# C2 –Optimisation in Computer Vision

## Part 3 –Segmentation
## Team no. 4

Members:
Miruna-Diana Jarda
Gunjan Paul
Diana Tat

01 **Problem Definition**

Define the topic

02 **Methods Approached**

Define the methods and criterias

03 **Code Implementation**

Implement the method in Python

04 **Results**

Segment our own examples

05 **Conclusions**

Discussions

## 1. Problem Definition

Image segmentation is the process of dividing an image into distinct, meaningful regions or segments. It can be approached in two ways: region-oriented, which groups pixels with similar visual properties into regions, and contour-oriented, which focuses on detecting object boundaries or edges.

The choice between these approaches depends on the image's characteristics and the specific analysis goal. Region-oriented methods work well for images with uniform regions, while contour-oriented methods excel in delineating precise object boundaries. Hybrid methods that combine both approaches are also used for more accurate results.Image segmentation is the process of dividing an image into distinct, meaningful regions or segments. It can be approached in two ways: region-oriented, which groups pixels with similar visual properties into regions, and contour-oriented, which focuses on detecting object boundaries or edges.

## Goal

**Goal**: Finding the different (meaningful) regions/segments of an image:

- This refers to the objective of dividing an image into distinct areas or regions, with each region containing objects or elements of significance. These regions should be meaningful in the context of the image's content. For example, in a photograph of a landscape, you might want to identify regions representing the sky, mountains, trees, and the ground as separate meaningful segments.

**Result**: Image that is a map of the different regions/segmentation:

- This outcome implies that after performing image segmentation, you will have a new image where each segment or region is highlighted or marked in a distinctive way. This "map" can be a representation of the original image where each meaningful region is colour-coded, outlined, or somehow visually distinguished from the rest of the image.

## 2. Methods Approached

We can deal with segmentation by applying **Mumford-Shah** model.  It is:

- Is a boundary-based segmentation technique. It seeks to find the optimal segmentation by identifying object boundaries while penalising the smoothness of the region.
- Suitable when we need precise boundaries and object shapes.
- It's robust to noise and can handle images with complex object shapes and contours.
- Can be computationally intensive and might not be the best choice for images with uniform regions.

# Mumford-Shah criterias

- 1) The segmentation results in an image.

  $f \rightarrow u$

- 2) The segmented image is like the original image.

  $\int_{\Omega} (f(x) - u(x))^2 \, dx$

- 3) The segmented regions are homogenous.

  $\int_{\frac{\Omega}{C}} |\nabla \ u(x)|^2 \, dx$

- 4) The segmented regions have smooth boundaries.

  $\arg_{u,C}(\mu Length(C))$

where C is the the segmentation boundary.

# Mumford-Shah

- Combining these criteria, the researchers formulated an objective function to be minimized:

$$\arg_{u,C}(\mu Length(C)) + \lambda \int_{\Omega} (f(x) - u(x))^2 \, dx + \int_{\frac{\Omega}{C}} |\nabla \ u(x)|^2 \, dx$$

- C was the edge set curve, and u was allowed to be discontinuous. Each term in the equation served a specific purpose: the first ensured the regularity of C, the second aimed to make u close to f, and the third was responsible for differentiating u within regions defined by C.
- However, the Mumford-Shah model was complex and computationally expensive. To simplify the process, the researchers decided to consider a piecewise constant formulation, introducing a fifth criterion:

# Chan-Vese method

- Mumford-Shah models are complicated and computationally expensive. As simplification, we consider a piecewise constant formulation and a fifth criteria:
  - The segmentation image has two regions

$$\arg_{u,C}(\mu Length(C)) + \lambda \int_{\Omega} (f(x) - u(x))^2 \, dx + \int_{\frac{\Omega}{C}} |\nabla \ u(x)|^2 \, dx$$

Right now we are shifting to Chan-Vese method which is inspired by the Mumford-Shah model.

By imposing the two regions, u is allowed to have only 2 values:

$$u(x) = \begin{cases} u_1(x), & \text{if } x \in C_1 \\ u_2(x), & \text{if } x \notin C_2 \end{cases}]$$

$\lambda$

# Chan-Vese method

So we can rewrite the equation as:

$$\arg_{u,C}(\mu Length(C)) + \ vArea(inside(C)) + \lambda_1 \int_{\text{inside}(C)} (f(x) - c_1)^2\, dx + \lambda_2 \int_{\text{outside}(C)} (f(x) - c_2)^2\, dx$$

To solve this problem we need to minimize the equation. By finding a local minimizer, we obtain a segmentation as the best two-phase piecewise approximation u of the image f.

To minimize the function we need to minimize over all set boundaries C.

C = $\{x \in \Omega : \varphi(x) = 0\}$

The inside and outside of C are distinguished by $\wp$ which is a level set function for a circle of radius r.

$$\varphi(x) = r - \sqrt{x_1^2 + x_2^2}$$

## Chan-Vese method

For a given C, there is more than one possible level set representation.

It $\varphi$ is a level set of $\Omega$, then so is any other function $\psi$ having the same sign:

$$sign(\psi(x)) = sign(\psi(x))$$

=> $\arg_{c_1, c_2, \varphi} \mu \int_\Omega \delta\left(\varphi(x)\left|\nabla\varphi(x)\right|\right)\, dx$ + $v \int_\Omega H\left(\varphi(x)\right) dx$ + $\lambda_1 \int_\Omega \left|f(x) - c_1\right|^2 H(\varphi(x))\, dx$ +

+ $\lambda_2 \int_\Omega \left|f(x) - c_2\right|^2 (1 - H(\varphi(x)))\, dx$

The final objective function they aimed to minimise included terms related to φ, Heaviside function (H), and Dirac mass (δ). H was used to regularise φ, and δ represented its distributional derivative.

# Chan-Vese method

$$H(t) = \begin{cases} 1 & \text{if } t \geq 0 \\ 0 & \text{if } t < 0 \end{cases}$$

$$\delta(t) = \frac{d}{dt} H(t)$$

Length of C is obtained as the total variation of $H(\varphi)$

$$Length(C) = \int_\Omega |\nabla H \varphi(x)| \, dx = \int_\Omega \delta(\varphi(x)) |\nabla \varphi(x)| \ , dx.$$

The minimization is solved by alternatingly updating $c_1$, $c_2$ and $\varphi$. For a fixed $\varphi$, the optimal values of $c_1$ and $c_2$ are the region averages.

## Chan-Vese method

We are obtaining:

$$c_1 = \frac{\int_\Omega f(x) H(\varphi(x))\, dx}{\int_\Omega H(\varphi(x))\, dx}$$

$$c_2 = \frac{\int_\Omega f(x)(1 - H(\varphi(x)))\, dx}{\int_\Omega (1 - H(\varphi(x)))\, dx}$$

For the minimization with respect to $\varphi$, H is regularized as:

$$H_\epsilon(t) = \frac{1}{2}\left(1 + \frac{2}{\pi}\arctan\left(\frac{t}{\epsilon}\right)\right)$$

and the Dirac function is:

$$\delta_\epsilon(t) := \frac{d}{dt} H_\epsilon(t) = \frac{\epsilon}{pi(\epsilon^2 + t^2)}$$

## Chan-Vese method

$$\varphi_{i,i}^{n+1} \leftarrow [\varphi_{i,i}^{n} + dt\delta_\epsilon(\varphi_{i,j}^{n})(A_{i,j,\varphi_{i+1,j}^{n}} + A_{i-1,j,\varphi_{i-1,j}^{n+1}} + B_{i,j,\varphi_{i,j+1}^{n}} + B_{i,j-1,\varphi_{i,j-1}^{n+1}} - \upsilon - \lambda_1(f_{i,j} - c_1)^2 + \lambda_2(f_{i,j} - c_2)^2 / [1 + dt\delta_\epsilon(\varphi_{i,j})(A_{i,j} + A_{i-1,j} + B_{i,j} + B_{i,j-1}]$$

In the end we obtain:

$$A_{i,j} = \frac{\mu}{\sqrt{\eta^2 + (\nabla_x + \varphi_{i,j})^2 + (\nabla_y^0 \varphi_{i,j})^2}} \qquad B_{i,j} = \frac{\mu}{\sqrt{\eta^2 + (\nabla_x^0 + \varphi_{i,j})^2 + (\nabla_y \varphi_{i,j})^2}}$$

## 3. Code implementation - PseudoCode

**Initialize:**

- Input the image to be segmented.
- Define an initial contour (C0) and initialize the level set function Φ accordingly.
- Set parameters: λ1, λ2 (weight parameters), μ (length penalty), ν (area penalty), ε (stopping criteria), and maximum number of iterations.
- Set reinitialization frequency (e.g., every 10 iterations).

**Algorithm:**

While the stopping criteria is not met and the number of iterations is less than the maximum:

1. Inside(C) = average intensity inside the current contour based on Φ.
2. Outside(C) = average intensity outside the current contour based on Φ.

# 3. Code implementation - PseudoCode

```
3. Compute the energy functional based on the current contour Φ:
   $Energy(\Phi) = \lambda_1 \int_{\text{inside}(\Phi)} |image(x,y) - \text{Inside}(\Phi)|^2 \, dxdy + \lambda_2 \int_{\text{outside}(\Phi)} |image(x,y) - \text{Outside}(\Phi)|^2 \, dxdy + \mu \cdot Length(\Phi) + \nu \cdot Area(\text{inside}(\Phi))$

4. Evolve the level set function Φ to minimize the energy.

5. If the current iteration is a multiple of the reinitialization frequency:
    - Reinitialize Φ to a signed distance function to maintain stability.

6. Update the contour C based on the evolved level set function Φ.
```

## 3. Code implementation - PseudoCode

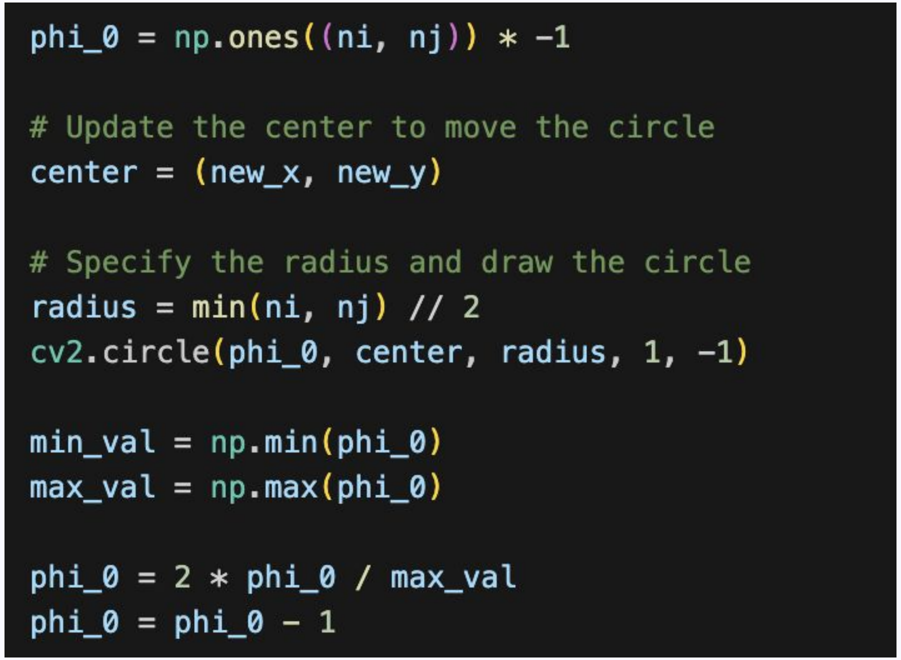
```
7. Check for convergence:
    - If the change in energy between iterations is less than ε, stop.
    - If the maximum number of iterations is reached, stop.
```

**Output:**
- Final contour (C) that segments the image.
- Optionally, segmented regions can be displayed or extracted based on the final contour.

## 3. Code implementation - phi

Initial $\varphi$

$$\varphi(x,y) = \begin{cases} 1, & \text{if } (x,y) \text{ is inside the circle} \\ -1, & \text{if } (x,y) \text{ is outside the circle} \end{cases}$$

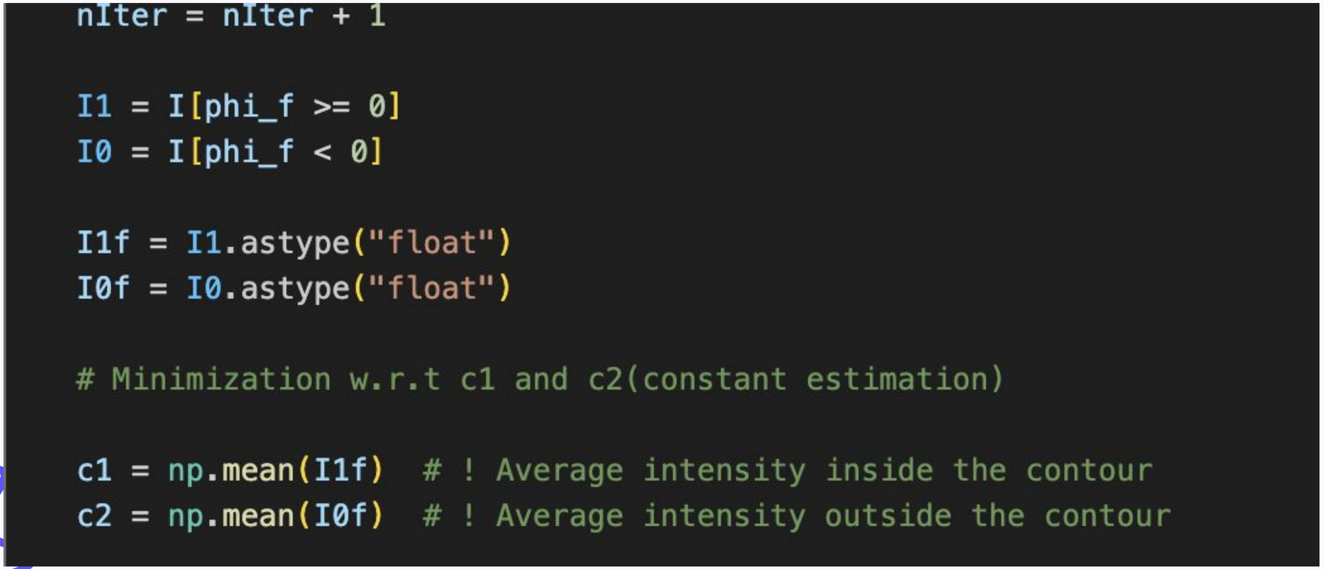


```python
phi_0 = np.ones((ni, nj)) * -1

# Update the center to move the circle
center = (new_x, new_y)

# Specify the radius and draw the circle
radius = min(ni, nj) // 2
cv2.circle(phi_0, center, radius, 1, -1)

min_val = np.min(phi_0)
max_val = np.max(phi_0)

phi_0 = 2 * phi_0 / max_val
phi_0 = phi_0 - 1
```

## 3. Code implementation - c1 & c2

$$c_1 = \frac{\int_\Omega f(x)H(\varphi(x))\,dx}{\int_\Omega H(\varphi(x))\,dx}$$

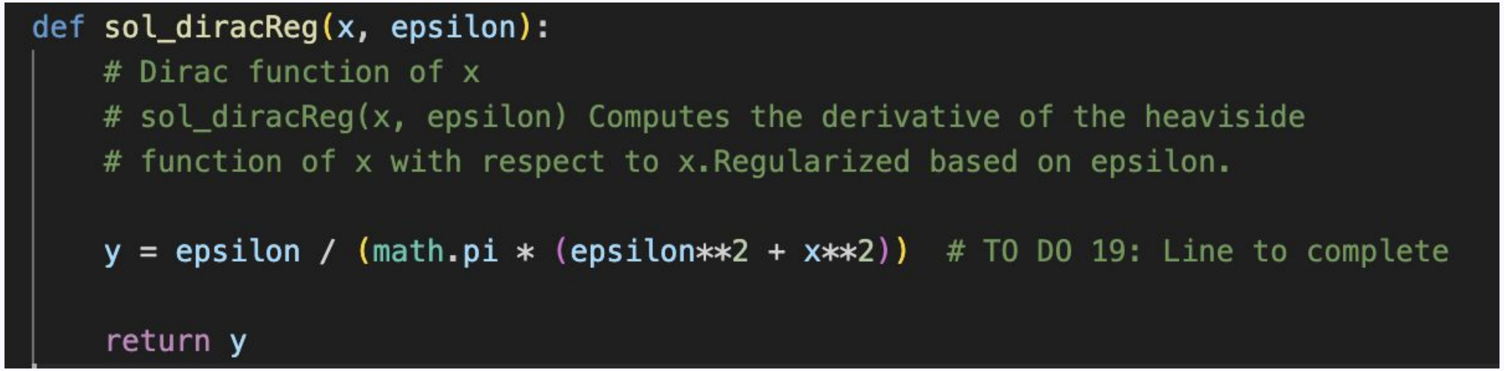
$$c_2 = \frac{\int_\Omega f(x)(1 - H(\varphi(x)))\,dx}{\int_\Omega (1 - H(\varphi(x)))\,dx}$$



```
    nIter = nIter + 1

    I1 = I[phi_f >= 0]
    I0 = I[phi_f < 0]

    I1f = I1.astype("float")
    I0f = I0.astype("float")

    # Minimization w.r.t c1 and c2(constant estimation)

    c1 = np.mean(I1f)  # ! Average intensity inside the contour
    c2 = np.mean(I0f)  # ! Average intensity outside the contour
```

## 3. Code implementation - Dirac

$$\delta_\epsilon(t) := \frac{d}{dt} H_\epsilon(t) = \frac{\epsilon}{pi(\epsilon^2 + t^2)}$$

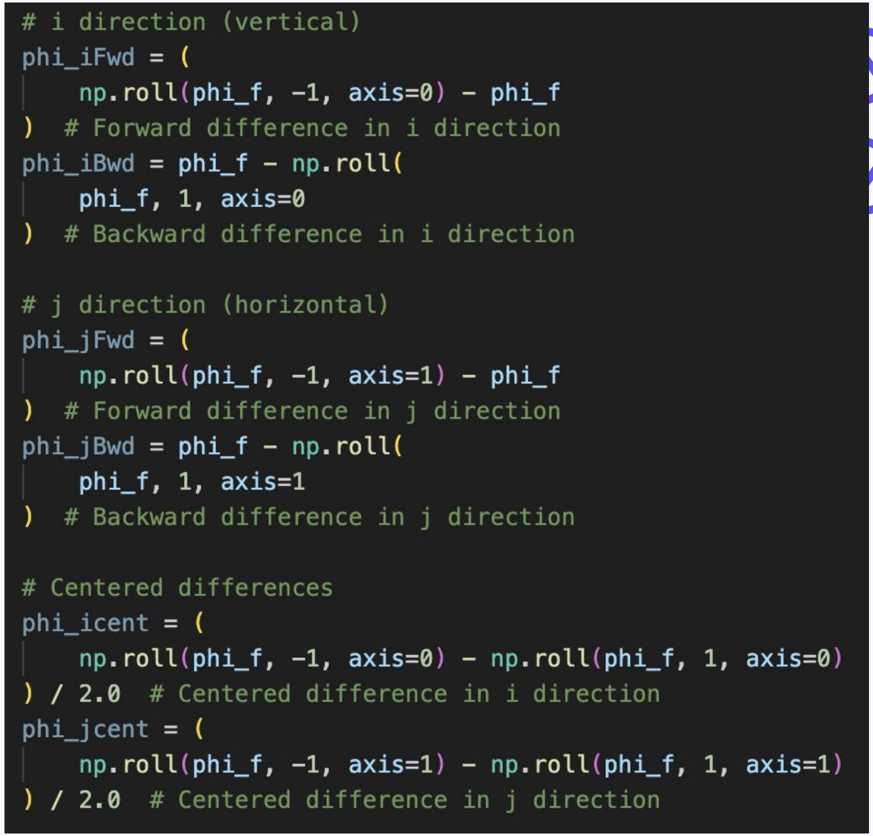


```python
def sol_diracReg(x, epsilon):
    # Dirac function of x
    # sol_diracReg(x, epsilon) Computes the derivative of the heaviside
    # function of x with respect to x.Regularized based on epsilon.

    y = epsilon / (math.pi * (epsilon**2 + x**2))  # TO DO 19: Line to complete

    return y
```

# 3. Code implementation - Derivatives

$$A_{i,j} = \frac{\mu}{\sqrt{\eta^2 + (\nabla_x + \varphi_{i,j})^2 + (\nabla_y^0 \varphi_{i,j})^2}}$$

$$B_{i,j} = \frac{\mu}{\sqrt{\eta^2 + (\nabla_x^0 + \varphi_{i,j})^2 + (\nabla_y \varphi_{i,j})^2}}$$

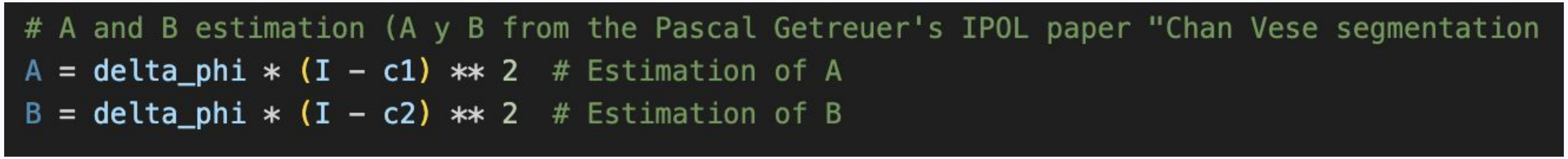


```python
# i direction (vertical)
phi_iFwd = (
    np.roll(phi_f, -1, axis=0) - phi_f
)  # Forward difference in i direction
phi_iBwd = phi_f - np.roll(
    phi_f, 1, axis=0
)  # Backward difference in i direction

# j direction (horizontal)
phi_jFwd = (
    np.roll(phi_f, -1, axis=1) - phi_f
)  # Forward difference in j direction
phi_jBwd = phi_f - np.roll(
    phi_f, 1, axis=1
)  # Backward difference in j direction

# Centered differences
phi_icent = (
    np.roll(phi_f, -1, axis=0) - np.roll(phi_f, 1, axis=0)
) / 2.0  # Centered difference in i direction
phi_jcent = (
    np.roll(phi_f, -1, axis=1) - np.roll(phi_f, 1, axis=1)
) / 2.0  # Centered difference in j direction
```

## 3. Code implementation - Derivatives

$$A_{i,j} = \frac{\mu}{\sqrt{\eta^2 + (\nabla_x + \varphi_{i,j})^2 + (\nabla_y^0 \varphi_{i,j})^2}}$$

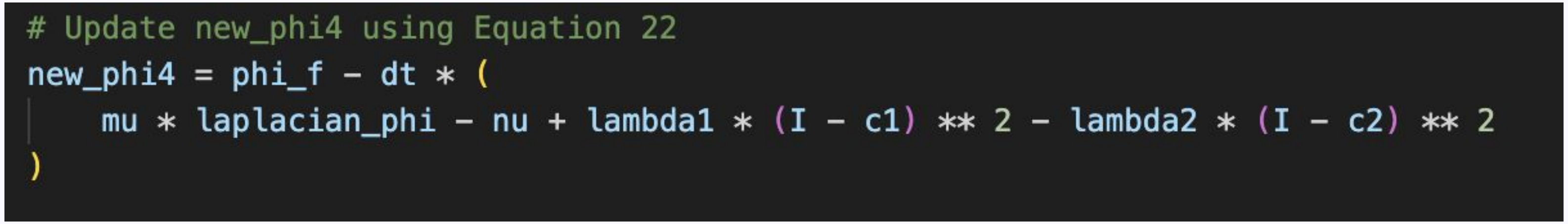
$$B_{i,j} = \frac{\mu}{\sqrt{\eta^2 + (\nabla_x^0 + \varphi_{i,j})^2 + (\nabla_y \varphi_{i,j})^2}}$$



```
# A and B estimation (A y B from the Pascal Getreuer's IPOL paper "Chan Vese segmentation
A = delta_phi * (I - c1) ** 2  # Estimation of A
B = delta_phi * (I - c2) ** 2  # Estimation of B
```

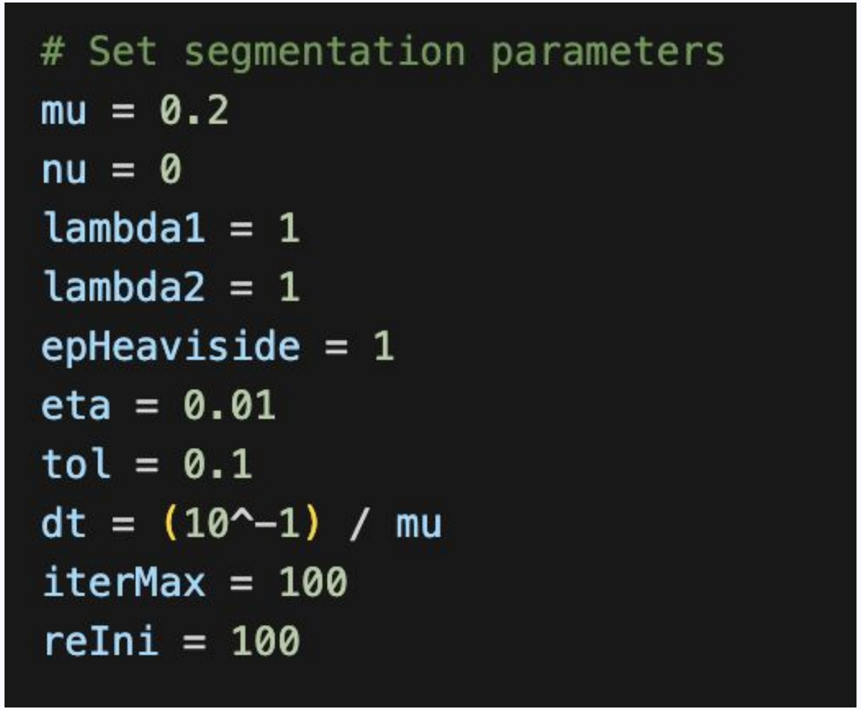
## 3. Code implementation - $\varphi$ (n+1)

$$\varphi_{i,i}^{n+1} \leftarrow [\varphi_{i,i}^{n} + dt\delta_{\epsilon}(\varphi_{i,j}^{n})(A_{i,j,\varphi_{i+1,j}^{n}} + A_{i-1,j,\varphi_{i-1,j}^{n+1}} + B_{i,j,\varphi_{i,j+1}^{n}} + B_{i,j-1,\varphi_{i,j-1}^{n+1}} - \upsilon - \lambda_1(f_{i,j} - c_1)^2 + \lambda_2(f_{i,j} - c_2)^2/[1 + dt\delta_{\epsilon}(\varphi_{i,j})(A_{i,j} + A_{i-1,j} + B_{i,j} + B_{i,j-1}]$$



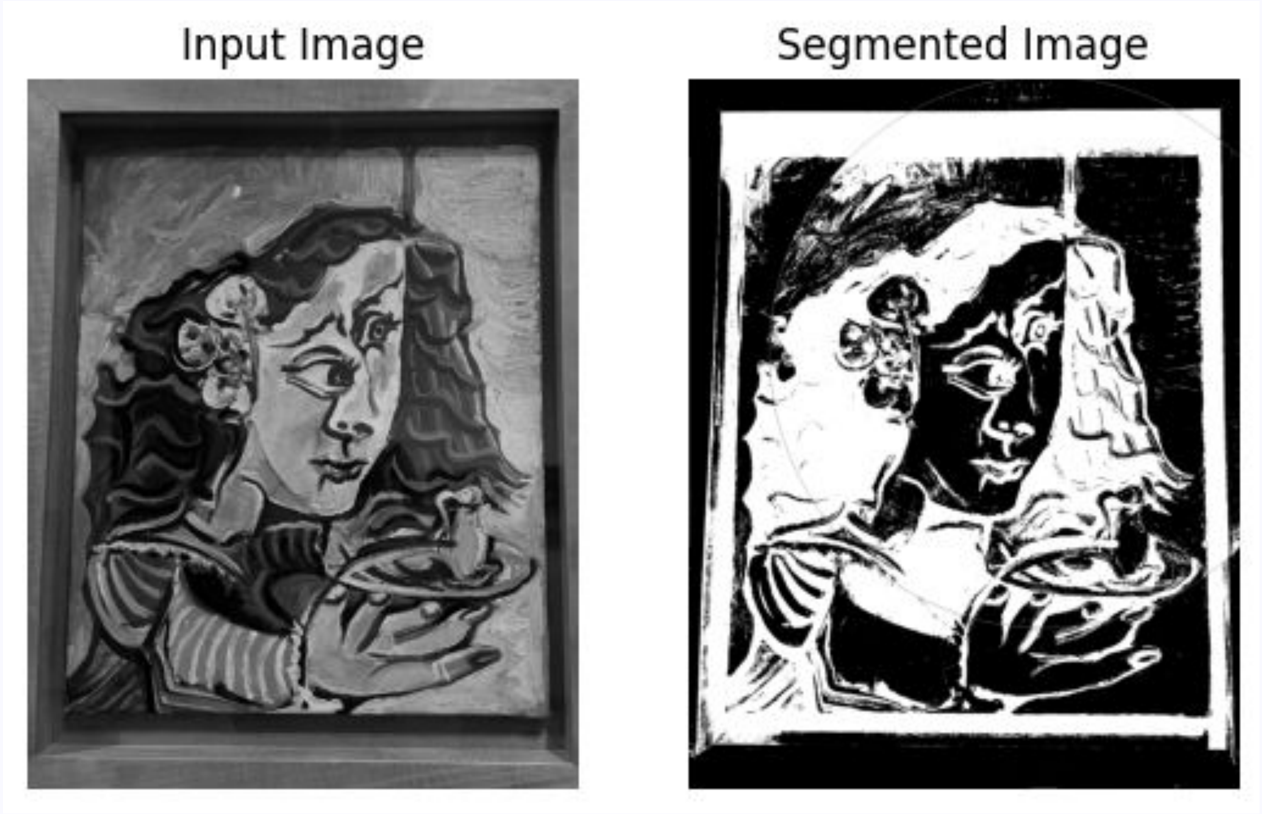
```
# Update new_phi4 using Equation 22
new_phi4 = phi_f - dt * (
    mu * laplacian_phi - nu + lambda1 * (I - c1) ** 2 - lambda2 * (I - c2) ** 2
)
```

## 3. Code implementation - Initializations

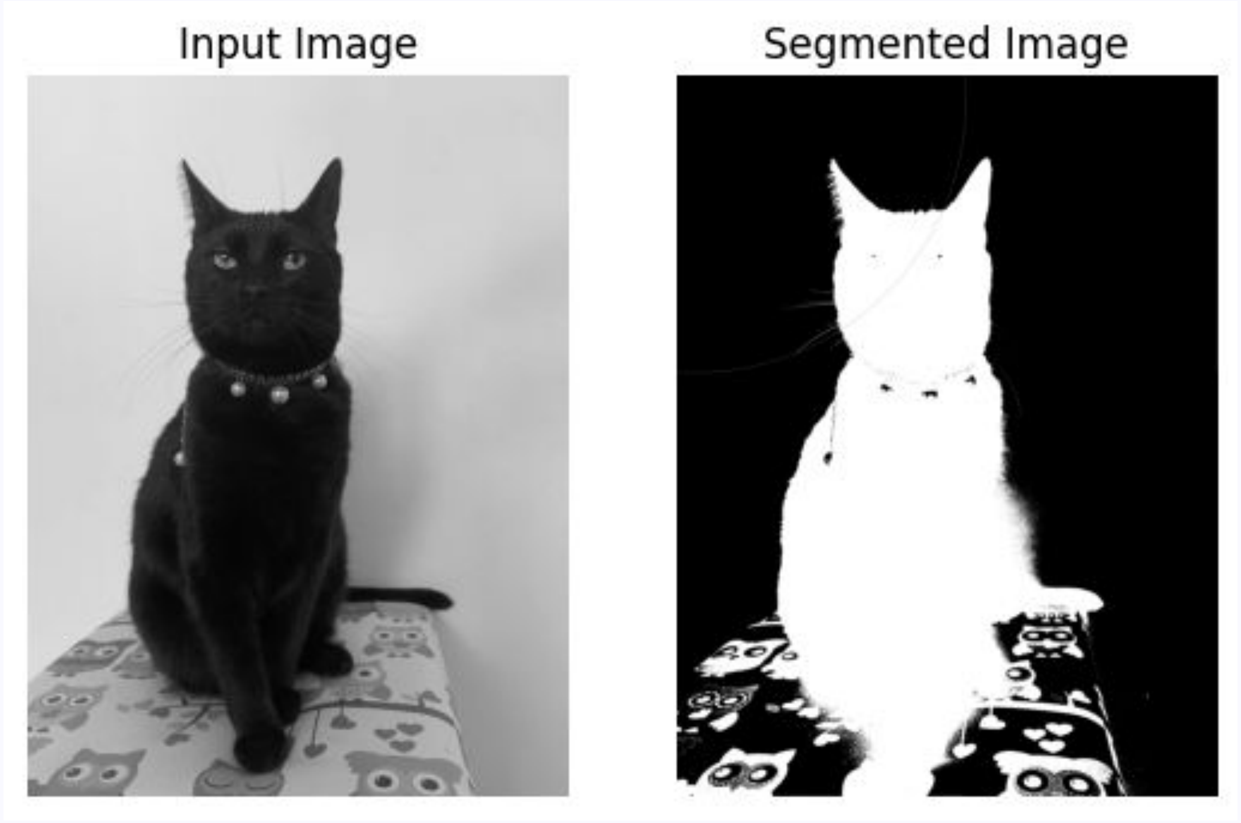


```
# Set segmentation parameters
mu = 0.2
nu = 0
lambda1 = 1
lambda2 = 1
epHeaviside = 1
eta = 0.01
tol = 0.1
dt = (10^-1) / mu
iterMax = 100
reIni = 100
```

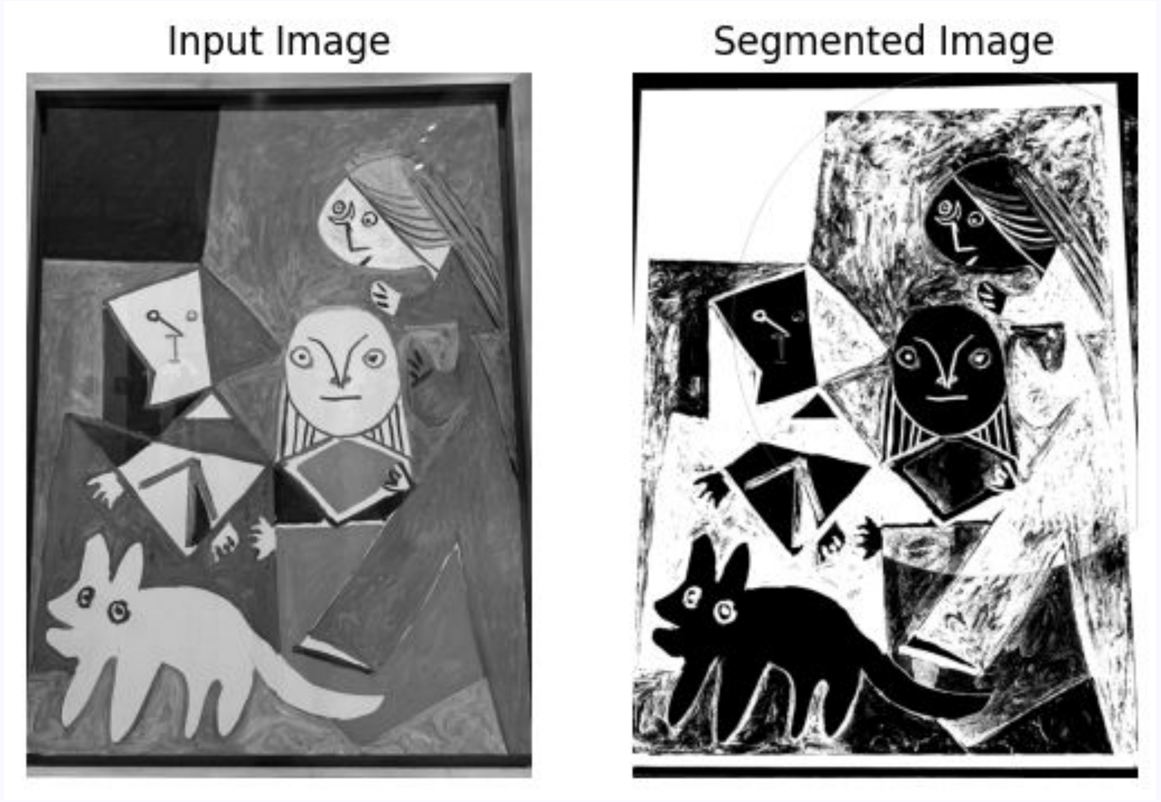
## 4. Results



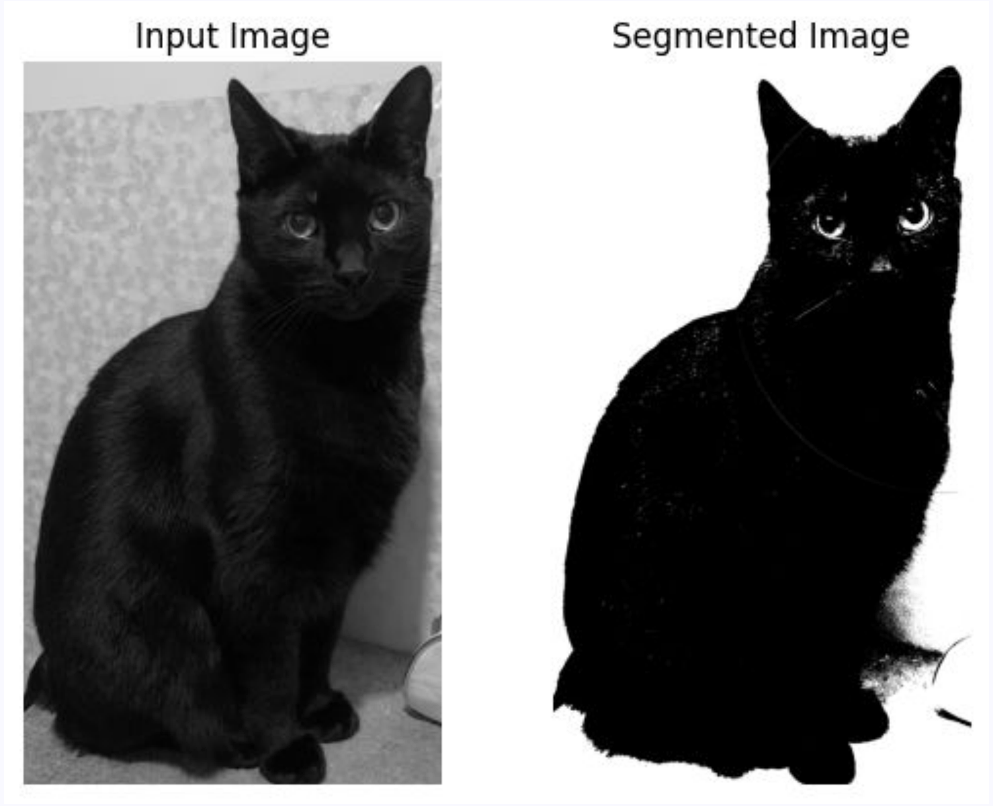
## 4. Results



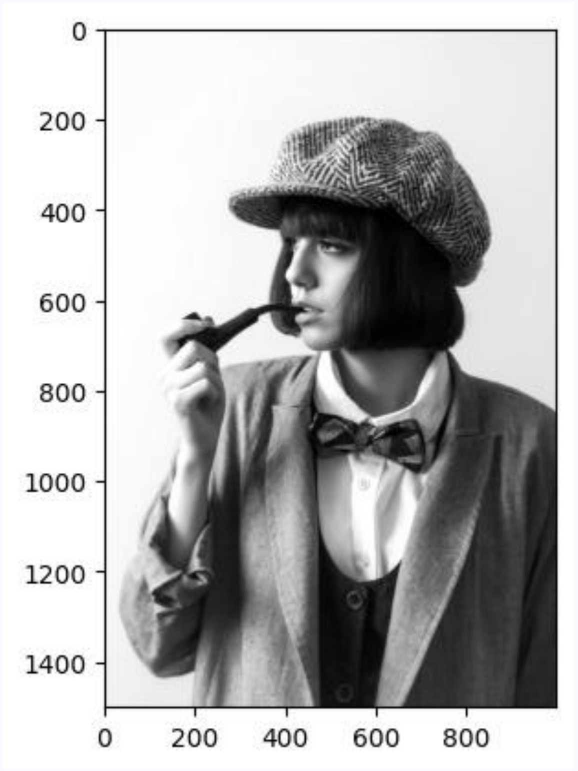
## 4. Results

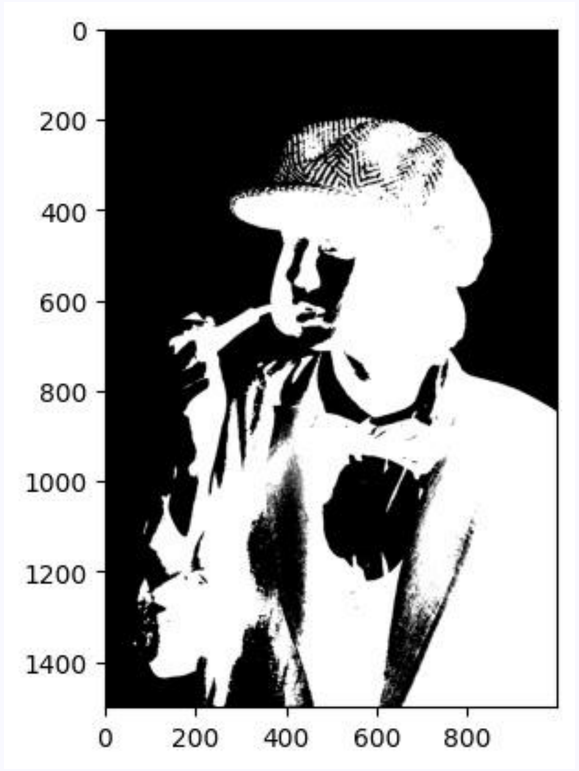


## 4. Results

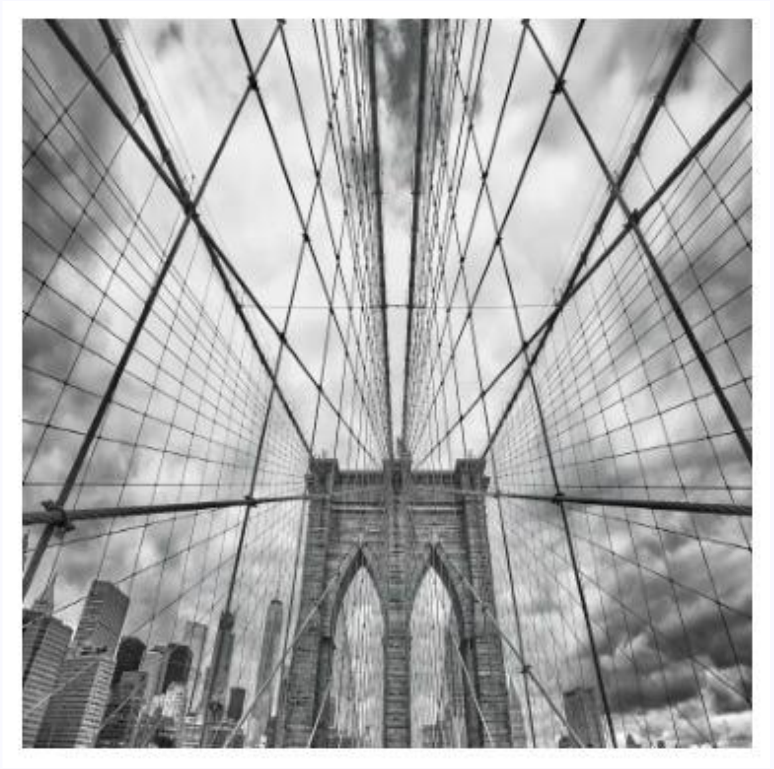


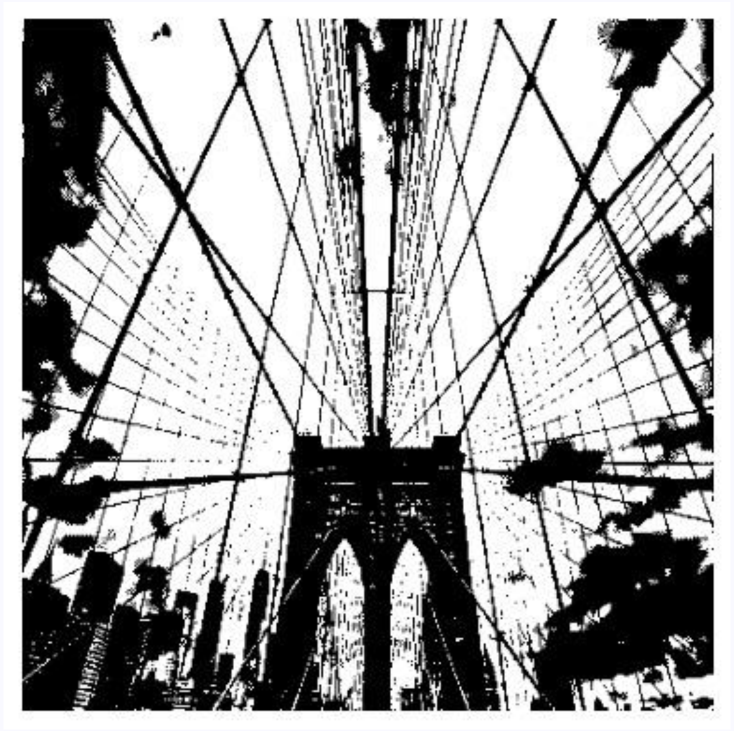
## 4. Results


0
200
400
600
800
1000
1200
1400
0
200
400
600
800

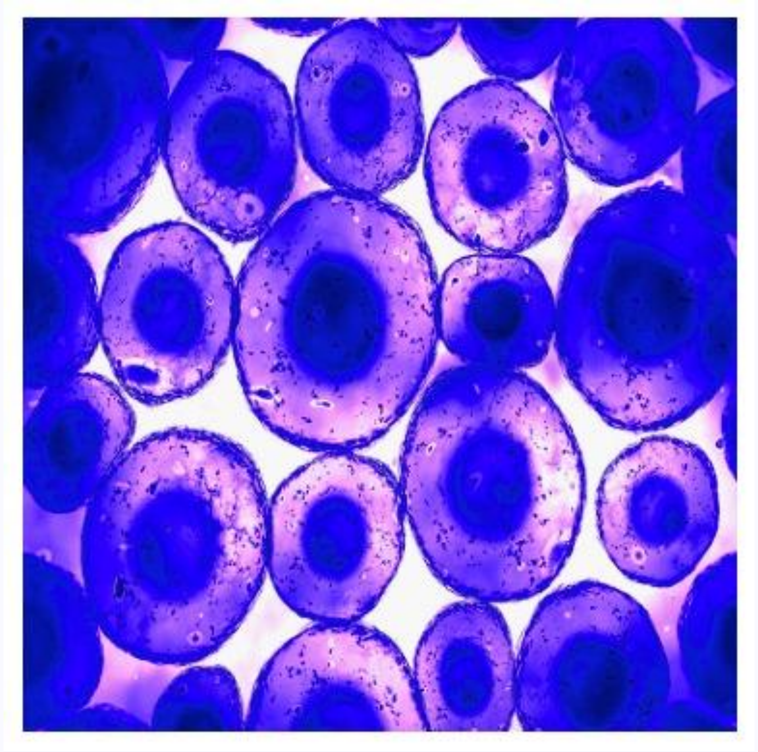
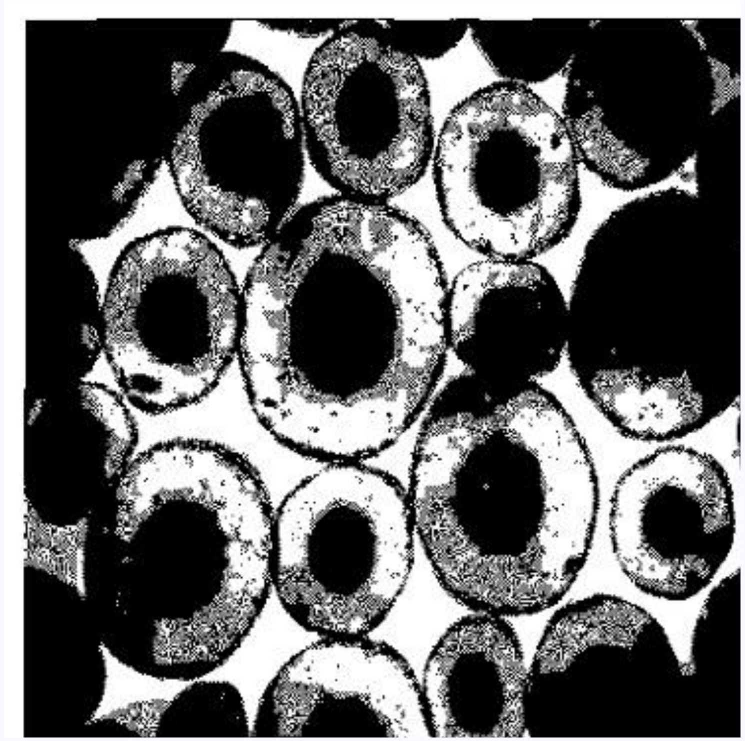

0
200
400
600
800
1000
1200
1400
0
200
400
600
800

## 4. Results

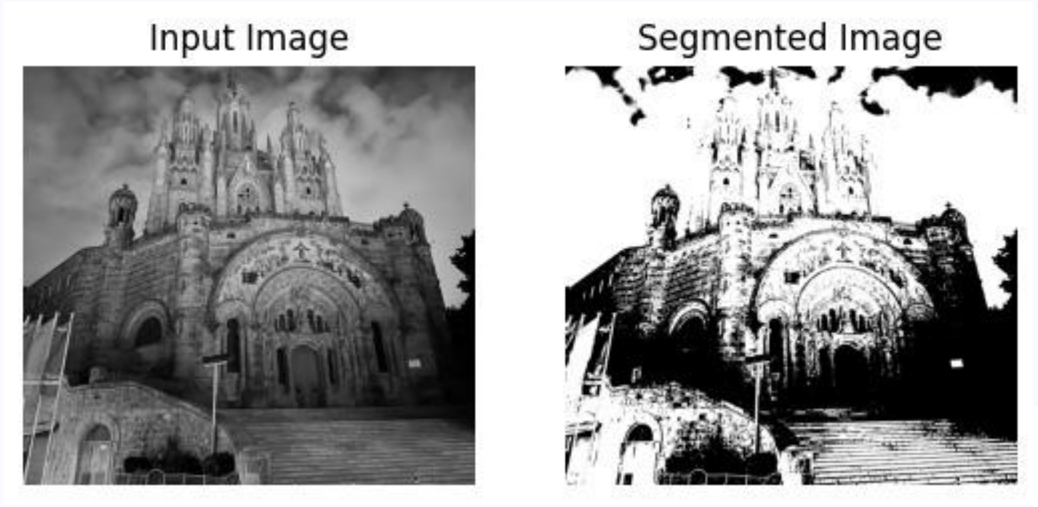


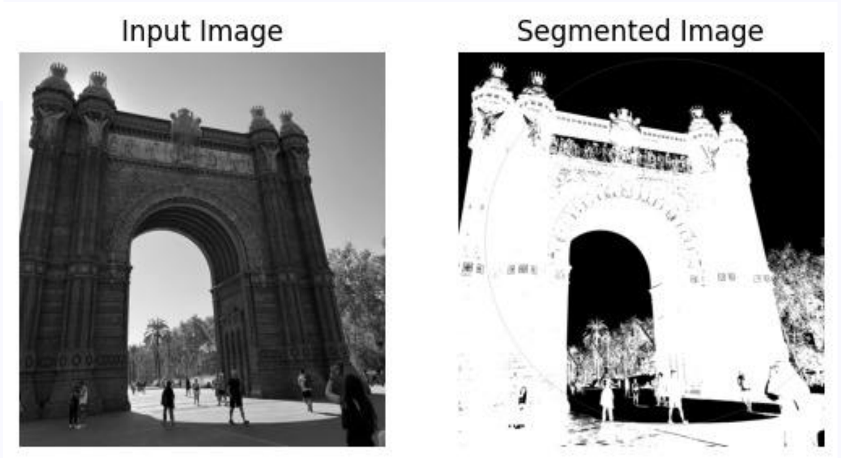


## 4. Results

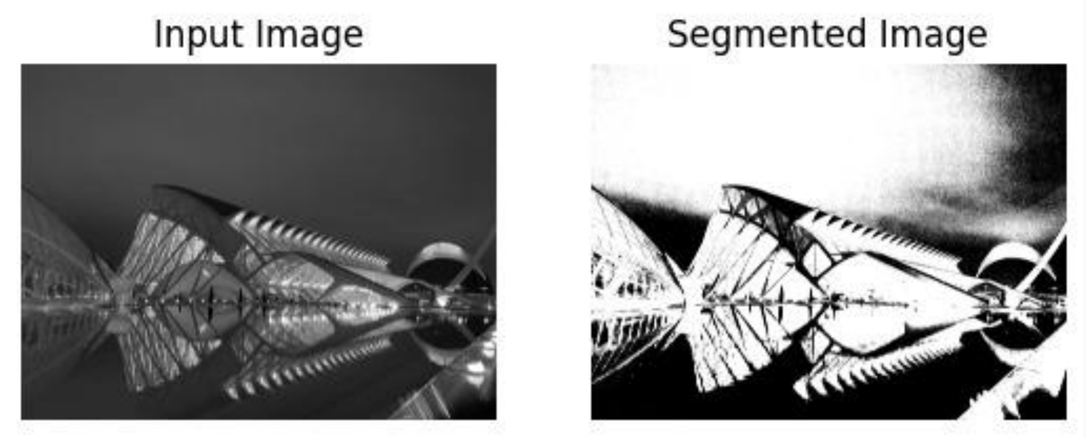



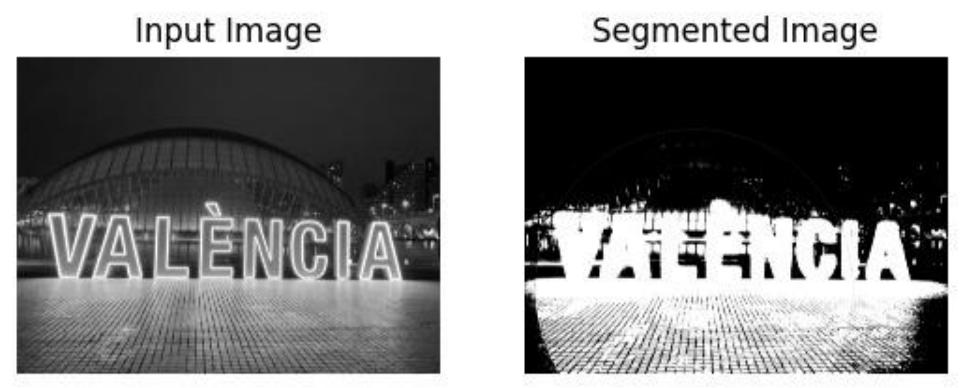
## 4. Results - not so good because of the method's limitations





## 4. Results - not so good because of the method's limitations





## 5. Conclusions

The Chan-Vese method aims to segment an object of interest from an image by evolving a contour to minimize an energy functional.

**Advantages:**

- It can segment objects with irregular shapes and topologies.
- It is relatively robust to noise in the image.
- It doesn't require prior knowledge or initialization of the object shape.
- It can be adapted for both grayscale and color images.

**Limitations:**

- Initialization Dependency: The segmentation result can be sensitive to the choice of the initial contour.
- Computational Cost: The level set evolution process can be computationally intensive, making it less suitable for real-time applications.
- Not Suitable for All Images: It may not work well for images with multiple objects of interest or complex scenes.

## 5. Conclusions

**Differences from Classical Segmentation Methods:**

- Active Contour vs. Classical Methods: Active contour methods like Chan-Vese evolve a contour to capture the object of interest, while classical methods (e.g., thresholding, edge detection) rely on fixed criteria to distinguish object from background.

- Adaptability: Active contour methods can adapt to object shapes and topologies, making them useful for segmenting objects with complex boundaries. Classical methods are often less flexible in handling irregular objects.

- Initialization: Active contour methods require initialization, while classical methods often do not.

- Noise Robustness: Active contour methods can be more robust to noise since they evolve the contour based on local image information. Classical methods may be sensitive to noise.

## 5. Conclusions

- Complexity: Active contour methods are more complex in terms of their underlying mathematics and algorithmic implementation compared to classical methods, which are simpler and more intuitive.

In summary, the Chan-Vese segmentation method is a powerful approach for segmenting objects with irregular shapes in noisy images. However, it has its limitations and may not always be the best choice, especially for real-time applications or images with multiple objects.

**GitHub link of our repository:**

https://github.com/gunjanmimo/C2-IMAGE-INPAITING