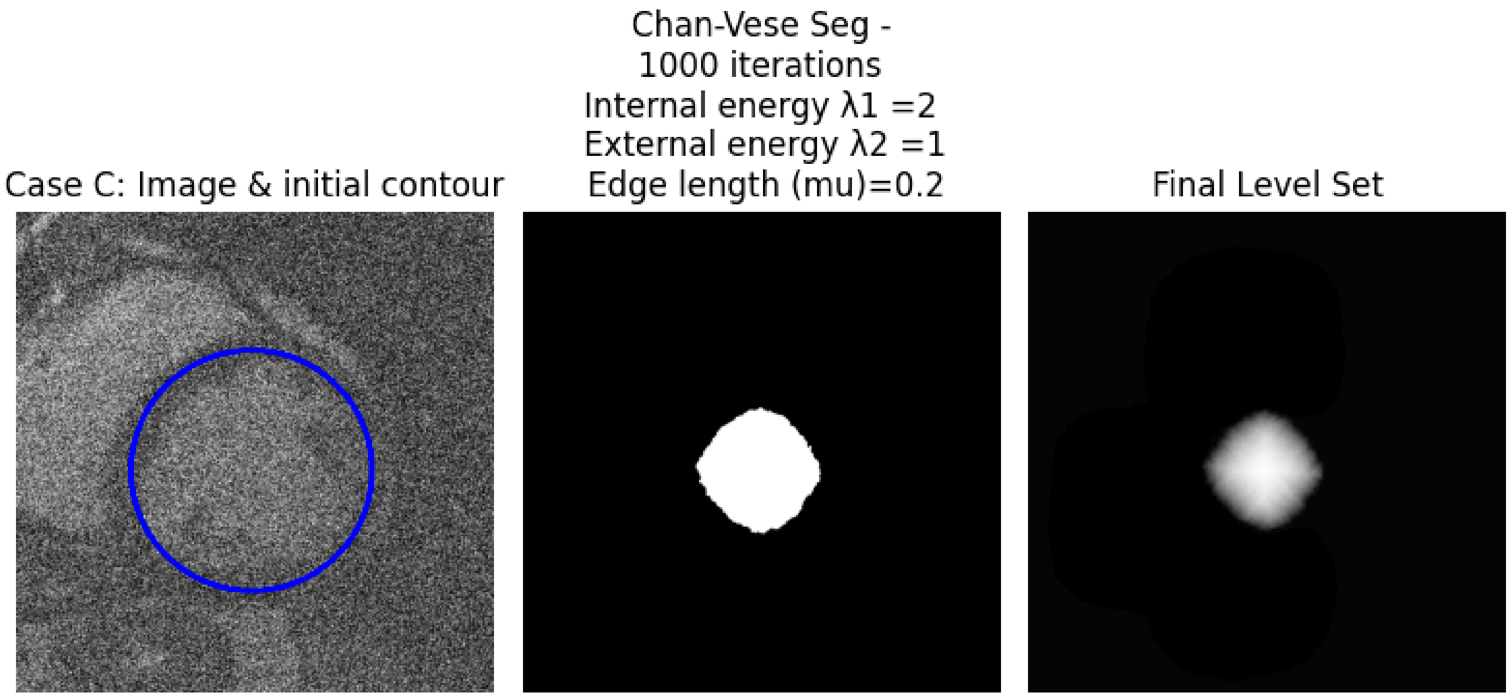
C. Initial contour enclosing the object (same as ii), but in this case the image is severely affected with Gaussian noise (please artificially add Gaussian noise to the image).

Question:	**Answer / Code / Image**

[**4A:** Show the results for the three cases (A,B,C) for the MRI image.]()

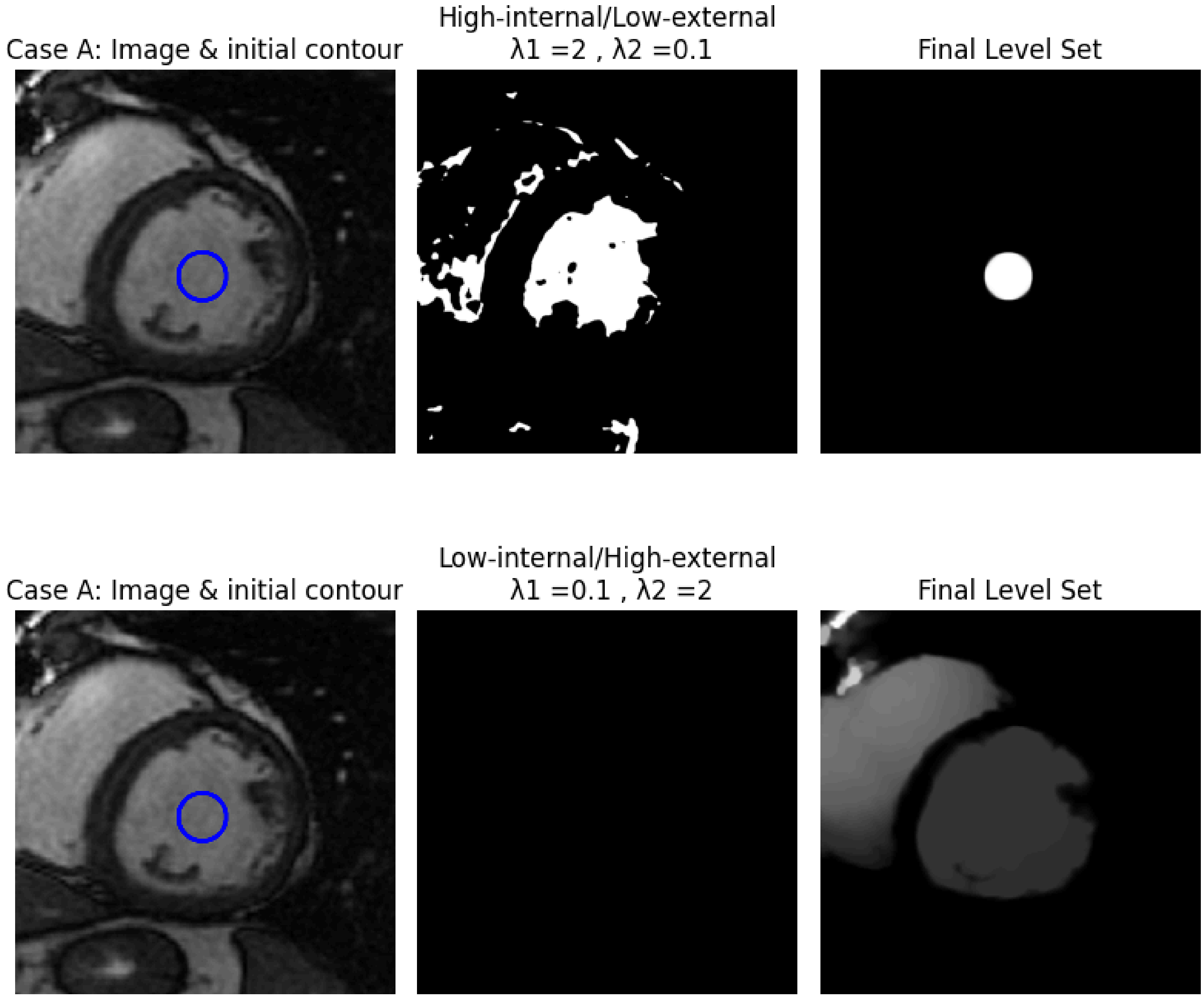
```
# Case     initial_contou  ,  img with/without noise, lambda1 and lambda2
Case_A = initial_contour_A , image_axial_thorax, "A", 1, 1
Case_B = initial_contour_B , image_axial_thorax, "B", 2, 1
Case_C = initial_contour_B , noisy_image_axial_thorax, "C", 2, 1
Case_D = initial_contour_A , noisy_image_axial_thorax, "D", 2, 1
Cases = [ Case_A, Case_B, Case_C, Case_D]

for case in Cases:
 initial_contour, image, idx, lambda1, lambda2 = case
 cv = chan_vese(
   image,
   mu=0.2,
   lambda1=lambda1,
   lambda2=lambda2,
   tol=1e-4,
   max_num_iter=1000,
   dt=0.1,
   init_level_set=initial_contour,
   extended_output=True,
 )
```


Case A: Image & initial contour
Chan-Vese Seg -
1000 iterations
Internal energy λ1 =1
External energy λ2 =1
Edge length (mu)=0.2
Final Level Set
Case B: Image & initial contour
Chan-Vese Seg -
1000 iterations
Internal energy λ1 =2
External energy λ2 =1
Edge length (mu)=0.2
Final Level Set

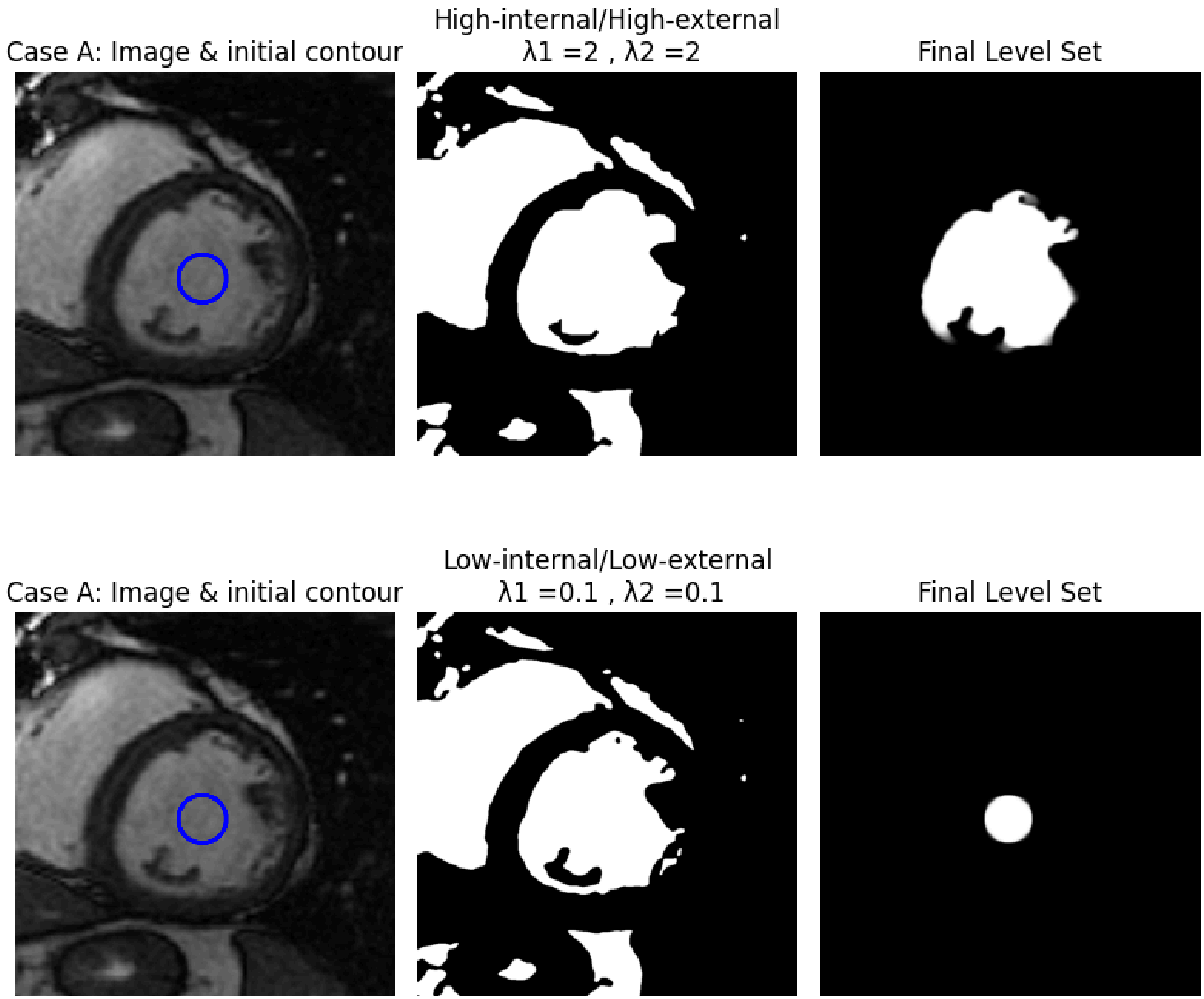
	
5B: The internal and external energies are balanced by a smoothness parameter in the Chan-Vase level sets. Use values of 0.5 and 2.5 for the 'smoothfactor' and obtain segmentation of the object	Since I'm using the python library, I don't have access to such a smoothness parameter. However, I can show the cases where Internal and external energies have different values. For each case (A and B) I have tested : High-internal/Low-external, Low-internal/-High-external, Low-internal/-Low-external, High-internal/-High-external No-internal/High-external, High-internal/No-external

	``
	energy = {
	   "High-internal/Low-external": [2,0.1],
	   "Low-internal/High-external": [0.1,2],
	   "High-internal/High-external": [2,2],
	   "Low-internal/Low-external": [0.1,0.1],
	}
	
	Case_A = initial_contour_A , image_axial_thorax, "A"
	Case_B = initial_contour_B , image_axial_thorax, "B"
	Cases = [Case_A, Case_B]
	
	for case in Cases:
	 for key, value in energy.items():
	   lambda1, lambda2 = value
	   initial_contour, image, idx= case
	   cv = chan_vese(
	     image,
	     mu=0.2,
	     lambda1=lambda1,
	     lambda2=lambda2,
	     tol=1e-4,
	     max_num_iter=1000,
	     dt=0.1,
	     init_level_set=initial_contour,
	     extended_output=True,
	   )
	``

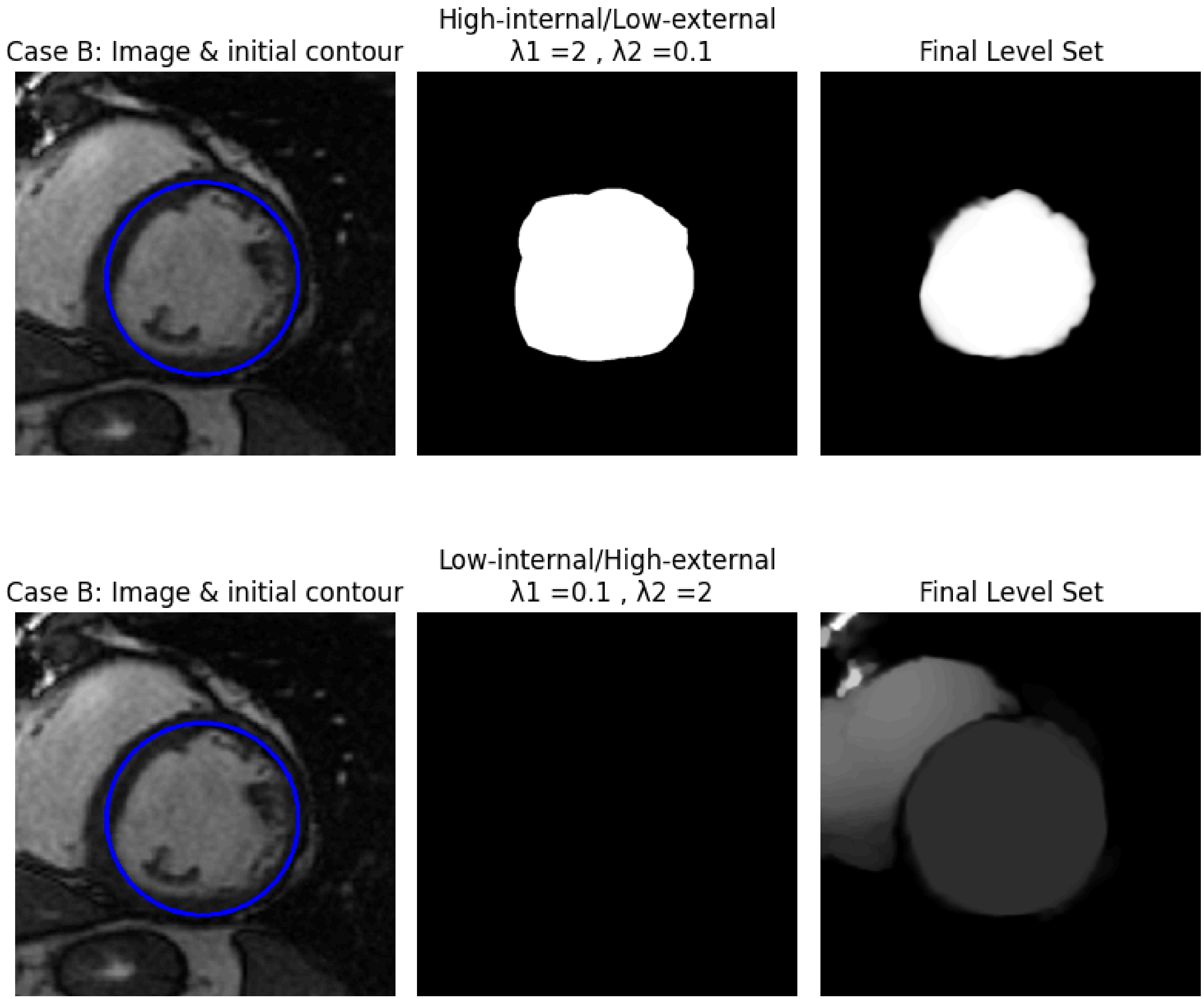
Case A

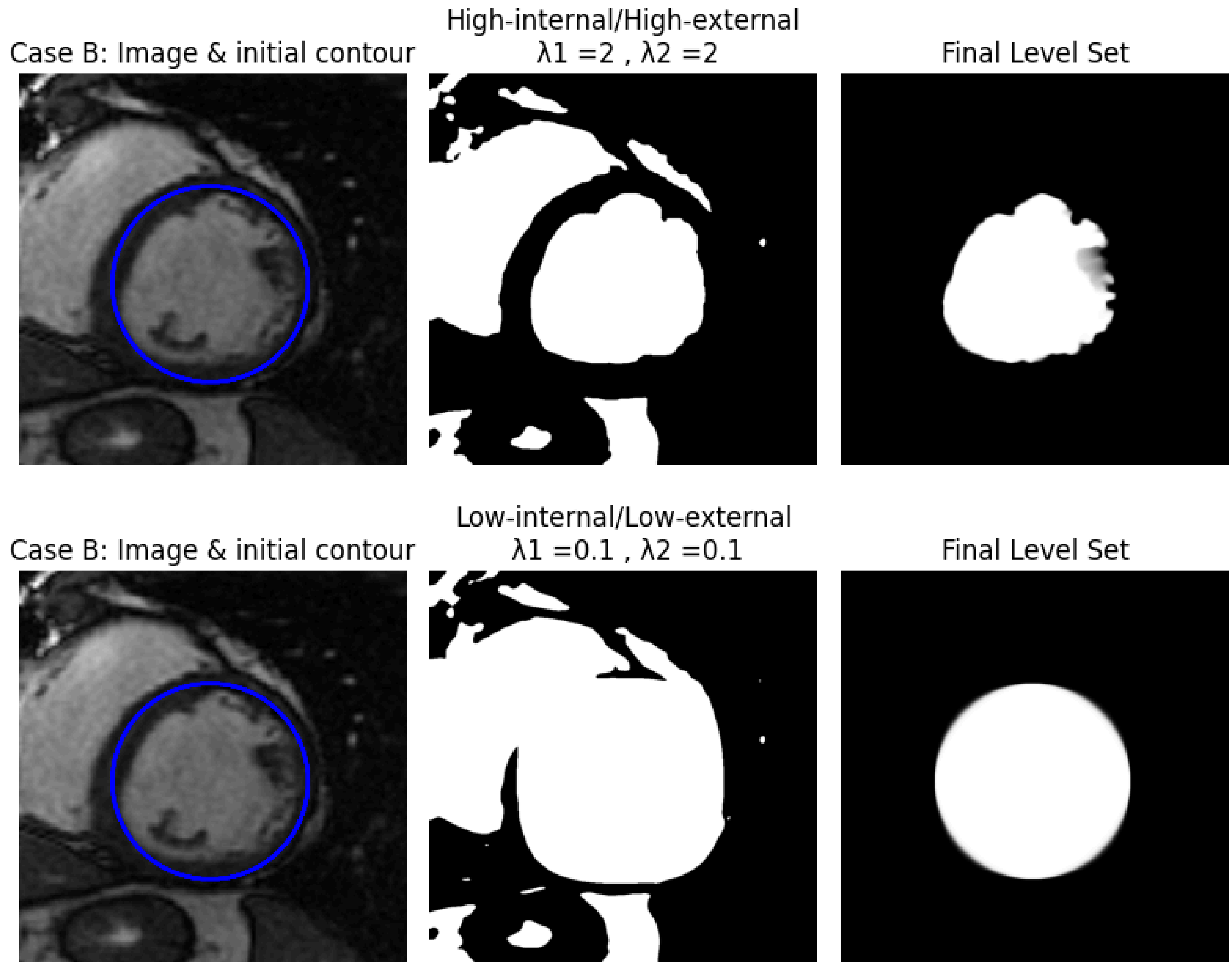


Case A: Image & initial contour
High-internal/Low-external
λ1 =2 , λ2 =0.1
Final Level Set
Case A: Image & initial contour
Low-internal/High-external
λ1 =0.1 , λ2 =2
Final Level Set


Case A: Image & initial contour
High-internal/High-external
λ1 =2 , λ2 =2
Final Level Set
Case A: Image & initial contour
Low-internal/Low-external
λ1 =0.1 , λ2 =0.1
Final Level Set

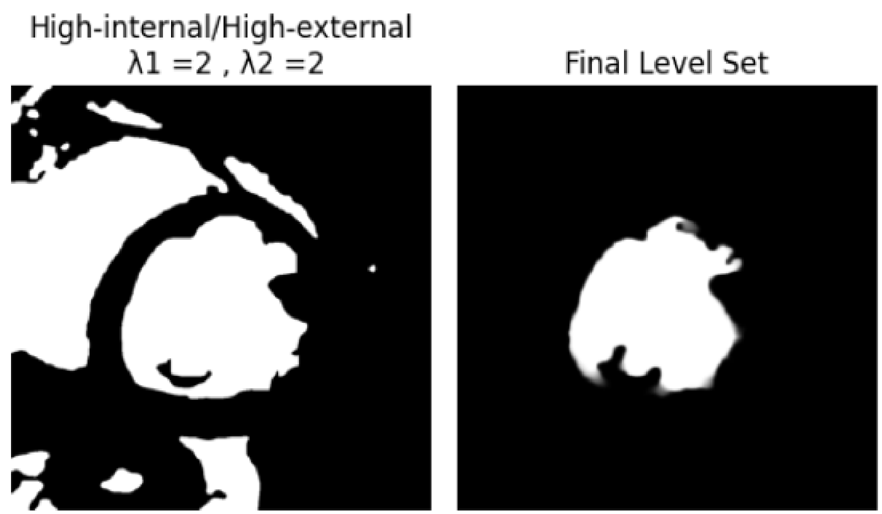
Case B


Case B: Image & initial contour
High-internal/Low-external
λ1 =2 , λ2 =0.1
Final Level Set
Case B: Image & initial contour
Low-internal/High-external
λ1 =0.1 , λ2 =2
Final Level Set

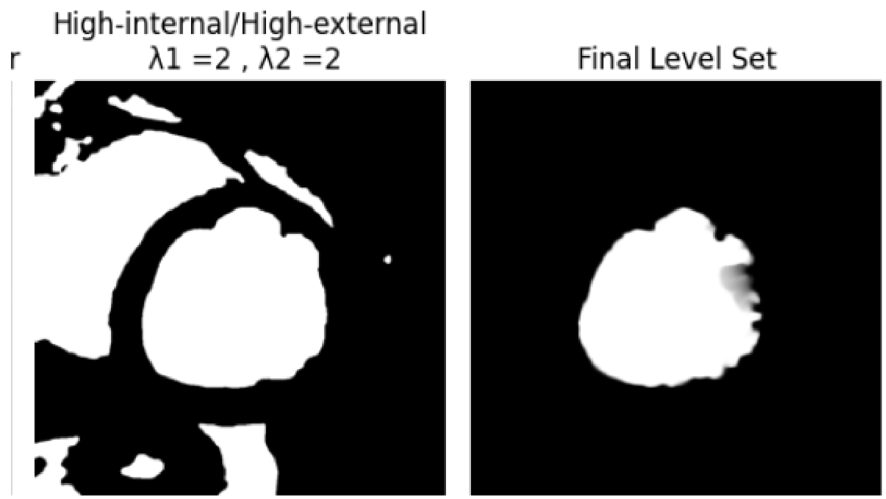
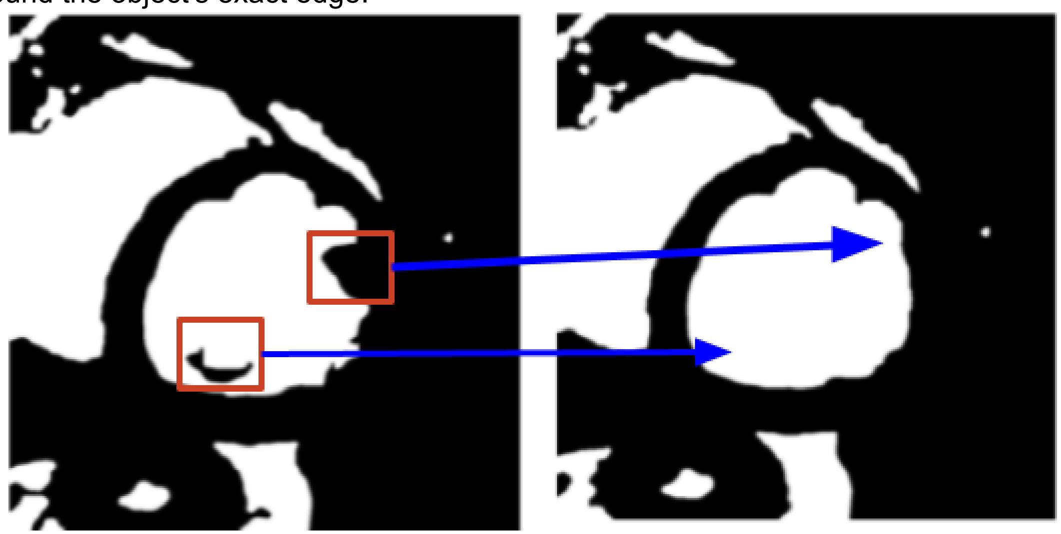

Case B: Image & initial contour
High-internal/High-external
λ1 =2 , λ2 =2
Final Level Set
Case B: Image & initial contour
Low-internal/Low-external
λ1 =0.1 , λ2 =0.1
Final Level Set

5C: Compare and discuss these segmentation strategies (i.e. detecting the object contour directly versus detecting a contour enclosing the object). What are the advantages and/or limitations? Has noise affected either strategy? [<100 words]

In case A, where the initial contour is within the object, I thought that it’s more rational to emphasize on external energy to push it to explore outside of contour. Consequently, the best result seem to be when we have equal/high energy in both sides:



In case B, where the initial contour is enclosing the object, I thought that equal values of internal and external energies are required to be able to catch epicardium boundary, including the boundary of the left ventricle blood pool, and it seems that I was right:

	 Detecting the **object contour directly** (like the blood pool boundary) is precise but sensitive to noise, as small variations can disrupt the boundary. In contrast, **detecting a contour enclosing the object** (like the epicardium boundary) is more robust to noise, as it captures larger structures. However, it may include unwanted regions and lacks precision around the object's exact edge.  Noise affects both strategies, but enclosing contours are generally more tolerant since they depend less on detailed edge information. In summary, direct contours provide accuracy, while enclosing contours offer noise resilience but may be less precise.
5D: From your observations, which version	The **direct contour detection** (e.g., segmenting the exact boundary of the blood pool) is more influenced by both initialization and noise. Starting with a small initial contour requires precise placement; otherwise, it may miss the

of level sets is more influenced by the initialization and by image noise? [<100 words].	target boundary or capture noise. Image noise disrupts this approach, leading to irregular boundaries. On the other hand, **enclosing contours** are less sensitive to initialization and noise, as they start outside the object and capture larger regions, making them more resilient to imperfections in the image.