UNIVERSITY OF Waterloo

Implementing

Active contours without edges

Tony F. Chan & Luminita A. Vese

A logic framework for active contours on multi-channel images

Berta Sandberg & Tony F. Chan

Project by: Sarah Nadi & Karim Ali

Dec. 13Th 2010 – CS 870

Outline

- Active contours without edges
  - Model
  - Implementation
  - Experimental results
- Logic framework for multichannel images
  - Model
  - Implementation
  - Experimental results
- Conclusions

Active Contour Models

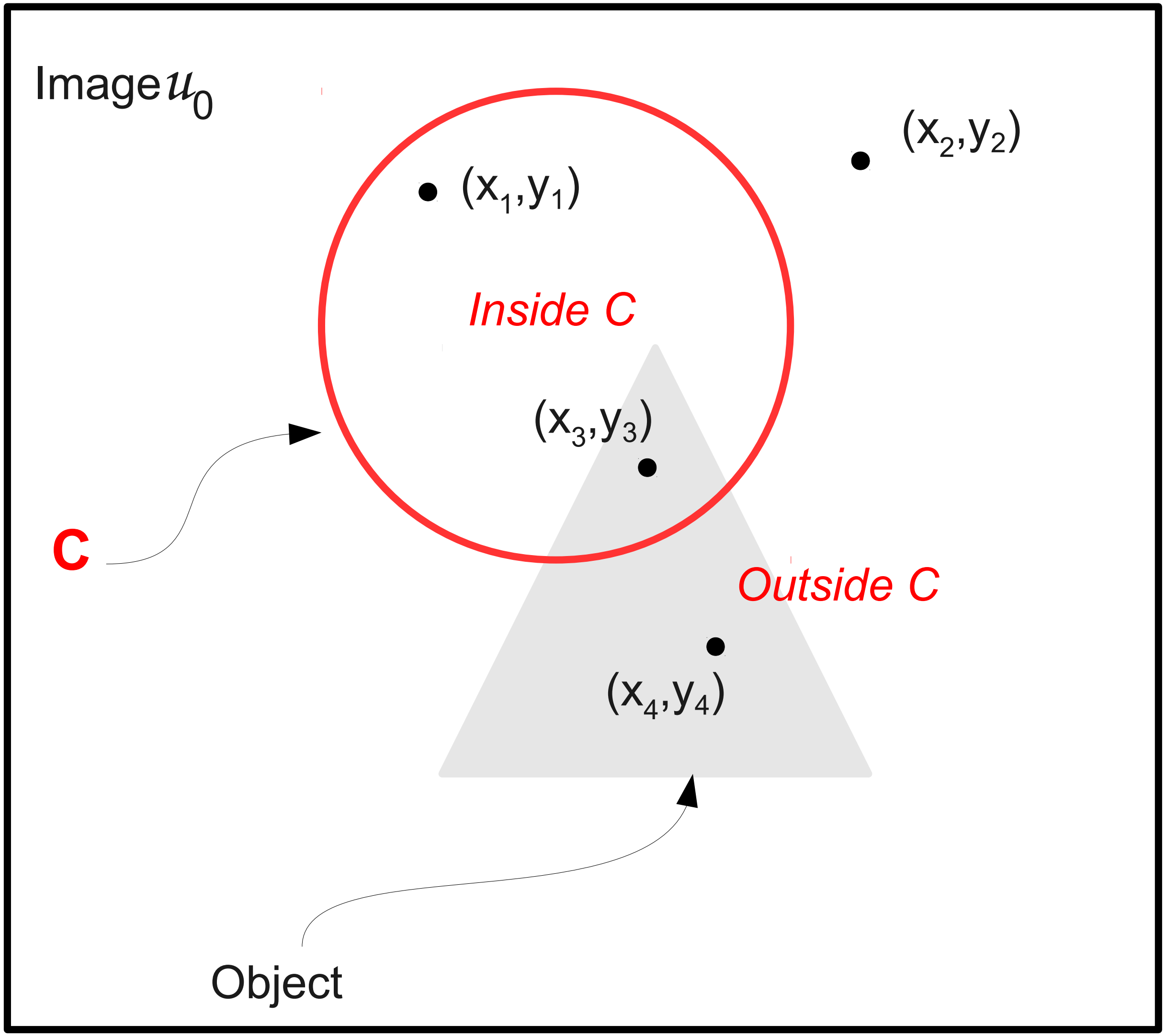
- Overview
  - Detect objects in an image
  - Use image gradient to evaluate the stopping condition

- Limitations
  - Can only detect edges defined by a gradient
    - Edges can be smoothed
    - Contour may pass through the boundary

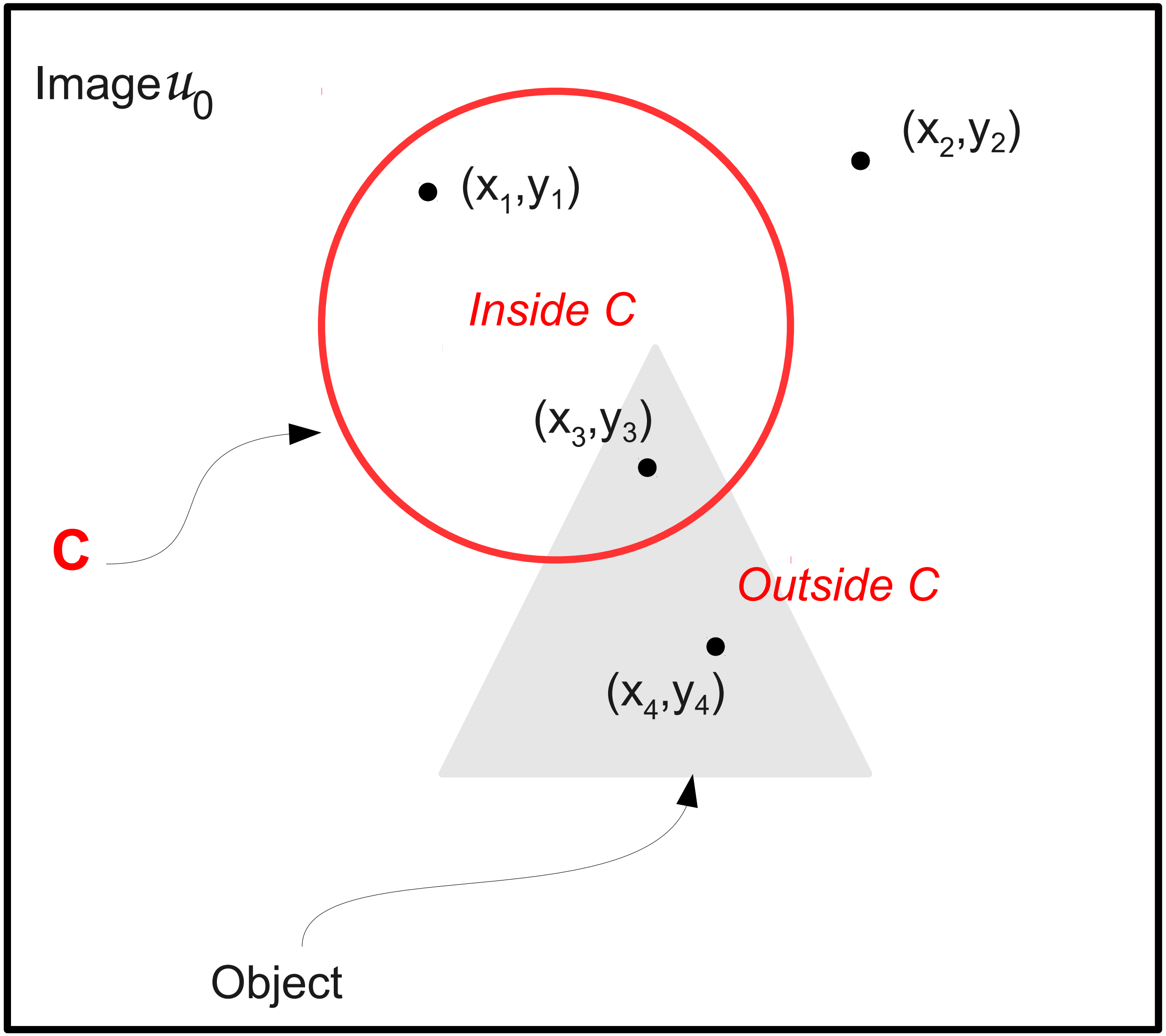
Active Contours Without Edges (Chan-Vese)

- Region based technique
- Stopping is based on Mumford-Shah segmentation techniques
  - No longer depends on gradient
  - Detects smooth or discontinuous boundaries
- Interior contours are automatically detected
- Initial curve can be anywhere in the image

Region Based View



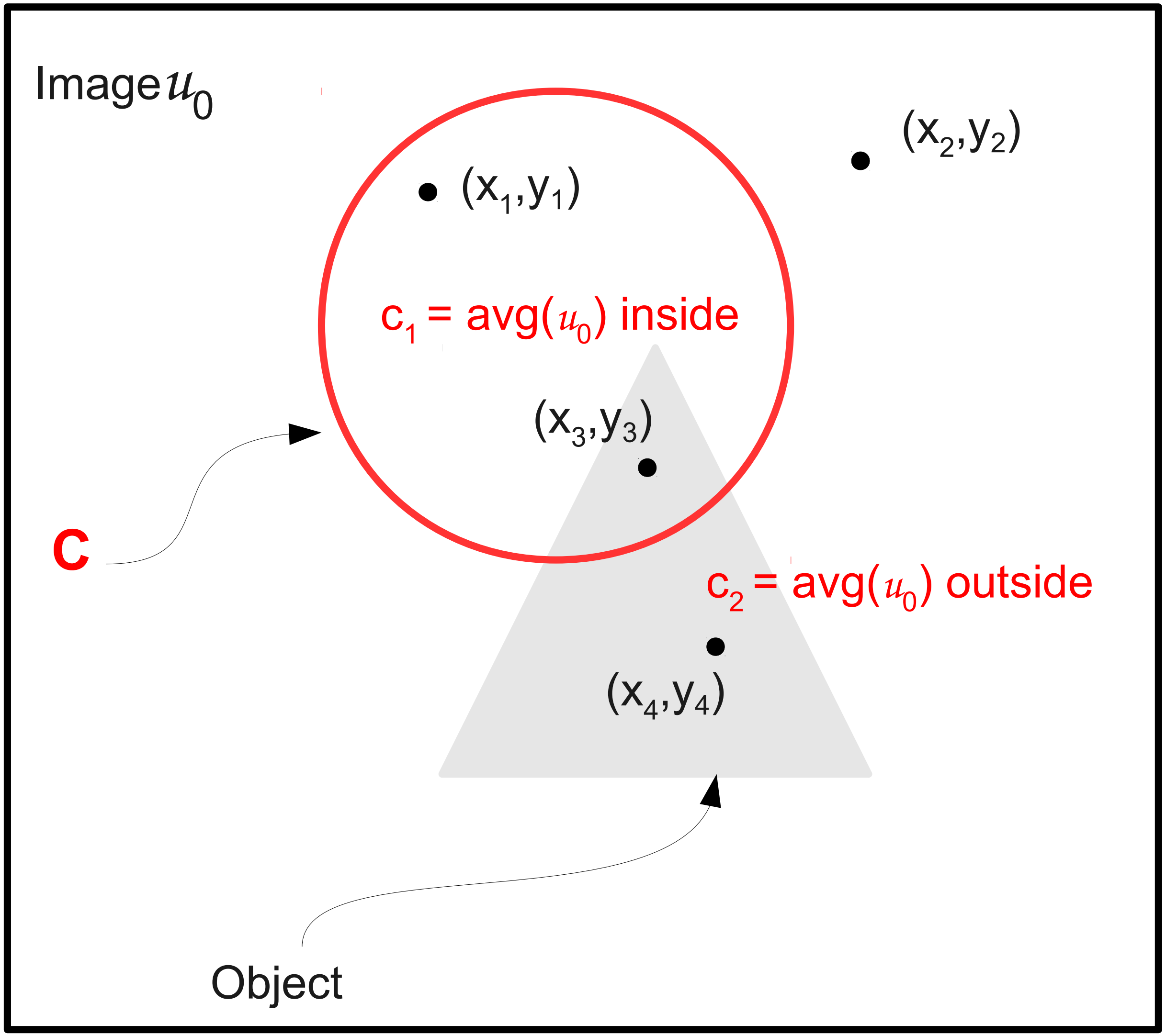
Region Based View



To fit C to our object, we **want**:

- (x_1,y_1) to be **outside** C
- (x_2,y_2) to be **outside** C
- (x_3,y_3) to be **inside** C
- (x_4,y_4) to be **inside** C

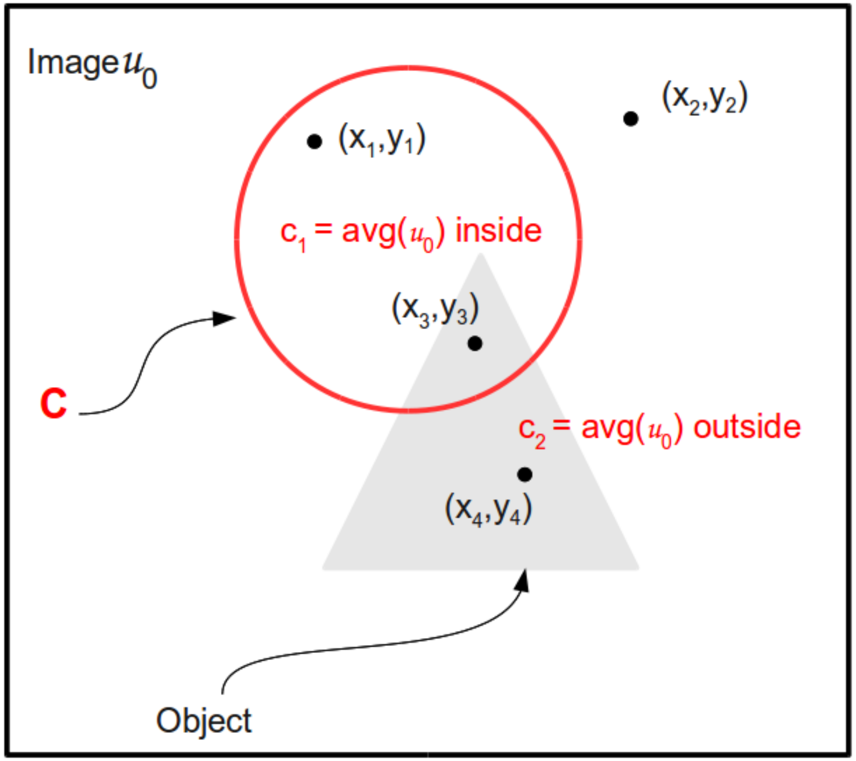
Region Based View



To fit C to our object, we **want**:

- (x_1,y_1) to be **outside** C
- (x_2,y_2) to be **outside** C
- (x_3,y_3) to be **inside** C
- (x_4,y_4) to be **inside** C

Curve Fitting



Fitting inside:

$$F_1(C) = \int_{inside(C)} |u_0(x,y) - c_1|^2 \, dx \, dy$$

Fitting outside:

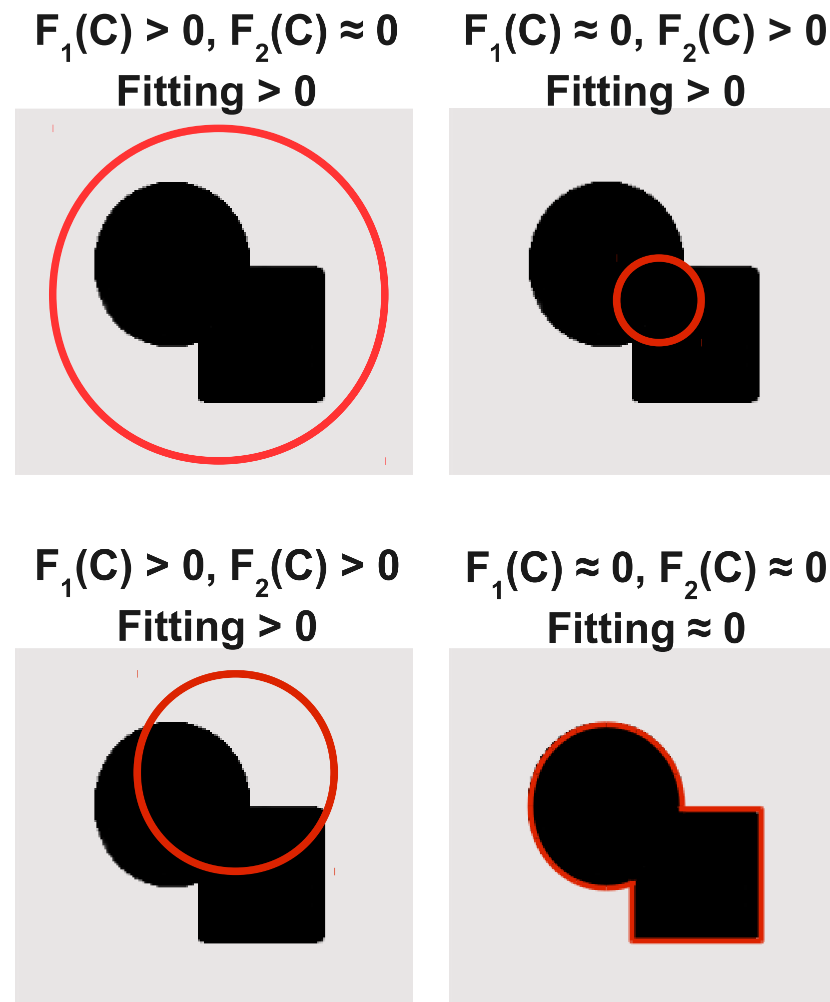
$$F_2(C) = \int_{outside(C)} |u_0(x,y) - c_2|^2 \, dx \, dy$$

Overall fitting of C:

$$\begin{aligned} F_1(C) + F_2(C) &= \int_{inside(C)} |u_0(x,y) - c_1|^2 \, dx \, dy \\ &+ \int_{outside(C)} |u_0(x,y) - c_2|^2 \, dx \, dy \end{aligned}$$

Intuition

- C should stop evolving when C = Object
- To achieve that: Fitting ≈ 0



Chan-Vese PDE

$$\frac{\partial \phi}{\partial t} = \delta_\varepsilon(\phi)[\mu \operatorname{div}(\frac{\nabla \phi}{|\nabla \phi|}) - \nu - \lambda_1(u_0 - c_1)^2 + \lambda_2(u_0 - c_2)^2]$$

- Initialize Φ^0 by Φ_0, n = 0
- Compute $c_1(\Phi^n)$ and $c_2(\Phi^n)$
- Solve the PDE to obtain Φ_{n+1}
- Reinitialize Φ locally to the signed distance function to the curve (optional)
- Check whether the solution is stationary. If not n = n + 1 and repeat

Implementation

- Finite difference discretization with an explicit time stepping scheme $\phi^{n+1} = \phi^n + \Delta t * \phi_t$

- $$\begin{aligned}\frac{\partial \phi}{\partial t} &= |\nabla \phi|[\mu \operatorname{div}(\frac{\nabla \phi}{|\nabla \phi|}) - \nu - \lambda_1(u_0 - c_1)^2 \\ &\quad + \lambda_2(u_0 - c_2)^2]\end{aligned}$$

- Stopping condition: $(\phi^{n+1} > 0) == (\phi_n > 0)$
- No reinitialization was necessary

Experimental Results

Evaluation Criteria

Criteria	Goal
Detecting Boundaries	Ability to correctly detect object boundaries of simple objects
Curve Position	Ability to correctly detect object boundaries irrespective of the initial curve position
Detecting Holes	Ability to detect holes in objects, and not simply stop on outside boundary
Blurred Images	Ability to correctly (as much as possible) detect object boundaries in blurred images
Noisy Images	Ability to correctly (as much as possible) detect object boundaries in noisy images
Parameter Settings	Ability to respond correctly to the different parameter settings

Detecting Boundaries

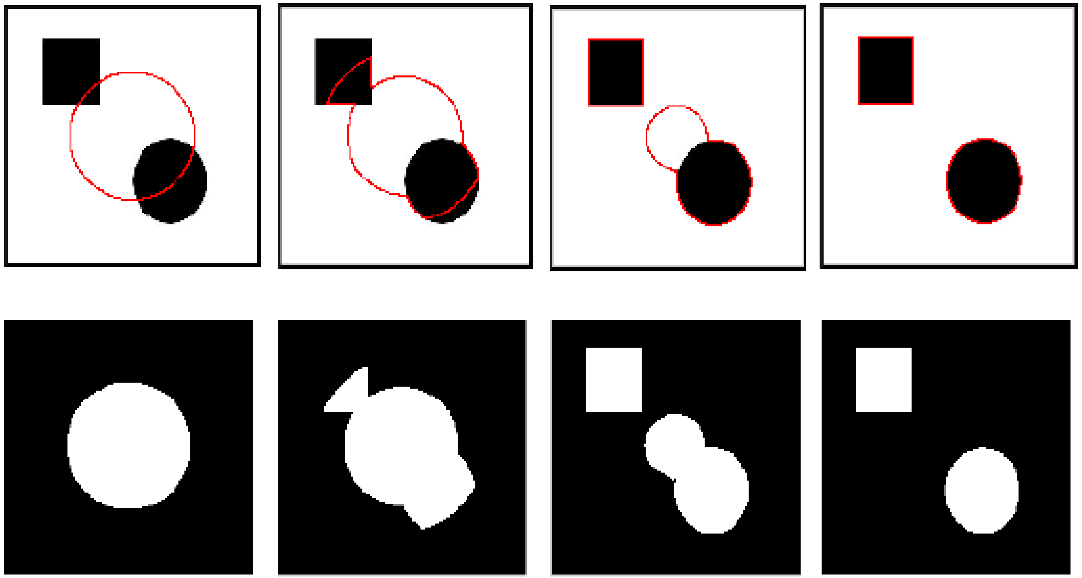


Figure 4. Successful detection of object boundaries. Top: the evolving curve (in red) over time where the first image shows the initial contour. Bottom: evolving segmentation over time until the objects are detected. Size = 300 x 300, $\phi_0(x, y) = -\sqrt{(x-150)^2 + (y-150)^2} + 75$, $\mu = 0.01$, no reinitialization, cpu = 2.9s, iterations = 7.

Detecting Boundaries *Cont'd*

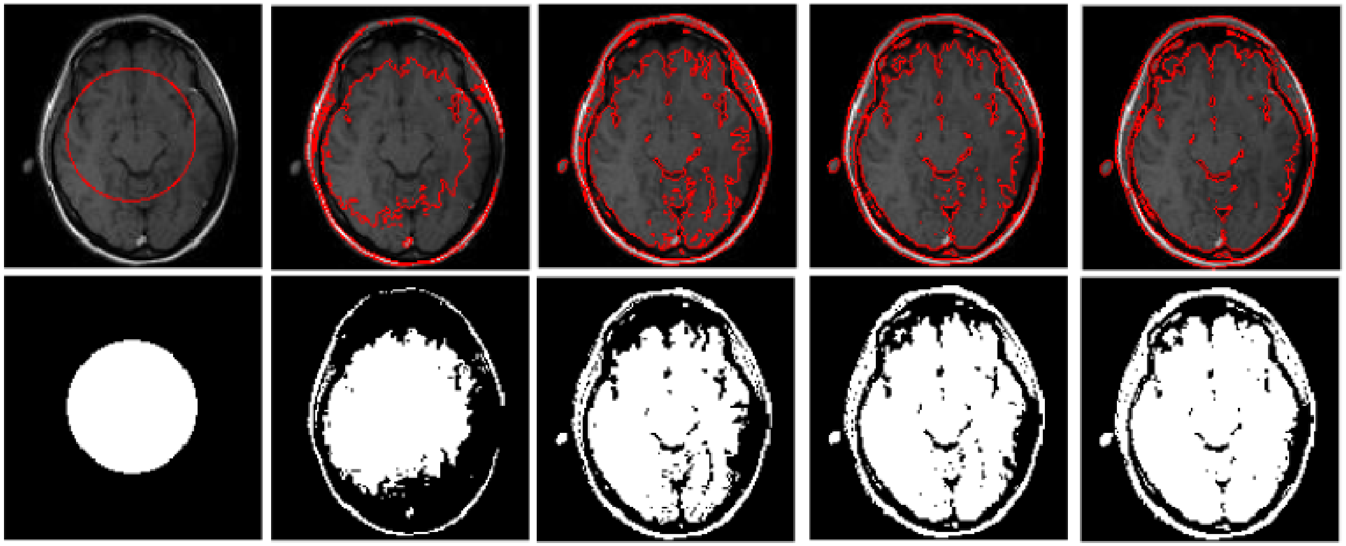


Figure 5. Successful detection of object boundaries. Top: the evolving curve (in red) over time where the first image shows the initial contour. Bottom: evolving segmentation over time until the object is detected. Size = 131 x 131, $\phi_0(x, y) = -\sqrt{(x - 65.6)^2 + (y - 65.5)^2} + 32.8$, $\mu = 0.01$, no reinitialization, cpu = 1.95 s, iterations = 7.

Detecting Boundaries *Cont'd*

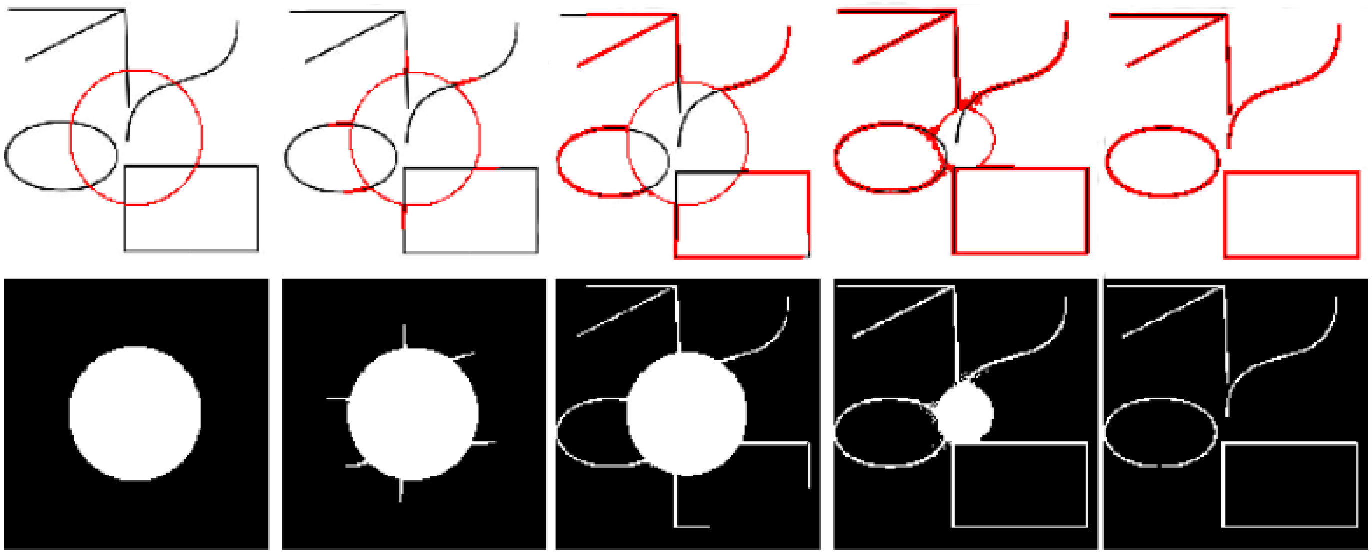


Figure 6. Successful detection of object with open boundaries. Top: the evolving curve (in red) over time where the first image shows the initial contour. Bottom: evolving segmentation over time until all objects are detected. Size = 300 x 300, $\phi_0(x, y) = -\sqrt{(x-150)^2 + (y-150)^2} + 75$, $\mu = 0.01$, no reinitialization, cpu = 5.19 s, iterations = 11.

Detecting Boundaries *Cont'd*

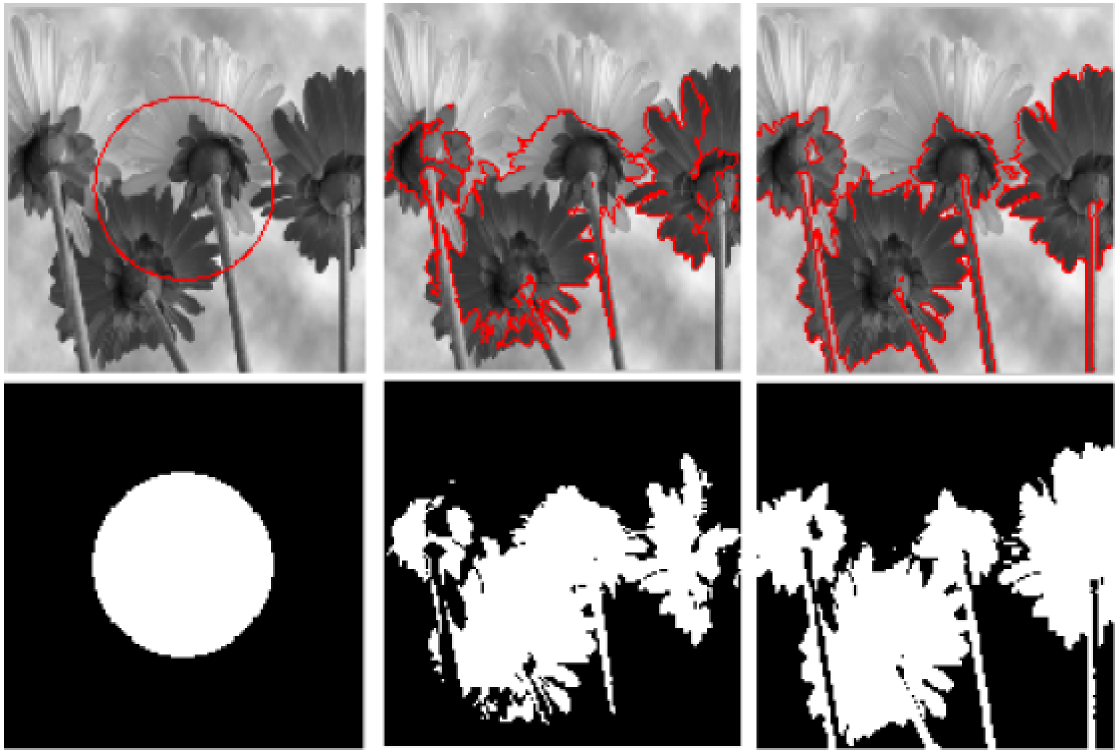


Figure 11. Inability to detect low intensities that do not have much variation from their background. Top: the evolving curve (in red) over time where the first image shows the initial contour. Bottom: evolving segmentation over time until the object is detected. Size = 300 x 300, $\phi_0(x, y) = -\sqrt{(x-150)^2 + (y-150)^2} + 75$, $\mu = 0.01$, no reinitialization, cpu = 1.35 s, iterations = 6.

Initial Curve Position

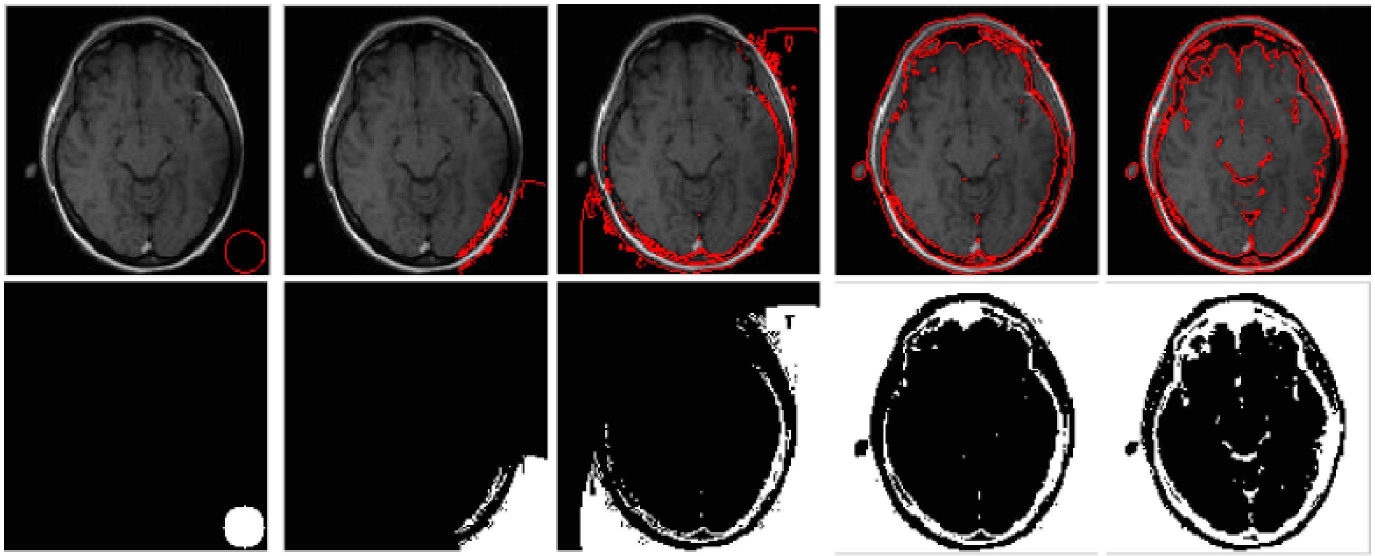


Figure 8. Initial curve position not overlapping any area of the object. Top: the evolving curve (in red) over time where the first image shows the initial contour. Bottom: evolving segmentation over time until the object is detected. Size = 300 x 300, $\phi_0(x,y) = -\sqrt{(x-120)^2+(y-120)^2}+10$, $\mu = 0.01$, no reinitialization, cpu = 2.01 s, iterations = 9.

Detecting Holes

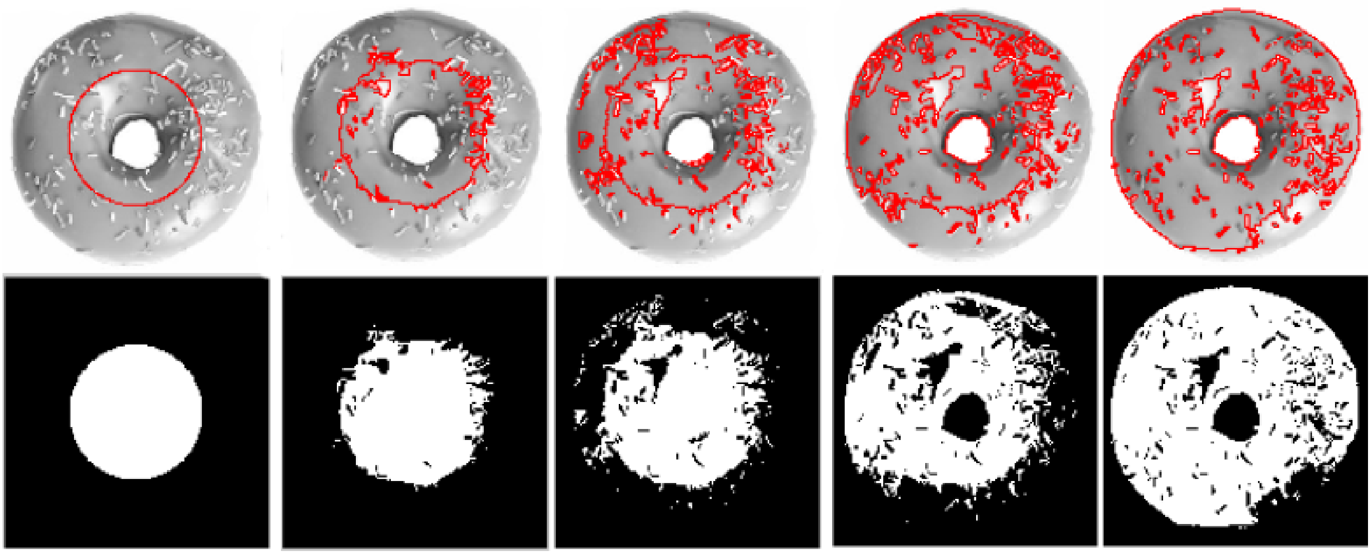


Figure 9. Successful detection of holes. Top: the evolving curve (in red) over time where the first image shows the initial contour. Bottom: evolving segmentation over time until the object is detected. Size = 131 x 131, $\phi_0(x, y) = -\sqrt{(x-65.5)^2 + (y-65.5)^2} + 32.8$, $\mu = 0.01$, no reinitialization, cpu = 8.12 s, iterations = 12.

Blurry Images

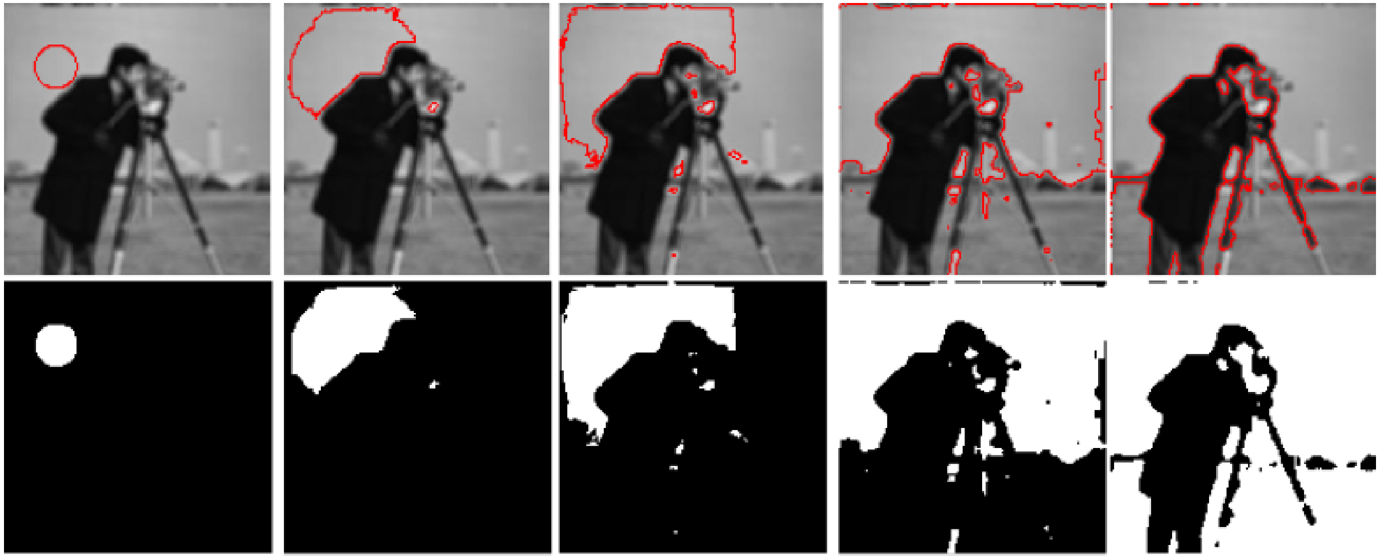


Figure 12. Ability to detect contours in blurry images. Top: the evolving curve (in red) over time where the first image shows the initial contour. Bottom: evolving segmentation over time until the object is detected. Size = 255 x 255, $\phi_0(x,y) = -\sqrt{(x-127.5)^2+(y-127.5)^2}+63.75$, $\mu = 0.01$, no reinitialization, cpu = 5.41 s, iterations = 15.

Noisy Images - Gaussian

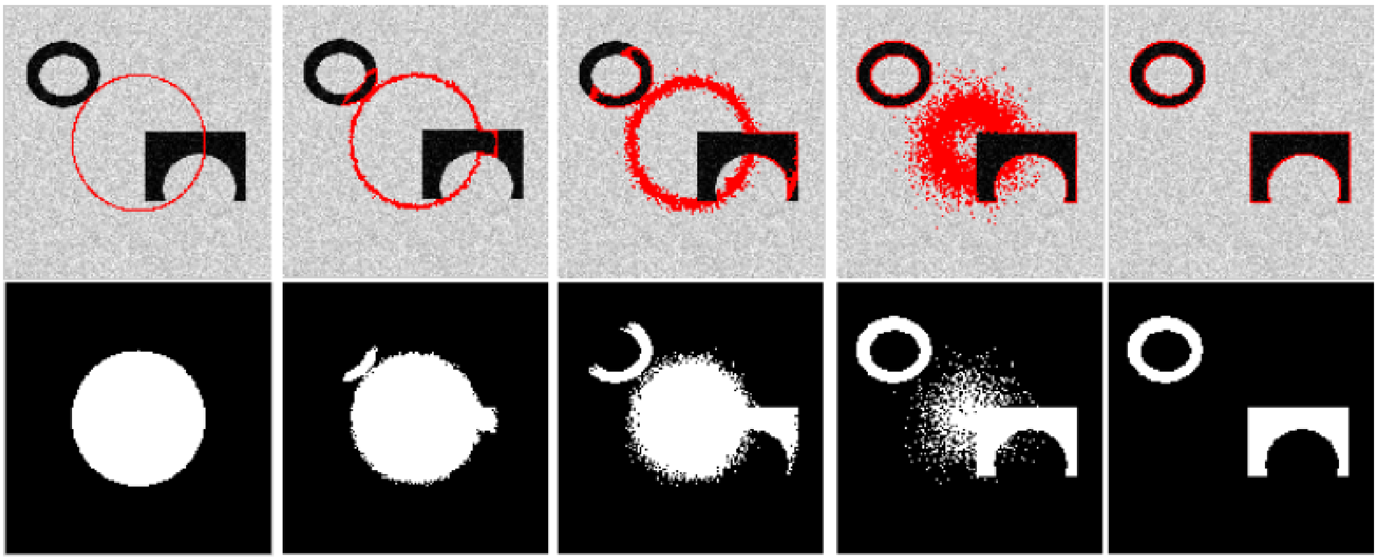


Figure 13. Ability to detect contours in an image with Gaussian noise. Top: the evolving curve (in red) over time where the first image shows the initial contour. Bottom: evolving segmentation over time until the object is detected. Size = 300 x 300, $\phi_0(x, y) = -\sqrt{(x-150)^2 + (y-150)^2} + 75$, $\mu = 0.01$, no reinitialization, cpu = 4.4 s, iterations = 7.

Noisy Images – Salt & Pepper

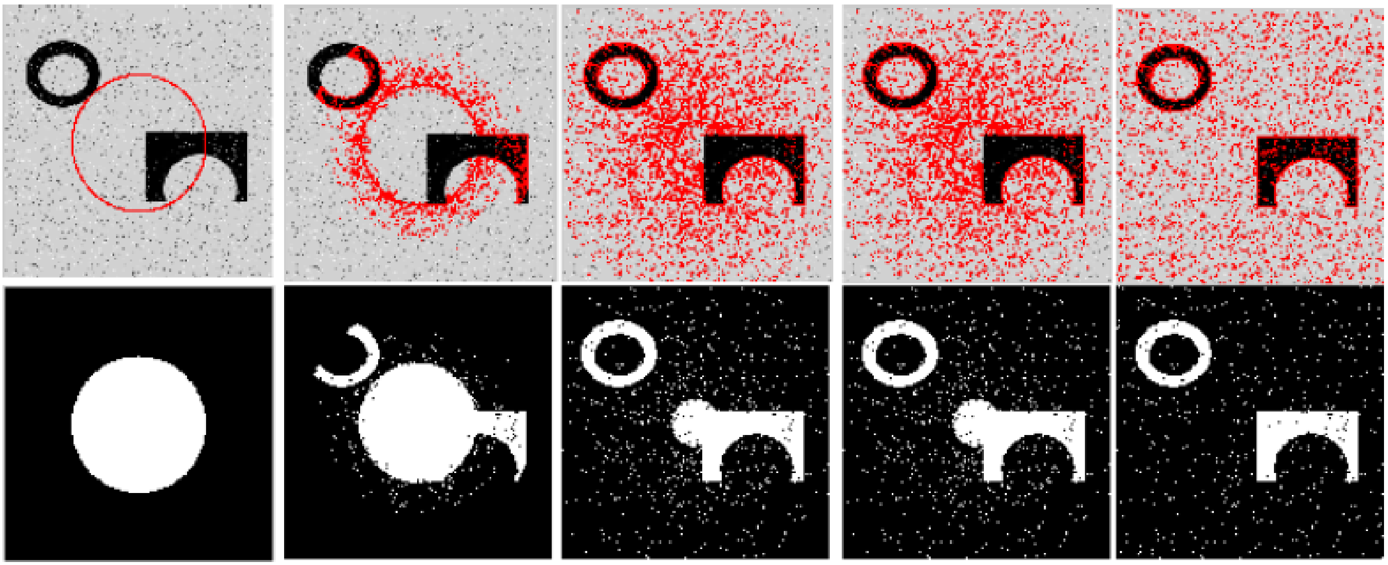
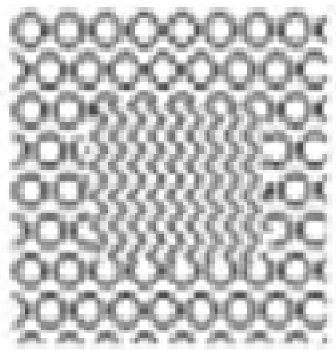


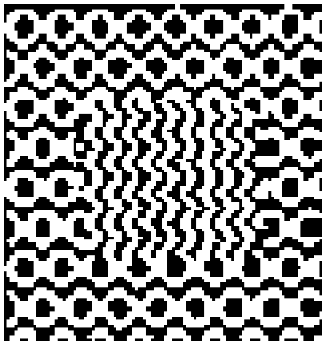
Figure 14. Figure 13, but with *Salt & pepper* noise. Noise was still detected despite increasing μ. Top: the evolving curve (in red) over time where the first image shows the initial contour. Bottom: evolving segmentation over time until the object is detected. Size = 300 x 300, $\phi_0(x, y) = -\sqrt{(x-150)^2 + (y-150)^2} + 75$, $\mu = 5$, no reinitialization, cpu = 12.27 s, iterations = 7.

Textured Images

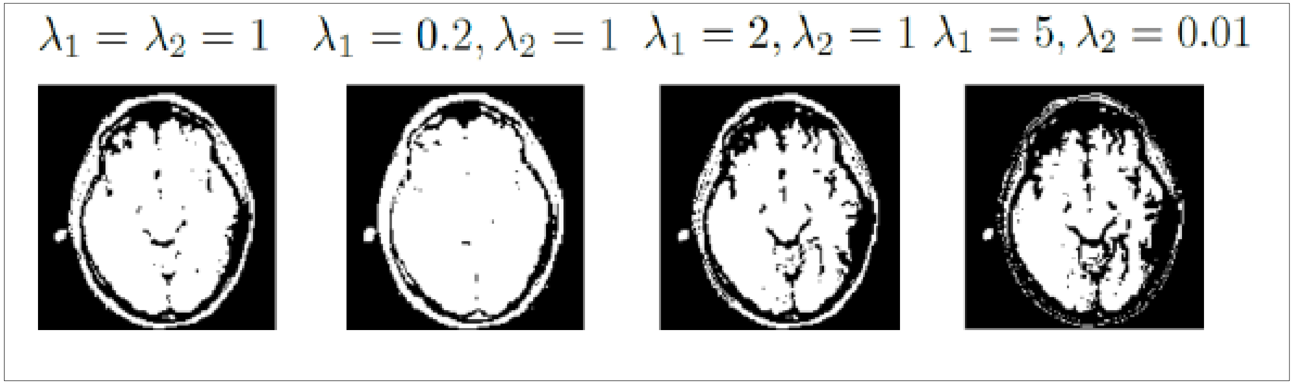
Image

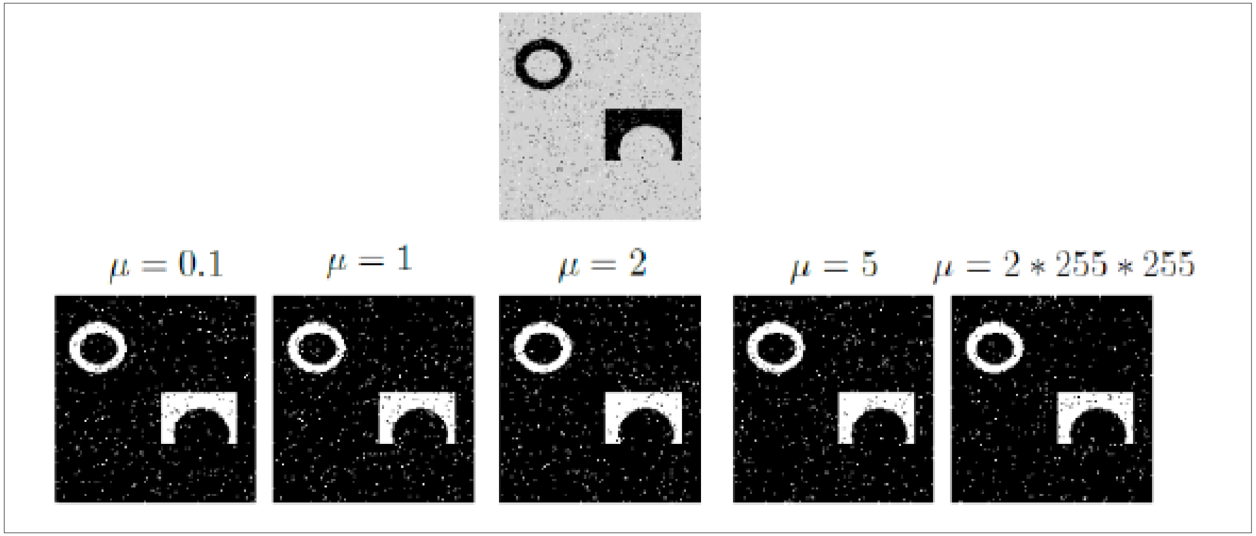


Segmentation



Varying Parameters





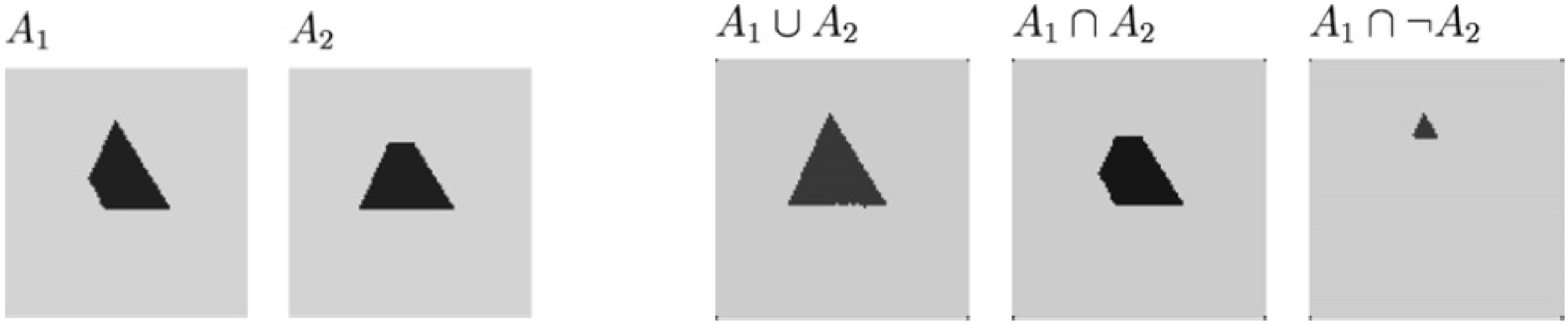
Summary of Results

- Able to detect all kinds of boundaries including holes and open boundaries
- Able to perform well with blurry images
- Able to perform well with Gaussian noise, but not “Salt & Pepper”
- Inability to detect objects with small variation from background
- Good response to change λ but not μ

Applying the Chan-Vese Model to Multi-channel Images

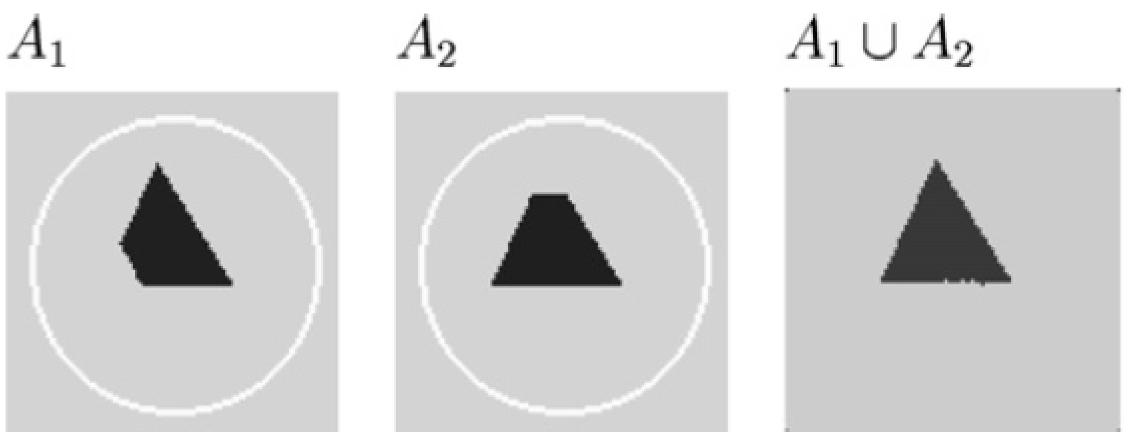
The Sandberg-Chan Framework

- Allows the user to produce any logical combination of object channels

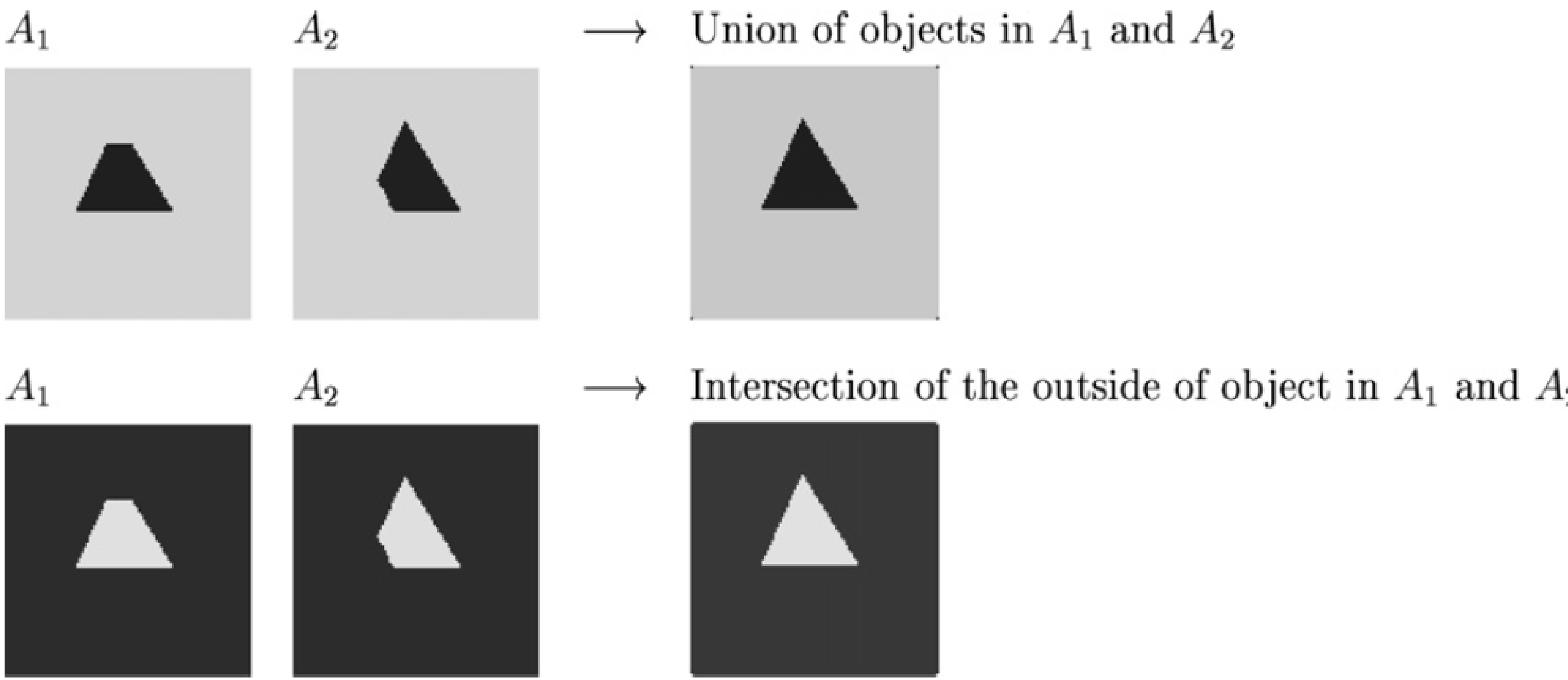


Proposed Solution

- Use the Chan-Vese model, and compare the contour fitting on each channel
  - Start with the same contour on each channel
  - Fit contour to the object on ALL channels according to the logic operator & based on regions



Region Based Logic Operations



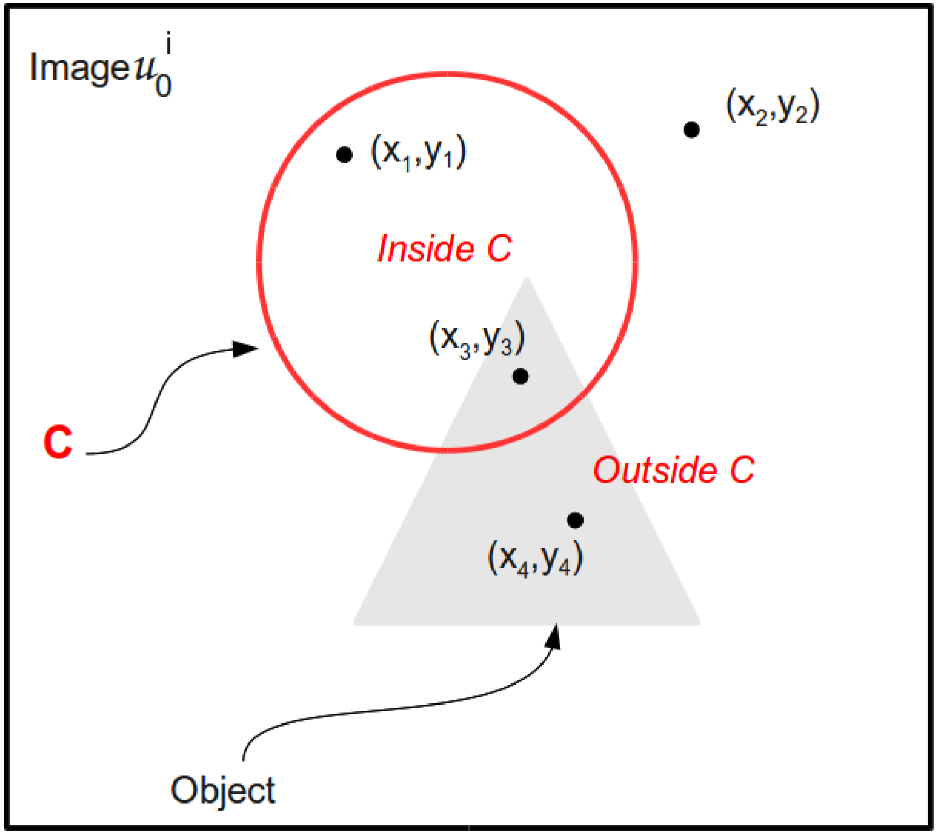
Logic Variables

$$z_i^{\text{in}}(u_0^i, x, y, C) = \begin{cases} 0 & \text{if}(x,y) \in C \text{ and } (x,y) \text{ inside the object in channel } i, \\ 1 & \text{otherwise,} \end{cases}$$

$$z_i^{\text{out}}(u_0^i, x, y, C) = \begin{cases} 1 & \text{if}(x,y) \notin C \text{ and } (x,y) \text{ is inside the object in channel } i, \\ 0 & \text{otherwise.} \end{cases}$$

- *Note:*

  0 is true because we want to minimize the fitting term



Level Set Formulation

- Objective function

$$F(\phi, c^+, c^-) = \mu|C(\phi)| + \lambda\Big[\int_\Omega f_{\text{in}}(z_1^{\text{in}}, \ldots, z_n^{\text{in}})H(\phi) + f_{\text{out}}\left(z_1^{\text{out}}, \ldots, z_n^{\text{out}}\right)(1 - H(\phi))\,\mathrm{d}x\Big].$$

- We want to minimize F

$$\frac{\partial \phi}{\partial t} = \delta(\phi)\left[\mu\nabla \cdot \left(\frac{\nabla\phi}{|\nabla\phi|}\right) - \lambda(f_{\text{in}}(z_1^{\text{in}}, \ldots, z_n^{\text{in}}) - f_{\text{out}}(z_1^{\text{out}}, \ldots, z_n^{\text{out}}))\right]$$

with the boundary condition $\frac{\delta(\phi)}{|\nabla\phi|}\frac{\partial\phi}{\partial\vec{n}} = 0$

Implementation

- Finite difference discretization with an explicit time stepping scheme $\phi^{n+1} = \phi^n + \Delta t * \phi_t$

- $\frac{\partial \phi}{\partial t} = |\nabla \phi| \mu \nabla (\frac{\nabla \phi}{|\nabla \phi|}) - \lambda (f_{in}(z_1^{in}, \ldots, z_n^{in}) - f_{out}(z_1^{out}, \ldots, z_n^{out}))$

- Stopping condition: $(\phi^{n+1} > 0) == (\phi_n > 0)$
- No reinitialization was necessary

Experimental Results

Evaluation Criteria

Criteria	Goal
Detecting Boundaries	Ability to correctly detect object boundaries of simple objects
Curve Position	Ability to correctly detect object boundaries irrespective of the initial curve position
Detecting Holes	Ability to detect holes in objects, and not simply stop on outside boundary
Blurred Images	Ability to correctly (as much as possible) detect object boundaries in blurred images
Noisy Images	Ability to correctly (as much as possible) detect object boundaries in noisy images
Union Operator	Ability to correctly obtain the union of two or more images
Intersection Operator	Ability to correctly obtain the intersection of two or more images
Complement	Ability to correctly obtain the union or intersection of two or more images containing complements
Parameter Settings	Ability to respond correctly to the different parameter settings

Simple Union & Intersection

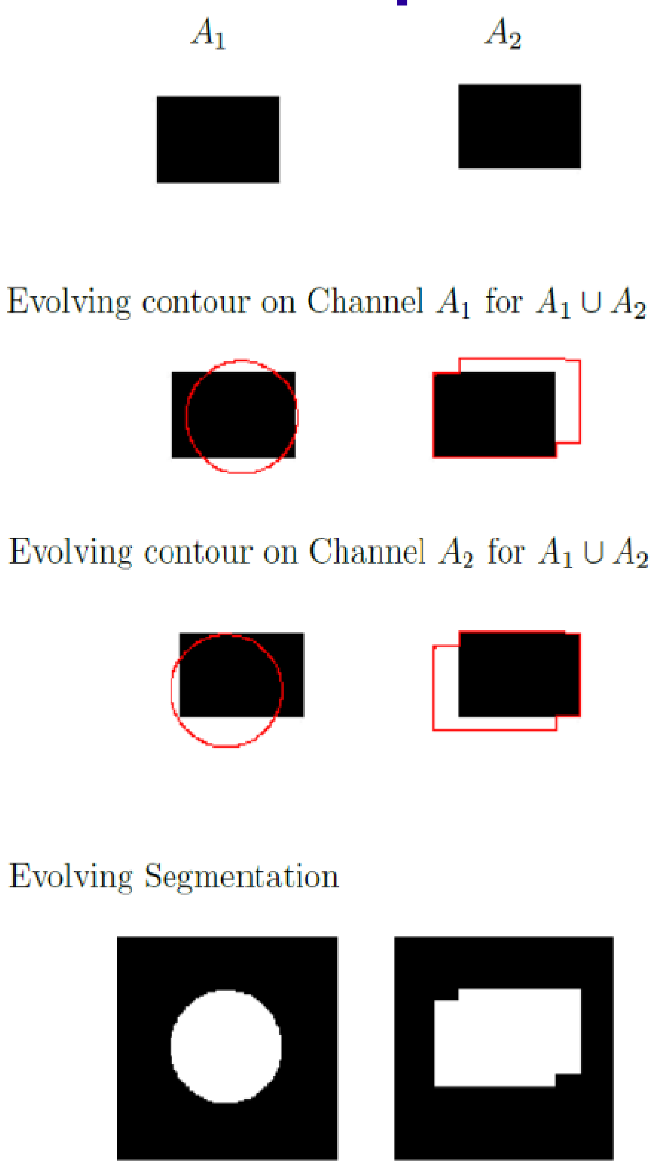


Figure 16. Simple union example. Size = 300 x 300, $\phi_0(x, y) = -\sqrt{(x-150)^2 + (y-150)^2} + 75$, no reinitialization, cpu = 1.32 s, iterations = 2.

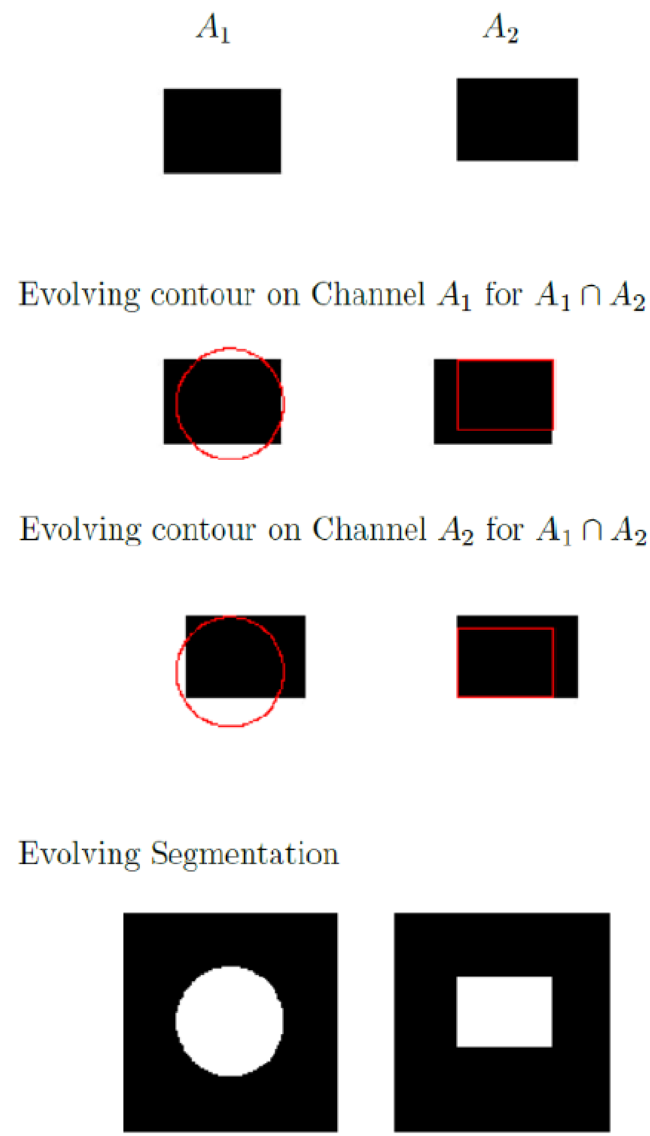


Figure 17. Simple intersection example. Size = 300 x 300, $\phi_0(x, y) = -\sqrt{(x-150)^2 + (y-150)^2} + 75$, no reinitialization, cpu = 1.83 s, iterations = 2.

Simple Union & Intersection – Flipped Intensities

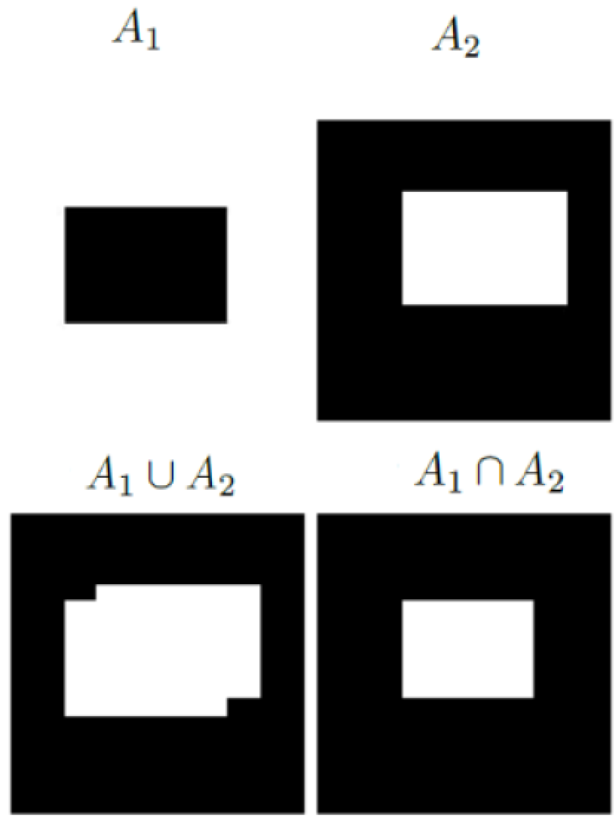


Figure 20. Segmentation for union and intersection of channels having reverse contrast.. Size = 300 x 300, $\phi_0(x, y) = -\sqrt{(x-150)^2 + (y-150)^2)} + 75$, no reinitialization. For union: cpu = 1.32 sec and iterations = 2. For intersection: cpu = 1.83 sec and iterations = 2

Occlusions

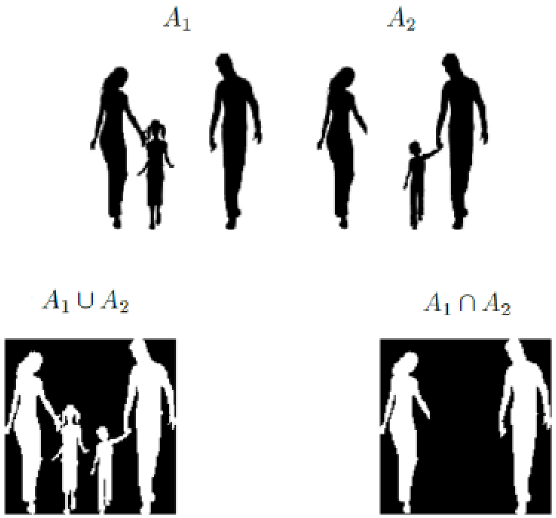
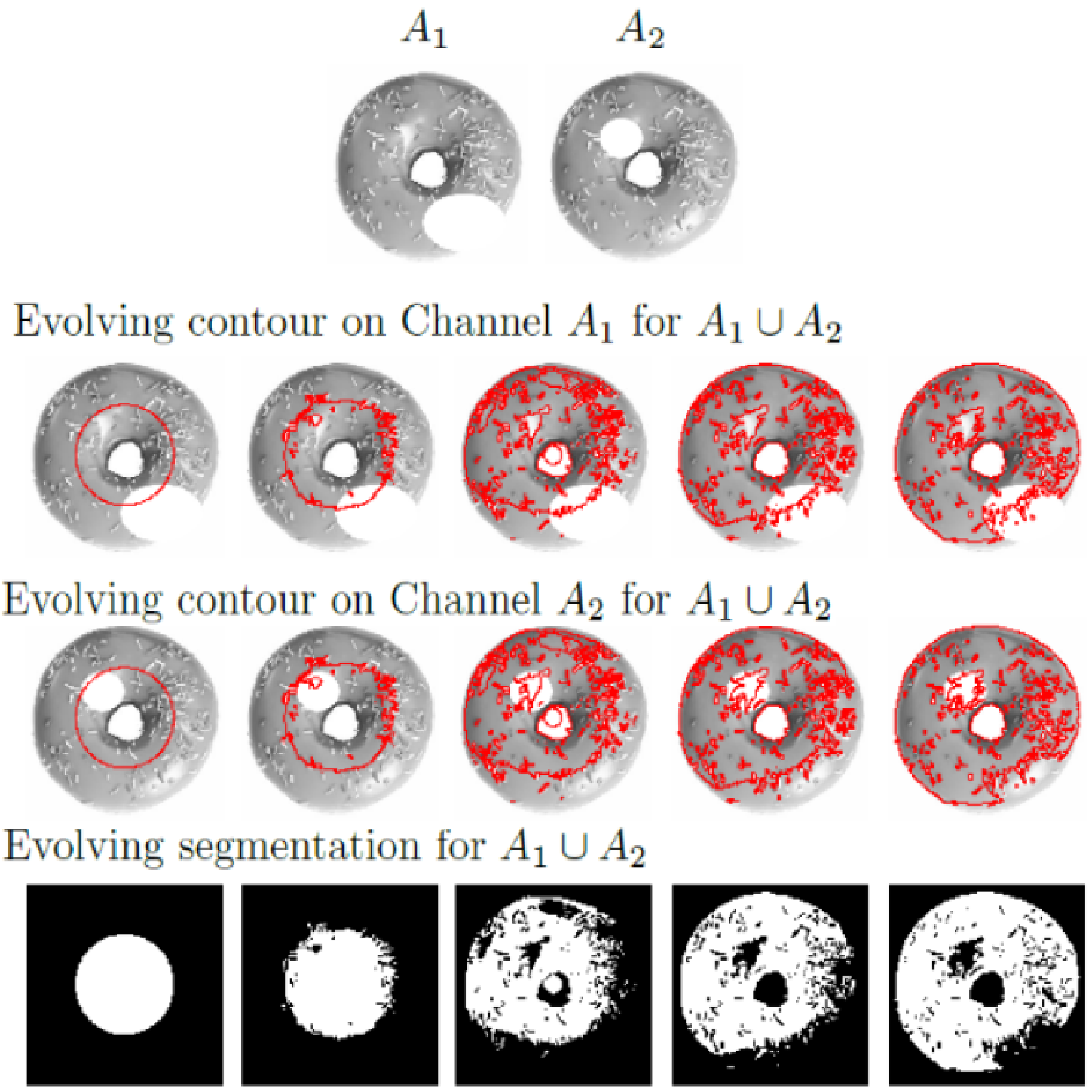
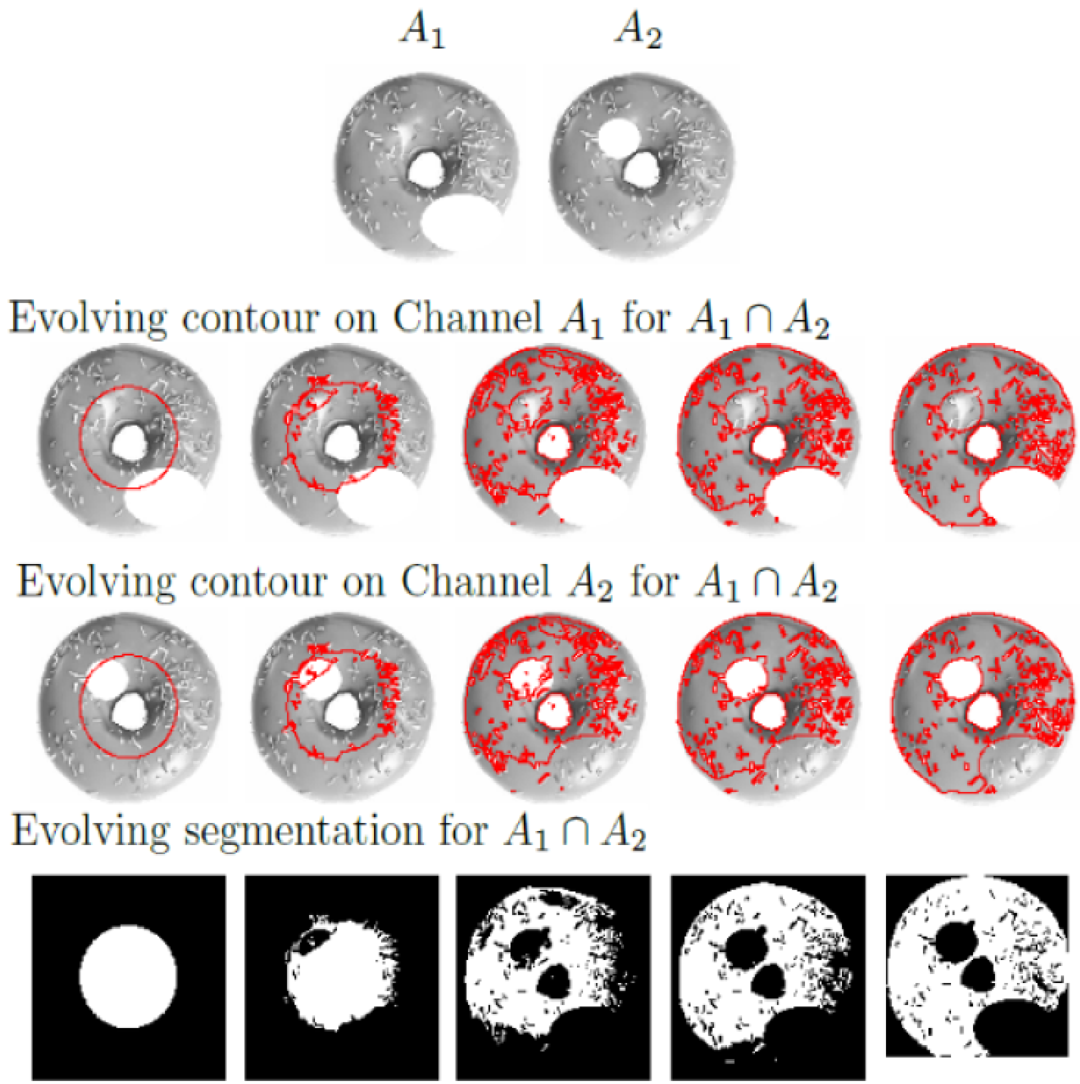


Figure 18. Combining missing information from two different channels. Size = 114 x 114. $\phi_0(x, y) = \sqrt{(x-57)^2 + (y-57)^2} - 28.5$, no reinitialization. Union: cpu = 4.01 s and iterations = 7. Intersection: cpu = 4.12 s and iterations = 7.

Union & Intersection with Holes



Union & Intersection with Holes



3 Channels

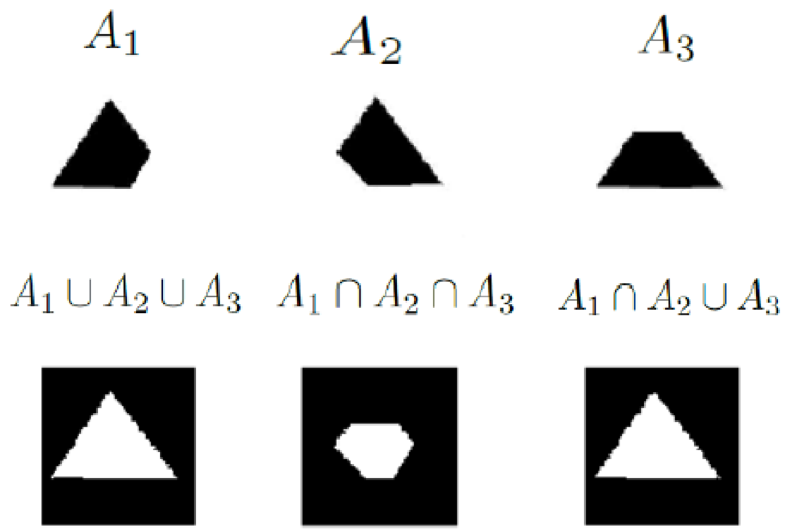


Figure 21. Logic operations performed on three channels. The figure shows the final segmentation in each case. Size = 200 x 200. $\phi_0(x,y) = -\sqrt{(x-100)^2 + (y-100)^2} + 50$, $\mu = 0.1$, no reinitialization. For union: cpu = 3.92 sec and iterations = 6. For intersection: cpu = 3.11 sec and iterations = 5. For intersection then union: cpu = 1.24 and iterations = 4.

Blurred Edges

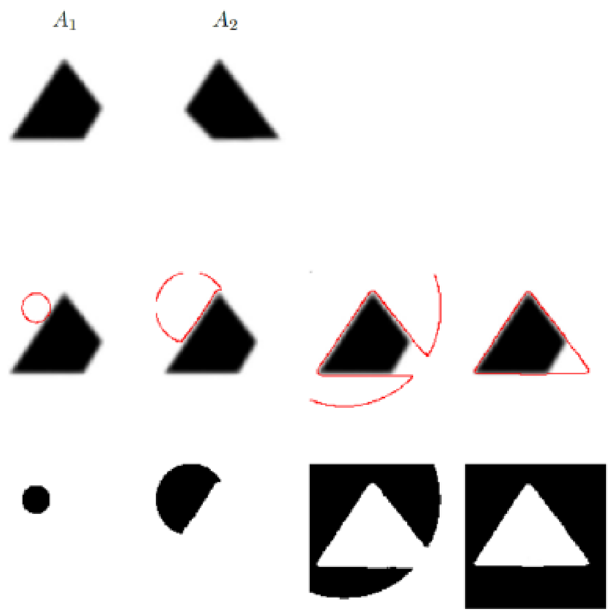


Figure 23. Successful union with blurred images. Top: the evolving curve for $A_1 \cup A_2$ (in red) over time where the first image shows the initial contour. Bottom: evolving segmentation over time until the union is detected. Size = 200 x 200, $\phi_0(x, y) = -\sqrt{(x-100)^2 + (y-100)^2} + 50$, no reinitialization, cpu = 6 s, iterations = 11.

Noisy Images

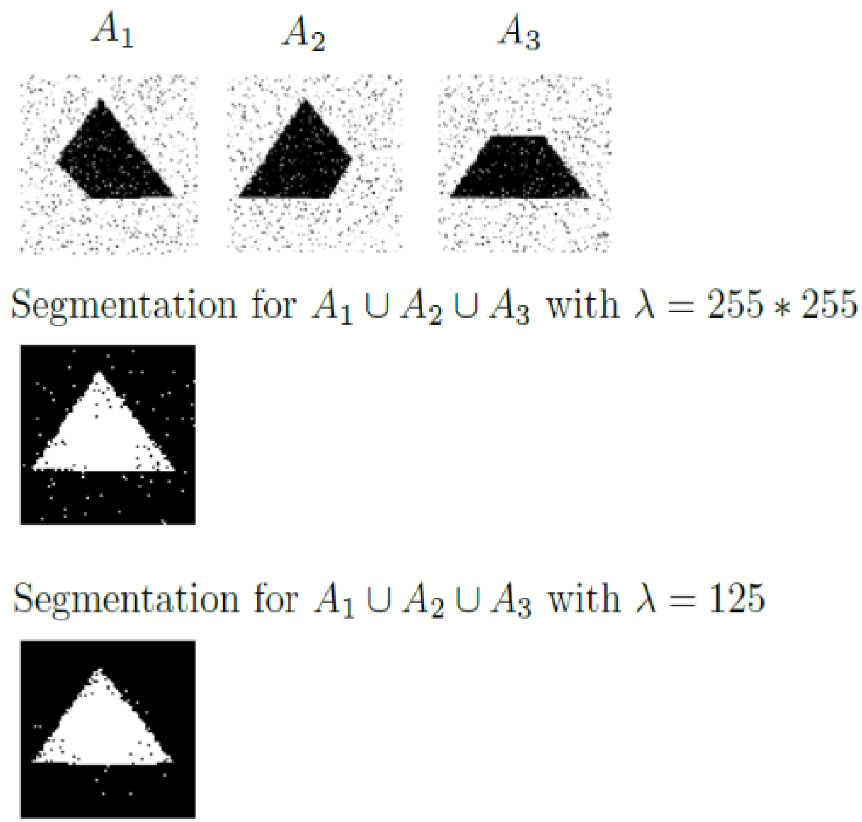
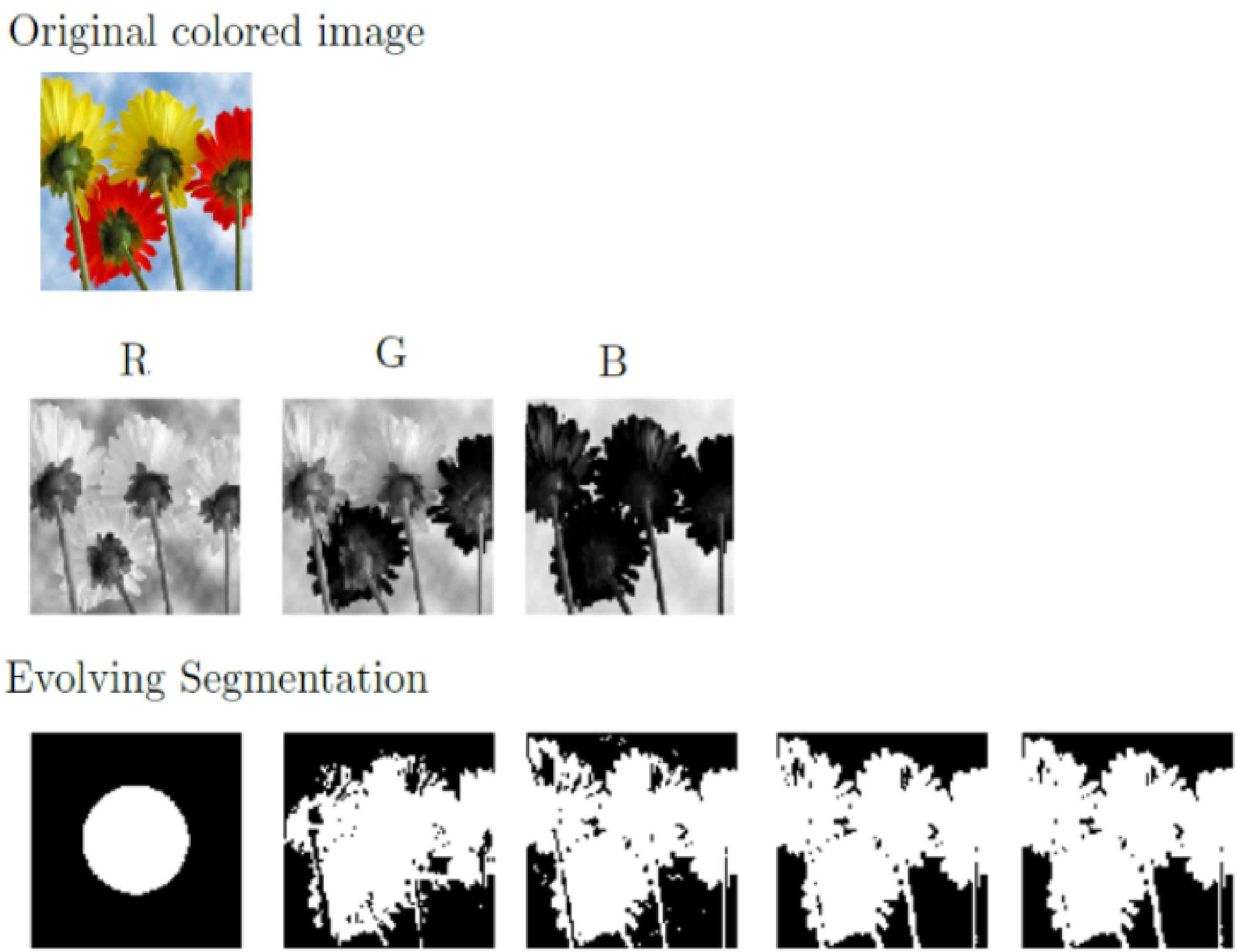


Figure 25. Union of noisy channels. Same image as Figure 21, but with *Salt & pepper'* noise with a variation of 0.1. Decreasing λ from 255*255 to 125 allowed less noise to be detected. Size = 200 x 200, $\phi_0(x, y) = -\sqrt{(x-100)^2 + (y-100)^2} + 50$, no reinitialization. Cpu = 3.17 sec and iterations = 5 when $\lambda = 255 * 255$. Cpu = 146.46 sec and iterations = 125 when $\lambda = 125$.

RGB Channels



Complement

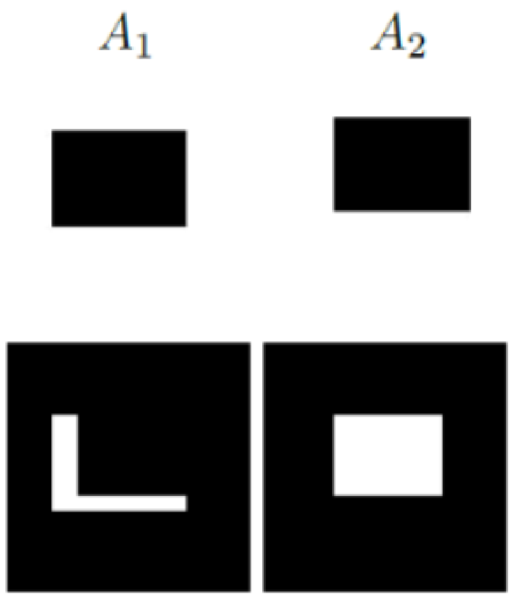


Figure 27. The complement operator causes the contour to go through an infinite loop between two segmentations: the correct segmentation, and the segmentation of $A_1 \cap A_2$. Bottom: correct and incorrect segmentation respectively

Defining Initial Phi

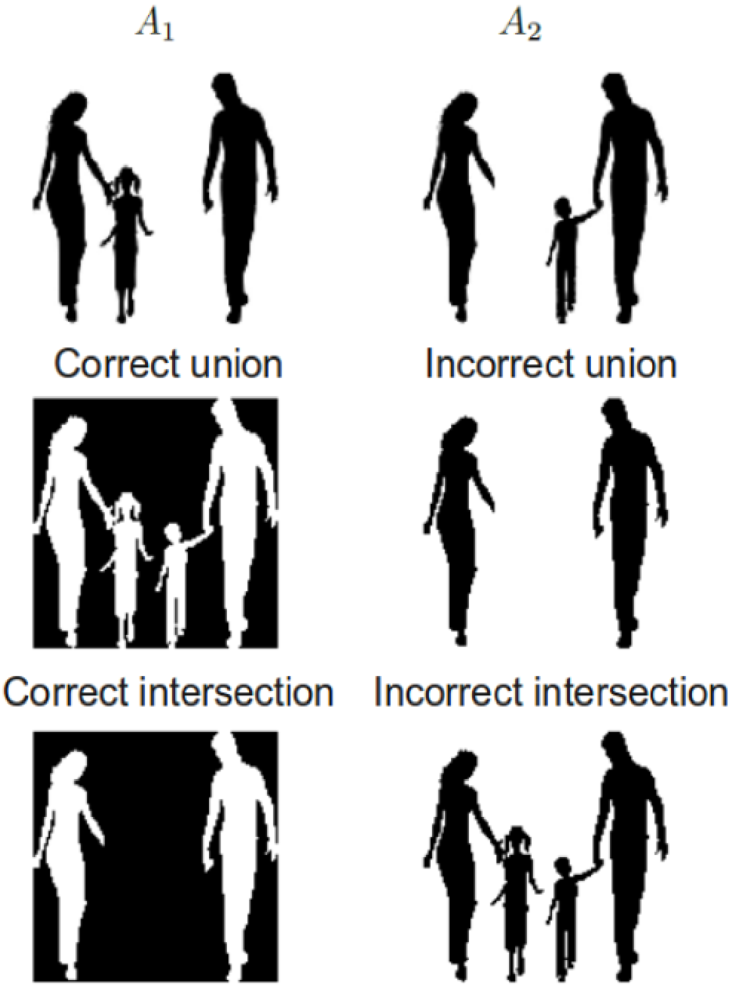


Figure 29. Union and intersection depending on the way ϕ is defined

Summary of Results

- Able to perform union & intersection on multiple channels
- Able to perform well with blurry images
- Able to perform well with noise
- Detecting objects in RGB images
- Good response to change in λ
- Complement not working
- Definition of initial phi varies results

Take Home Message

- Chan-Vese model
  - Detects objects without edge information
  - Must have variations in intensities to work
- Sandberg-Chan logic framework
  - Can combine multiple images using different logic operations

Thank you

???