Analysis of Chan-Vese Segmentation

Yuhang Chen

April 2020

1 Introduction

Segmentation is the process of partitioning a digital image into multiple meaningful sections which is useful in many areas. There are many classical methods to implement image segmentation such as thresholding according to the image intensity, clustering and edge detection like Canny edge detecter.

While many segmentation methods rely heavily in some way on edge detection, the Chan-Vese model for active contours ignores edges completely[1]. The model is based on an energy minimization problem, which can be reformulated in the level set formulation, leading to an easier way to solve the problem. And it is based on Mumford-Shah functional for segmentation[2].

In this paper, we will derive the mathematical methods used in this algorithm and the numerical implemetation of this method. We will also do some experiments on different images and different pair of parameters to test performance of this algorithm and talk about the strengths, weaknesses and potential future improvements.

2 The Chan-Vese Model

2.1 Energy Functional

The Chan–Vese method is a region-based active contour method inspired by the Mumford–Shah model. It minimizes an energy which can be seen as a particular case of the minimal partition problem. As a result, we have to define an energy functional and at each time stamp, the goal of the Chan-Vese model is to progress to a lower energy which means it has to minimize the energy in both sides of the contour.

The energy functional used in Chan-Vese model can be defined as follows:[1]

$$E(C,u,v) = \lambda_1 \iint_R (I-u)^2 dxdy + \lambda_2 \iint_{R^C} (I-u)^2 dxdy + \mu \int_C ds + \nu \iint_R dxdy \tag{1}$$

The energy functional is represented by four terms where I denotes the image, u and v are the average intensity respectively inside and outside of the contour, C is the contour, R is the region inside of the contour and R^C is the region outside of the contour. $\lambda_1, \lambda_2, \mu, \nu$ are fixed parameters which are chosen by us. The overall goal is to minimize all of the four terms at each time step. The first term is the energy inside of the contour and the second term is the energy outside of the contour. These two terms represent the discrepancy between the image I and the two constants u

and v, to minimize these two terms, we want to keep the regions as uniform as possible. The third term penalize the length of the contour which will keep the contour as short and smooth as possible to avoid complicated curves and help us dealing with noisy situations. The fourth term penalize the enclosed area of the contour to control its size.

In order to find out the change of contour C at each time step, we need to derive the gradient descent C_t of the energy functional E which will be discussed in the next section.

2.2 Gradient Descent

The gradient descent of a function is defined as follows:

$$C_t = -\nabla E \tag{2}$$

Suppose C varies in time, then $E(C(t))$ is a time-vary energy and we can take the derivative in time as follows:

$$\frac{d}{dt}E(C(t)) = \frac{\partial E}{\partial C_t} =< C_t, \nabla E >= \int_C (C_t \cdot \nabla E)ds \tag{3}$$

From (3), we can see that the time derivative of the energy functional can be represented as the inner product of C_t and ∇E. In this case, once we isolate C_t from the integral, we will be able to extract ∇E.

In order to simplify, in this section, we will use f_{in} to represent $(I-u)^2$, and f_{out} to represent $(I-v)^2$. So the first term of the energy functional (1) can be written as $\iint_R f_{in}dxdy$. And we will focus on the first term of the energy functional since once we have derived the first term, we can reform the integral outside the region as the integral over the whole area minus the integral inside the region. And since the integral over the whole area doesn't depend on the curve, we have:

$$\iint_{R^C} f_{out}dxdy = \iint_{\Omega} f_{out}dxdy - \iint_R f_{out}dxdy = 0 - \iint_R f_{out}dxdy = -\iint_R f_{out}dxdy \tag{4}$$

In order to distinguish the energy functional derived above, we will denote the first term of the energy functional as follows:

$$\tilde{E}(C) = \iint_R f_{in}dxdy \tag{5}$$

Recall the divergence theorem:

$$\int_C \vec{F} \cdot N ds = \iint_R \nabla \cdot \vec{F} dxdy \tag{6}$$

In this section, we want to use the backward of the divergence theorem. Suppose we choose $\vec{F}$ such that $\nabla \cdot \vec{F} = f_{in}$, then we have:

$$\tilde{E}(C) = \iint_R f_{in}dxdy = \iint_R \nabla \cdot \vec{F} dxdy = \int_C \vec{F} \cdot N ds \tag{7}$$

Now, we can take the time derivative of $\tilde{E}(C)$ as:

$$\frac{d\tilde{E}(C)}{dt} = \frac{d}{dt}\int_C \vec{F} \cdot N ds \tag{8}$$

Since the variable s is time-dependent, we have to replace it with some other variable that is time-independent. Obviously, if a curve is evolving, and we look at the continuum of the curves over time, we can parameterize each one from 0 to 1. We can arbitrarily put any point on one curve in correspondence with any point on the other curve so that we can come up with the parameterization that is independent of time. In this case, we will replace ds with $||C_p||dp$ and we have:

$$\frac{d\tilde{E}(C)}{dt} = \frac{d}{dt}\int_C \vec{F}\cdot N ds = \frac{d}{dt}\int_0^1 \vec{F}\cdot N||C_p||dp \tag{9}$$

Now, we can take the derivative inside as:

$$\frac{d}{dt}\int_0^1 \vec{F}\cdot N||C_p||dp = \int_0^1 [(\frac{\partial}{\partial t}\vec{F})N||C_p|| + \vec{F}\cdot(\frac{\partial}{\partial t}N)||C_p|| + \vec{F}\cdot N(\frac{\partial}{\partial t}||C_p||)]dp \tag{10}$$

For the third term, we have:

$$\frac{d}{dt}||C_p|| = \frac{d}{dt}\sqrt{C_p\cdot C_p} = \frac{C_{pt}\cdot C_p}{||C_p||} \tag{11}$$

For the first term, since $\vec{F}$ is a vector, the derivative of $\vec{F}$ will look like:

$$\frac{\partial}{\partial t}\vec{F} = \left[\frac{\partial \vec{F}}{\partial \vec{x}}\right] C_t \tag{12}$$

where $\left[\frac{\partial \vec{F}}{\partial \vec{x}}\right]$ is a Jacobian matrix: $\begin{bmatrix} \frac{\partial}{\partial x}F_1 & \frac{\partial}{\partial y}F_1 \\ \frac{\partial}{\partial x}F_2 & \frac{\partial}{\partial y}F_2 \end{bmatrix}$.

For the second term, we can write the normal as a 90° rotation of the tangent:

$$N = JT = J\frac{C_p}{||C_p||} \tag{13}$$

where $J = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$ and $T = \frac{C_p}{||C_p||}$. Now, we take the time derivative to the above equation on both sides as:

$$\begin{aligned} \frac{\partial}{\partial t}N &= \frac{\partial}{\partial t}(J\frac{C_p}{||C_p||}) = J(\frac{\partial}{\partial t}\frac{C_p}{||C_p||}) = J(\frac{C_{pt}}{||C_p||} - \frac{C_p}{||C_p||^2}||C_p||_t) \\ &= \frac{1}{||C_p||}J(C_{pt} - T(C_{pt}\cdot T)) = \frac{1}{||C_p||}J(N(C_{pt}\cdot N)) \\ &= \frac{-T(C_{pt}\cdot N)}{||C_p||} \end{aligned} \tag{14}$$

Combining these together, we now have:

$$\begin{aligned} \frac{d}{dt}\int_0^1 \vec{F}\cdot N||C_p||dp &= \int_0^1 [(\left[\frac{\partial \vec{F}}{\partial \vec{x}}\right] C_t)N||C_p|| - \vec{F}(\frac{-T(C_{pt}\cdot N)}{||C_p||})||C_p|| + \vec{F}\cdot N(\frac{C_{pt}\cdot C_p}{||C_p||})]dp \\ &= \int_0^1 [(\left[\frac{\partial \vec{F}}{\partial \vec{x}}\right] C_t)N||C_p|| - \vec{F}\cdot T(C_{tp}\cdot N) + \vec{F}\cdot N(C_{tp}\cdot T)]dp \\ &= \int_0^1 [(\left[\frac{\partial \vec{F}}{\partial \vec{x}}\right] C_t)N - (\vec{F}\cdot T)(C_{ts}\cdot N) + \vec{F}\cdot N(C_{ts}\cdot T)]ds \end{aligned} \tag{15}$$

So far, we can obtain:

$$\begin{aligned}\frac{d\tilde{E}(C)}{dt} &= \int_C [(\left[\frac{\partial \vec{F}}{\partial \vec{x}}\right] C_t)N + C_t((\vec{F}\cdot T)N)_s - C_t((\vec{F}\cdot N)T)_s]ds \\ &= \int_C C_t[\left[\frac{\partial \vec{F}}{\partial \vec{x}}\right]^T N + ((\vec{F}\cdot T)N)_s - ((\vec{F}\cdot N)T)_s]ds\end{aligned} \tag{16}$$

where inside the square brackets is the gradient of energy $\tilde{E}$ which is not dependent on t anymore, to further simplify the gradient, we have:

$$\nabla \tilde{E} = \left[\frac{\partial \vec{F}}{\partial \vec{x}}\right]^T N + (\vec{F}_s\cdot T)N + (\vec{F}\cdot T_s)N + (\vec{F}\cdot T)N_s - (\vec{F}_s\cdot N)T - (\vec{F}\cdot N_s)T - (\vec{F}\cdot N)T_s \tag{17}$$

Noting that $\vec{F}_s = \frac{\partial}{\partial s}\vec{F}(C(s,t)) = \left[\frac{\partial \vec{F}}{\partial \vec{x}}\right]T$, $T_s = -\kappa N$ where κ is the curvature of the contour and $N_s = JT_s = -\kappa JN = \kappa T$. We can rewrite the gradient as:

$$\begin{aligned}\nabla \tilde{E} = &\left[\frac{\partial \vec{F}}{\partial \vec{x}}\right]^T N \\ &+ (T^T\left[\frac{\partial \vec{F}}{\partial \vec{x}}\right]T)N - \kappa(\vec{F}\cdot N)N + \kappa(\vec{F}\cdot T)T \\ &- (N^T\left[\frac{\partial \vec{F}}{\partial \vec{x}}\right]T)T - \kappa(\vec{F}\cdot T)T + \kappa(\vec{F}\cdot N)N \\ &= \left[\frac{\partial \vec{F}}{\partial \vec{x}}\right]^T N + (T^T\left[\frac{\partial \vec{F}}{\partial \vec{x}}\right]T)N - (N^T\left[\frac{\partial \vec{F}}{\partial \vec{x}}\right]T)T \\ &= (T^T\left[\frac{\partial \vec{F}}{\partial \vec{x}}\right]^T N)T + (N^T\left[\frac{\partial \vec{F}}{\partial \vec{x}}\right]^T N)N + (T^T\left[\frac{\partial \vec{F}}{\partial \vec{x}}\right]T)N - (N^T\left[\frac{\partial \vec{F}}{\partial \vec{x}}\right]T)T \\ &= (N^T\left[\frac{\partial \vec{F}}{\partial \vec{x}}\right]N + T^T\left[\frac{\partial \vec{F}}{\partial \vec{x}}\right]T)N\end{aligned} \tag{18}$$

Now after we get rid of the tangential component, only the normal component of the gradient influences the contour, which matches the behavior of a geometric functional. Both T and N form an orthonormal basis and $\left[\frac{\partial \vec{F}}{\partial \vec{x}}\right]$ is diagonalizable. Recall that $a^T Aa + b^T Ab = trace(A)$ where a, b are part of an orthonormal basis and A is diagonalizable. Applying this to (18), we have:

$$\nabla \tilde{E} = trace(\left[\frac{\partial \vec{F}}{\partial \vec{x}}\right])N = trace(\begin{bmatrix}\frac{\partial}{\partial x}F_1 & \frac{\partial}{\partial y}F_1 \\ \frac{\partial}{\partial x}F_2 & \frac{\partial}{\partial y}F_2\end{bmatrix})N = (\frac{\partial}{\partial x}F_1 + \frac{\partial}{\partial y}F_2)N = (\nabla\cdot\vec{F})N \tag{19}$$

Using the original definition of the function $f_{in} = \nabla\cdot\vec{F}$, we finally arrive at:

$$\nabla \tilde{E} = f_{in}N \tag{20}$$

For (4), we can use the same method derived above and obtain:

$$\nabla \tilde{E} = -f_{out}N \tag{21}$$

According to above, we know that the gradient of the fourth term is similar to the first and second term, so we have:

$$\nabla \tilde{E} = N \tag{22}$$

Now, we only have the third term left, based on our derivation above, we can easily obtain that:

$$\frac{d\tilde{E}(C)}{dt} = \frac{d}{dt}\int_C ds = \frac{d}{dt}\int_0^1 ||C_p||dp = \int_0^1 (\frac{C_{pt}\cdot C_p}{||C_p||})dp = \int_0^1 (C_{pt}\cdot T)dp = \int_0^1 (C_{tp}\cdot T)dp \tag{23}$$

Applying integration by parts, we now have:

$$\frac{d\tilde{E}(C)}{dt} = -\int_0^1 (C_t \cdot T_p)dp = -\int_0^1 (C_t \cdot T_s)ds \tag{24}$$

Extracting $\nabla \tilde{E}$ this, we get:

$$\nabla \tilde{E} = -T_s = -\kappa N \tag{25}$$

Combining these four terms together, we now have the gradient of the energy functional (1) defined by Chan-Vese model as:

$$\nabla E = \lambda_1(I-u)^2N - \lambda_2(I-v)^2N - \mu\kappa N + \nu N \tag{26}$$

And the gradient descent will be represented as:

$$C_t = -\nabla E \tag{27}$$

2.3 Level-set Functions

The minimization problem requires minimizing over all set boundaries C. This is accomplished by applying the level-set technique, instead of manipulating C explicitly, it is represented as the level set function ψ by the relationship:

$$C = \{\begin{pmatrix} x \\ y \end{pmatrix} : \psi(x,y) = constant\} \tag{28}$$

Suppose $\psi(x,y,t)$ is the time evolving implicit representation of $C_{p,t} = \begin{pmatrix} x(p,t) \\ y(p,t) \end{pmatrix}$ which means $\psi((C_{p,t}),t) = constant$. Now, let's take the total derivative in time, we have:

$$\frac{d}{dt}\psi((C_{p,t}),t) = 0 \tag{29}$$

Implementing the chain rule, we have:

$$\nabla\psi \cdot C_t + \psi_t = 0 \tag{30}$$

After we rearrange the equation above, we have:

$$\psi_t = -\nabla\psi \cdot C_t \tag{31}$$

Inserting the result of C_t we have calculated in (26) and (27) and using the value of curvature $\kappa = -\nabla \cdot (\frac{\nabla\psi}{||\nabla\psi||})$, we now have the level-set function:

$$\psi_t = \lambda_1(I-u)^2||\nabla\psi|| - \lambda_2(I-v)^2||\nabla\psi|| + \mu\nabla\cdot(\frac{\nabla\psi}{||\nabla\psi||})||\nabla\psi|| + \nu||\nabla\psi|| \tag{32}$$

3 Numerical Approximation

3.1 Discretization Scheme

In order to apply the level-set function to an image, a discretization scheme needed to be defined for the derivatives in (32). Looking at the equation derived above, we can see that except for the third term, the other three terms all have the similar form. So let's first look at the third term of the level-set function. In order to distinguish the level-set equation of the desired image and expand the general form of this part, we rewrite this term as follows:

$$\tilde{\psi_t} = \alpha\nabla\cdot(\frac{\nabla\psi}{||\nabla\psi||})||\nabla\psi|| = \alpha\frac{\psi_x^2\psi_{yy} - 2\psi_x\psi_y\psi xy + \psi_y^2\psi_{xx}}{\psi_x^2 + \psi_y^2} \tag{33}$$

This formula involves the first and second order partial derivatives which can be best represented using the central differences. The first order partial derivatives are:

$$\begin{aligned} \psi_x(x,y,t) &= \frac{\psi(x+\Delta x,y,t) - \psi(x-\Delta x,y,t)}{2\Delta x} \\ \psi_x(x,y,t) &= \frac{\psi(x,y+\Delta y,t) - \psi(x,y-\Delta y,t)}{2\Delta y} \end{aligned} \tag{34}$$

And the second order partial derivatives are:

$$\begin{aligned} \psi_{xx}(x,y,t) &= \frac{\psi(x+\Delta x,y,t) - 2\psi(x,y,t) + \psi(x-\Delta x,y,t)}{\Delta x^2} \\ \psi_{yy}(x,y,t) &= \frac{\psi(x,y+\Delta y,t) - 2\psi(x,y,t) + \psi(x,y-\Delta y,t)}{\Delta y^2} \\ \psi_{xy}(x,y,t) &= \frac{\psi(x+\Delta x,y+\Delta y,t) - 2\psi(x,y,t) + \psi(x-\Delta x,y-\Delta y,t)}{\Delta x\Delta y} \end{aligned} \tag{35}$$

In order to maitain the stability of the approximation, we also have to intorduce the CFL condition as:

$$\alpha\Delta t \le \frac{1}{2}\Delta x^2, \alpha\Delta t \le \frac{1}{2}\Delta y^2 \tag{36}$$

For the other three terms, first, we cant rewrite the level-set equation as follows:

$$\tilde{\psi_t} = \beta||\nabla\psi|| \tag{37}$$

Noting that the term's approximation depends on the sign of its coefficient and the signs of the approximation schemes used in it. So an upwind entropy scheme is needed:

$$\beta||\nabla\psi|| = \begin{cases} \beta\sqrt{min^2(D_x^-\psi,0)+max^2(D_x^+\psi,0)+min^2(D_y^-\psi,0)+max^2(D_x^+\psi,0)}, & \beta > 0 \\ \beta\sqrt{min^2(D_x^+\psi,0)+max^2(D_x^-\psi,0)+min^2(D_y^+\psi,0)+max^2(D_x^-\psi,0)}, & \beta < 0 \end{cases} \tag{38}$$

Both the backward and forward difference are used in the calculation of the norm of the level-set gradient. The forward differences used in (38) can be represented as:

$$\begin{aligned} D_x^+\psi &= \frac{\psi(x+\Delta x,y,t)-\psi(x,y,t)}{\Delta x} \\ D_y^+\psi &= \frac{\psi(x,y+\Delta y,t)-\psi(x,y,t)}{\Delta y} \end{aligned} \tag{39}$$

And the backward differences can be represented as:

$$\begin{aligned} D_x^-\psi &= \frac{\psi(x,y,t)-\psi(x-\Delta x,y,t)}{\Delta x} \\ D_y^-\psi &= \frac{\psi(x,y,t)-\psi(x,y-\Delta y,t)}{\Delta y} \end{aligned} \tag{40}$$

As a result, the discretization will now be calculated depending on the sign of the coefficient and the signs of both differencing schemes. And the CFL condition is:

$$\beta\Delta t \leq \frac{1}{\sqrt{2}}\Delta x, \beta\Delta t \leq \frac{1}{\sqrt{2}}\Delta y \tag{41}$$

Since we use two different discretization schemes, we have two CFL conditions needed to be satisfied. Therefore, for the overall discretization schemes, we have to fulfill the most restrictive CFL conditions derived above. In this case, we have to fulfill the CFL conditions in (36).

3.2 Algorithm in Pseudo Code

The basic implementation of the Chan-Vese model can be described as follows:

1. Initialize $\lambda_1, \lambda_2, \mu, \nu, \Delta t$.
2. Initialize ψ^0 with a desired shape and location for the contour.
3. Use the region defined by the current contour to compute the average intensity μ, ν.
4. Calculate ψ_t using (33) and (38) and μ, ν.
5. Calculate $\psi^{n+1} = \psi^n + \Delta t \cdot \psi_t$ to evolve contour.
6. Check whether the solution is stationary, if not, repeat step 3 - 5, else stop.

In this paper, in order to simplify the code, we set $\Delta x = \Delta y = 1$. Applying this to the CFL condition in (36), we have $\Delta t \leq 0.5$. So we set $\Delta t = 0.1$. For the remaining parameters in step 1, we will first set the default value as $\lambda_1 = \lambda_2 = 1, \mu = 0.5, \nu = 0$. We will then discuss the results of choosing different pairs of parameters.

4 Experimental Results

4.1 Performance on Regular Images

First, we applied the Chan-Vese model with default parameters to a picture of "Leonardo" and recorded the contour derived by each iteration as shown in Figure 1.

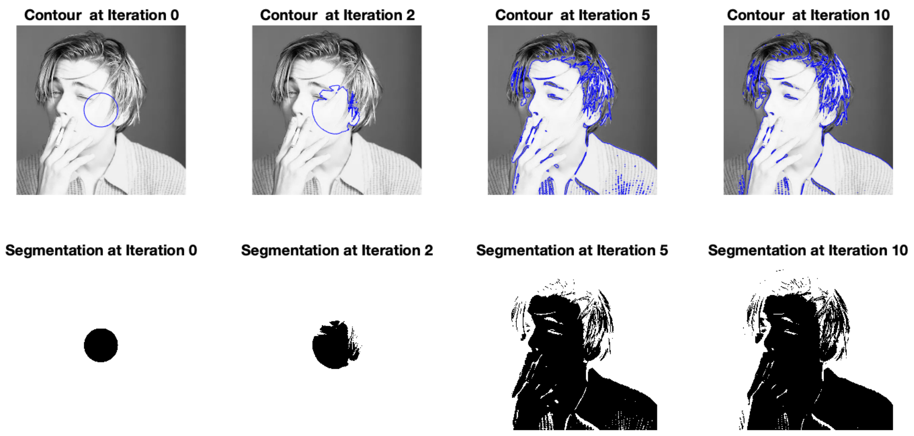

Contour at Iteration 0
Contour at Iteration 2
Contour at Iteration 5
Contour at Iteration 10
Segmentation at Iteration 0
Segmentation at Iteration 2
Segmentation at Iteration 5
Segmentation at Iteration 10

Figure 1: Chan-Vese Algorithm on "Leonardo"

From the evolution of the contour, we can see that only after five iterations, we can obtain an overall contour of the image. And the result is quite good.

Since in Chan-Vese model, at each step, our goal is to minimize the energy funcitonal (1), we can further examine this model, we then look at the evolution of energy after each iteration. As shown in Figure 2, we can see that there is a sharp drop between the third and fifth iterations which corresponds to the fully evolving of the contour, we can also see that after six rounds of iterations, the energy becomes stable.

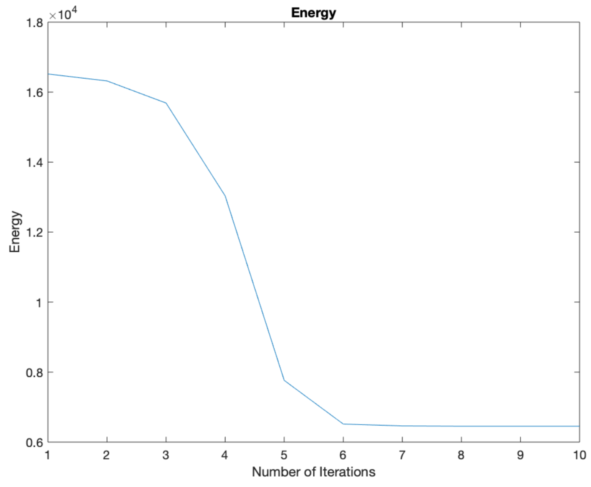


Figure 2: Energy Evolution on "Leonardo"

Now, let's implement the Chan-Vese Algorithm on a texture picture of stars. As shown in Figure 3, we can see that the Chan-Vese algorithm explicitly contour the edge of each stars which shows that this algorithm works best under the situation when the object can be clearly separated from the background.

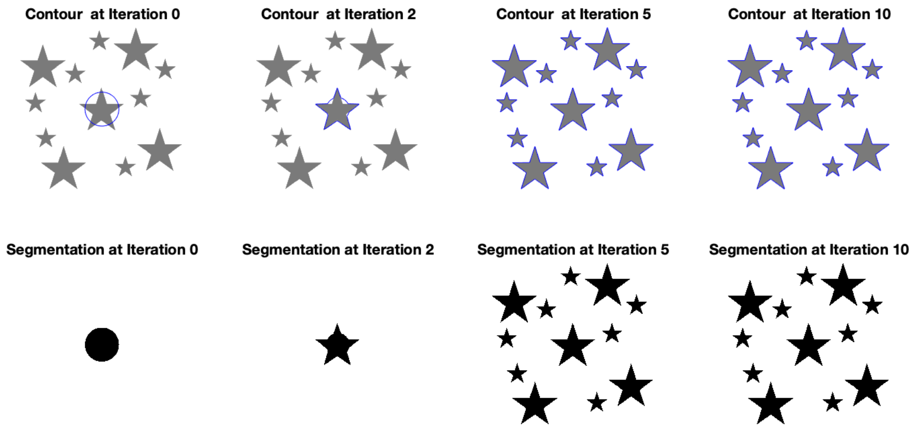

Contour at Iteration 0
Contour at Iteration 2
Contour at Iteration 5
Contour at Iteration 10
Segmentation at Iteration 0
Segmentation at Iteration 2
Segmentation at Iteration 5
Segmentation at Iteration 10

Figure 3: Chan-Vese Algorithm on "Stars"

The energy evolution shown in Figure 4 also shows that Chan-Vese Algorithm can quickly reach the minimal of the energy.

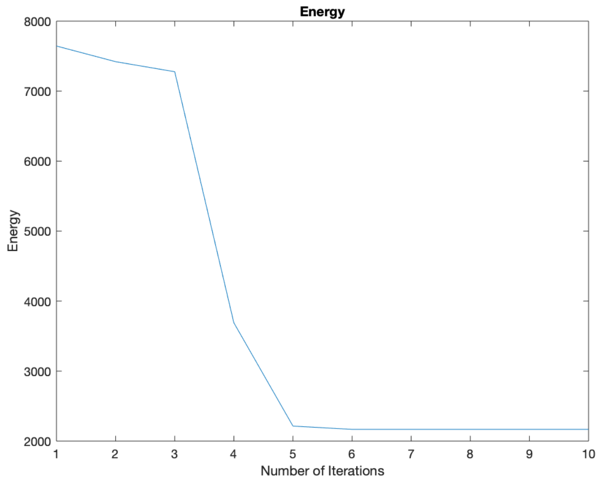


Figure 4: Energy Evolution on on ”Stars”

4.2 Performance on Nosiy Images

In order to further test the practicality of this algorithm, we run some tests on noisy images, and here are the results. As shown in Figure 5 and Figure 6, we added Gaussian noise and Salt & Pepper noise separately to the image ”Leonardo”.

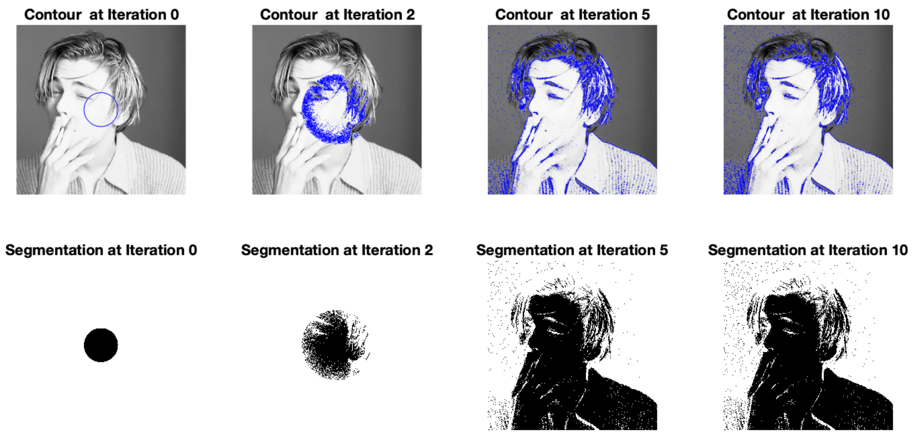
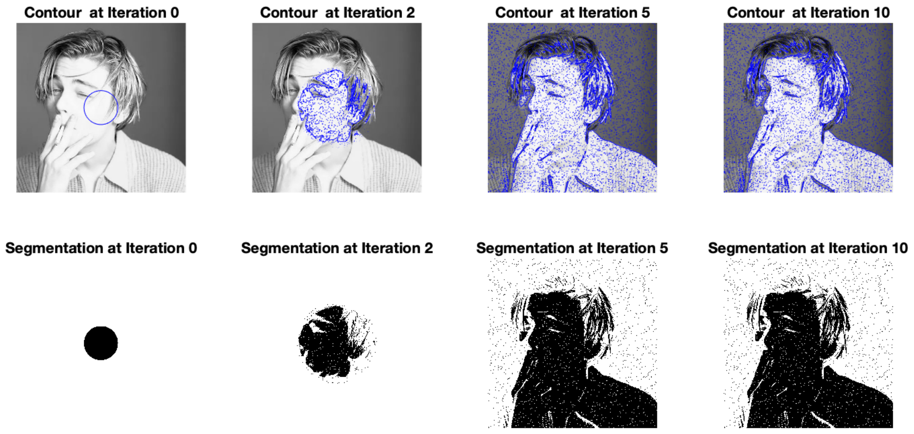

Contour at Iteration 0
Contour at Iteration 2
Contour at Iteration 5
Contour at Iteration 10
Segmentation at Iteration 0
Segmentation at Iteration 2
Segmentation at Iteration 5
Segmentation at Iteration 10

Figure 5: Chan-Vese algorithm on ”Leonardo” with Gaussian Noise

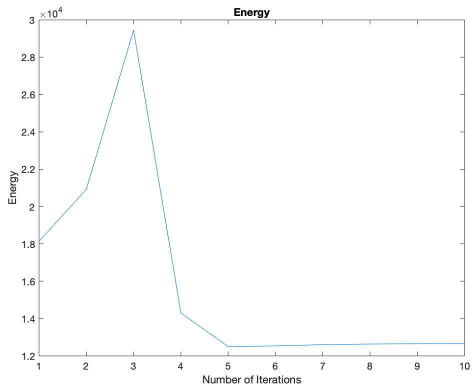


Contour at Iteration 0
Contour at Iteration 2
Contour at Iteration 5
Contour at Iteration 10
Segmentation at Iteration 0
Segmentation at Iteration 2
Segmentation at Iteration 5
Segmentation at Iteration 10

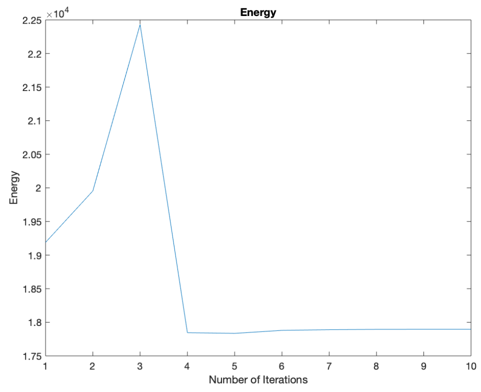
Figure 6: Chan-Vese algorithm on "Leonardo" with Salt & Pepper Noise

We can see from the results that although there are some undesirable effects due to the noise, we can still obtain the overall contour of the image. We can also see that different kinds of noise have different impacts on this algorithm, when adding Salt & Pepper noise to the image, the final contour enclosed more noise pixels and the result is worse than adding Gaussian noise.

As for the energy evolution shown in Figure 7, different from the energy evolution of the regular picture, we can see that the energy first go up rapidly and then go down sharply before reaching a steady state due to noise pixels. The final minimal energy of "Leonardo" with Gaussian noise is also lower than "Leonardo" with Salt & Pepper noise.



(a) Energy on "Leonardo" with Gaussian Noise



(b) Energy on "Leonardo" with Salt & Pepper Noise

Figure 7: Energy Evolution on "Leonardo" with Salt & Pepper Noise

4.3 Effect of Varying μ

In Chan-Vese algorithm, parameter μ adjusts the length penalty, which balances between fitting the input image more accurately (smaller μ) versus producing a smoother boundary (larger μ) as shown in Figure 8.

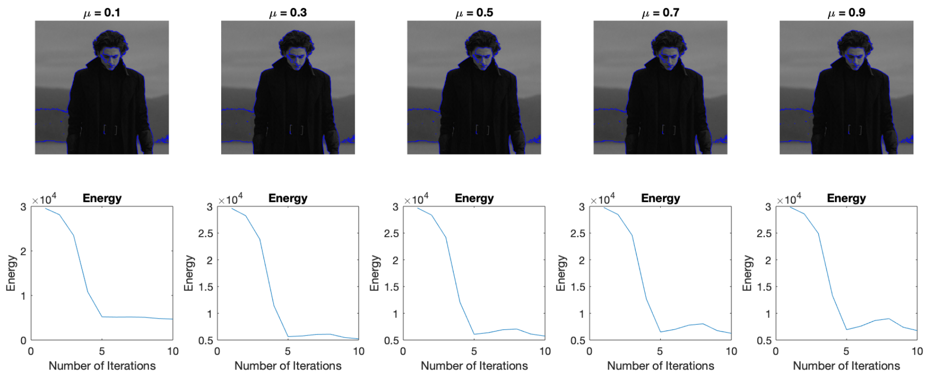


Figure 8: Chan-Vese algorithm with different μ on "Timothee"

4.4 Effect of Varying ν

Parameter *nu* sets the penalty (or reward, if $\nu < 0$) for area inside the contour. Noting that this parameter is meaningful only when there is a prescribed inside and outside of the segmentation boundary. In Figure 9, the evolution is shown with five different values of ν. When ν is too negative, the boundary expands to fill the full domain, and when ν is too positive, the boundary shrinks until it vanishes.

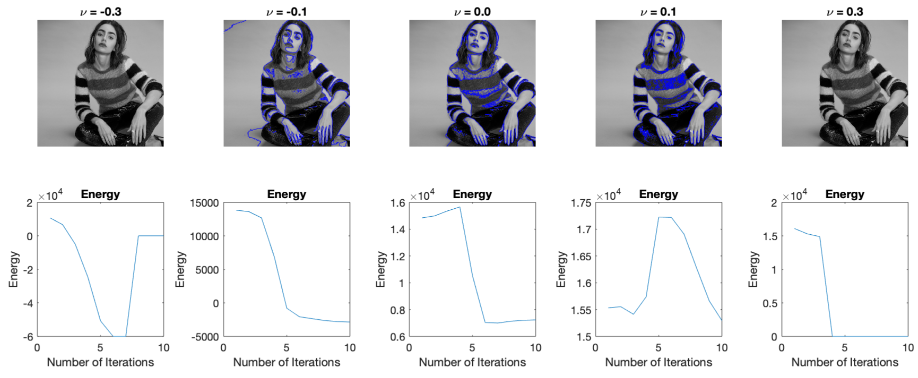


Figure 9: Chan-Vese algorithm with different ν on "Lily"

4.5 Effect of Varying Number of Initialization Curves

For above experiments, we only used a simple circular level-set for initialization, and in this section, we tested effects of different number of curves used for initialization as shown in Figure 10. Although the final minimal tends to be the same after several iterations, the number of curves used for initialization has a great impact on the number of iterations needed to derive the contour.

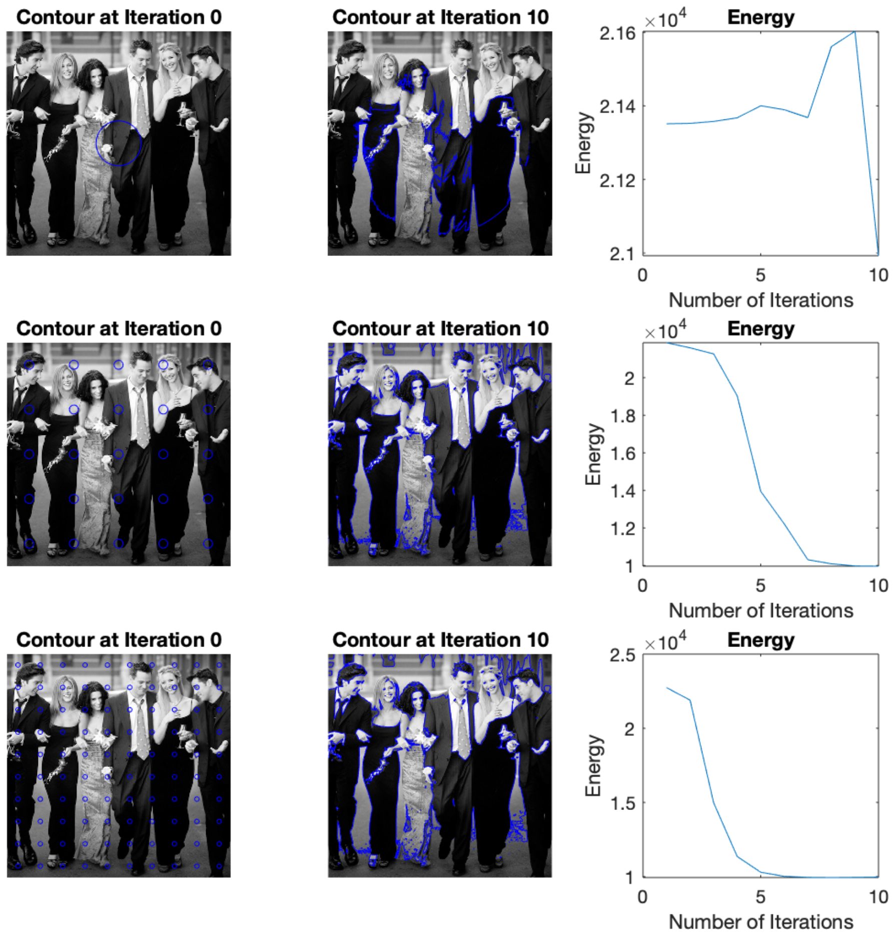


Figure 10: Chan-Vese algorithm with different initialization circles on ”Friends”

5 Conclusions

5.1 Strengths and Weaknesses

Overall, the Chan-Vese algorithm seems to be a value tool in the area of image segmentation. It can quickly find the contour and it relies on global properties like intensities and region areas, rather than just taking into account local properties, such as gradients. However, this makes it difficult for the Chan-Vese algorithm to deal with noisy pictures. And, if we take the length of the contour and the region enclosed by the contour into consideration, these penalty or reward may have negative effects on the results which is not what we want in the first place.

5.2 Potential Improvements

Many implementations of the Chan-Vese algorithm including a reinitialization step since the Chan–Vese model generally has multiple local minimizers. By using an initial boundary, specific objects in the image can be segmented. And reinitialization may need to be used to avoid including separate objects and allow it to maintain a single closed curve.

Also, in order to refine the algorithm, some implementations use values that were already computed to decrease the computing time of the next values and reduce the time of computing PDE equations at each loop which my save a lot of time[3].

The Chan-Vese algorithm can also be used to segment color images which was not discussed in this paper.

So if there was more time, we will focus on the three problems above and try to do more experiments to test the performance of this algorithm on different kinds of images.

References

[1] T. F. Chan and L. A. Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10(2):266–277, 2001.

[2] David Mumford. Optimal approximation by piecewise smooth functions and associated variational problems. *Commun. Pure Applied Mathematics*, pages 577–685, 1989.

[3] Zygmunt L Szpak and Jules R Tapamo. Further optimizations for the chan-vese active contour model. In *High Performance Computing and Simulation Conference*, pages 272–280. Citeseer, 2008.