

An Implementation of the Active Contours without Edges Model and the Logic Framework for Active Contours on Multi-channel Images

Karim Ali and Sarah Nadi
{karim, snadi}@cs.uwaterloo.ca
CS870 - Fall 2010

David R. Cheriton School of Computer Science
University of Waterloo

Abstract

In this project, we provide an implementation for the Chan-Vese model for active contours without edges, and the Sandberg-Chan logic framework for active contours on multi-channel images. The Chan-Vese model is a special case of the more general Mumford-Shah functional for segmentation and level sets. It differs from other active contour models in that it is not edge dependent, therefore it is more capable of detecting objects whose boundaries may not be defined by a gradient (e.g. blurry images, noisy images). The Sandberg-Chan model builds on top of the Chan-Vese model by allowing the contour to evolve simultaneously on multiple channels. It also offers a set of logic operations that can be applied to those channels (union, intersection, complement, and their combinations). We evaluate our implementation based on several criteria, such as: independence of initial curve position, successful detection of holes, performance with blurred and noisy images, and response to scaling parameters.

1. Introduction

Image segmentation or boundary detection is a very important problem in the area of Image Processing, and has received a lot of attention in the past. The classical Active Contour model (or Snakes) proposed by Kass et al. [3] was the first model to use the idea of energy minimization to attract a contour to the edges of the objects in an image. The Snakes model was very successful and variations of it were highly adopted later on (E.g. [1]). Most of these models use the level set formulation for propagating fronts to evolve the curve. However, these models highly depend on motion defined by the gradient of the image which leads to poor performance in smoothed edges. To overcome these limitations, Chan and Vese [2] propose an active contour model that does not depend on the edges (i.e the gradient)

for propagating the curve to detect the boundary of the object. Instead, they use a region-based approach based on the Mumford-Shah model [4] to divide the image into two regions: one inside the propagating curve, and one outside. The curve is at the boundary of the object if there is no difference in intensities inside the curve as well as outside the curve.

The Active Contours without Edges model proposed by Chan and Vese (referred to as Chan-Vese model throughout the rest of this paper) is very successful in detecting objects even in noisy or blurry images. It can also detect holes in objects which is usually a limitation in previous models. As an extension to this work, Sandberg and Chan [5] propose a logic framework (referred to as the Sandberg-Chan model throughout the rest of this paper) that performs logic operations on multiple images according to the curve propagation proposed in the Chan-Vese model. To achieve that, previous models usually had two steps. They would either first segment the object in each channel separately then combine the segmented objects according to the logic operation through bitwise operations [6, 7] or they would apply logic operations to the different images then segment the resulting image. The first approach is very costly, and the second approach requires a lot of prior knowledge about the intensities of each image. To overcome these drawbacks, the Sandberg-Chan model is based on the idea of fitting a single contour to the object on all channels according to the logic operator, and based on regions.

In this paper, we report on our findings after implementing each of these models. We implement both models in Matlab, and experiment with several images. We explain the details of our implementation as well as our results. The rest of this paper is organized as follows: Section 2 provides brief background about each of the two models implemented in this paper. Section 3 then explains our implementation. In this section, we explain how we implemented the models and any variations from the original papers. Sec-

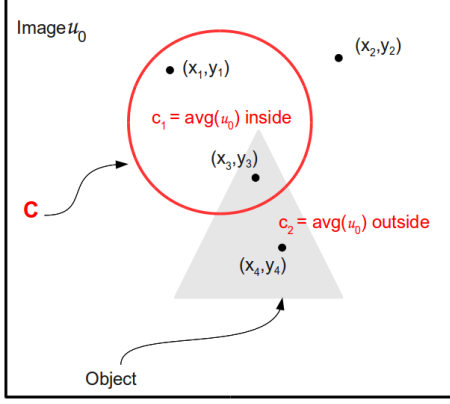


Figure 1. Region Based Model

tion 4 explains our evaluation criteria, and presents the results we obtained. Section 5 discusses some of the difficulties we faced during implementation. Section 6 concludes this paper by summarizing our findings.

2. Background

2.1. Chan-Vese Model

The Chan-Vese model [2] is a region based model for detecting objects in an image. It is based on a restriction of the Mumford-Shah model which divides an image into regions and represents each region by a piecewise constant (also called the minimal partition problem). Figure 1 shows this region based model. The figure shows an image u_0 which has a gray triangular object in it. The red curve, C , is the initial contour used to detect this object. The main idea behind the model is that the curve divides the image into two regions: that inside the curve and that outside. Each region is represented by a constant, c , which is the average intensity of the image values in each region. In order for the curve to fit the object, there must be minimal variation of the intensities inside the curve as well as outside. In other words, this turns into a minimization problem of the difference of intensities inside (fitting term one, F_1) plus those outside (fitting term two, F_2). For example, in Figure 1, the point (x_1, y_1) will have to be outside the curve in order to minimize the difference between the points inside the curve. Similarly, the point (x_4, y_4) will have to be inside the curve. Figure 2 shows all the possibilities of the curve's fitting according to its position with respect to the object.

More formally, the Euler-Lagrange equation (as derived in the original paper) representing the time motion of the curve C is shown in Equation 1.

$$\frac{\partial \phi}{\partial t} = \delta_\varepsilon(\phi) [\mu \operatorname{div} \left(\frac{\nabla \phi}{|\nabla \phi|} \right) - \nu - \lambda_1 (u_0 - c_1)^2 + \lambda_2 (u_0 - c_2)^2] \quad (1)$$

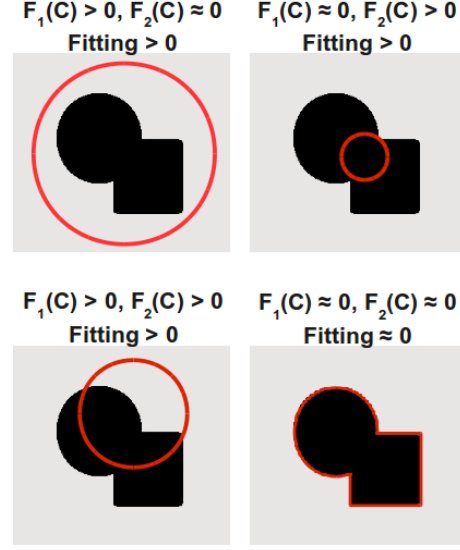


Figure 2. Chan-Vese Curve Fitting

The parameter μ is a scaling parameter for the length of the curve represented in terms of curvature. The smaller μ is, the more the length of the curve can increase without penalizing the minimization. This allows the model to detect smaller objects and holes. The larger μ is, the less freedom there is for the curve to increase in length, and thus, it will only be able to detect larger objects. The parameter ν is also a scaling term for the area of the curve. However, the authors do not use the area term in the Euler-Lagrange derivation, and always set ν to 0. It seems that μ is sufficient to scale the curve according to the objects that need to be detected. Finally, λ_1 and λ_2 are weighting parameters for the forces inside the curve and outside the curve respectively. Since we want to give both forces equal weight, the authors set $\lambda_1 = \lambda_2 = 1$ in all their experiments.

2.2. Sandberg-Chan Model

The main goal of the Sandberg-Chan model [5] is to perform logic operations on a combination of different images accurately and efficiently. For example, finding the union of two images where different parts of the object are occluded in each image so that a complete object can be obtained. This example will be shown later in Figure 18. In order to do that, they use the Chan-Vese model to detect objects, and simultaneously include the logic operations in the minimization problem. Since they are using the Chan-Vese model, the problem must be viewed in terms of regions as well. Accordingly, they define the two main logic operations (union and intersection) in terms of regions where the union of two images is the union of the inside of the curve with respect to the images plus the intersection of the outside of the curve with respect to the images. For example, Figure 3 [5] shows that the union of A_1 and A_2 (in terms of

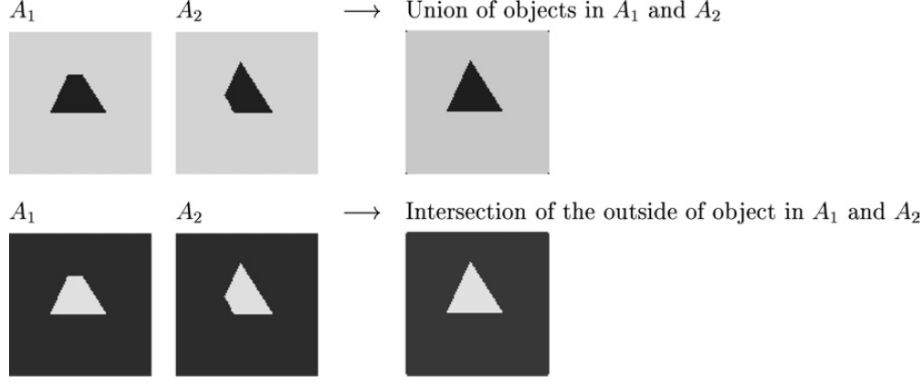


Figure 3. Region Based View Union

its shape) can be obtained by taking the union of the insides of the object or the intersection of the outsides. Similarly, the intersection of two images would be the intersection of the insides plus the union of the outsides. Thus, in order to obtain accurate results irrespective of the object's intensity outside versus inside, we should consider *both* the logic operation required for the region inside the object as well as that outside.

More formally, the authors define two logic variables z_i^{in} and z_i^{out} to denote whether a point (x, y) should be in the moving curve C or not with respect to image i . Since we are trying to minimize the fitting of the curve, they use 0 to denote *true* and 1 to denote *false* (the reverse of the usual convention). Following the Chan-Vese model, they represent each of the regions inside and outside the curve by a constant c . In this paper, they represent c_{in} as c_+ and c_{out} as c_- . Equations 2 and 3 show how to calculate z_{in} and z_{out} respectively in terms of the Chan-Vese model. We note here that in the original paper there was a typo in these equations where they divide by the maximum intensity of each image instead of the maximum intensity squared. However, in order to have z_{in} and z_{out} have values from 0 to 1 that represent logic values, we need to divide by the maximum intensity squared. The equations shown below have been corrected for that.

$$z_i^{in}(u_0^i, x, y, C) = \frac{|u_0^i - c_+|^2}{(\max_{(x,y) \in u_0^i} u_0^i)^2} \quad (2)$$

$$z_i^{out}(u_0^i, x, y, C) = \frac{|u_0^i - c_-|^2}{(\max_{(x,y) \in u_0^i} u_0^i)^2} \quad (3)$$

In order to perform logic operations on z_{in} and z_{out} , the authors introduce interpolation functions that mimic the behavior of the regular truth table, but for continuous values between 0 and 1. The union and intersection functions for two variables are shown in Equations 4 and 5 respectively. These equations can be simply extended to any number of variables.

$$f_{\cup} = (z_1 \cdot z_2)^{1/2} \quad (4)$$

$$f_{\cap} = 1 - ((1 - z_1) \cdot (1 - z_2))^{1/2} \quad (5)$$

Based on these functions, the Euler-Lagrange equation for any logic operation is shown in Equation 6. According to the desired logic operation, f_{in} and f_{out} will be specified. For example, if we are doing the union of the images, then we will need to perform a union operation on the insides. Thus, f_{in} will be replaced by Equation 4. Similarly, we will need to do the intersection of the outsides so f_{out} will be replaced by Equation 5.

$$\frac{\partial \phi}{\partial t} = \delta(\phi) [\mu \nabla \cdot \left(\frac{\nabla \phi}{|\nabla \phi|} \right) - \lambda (f_{in}(z_1^{in}, \dots, z_n^{in}) - f_{out}(z_1^{out}, \dots, z_n^{out}))] \quad (6)$$

3. Implementation

During implementation, we tried to stick to the authors' formulas and guidelines as much as possible. In this section we explain what methods we use to numerically solve the PDEs. For the chan-vese model, we use the PDE given in Equation 9 in the original paper [2] which is shown in Equation 1. We, first, tried to implement the dirac delta function, $\delta_\epsilon \phi$, as defined in the paper. However, it resulted in worse segmentations. Accordingly, we chose to use $|\nabla \phi|$ instead of the dirac delta function as this was indicated as a valid alternative by the authors. To calculate c_1 and c_2 , we simply calculated the mean of the values inside ϕ (specifically where $\phi > 0$) and the mean of the values outside ϕ (specifically where $\phi < 0$) respectively. To actually solve the PDE, we chose to use an explicit time stepping scheme for the finite difference discretization. That is, in each time step $\phi^{n+1} = \phi^n + \Delta t * \text{phi}_t$. This was simpler to implement, and although Δt must be chosen carefully to satisfy the stability condition ($\Delta t \leq (1/\text{imageSize})^2$), we did

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

Table 1. Evaluation Criteria

not suffer from performance problems due to this restriction.

For the Sandberg-Chan model, we used the PDE given in original paper as well, and shown in Equation 6. Again, we used $|\nabla \phi|$ instead of the dirac delta function. We calculated z_{in} and z_{out} according to Equations 2 and 3 shown above which have been corrected for the typo in the original paper. For all other equations and calculations necessary in the model, we closely followed the original paper in our implementation. We also used an explicit time stepping scheme in this model.

We use the same stopping condition for both model. That is, to check if we have reached the stationary solution, we check if more points have been added inside the contour or not. More formally, we check if $(\phi^{n+1} > 0) == (\phi_n > 0)$. If the contour did not move, then new points will be added to its inside, and this condition will be satisfied.

4. Experimental Results

In order to make sure we correctly implemented the models, we had several evaluation criteria. Table 1 summarizes these criteria, and explains the goals of each. Some of these criteria apply for both models, while other apply to one or the other. For the rest of this section, we proceed by presenting our experimental results for both models. For each model, we proceed in the order of these criteria starting with the simplest cases, and incrementally challenging the model. Unless otherwise stated, all the segmentation was performed on a gray scale version of the original image. All the images in the Chan-Vese model were run on an Intel Core 2 Duo processor with 2.00 GHz and a 2GB RAM and all the images in the Sandberg-Chan model were run on a Intel Centrino Duo 2.0 processor and a 1GB RAM computer. As shown in the results, we tried different image sizes to ensure that our solution converges.

4.1. Chan-Vese Model Results

We present the results from the Chan-Vese model in this section, and show which criteria were met and which were not. Unless otherwise specified, we set $\lambda_1 = \lambda_2 = 1$.

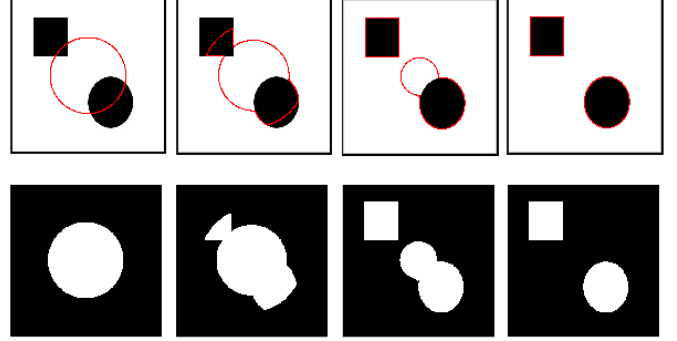


Figure 4. Successful detection of object boundaries. Top: the evolving curve (in red) over time where the first image shows the initial contour. Bottom: evolving segmentation over time until the objects are detected. Size = 300 x 300, $\phi_0(x, y) = -\sqrt{(x-150)^2 + (y-150)^2} + 75$, $\mu = 0.01$,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

Successful Detection of Boundaries

Figure 4 shows that the implemented model can successfully detect object boundaries in a simple image. In this case, the evolving curve was successfully able to detect the boundaries of both objects. Figure 5 shows a slightly more complicated example where the evolving contour successfully detects the contour of the brain. Note that there is a tiny area near the center to the left of the brain that was also successfully detected. To ensure that open boundaries (e.g. lines) can also be detected, we used an image with several shapes shown in Figure 6.

Independence of Initial Curve Position

To ensure that the initial position of the curve does not affect the final segmentation, we tested two other positions for the initial curve for the same brain image used in Figure 5. In Figure 5, the initial contour was completely overlapping the object. Accordingly, we tried two other positions. The first one is shown in Figure 7 where the initial contour only partially overlaps with the object. The second one is shown in Figure 8 where the initial contour does not overlap the ob-

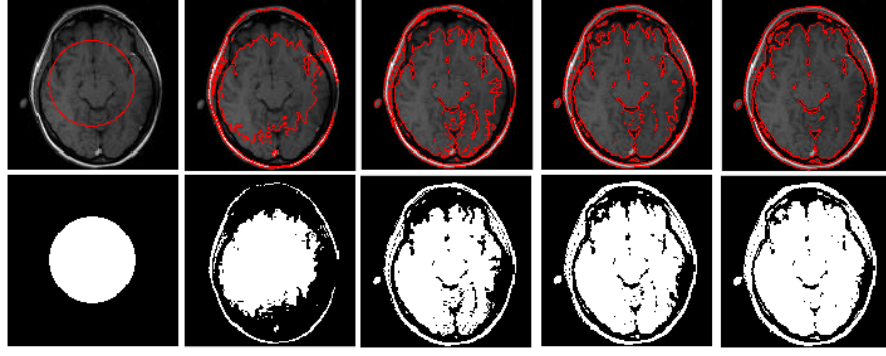


Figure 5. Successful detection of object boundaries. Top: the evolving curve (in red) over time where the first image shows the initial contour. Bottom: evolving segmentation over time until the object is detected. Size = 131 x 131, $\phi_0(x, y) = -\sqrt{(x - 65.6)^2 + (y - 65.5)^2} + 32.8$, $\mu =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

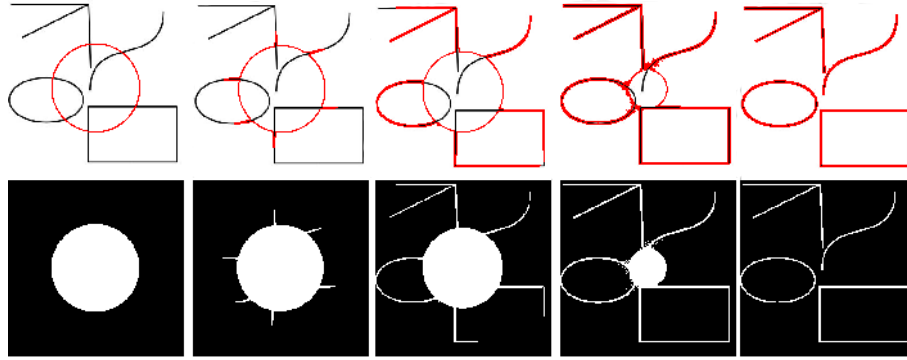


Figure 6. Successful detection of object with open boundaries. Top: the evolving curve (in red) over time where the first image shows the initial contour. Bottom: evolving segmentation over time until all objects are detected. Size = 300 x 300, $\phi_0(x, y) = -\sqrt{(x - 150)^2 + (y - 150)^2} + 75$,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

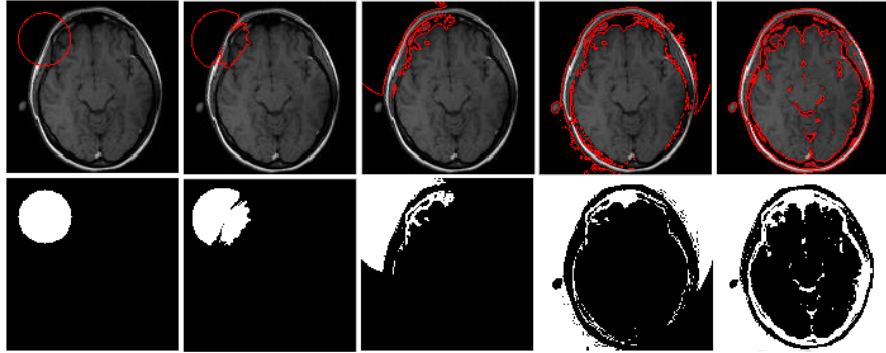


Figure 7. Initial curve position partially overlapping object. Top: the evolving curve (in red) over time where the first image shows the initial contour. Bottom: evolving segmentation over time until the object is detected. Size = 131 x 131, $\phi_0(x, y) = -\sqrt{(x - 30)^2 + (y - 30)^2} + 20$, $\mu = 0.01$, no reinitialization, cpu = 2.26 s, iterations = 9.

ject in any part. Visually, the obtained segmentation was the same with a slightly higher number of iterations and CPU time. In all three cases, the brain was correctly segmented. We compared the area segmented as the brain in each case to make sure the same segmentation was obtained, and in each case, we got the same area of 0.5268. Note, however,

that when part of the curve lies outside the object, the definition of inside and outside changes since the curve becomes an open curve at one point. This should not make a difference as long as the contour lies correctly at the boundaries of the object.

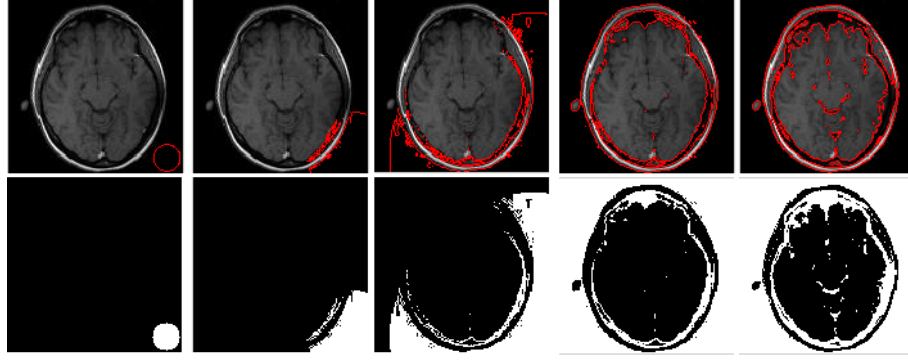


Figure 8. Initial curve position not overlapping any area of the object. Top: the evolving curve (in red) over time where the first image shows the initial contour. Bottom: evolving segmentation over time until the object is detected. Size = 300 x 300, $\phi_0(x, y) = -\sqrt{(x - 120)^2 + (y - 120)^2}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

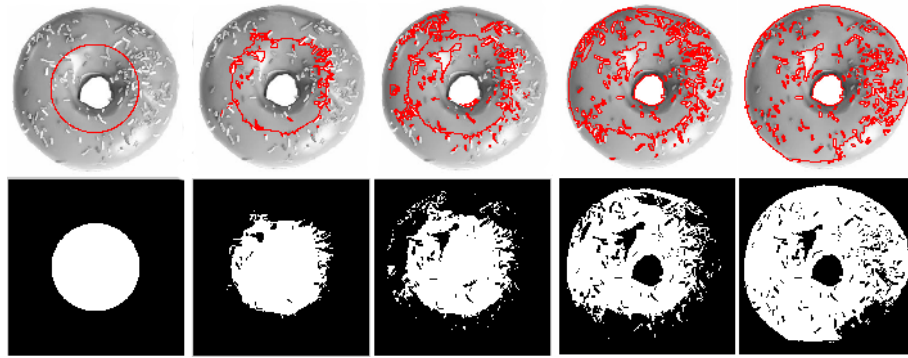


Figure 9. Successful detection of holes. Top: the evolving curve (in red) over time where the first image shows the initial contour. Bottom: evolving segmentation over time until the object is detected. Size = 131 x 131, $\phi_0(x, y) = -\sqrt{(x - 65.5)^2 + (y - 65.5)^2} + 32.8$, $\mu = 0.01$, no reinitialization,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

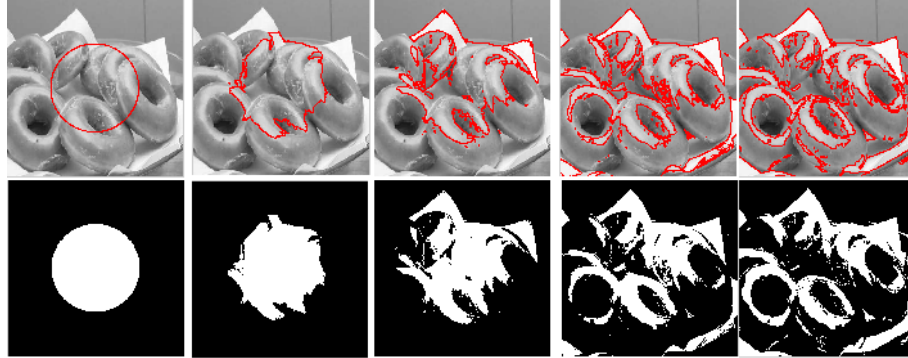


Figure 10. Successful detection of multiple holes. Top: the evolving curve (in red) over time where the first image shows the initial contour. Bottom: evolving segmentation over time until the object is detected. Size = 300 x 300, $\phi_0(x, y) = -\sqrt{(x - 150)^2 + (y - 150)^2} + 75$, $\mu = 0.01$, no reinitialization, cpu = 15.07 s, iterations = 23.

Successful Detection of Holes

In order to make sure the contour does not simply stop at the outside boundary of an object and ignore any details inside such as holes, we tried two donut images. Figure 9 shows how the curve is able to detect the hole in the donut. Additionally, it is able to detect the various sprinkles on it. We

note that the bottom right boundary is not very exact, and this is because the lighting effect in the image makes the intensity of this area very close to that of the background. This is a known drawback of the Chan-Vese model since it only divides the image into two regions, and requires a big variation in intensities for the curve to change. Another example showing this limitation is shown in Figure 11

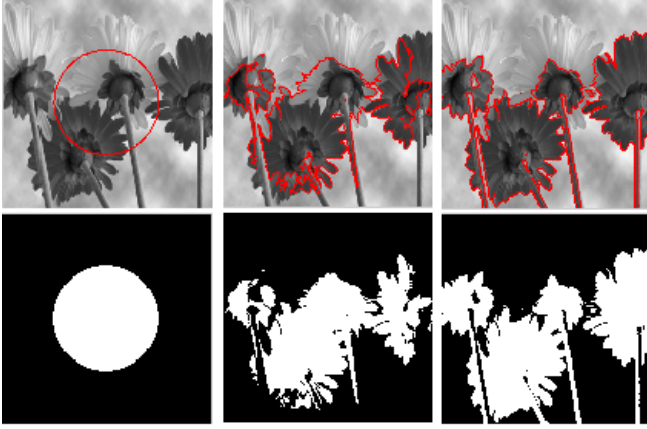


Figure 11. Inability to detect low intensities that do not have much variation from their background. Top: the evolving curve (in red) over time where the first image shows the initial contour. Bottom: evolving segmentation over time until the object is detected. Size = 300×300 , $\phi_0(x, y) = -\sqrt{(x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

where the flower parts with very low intensities are not detected. We will show that we can overcome this limitation with the Sandberg-Chan model for this particular example. Figure 10 shows that multiple holes in an image can also be successfully detected.

Reasonable Performance with Blurred Images

Figure 12 shows how the contour behaves in a blurry image. The photographer was successfully detected. Additionally, other objects in the image such as the tripod were found. However, objects with a very light intensity were again not detected.

Reasonable Performance with Noisy Images

Figure 13 shows how two objects in a noisy image were successfully detected. In this image, we added a Gaussian noise with mean = 0 and variation = 0.01 variance using Matlab's built-in noise function. The image shows that although some of the noise was detected in intermediate iterations, the contour continued to evolve until it was only surrounding the desired objects. Unfortunately, the effect of noise was not completely ignored in all cases. For example, Figure 14 shows that for the same image, but with *Salt & pepper* noise, the noise was detected as objects. This should not have been the case since we used a large value $\mu = 5$ for the length scaling parameter. However, varying μ does not have the intended effect as explained in the next paragraph.

Successful Response to Parameter Settings

There are mainly three parameters in this model that can be varied: μ , λ_1 , λ_2 . We would usually want both λ_1 and λ_2 to be equal to 1 to indicate that we care both about the difference of intensities inside and outside. A smaller weighting for one of them would mean that we would ignore some of the variation in intensities in this area. A larger weighting means that we are magnifying the difference in intensities in that area (i.e. we want to detect any minor changes). We show an example of these variations in Figure 15 where we vary λ_1 and λ_2 . When we set $\lambda_1 = 0.2$ and $\lambda_2 = 1$, less details within the interior of the brain is detected. When we set $\lambda_1 = 2$ and $\lambda_2 = 1$, we can see that more details within the brain are detected. When we set $\lambda_1 = 5$, and $\lambda_2 = 0.01$ (a very extreme case), we can see that parts of outside boundaries are not well detected, while many details are detected within the brain.

The second parameter, μ controls how much we allow the length of the curve to increase. If μ is small, it means that the curve length can increase to detect multiple smaller objects without penalizing our minimization problem. On the other hand if μ is large, it means that any change in the curve length will be scaled up, and thus will limit the curve expansion to keep the force to a minimum. Unfortunately, we were not able to see this effect in our experiments. We tried varying μ to be able to detect groupings of objects instead of the individual objects, but the segmentation was invariant to μ . We give more details about how we tried to handle this, and why we believe it is not working in Section 5.

4.2. Sandberg-Chan Model Results

For the Sandberg-Chan model, the successful detection of boundaries criterion is implicitly included in the logic operations functionality. We therefore, show the other criteria. Unless otherwise specified, $\mu = 0.1$ and $\lambda = 255 * 255$ in the experiments below.

Logic Operators: union and intersection

To ensure that the union operator is working properly, we tried it on a very simple example of two rectangles shown in Figure 16. The union of the two rectangles was successfully detected. Similarly, Figure 17 shows the intersection for the same two simple rectangles.

A practical usage of this logic operations framework is to recover missing parts from different images, and combine them to produce a more complete image. We show this in Figure 18 which shows two different version of a family image with a child missing in each image. When the union is applied to these two channels, the complete family image is recovered.

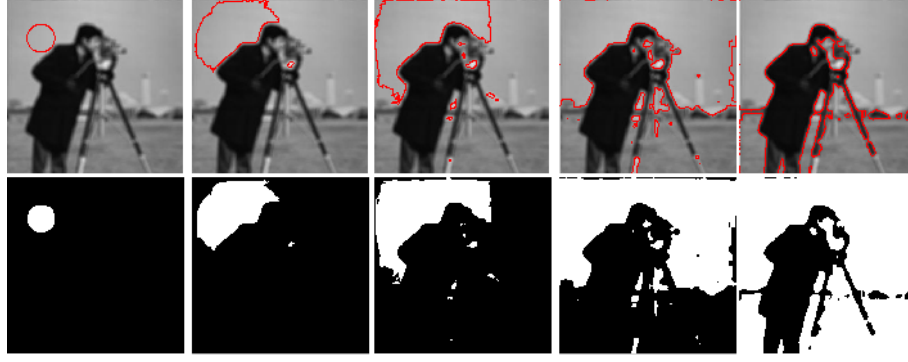


Figure 12. Ability to detect contours in blurry images. Top: the evolving curve (in red) over time where the first image shows the initial contour. Bottom: evolving segmentation over time until the object is detected. Size = 255 x 255, $\phi_0(x, y) = -\sqrt{(x - 127.5)^2 + (y - 127.5)^2} + 63.75$, $\mu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

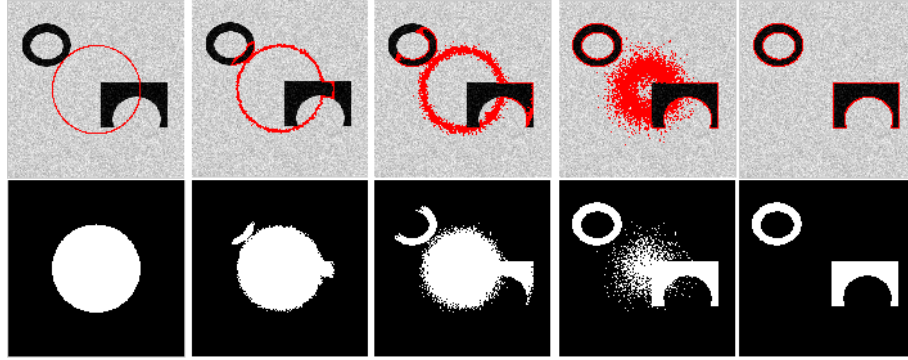


Figure 13. Ability to detect contours in an image with Gaussian noise. Top: the evolving curve (in red) over time where the first image shows the initial contour. Bottom: evolving segmentation over time until the object is detected. Size = 300 x 300, $\phi_0(x, y) = -\sqrt{(x - 150)^2 + (y - 150)^2}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

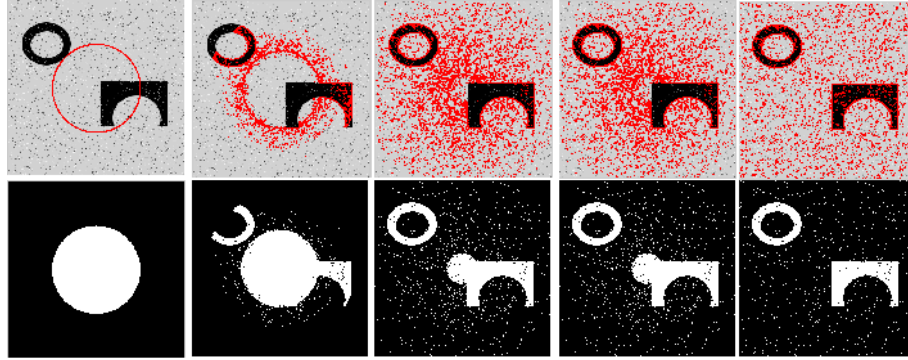


Figure 14. Figure 13, but with *Salt & pepper* noise. Noise was still detected despite increasing μ . Top: the evolving curve (in red) over time where the first image shows the initial contour. Bottom: evolving segmentation over time until the object is detected. Size = 300 x 300, $\phi_0(x, y) = -\sqrt{(x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

Figure 19 shows a more complicated example for both the intersection and union with the donut image shown in the previous section. We used two versions of the donut with each one having a different part occluded. The results of the union and intersection operators are shown in the figure. Additionally, this example shows that the Sandberg-

Chan model can still successfully detect holes despite the addition of the logic operations.

We also wanted to make sure that the way the intensities in each channel are defined does not matter. Figure 20 shows the union and intersection performed on two channels of opposite intensities inside and outside the objects.

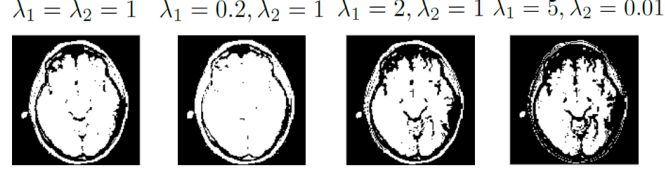


Figure 15. Effect of varying λ_1 which controls how much details are detected inside the contour. The figure shows the final segmentation in each case.

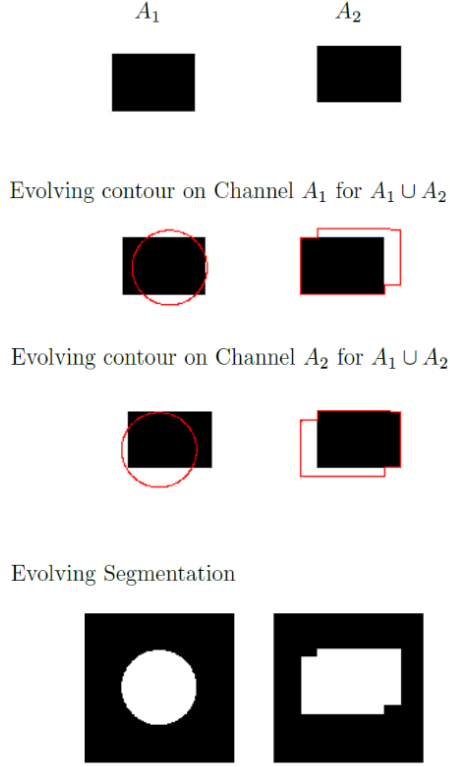


Figure 16. Simple union example. Size = 300 x 300, $\phi_0(x, y) = -\sqrt{(x - 150)^2 + (y - 150)^2} + 75$, no reinitialization, cpu = 1.32 s, iterations = 2.

The segmentation was indeed contrast invariant as claimed by the authors. Finally, we show that our implementation can be extended to more than two channels. Figure 21 shows the union, intersection, and the intersection of the union of three different channels.

Independence of Initial Curve Position

Similar to the previous model, we need to ensure that the position of the initial curve does not alter our results. Figure 22 shows the segmentation results obtained for the same image used in Figure 16, but with a different position for the initial curve. Both segmentation are the same.

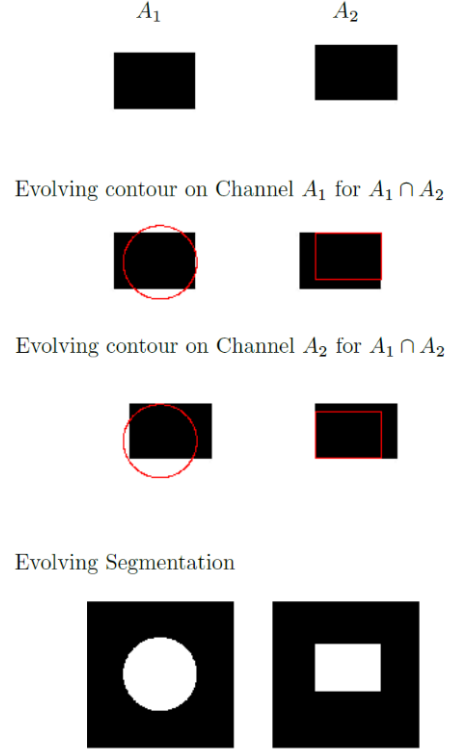


Figure 17. Simple intersection example. Size = 300 x 300, $\phi_0(x, y) = -\sqrt{(x - 150)^2 + (y - 150)^2} + 75$, no reinitialization, cpu = 1.83 s, iterations = 2.

Reasonable Performance with Blurred Images

Figure 23 shows the union performed on two blurry triangles. We blurred the triangles used in Figure 21, and performed the union operation on them. The full triangle was segmented (the union) despite the blurred edges.

Reasonable Performance with Noisy Images

To be able to perform well in noisy images, λ must be decreased in order to ignore the noise. We added *Salt & pepper* noise with a variation of 0.1 to the images shown in Figure 25, and tried to perform the union. In the first case shown, we set $\lambda = 255 * 255$ (i.e $\max(u)^2$) which is the default setting used for clean images, and in the second case, we set $\lambda = 125$. We show the final segmentation for both

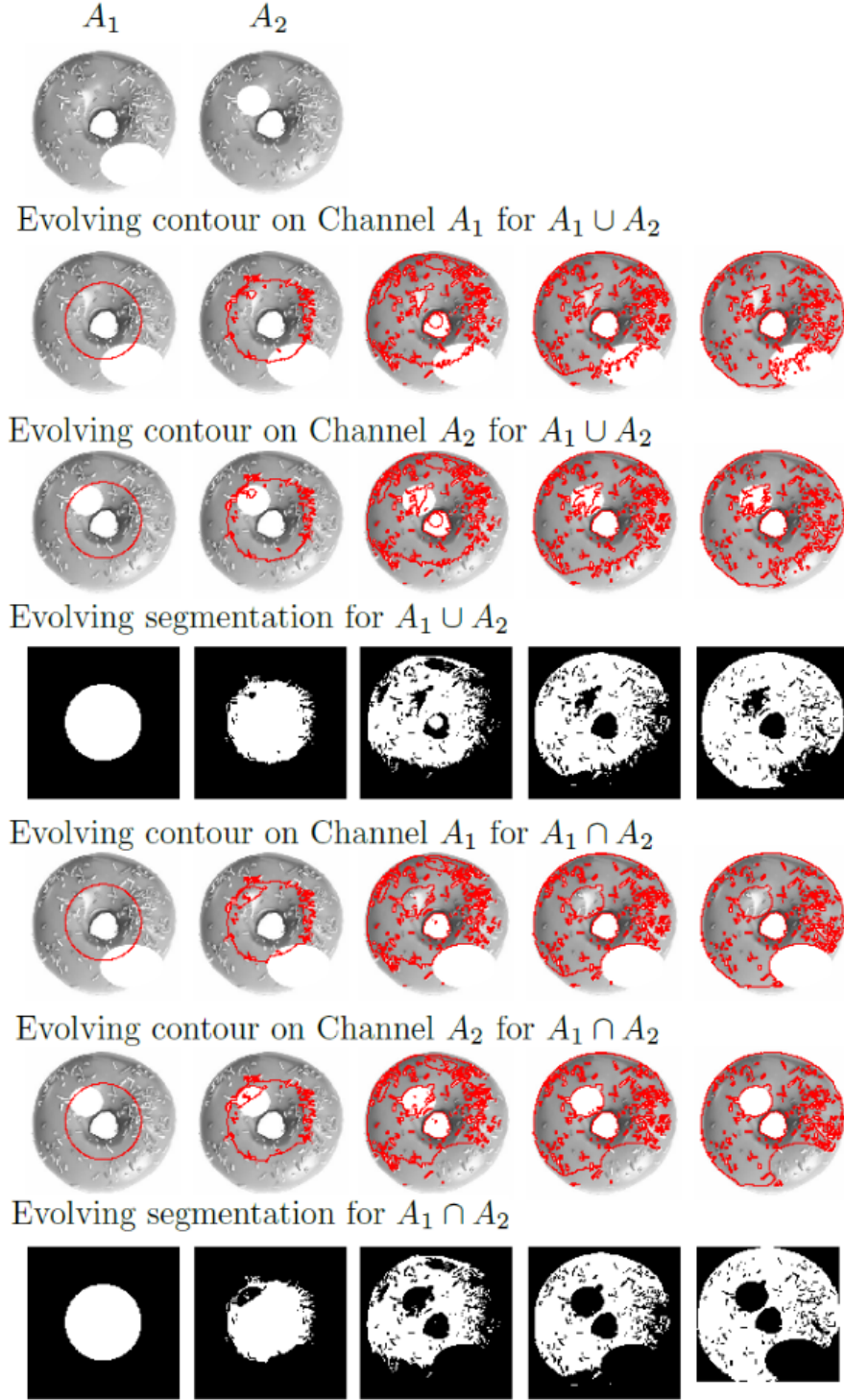


Figure 19. Performing union and intersection on images with holes. Size = 300 x 300. $\phi_0(x, y) = -\sqrt{(x - 150)^2 + (y - 150)^2} + 75$, no reinitialization. For union: cpu = 8.63 sec and iterations = 10. For intersection: cpu = 7.68 sec and iterations = 10.

cases. In the first case, the noise was detected, while in the second case, the noise was ignored since λ was small. This

shows that we also satisfy the response to parameter settings criteria since λ was the only variation mentioned by the au-

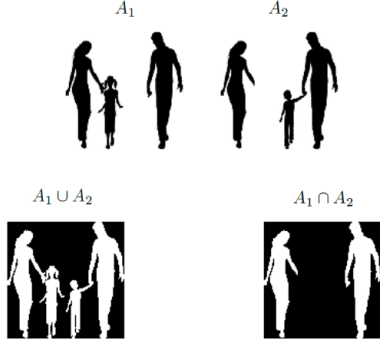


Figure 18. Combining missing information from two different channels. Size = 114 x 114. $\phi_0(x, y) = \sqrt{(x - 57)^2 + (y - 57)^2} - 28.5$, no reinitialization. Union: cpu = 4.01 s and iterations = 7. Intersection: cpu = 4.12 s and iterations = 7.

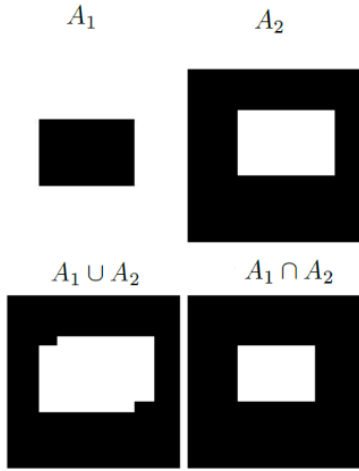


Figure 20. Segmentation for union and intersection of channels having reverse contrast.. Size = 300 x 300, $\phi_0(x, y) = -\sqrt{(x - 150)^2 + (y - 150)^2} + 75$, no reinitialization. For union: cpu = 1.32 sec and iterations = 2. For intersection: cpu = 1.83 sec and iterations = 2

thors in the original paper [5]. We note that the contour evolved much more slowly with a smaller λ , and thus took much more CPU time, and a larger number of iterations.

Complement

Unfortunately, the complement did not work properly in our implementation. We closely followed the definitions in the original paper where $z_i^{in'} = 1 - z_i^{in}$ and $z_i^{out'} = 1 - z_i^{out}$. The result was actually successfully detected, but continuous flickering occurred, and the curve did not stop evolving. We discuss this results in more details in Section 5.

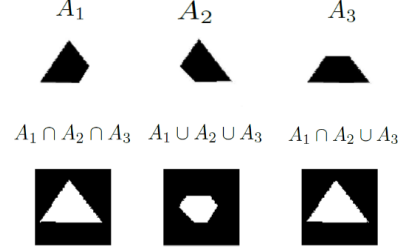


Figure 21. Logic operations performed on three channels. The figure shows the final segmentation in each case. Size = 200 x 200. $\phi_0(x, y) = -\sqrt{(x - 100)^2 + (y - 100)^2} + 50$, $\mu = 0.1$, no reinitialization. For union: cpu = 3.92 sec and iterations = 6. For intersection: cpu = 3.11 sec and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

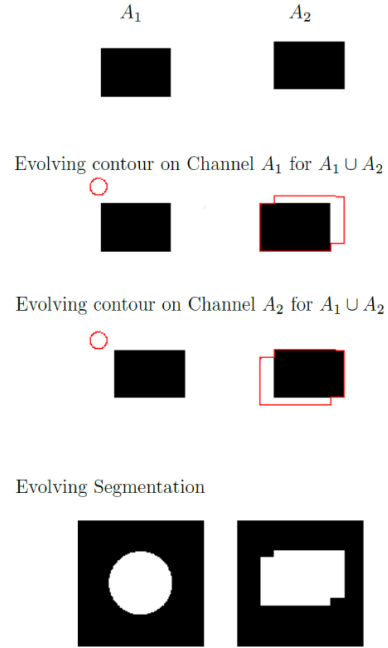


Figure 22. Same example shown in Figure 16, but with different initial curve position. Initial curve does not overlap with either objects. Same segmentation is obtained. Size = 300 x 300, $\phi_0(x, y) = -\sqrt{(x - 150)^2 + (y - 150)^2} + 75$, no reinitialization, cpu = 4.43s, 4 iterations.

Other Tests

Since the Sandberg-Chan model can successfully detect objects in one channel which are not in another when performing the union, we were curious to know whether this would allow us to perform successful segmentation on colored images by combining the information in the three channels R,G,B. Accordingly, we used the image in Figure 11 which the Chan-Vese model was not able to completely segment. We extracted three channels from the colored image: Red, Green, Blue, and performed the union on the three chan-

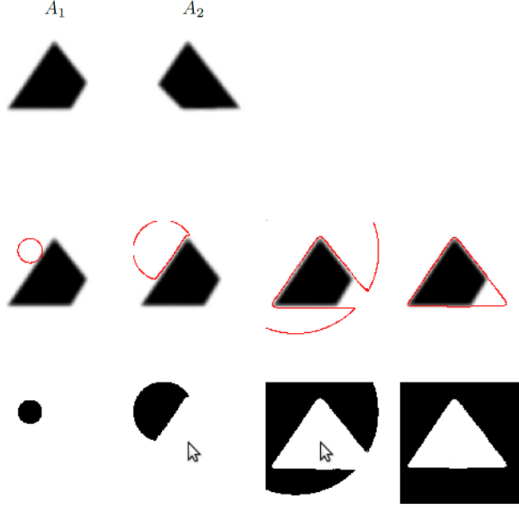


Figure 23. Successful union with blurred images. Top: the evolving curve for $A_1 \cup A_2$ (in red) over time where the first image shows the initial contour. Bottom: evolving segmentation over time until the union is detected. Size = 200 x 200, $\phi_0(x, y) = -\sqrt{(x-100)^2 + (y-100)^2} + 50$, no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

nels. The results are shown in Figure 24. The complete contour of the flowers was successfully detected unlike that obtained in Figure 11.

5. Difficulties and Discussion

We have faced many challenges throughout the implementation of this project. Some of the challenges caused major delays in the implementation cycle and hindered us from improving our code and adding more features to it.

5.1. Varying μ

μ is a scaling factor that is used in both the Chan-Vese model and the Sandberg-Chan model to control the weight given to the length of the contour. Therefore, varying the value of μ will affect the ability to segment objects in noisy images, especially when the noise is similar to Matlab’s *salt and pepper* noise. However, varying μ in our implementation does not really have any impact on the length of the contour. Figure 26 shows the segmentation of an object in a noisy image for different values for μ . The resulting segmentation is the same for all values of μ . It is worth noting that this challenge kept us busy for over two days, after which we decided we should move on to other parts of the project.

5.2. Computing the Complement

The Sandberg-Chan model supports the complement logic operator, so one can segment the logic object ($A \cap \neg B$)

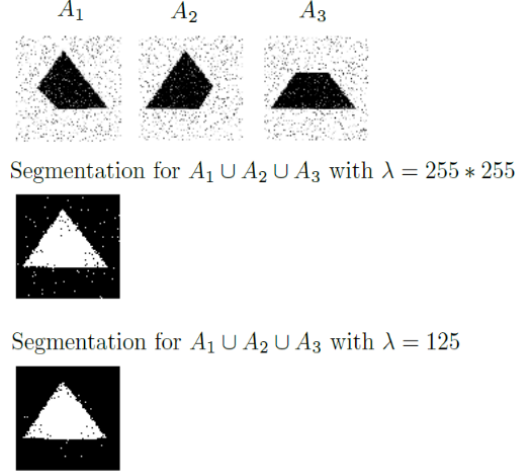


Figure 25. Union of noisy channels. Same image as Figure 21, but with *Salt & pepper* noise with a variation of 0.1. Decreasing λ from 255*255 to 125 allowed less noise to be detected. Size = 200 x 200, $\phi_0(x, y) = -\sqrt{(x-100)^2 + (y-100)^2} + 50$, no reinitialization. Cpu = 3.17 sec and iterations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

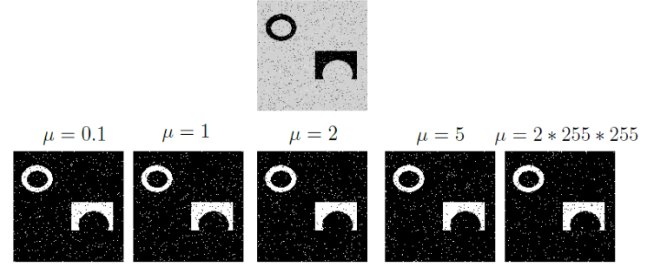


Figure 26. μ does not have any effect on the length of the contour. *Salt and pepper* noise was added to the original image for testing purposes. Top: original image with noise. Bottom: segmentation for various values of μ .

(i.e. segment what is in channel A and not in channel B). The model defines the complement of the object in channel i as follows:

$$z_i^{in'} = 1 - z_i^{in}, z_i^{out'} = 1 - z_i^{out}. \quad (7)$$

Although the definition is very simple, it did not work in practice. We implemented the complement logic operator as defined in Equation 7, and the segmentation does not stop at the required result. It rather keeps evolving after reaching the required solution, then keeps flipping between two segmentations: the correct segmentation, and the segmentation of $A \cap B$. Figure 27 shows this scenario. We tried to debug the reasons for this problem for three days.

Original colored image



R



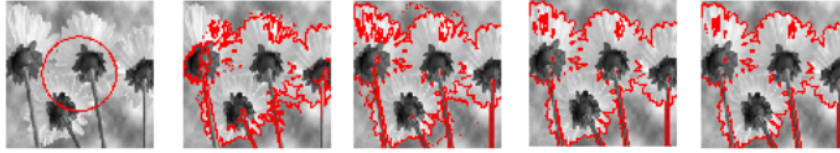
G



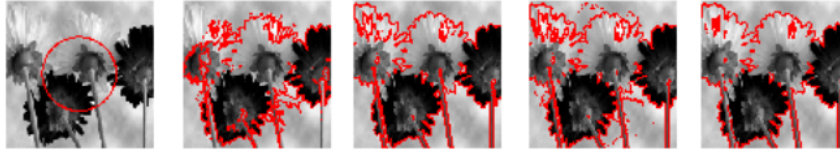
B



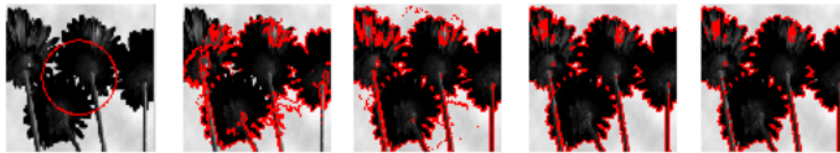
Evolving contour on R Channel for $R \cup G \cup B$



Evolving contour on G Channel for $R \cup G \cup B$



Evolving contour on B Channel for $R \cup G \cup B$



Evolving Segmentation

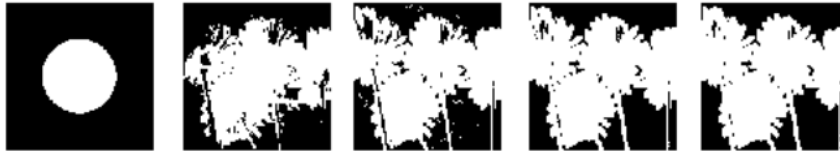


Figure 24. Ability to detect the full contour on a colored image with low variation between foreground and background by taking the union of the RGB channels. Full contour detected as opposed to Figure 11. Size = 300 x 300, $\phi_0(x, y) = -\sqrt{(x - 150)^2 + (y - 150)^2} + 75$, no reinitialization, size = cpu = 1.73 sec, iterations = 7.

5.3. Defining the initial contour ϕ_0

Both the Chan-Vese model and the Sandberg-Chan model define the initial contour (ϕ_0) as a signed distance function where ϕ_0 is positive inside the contour C , zero on

the boundary of C (∂C), and negative outside the contour C . Figure 28 illustrates those assumptions.

Since we model our initial contour as a circle, we have to modify the regular equation of the circle to match those

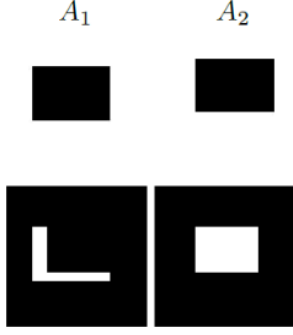


Figure 27. The complement operator causes the contour to go through an infinite loop between two segmentations: the correct segmentation, and the segmentation of $A_1 \cap A_2$. Bottom: correct and incorrect segmentation respectively

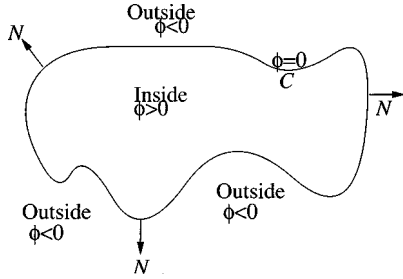


Figure 28. Assumptions about the way ϕ_0 is defined.

assumption. Therefore, we initialize ϕ_0 as follows:

$$\phi_0(x, y) = radius - \sqrt{(x - c_x)^2 + (y - c_y)^2}, \quad (8)$$

where c_x and c_y are the x-coordinate and y-coordinate of the center of the circle respectively.

Although this definition of ϕ_0 is compatible with most of the images we used to test our implementation, it did fail to correctly segment the objects in some images. In fact, the results were the opposite of the used logic operator (i.e. the output is the intersection of the channels, while it should be the union). After spending quite sometime to resolve this problem and checking what might be the source of the error, we tried using the regular equation of a circle to initialize ϕ_0 (Equation 9). We were very intrigued to notice that this modification resulted in the correct segmentation of the input channels. Figure 29 shows the final segmentation using both definitions for ϕ_0 .

$$\phi_0(x, y) = \sqrt{(x - c_x)^2 + (y - c_y)^2} - radius, \quad (9)$$

where c_x and c_y are the x-coordinate and y-coordinate of the center of the circle respectively.

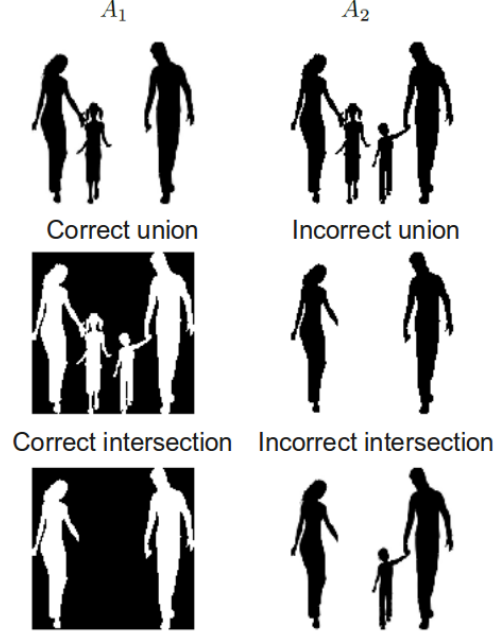


Figure 29. Union and intersection depending on the way ϕ is defined

5.4. Mistyped Equations

As previously mentioned in Section 2.2, there is a typo in the equations that define z_{in} and z_{out} in the Sandberg-Chan model (equation 3 in their paper). In addition, they have another typo in the expansion of the the Euler-Lagrange equation for the PDE that models the problem. Both typos consumed over a week of our time until we were able to find out what are the correct equations in both cases. We could have spent this time overcoming other challenges that we faced while implementing both models.

6. Conclusion

In this project, we have provided an implementation for the Chan-Vese model for active contours without edges and the Sandberg-Chan logic framework for multi-channel images, both written in Matlab. Our experiments have shown that the Chan-Vese model is effective for various types of images. In specific, it is useful when an edge-based segmentation is not applicable (i.e. edges do not depend on a gradient). However, our implementation fails to respond to different values for the contour length scaling parameter, μ . This affects the ability of our implementation to handle Matlab's *Salt & pepper* noise. On the other hand, changing the values for the intensity weight parameters λ_1 and λ_2 affects the amount of details obtained inside and outside the segmented object respectively.

We have also shown that our implementation for the Sandberg-Chan logic framework handles different logic

combinations for multiple channels. We have also experimented using it as a segmentation tool for colored images by modeling an input colored images as a 3-channels (red, green, blue). However, our implementation for the *complement* logic operation seems to be missing a hidden trick in the Sandberg-Chan logic framework.

References

- [1] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *International journal of computer vision*, 22(1):61–79, 1997.
- [2] T. Chan and L. Vese. Active contours without edges. *IEEE Transactions on image processing*, 10(2):266–277, 2001.
- [3] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988.
- [4] D. Mumford and J. Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on pure and applied mathematics*, 42(5):577–685, 1989.
- [5] B. Sandberg and T. Chan. A logic framework for active contours on multi-channel images. *Journal of Visual Communication and Image Representation*, 16(3):333–358, 2005.
- [6] G. Sapiro. Color Snakes* 1. *Computer Vision and Image Understanding*, 68(2):247–253, 1997.
- [7] S. Zhu and A. Yuille. Region competition: Unifying snakes, region growing, and Bayes/MDL for multi-band image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(9):884–900, 2002.

Appendices

A. Division of Labor

We both implemented the Chan-Vese model together since this was the starting point of the project. Sarah did most of the initial implementation for the Sandberg-Chan model by defining all the necessary functionalities for the logic operations for two images. Karim then extended it to multiple images. Karim also tested the Sandberg-Chan model, and took most of the snapshots in that part for the paper. Sarah tested the Chan-Vese model, and took all its snapshots for the paper. We both worked on finding an appropriate stopping condition for the contour evolution, and invested a lot of time to debug the problems we faced that were described in Section 5.