PDEs Image Proc &Vision Research Report

Xiyu Ouyang

June 3, 2018

1 Introduction

Active contour models (ACM) is one of the most popular methods in object detecting and image segmentation due to its two major advantages: first, it can achieve better accuracy of segmentation and second, it can be easily formulated as a minimization energy problem. In general, active contours models can be classified into two types: geometric active contours and parametric active contours. Geometric active contours, which is based on curve revolution and level sets, is using geometric criteria to evaluate the curves. On the other hand, in parametric active contour, the initial curve will change the shape of itself to fit the true boundary of the detected object under the influence of the internal and external energy. The paper will explain in details one particular parametric active contour called snake, and implement snake algorithm for detecting bound of object. After that several corresponding simulations, comparisons and discussions are followed.

2 Active Contours Methods

2.1 Snake Algorithm

Snake Algorithm is a energy optimization based curve evolution model first introduced by Kass et al. In Kass et al's first efforts on active contours, he defined the curves as an image domain that can move under the influence of the internal forces as well as external forces, where internal forces come from the curve itself and the external forces come from the image data. The internal splines forces impose a piecewise smoothness constraint on the moving curve while the external constraints forces push the splines (by minimizing the energy)towards features such as lines and edges. Then the problem is converted to minimming the energy function:

$$E_{snake} = \int_0^1 E_{int}(C(s)) + E_{ext}(C(s))\ ds \tag{1}$$

where

$$C(s) = (x(s), y(s))$$

is the parametric representation of the curve with parameter $s \in (0, 1)$. More often, the snake algorithm is used to segment an object, so it is often used in a closed contour, ie, $s(0) = s(1)$. The total energy is composed of two components: Internal energy and external energy. A common external energy is the inverse or opposite of the gradient magnitude: this is because the external energy function usually stands for a potential attraction field onto the edge of the object, and due to the fact that we would like to minimize the energy, we need to take the opposite value: lower energies near the edges, higher energies anywhere else. External energy is usually defined as:

$$E_{ext} = -\gamma(s) \int_a^b ||\nabla I||^2 (C(s))\ ds \tag{2}$$

where a and b are two distinct points on the contour (if it is a closed contour then a = b), γ can be both constant or non - constant. Later in our implementation, we will apply a Gaussian filter to the edge to represent norm of the gradient of the filtered image, which produces a more smoothed gradient around the edge. Hereafter, the paper uses to $||\nabla I||$, to represent norm of the gradient of the filtered image, $||\nabla(G * I)||$, where G stands for Gaussian filter.

For the internal energy, it is affected by smoothness and curvature. So it is composed by two terms. The first one is a function regrading smoothness or elasticity so it should be a function of the first derivative of our contour. it is defined as

$$E_{int} = \frac{1}{2}(\alpha(s)|C_s(s)|^2 + \beta(s)|C_{ss}(s)|^2) \tag{3}$$

where α and β can be both constant or non - constant. The first term is magnitude of the first derivative, which is larger for longer snakes, and the second term is the magnitude of the second derivative, which is larger for sharper bends. By minimizing the energy function, the original curve got shortened and smoothed. For instance, if we minimize the energy function of a curve which has fixed end points, we will eventually get a straight line.

2.2 Euler Lagrange Minimization

To minimize the following energy function:

$$E_{snake} = \frac{1}{2}(\alpha(s)|C_s(s)|^2 + \beta(s)|C_{ss}(s)|^2) - \gamma(s) \int_a^b ||\nabla I||^2(C(s))\ ds \tag{4}$$

we need to use Euler - Lagrange equations. if α and β are constants, by using Euler - Lagrange, we need to solve:

$$L_s - \frac{\partial C_s}{\partial s} + \frac{\partial C_{ss}}{\partial s^2} = 0$$

where

$$L_s = \gamma(s)\nabla(||\nabla I||^2)(C(s))$$
$$\frac{\partial C_s}{\partial s} = \frac{1}{2} * 2 * \alpha(s)C_{ss}(s)$$
$$\frac{\partial C_{ss}}{\partial s^2} = \frac{1}{2} * 2 * \beta(s)C_{ssss}(s)$$

Then we convert the Lagrange into a time dependent partial differential system where the contour function C becomes a time dependent function $C(s,t)$. Solving it gives

$$\alpha(s)C_{ss}(s) - \beta(s)C_{ssss}(s) + \gamma(s)\nabla(||\nabla I||^2)(C(s)) = C_t(s,t) \tag{5}$$

If we split the vector C into its two components x and y, we will get the following two independent equations:

$$\alpha(s)x_{ss}(s) - \beta(s)x_{ssss}(s) + \gamma(s)\nabla(||\nabla I||^2)(x(s)) = x_t(s,t) \tag{6}$$
$$\alpha(s)y_{ss}(s) - \beta(s)y_{ssss}(s) + \gamma(s)\nabla(||\nabla I||^2)(y(s)) = y_t(s,t) \tag{7}$$

If α and β are not constants, we can use the discrete formulas of the energy function, which can be written as:

$$E_{snake} = \sum_{i=1}^{n} E_{int}(i) + E_{ext}(i)$$

Similarily, minimize the above energy function we get the following Euler Equations in matrix form:

$$Ax + f_x(x,y) = 0$$
$$Ay + f_y(x,y) = 0$$

For the purpose of this paper, we won't discuss the situation where α and β are not constants.

2.3 Spatial Discretization

Next, we use gradient descent method to solve the above continuous minimization problem by converting the snake s into a function of time (After a long time when the snake has converged to a minimum, the derivative with respect to time will be zero, so the equation is satisfied):

$$C_t(s,t) = \alpha(s)C_{ss}(s,t) - \beta(s)C_{ssss}(s,t) + \gamma(s)\nabla(||\nabla I||^2)(C(s)) \tag{8}$$

If the parameter C is discretized, then we can implement the algorithm numerically. Recall the Taylor series approach of finite differencing:

$$f(x+2\Delta x) = f(x) + 2\Delta x f'(x) + 2\Delta x^2 f''(x) + \frac{4}{3}\Delta x^3 f'''(x) + \frac{2}{3}\Delta x^4 f^{(4)}(x) + O(\Delta x^5)$$

$$f(x-2\Delta x) = f(x) - 2\Delta x f'(x) + 2\Delta x^2 f''(x) - \frac{4}{3}\Delta x^3 f'''(x) + \frac{2}{3}\Delta x^4 f^{(4)}(x) + O(\Delta x^5)$$

$$f(x+\Delta x) = f(x) + \Delta x f'(x) + \frac{1}{2}\Delta x^2 f''(x) + \frac{1}{6}\Delta x^3 f'''(x) + \frac{1}{24}\Delta x^4 f^{(4)} + O(\Delta x^5)$$

$$f(x-\Delta x) = f(x) - \Delta x f'(x) + \frac{1}{2}\Delta x^2 f''(x) - \frac{1}{6}\Delta x^3 f'''(x) + \frac{1}{24}\Delta x^4 f^{(4)} + O(\Delta x^5)$$

From above equations, by using second order forward time central difference (FTCS), we can calculate the second derivative and the forth derivative:

$$f''(x) = \frac{f(x+\Delta x) - 2f(x) + f(x-\Delta x)}{\Delta x^2}$$

$$f^4(x) = \frac{f(x+2\Delta x) - 4f(x+\Delta x) + 6f(x) - 4f(x-\Delta x) + f(x-2\Delta x)}{\Delta x^4}$$

Similar idea, after discretization, we will have the following for x:

$$x_{ss}(s,t) = \frac{x(s+\Delta s,t) - 2x(s,t) + x(s-\Delta s,t)}{\Delta s^2} \tag{9}$$

$$x_{ssss}(s,t) = \frac{x(s+2\Delta s,t) - 4x(s+\Delta s,t) + 6x(s,t) - 4x(s-\Delta s,t) + x(s-2\Delta s,t)}{\Delta s^4} \tag{10}$$

And for y:

$$y_{ss}(s,t) = \frac{y(s+\Delta s,t) - 2y(s,t) + y(s-\Delta s,t)}{\Delta s^2} \tag{11}$$

$$y_{ssss}(s,t) = \frac{y(s+2\Delta s,t) - 4y(s+\Delta s,t) + 6y(s,t) - 4y(s-\Delta s,t) + y(s-2\Delta s,t)}{\Delta s^4} \tag{12}$$

The last term

$$\gamma(s)\nabla(||\nabla I||^2)(C(s,t)) \tag{13}$$

is just the spatial derivative of the gradient of the input image.

2.4 Temporal Discretization

We also need to compute the temporal derivative of $x_t(s,t)$ and $y_t(s,t)$. Since the snake is evolving through the time, it is reasonable to use two adjacent time spots (for instance, the difference between the snake when it was at 10 second and 9 second) in forward direction. i.e, forward time differences. If we apply this forward Euler finite differnece on x and y function we will obtain:

$$x_t(s,t) = \frac{x(s,t+\Delta t) - x(s,t)}{\Delta t} \tag{14}$$

$$y_t(s,t) = \frac{y(s,t+\Delta t) - y(s,t)}{\Delta t} \tag{15}$$

2.5 CFL Condition

In this section ,we will apply DFT-based Von Neumann analysis to determine the stability of the scheme and the necessary CFL condition. Due to the limited knowledge and the purpose of this paper, we only look at snake energy without the last term (i.e, the external energy term). we have

$$\begin{aligned}\frac{x(s,t+\Delta t)-x(s,t)}{\Delta t} &= \alpha * \frac{x(s+\Delta s,t)-2x(s,t)+x(s-\Delta s,t)}{\Delta s^2} \\ &- \beta * \frac{x(s+2\Delta s,t)-4x(s+\Delta s,t)+6x(s,t)}{\Delta s^4} \\ &- \beta * \frac{-4x(s-\Delta s,t)+x(s-2\Delta s,t)}{\Delta s^4}\end{aligned} \tag{16}$$

Applying DFT to both sides, we transform the variables from space domains to frequency domains ω. Then the update yields:

$$\begin{aligned}X(\omega,t+\Delta t)-X(\omega,t) &= \frac{\alpha\Delta t}{\Delta s^2}\big(X(\omega+\Delta\omega,t)-2X(\omega,t)+X(\omega-\Delta\omega,t)\big) \\ &- \frac{\beta\Delta t}{\Delta s^4}\big(X(\omega+2\Delta\omega,t)-4X(\omega+\Delta\omega,t)+6X(\omega,t)\big) \\ &- \frac{\beta\Delta t}{\Delta s^4}\big(4X(\omega-\Delta\omega,t)+X(\omega-2\Delta s,t)\big)\end{aligned} \tag{17}$$

Recall that in frequency domain,

$$\begin{aligned}X(\omega+\Delta\omega,t) &= X(\omega,t)e^{i\omega\Delta s} \\ X(\omega-\Delta\omega,t) &= X(\omega,t)e^{-i\omega\Delta s}\end{aligned}$$

plug above equations into (17), then we will have:

$$\begin{aligned}X(\omega,t+\Delta t)-X(\omega,t) &= \frac{\alpha\Delta t}{\Delta s^2}X(\omega,t)(e^{i\omega\Delta s}+e^{-i\omega\Delta s})-\frac{2\alpha\Delta t}{\Delta s^2}X(\omega,t) \\ &- \frac{\beta\Delta t}{\Delta s^4}X(\omega,t)(e^{i2\omega\Delta s}+e^{-i2\omega\Delta s})+\frac{4\beta\Delta t}{\Delta s^4}X(\omega,t)(e^{i\omega\Delta s}+e^{-i\omega\Delta s}) \\ &- \frac{6\beta\Delta t}{\Delta s^4}X(\omega,t)\end{aligned} \tag{18}$$

In the end, apply Euler's formula we get

$$\frac{X(\omega, t+\Delta t)}{X(\omega, t)} = 1 + \frac{2\alpha\Delta t}{\Delta s^2}(cos(\omega\Delta s) - 1) - \frac{2\beta\Delta t}{\Delta s^4}(cos(2\omega\Delta s) - 4cos(2\omega\Delta s) + 3) \quad (19)$$

For the system to be stable, we need

$$|\frac{X(\omega, t+\Delta t)}{X(\omega, t)}| \leq 1$$

Let $u = \frac{\alpha\Delta t}{\Delta s^2}$, and $v = \frac{\beta\Delta t}{\Delta s^4}$, then equation (19) becomes:

$$\frac{X(\omega, t+\Delta t)}{X(\omega, t)} = 1 + 2u(cos(\omega\Delta s)) - 2u - 6v - 2v(cos(2\omega\Delta s)) + 8v(cos(2\omega\Delta s)) \quad (20)$$

Rearrange (20) we have the following:

$$|1 - 2u - 6v + 2u(cos(\omega\Delta s)) + 6v(cos(2\omega\Delta s))| \leq 1 \quad (21)$$

Rearrange the equation we have:

$$|1 - 2u(1 - cos(\omega\Delta s)) - 6v(1 - cos(\omega 2\Delta s))| \leq 1 \quad (22)$$

or,

$$0 \leq 2u(1 - cos(\omega\Delta s)) + 6v(1 - cos(\omega 2\Delta s)) \leq 2 \quad (23)$$

we know that $(1-cos(\omega\Delta s)) \in (0,2)$ and $(1-cos(\omega 2\Delta s)) \in (0,2)$. So (23) is always greater than zero. The worst case of RHS is when $(1 - cos(\omega\Delta s))$ and $(1 - cos(\omega 2\Delta s))$ are all equal to 2. Plug into (23) we have

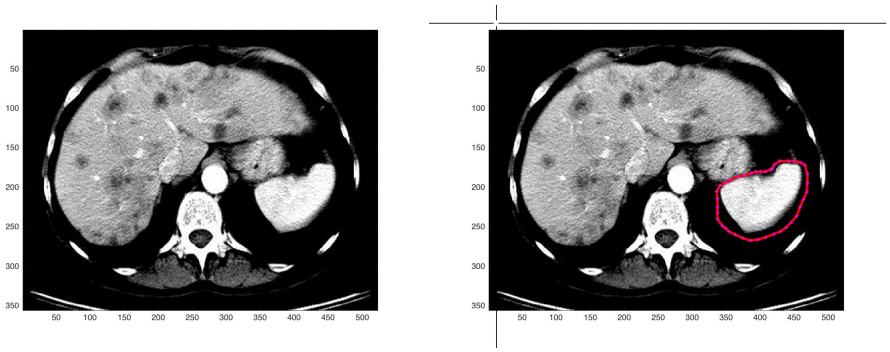
$$2u + 6v \leq 1 \quad (24)$$

So the CFL condition is :

$$\frac{2\alpha\Delta t}{\Delta s^2} + \frac{6\beta\Delta t}{\Delta s^4} \leq 1 \quad (25)$$

3 Results

This section provides experimental results, followed by some discussions on the corresponding observations. For the purpose of showing the results, we use a CT scan image of human lung as the input data. To obtain the result, we first ask user to select the initial points, so that the desired part will be segmented by closed contour:



(a) Original CT Image (b) Image After Segmentation

Figure 1: User Segments the Selected Object

Then we apply the Gaussian filter to the original gray scale CT scan image. After that, we solve the PDE for the evolution of the contour, and we stop when the energy has reached to a local minimum. Figure 2 show a sequence of image results if we run snake algorithm. We can see from Figure 2, the initial points converge to the edge of the selected piece, and eventually match the edge, which is exactly what we want. But we also notice that in some parts, the contour doesn't appear really smooth even after applying Gaussian filter. This is due to the value of α, β and γ. Different values of α, β and γ determine the shape of the contour, and we will discuss their effect in the following sections.

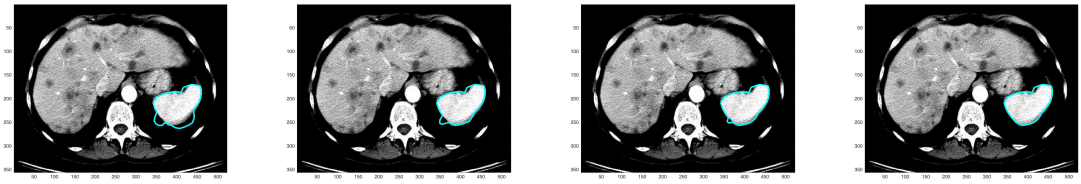
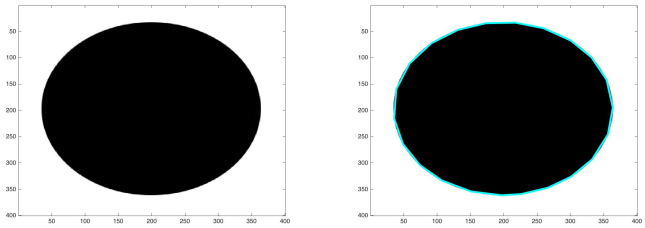


Figure 2: Evolution Of the Contour

3.1 Effect of constant α

In order to see the effect of α, we set β to be very small (In this case, 0.001, almost equal to zero) so that the results would be more clear. For the simplicity of illustration, we will use an image of a circle to demonstrate our points. In the first attempt, we set α equals to 3 and in the second attempt, we set α equals to 0.05. Other than that, every other parameters are the same, including the number of selection points which is set to 21. Figure 3 shows the original image and the final converged contour. In both case, the final converged contour is approximately the same, and in both case we get desired results. But when we look at the data of the time for the snake to converge in Table 1, we can see that in both case, starting time is the same, starting lengths and ending length are very close, but different α brings different converge time. For larger α ($\alpha = 3$), the converging time is much more shorter compared to the smaller α ($\alpha = 0.05$). This is because α controls the internal energy function's sensitivity to the amount of stretch in the snake. It penalizes changes in distances between points in the contour. The larger the α, the more sensitive of the function, the quicker it converges.



(a) Original Image (b) Converged Contour

Figure 3: Evolution Of the Contour

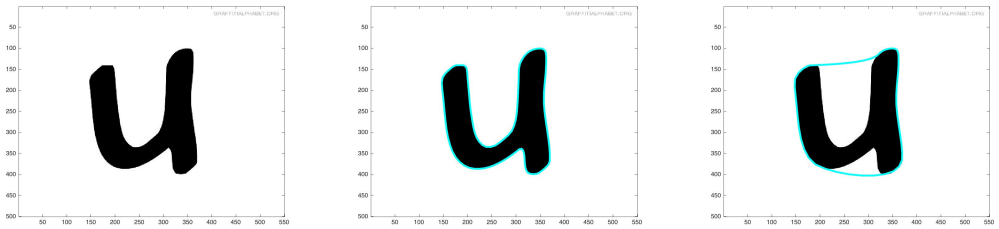
Table 1: Snake Converge Time

Case	Starting Time#1	Ending Time#2	Starting Length#3	Ending Length#4
1	1.000e-04	0.0032	1.1857e+03	985.8674
2	1.000e-04	0.0311	1.1951e+03	985.7681

Starting time	Ending time	Starting length	Ending length
Time when the initial contour starts to evolve	Time when the contour stops to evolve	The overall length of all user selected segments when the contour starts to evolve	Then overall length of all user selected segments when the contour is completely converged

3.2 Effect of constant β

In this section, we discuss the effect of the parameter β. Figure 4 represents a sequences of images showing that the snake evolution with a relatively high value of β (0.005), and a relatively low value of β (0.001). As we can tell, the final contour of $\beta = 0.005$ shows that the snake has not been able to converge to the edge as well as when we set $\beta = 0.001$. For images which have convex shape, large value of beta works just fine. But if the input image has concave shape, or at least concave corner, then large value of beta won't fit the edge very precisely. This behavior is due to the fact that the value of β controls the smoothness of the snake, and doesn't bend around the corner unless the external energy is relatively large enough to govern the evolution. For the purpose of illustration, we will use an image of "u" since it has sharp concave corner.



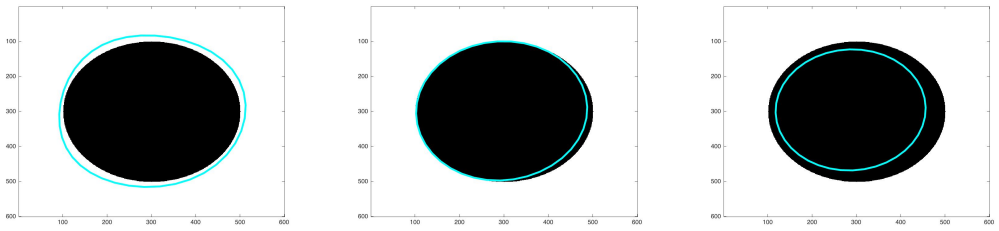
(a) Original Image (b) $\beta = 0.001$ (c) $\beta = 0.005$

Figure 4: Comparison Of the Final Snake Contour With Different β Value

3.3 Effect of constant γ

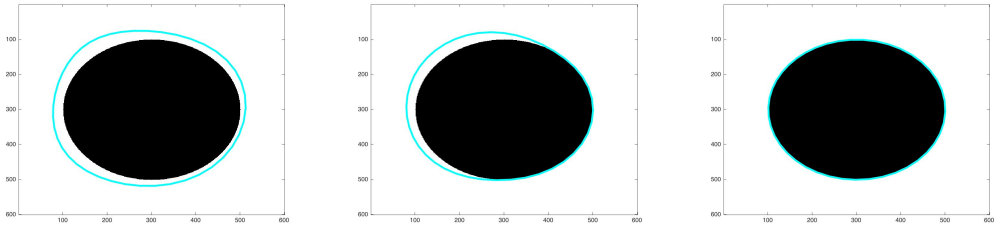
In this section, we experiment with different constants γ. As we can see when γ is equal to 100000, the gradient field of the snakes can't compete with the potential attraction field onto the edge of an object, thus the snake will finally pass the edge of the object, shrink into a dot and disappear. But if γ is ten times larger, then the gradient field, or

the absolute value of the local minimum is large enough to up against the attraction field so that we can get a clear edge, and after that snake will be stabilized. For the purpose of illustration, we just use an image of a circle here. The results are shown in Figure 5, 6.



(a) Snake approaches the edge (b) Snake hits the edge (c) Snake passes the edge

Figure 5: Evolution Of the Snake When $\gamma = 100000$



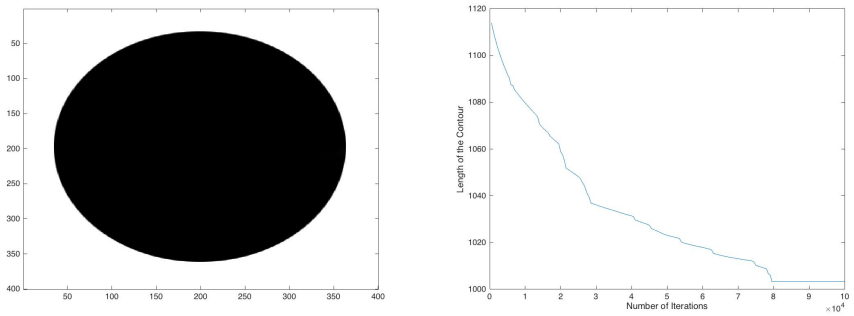
(a) Snake approaches the edge (b) Snake hits the edge (c) Snake is stablized

Figure 6: Evolution Of the Snake When $\gamma = 1000000$

3.4 Analytic analysis

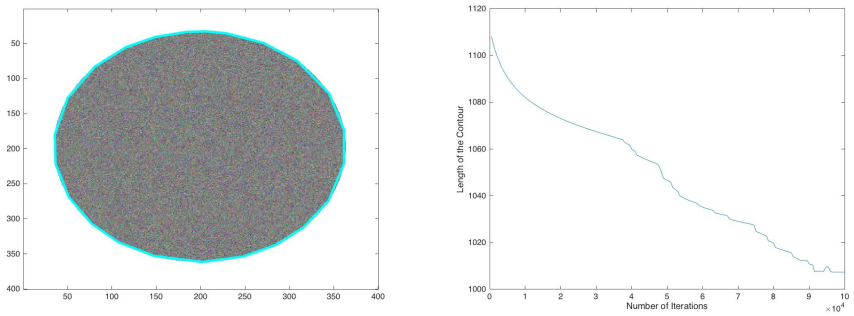
In this section, we explore the effects of noise on the image and the corresponding effects on snake algorithm. For the simplicity of illustration, we use an image of a circle to represent our results. First, we use an un-processed image and plot the correlation between the length of the contour and the number of iterations. As we can see from Figure 7, when iteration is around $8 * 10^4$, the contour stopped evolving. Next, we add noise to the original image. In figure 8, the original image was added Gaussian noise with density equals to 0.5. We can see from the plot, for the original image without noise, it needs longer time for the contour to converge. But when we apply gaussian noise, number of iterations increases from $8 * 10^4$ to $9 * 10^4$, which make sense since adding noise increase the difficulty of getting converged to the boundary, thus it need more iterations to achieve this. To analytically analyze the impact that noise can potentially has, we could also use segmentation error.

The segmentation error is estimated in terms of pixel difference. The comparison of two different binary images can be made by pixel XOR operation. For the purpose of the paper we won't discuss this topic in depth here.



(a) Original Image (b) Contour length VS Converging time

Figure 7: Length - Time plot without Noise



(a) Gaussian noise (I = 0.5) (b) Contour length VS Converging time

Figure 8: Length -Time plot with Noise

4 Conclusion

Snake method is without doubt, very powerful and meanwhile intuitively straightforward, but it also has significant downsides – First, the energy function is parameterization dependent, ie, it is not geometric. Different solutions can be obtained by changing the parameterization while keeping the initial curve. Second, the evolving contour has difficulty handling the changes in topology: multiple objects, non-convex objects, can not be

detected effectively. This is demonstrated in section 3.2. Third, the evolving curve has the tendency to get stuck at subtle edges, creating undersized local minimum, which makes the model sensitive to the initialization and noise.This is also demonstrated in section 3.4. If I had more time for the project, i can implement external energy with more details, for instance, includes line functional and termination functional into the energy and see what's happen. In the future, if I still need to solve a segmentation problem, I might try different approaches such as level set based implicit curve evolution approaches, which can overcome some of the major drawbacks of the classic snake algorithm. Moreover, i will think about how to optimize the algorithm so that the run time can be shorter than what I have right now.