Image Segmentation Using Chan-Vese Algorithm

ECE 6560

Partial Differential Equations for Image Processing and Computer Vision

Iyer Vidhya Subramanian

Professor: Anthony Yezzi

April 28 2017

1. Introduction

One of the most important and ubiquitous tasks in image analysis is segmentation. Segmentation is the process of partitioning an image into a set of distinct regions, which are different in some important qualitative or quantitative way. This is a critical intermediate step in all high-level object-recognition tasks. For example, if we want to locate the face of a particular person in an image of a crowd, we must first determine which parts of the image correspond to human faces. Other common segmentation tasks include segmenting written or typeset characters on a page, segmenting tumors from healthy brain tissue in an MRI image, etc. [2]

There are a wide variety of approaches to the segmentation problem. One of the popular approaches is active contour models or snakes which was first introduced by Terzopoulos. [5] The basic idea is to start with a curve around the object that is to be detected, the curve moves towards its inward normal and has to stop on the boundary of that object. [3] However, all the active contour models are depending on the gradient of the given image to stop the evolution of the curve. Therefore, these models can only detect objects with edges defined by a gradient.

Based on the Mumford-Shah functional for segmentation [6], Chan and Vese [1] proposed a new level set model for active contours to detect objects whose boundaries are not necessarily defined by a gradient. This is a powerful and flexible method which can segment many types of images, including some that would be quite difficult using thresholding or gradient based methods. In this paper, we will discuss and analyze the Chan-Vese model, explore the quantitative and qualitative analysis and discuss the weaknesses found in this model.

2. The Chan-Vese Model

Energy Functional:

Let I be the image, C be the contour, R be the region inside C and RC be the region outside of C. u denotes the average intensity inside C and v is the average intensity outside C.

The energy functional used in the Chan-Vese model is given as follows:

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

where λ_1, $\lambda_2 > 0$, $\mu \geq 0$, and $\nu \geq 0$ are fixed scaling parameters.

The object of the Chan Vese algorithm is to minimize this energy functional $E(C,u,v)$ both inside and outside the contour. We are looking for each term of the energy functional C,u,v in this equation to minimize. Looking at the equation, the first two terms $\lambda_1 \iint_{\mathrm{R}}(I-u)^2 dxdy + \lambda_2 \iint_{\mathrm{RC}}(I-v)^2 dxdy$ minimize the energy on both sides of the contour keeping the regions uniform. The third term $\mu \int_C ds$ is for penalizing the length of the contour, s being the arc length. The last term $\nu \iint_{\mathrm{R}} dxdy$ (where $dxdy$ is dA) penalizes the area of the contour.

Gradient Descent and Level Set Functions :

We now derive the gradient descent of the contour to learn how it changes with each time step.

The gradient descent is given as:

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

The time derivative of the energy functional $\frac{d}{dt}E(C)$ can be given as the inner product of ∇E and C_t which again can be defined as the L_2 continuous norm defined over the contour.

$$\begin{aligned} \frac{d}{dt}E(C) &= \langle C_t, \nabla E \rangle \\ \frac{d}{dt}E(C) &= \int_C (C_t \cdot \nabla E) ds \end{aligned} \tag{3}$$

Let f_{in} be a non-linear function integrated over the region R inside the contour where $f_{in} = \nabla \cdot \vec{F}$:

$$\hat{E}(C) = \iint_{\mathrm{R}} f_{in} dxdy \tag{4}$$

Applying the Divergence Theorem, we get:

$$\hat{E}(C) = \iint_{\mathrm{R}} f_{in} dxdy = \iint_{\mathrm{R}} (\nabla \cdot \vec{F}) dxdy = \int_C \vec{F} \cdot N ds \tag{5}$$

Where N is the normal component of the contour. We now need to replace the time-dependent variable s with the time-independent variable p so we can move the derivative inside the integral. The relationship between s and p can be given as $\|C_p\| dp = ds$.

$$\hat{E}(C) = \int_C \vec{F} \cdot N ds = \int_0^1 \vec{F} \cdot N \|C_p\| \, dp \tag{6}$$

Now, as we removed the time-dependent variables, we can take the derivative inside the integral and take the time derivative of the integrand:

$$\frac{d}{dt}\hat{E}(C) = \frac{d}{dt}\int_0^1 \left(\vec{F} \cdot N \|C_p\|\right) dp$$

$$= \int_0^1 \frac{d}{dt}\left(\vec{F} \cdot J \frac{C_p}{\|C_p\|}\|C_p\|\right) dp$$

Using the relationship $N = JT$ where $T = \frac{C_p}{\|C_p\|}$ and J is a constant matrix [0 1; -1 0] and T is the tangential component, we simplify as:

$$\frac{d}{dt}\hat{E}(C) = \int_0^1 \frac{d}{dt}\left(\vec{F} \cdot JC_p\right) \tag{7}$$

Now we take the derivative of $(\vec{F} \cdot JC_p)$ with respect to time:

$$\frac{d}{dt}\hat{E}(C) = \int_0^1 \left(\frac{d\vec{F}}{dt} \cdot JC_p + \vec{F} \cdot JC_{pt}\right) dp$$

We then change the order of derivation for C_{pt} to C_{tp} instead

$$\frac{d}{dt}\hat{E}(C) = \int_0^1 \left(\frac{d\vec{F}}{dt} \cdot JC_p + \vec{F} \cdot JC_{tp}\right) dp$$

And by substituting the Jacobian $\left[\frac{\partial \vec{F}}{\partial X}\right] C_t$ for $\frac{d\vec{F}}{dt}$:

$$\frac{d}{dt}\hat{E}(C) = \int_0^1 \left(\left[\frac{\partial \vec{F}}{\partial X}\right] C_t \cdot JC_p + \vec{F} \cdot JC_{tp}\right) dp \tag{8}$$

We can now switch back to our time dependent variable s using these relationships $\frac{d}{dp} = \frac{d}{ds}\|C_p\|$ and $C_p = C_s\|C_p\|$ and $C_{tp} = C_{ts}\|C_p\|$:

$$\frac{d}{dt}\hat{E}(C) = \int_0^1 \left(\left[\frac{\partial \vec{F}}{\partial X}\right] C_t \cdot JC_s\|C_p\| + \vec{F} \cdot JC_{ts}\|C_p\|\right) dp$$

$$= \int_0^1 \left(\left[\frac{\partial \vec{F}}{\partial X}\right] C_t \cdot JC_s + \vec{F} \cdot JC_{ts}\right) ds \tag{9}$$

We now apply integration by parts to the second term to isolate C_t :

$$\frac{d}{dt}\hat{E}(C) = \int_0^1 \left(\left[\frac{\partial \vec{F}}{\partial X}\right] C_t \cdot JC_s - \left[\frac{\partial \vec{F}}{\partial X}\right] C_s \cdot JC_t \right) ds$$

$$= \int_0^1 C_t \cdot \left(\left[\frac{\partial \vec{F}}{\partial X}\right]^T JC_s + J\left[\frac{\partial \vec{F}}{\partial X}\right] C_s \right) ds \tag{10}$$

The above equation is of the form $\frac{d}{dt}E(C) = \int_C (C_t \cdot \nabla E) ds$ and hence we can extract $\nabla \hat{E}$ from this as:

$$\nabla \hat{E} = \left[\frac{\partial \vec{F}}{\partial X}\right]^T JC_s + J\left[\frac{\partial \vec{F}}{\partial X}\right] C_s$$

As N = JC_s and T = C_s, we substitute them in the $\nabla \hat{E}$ functional:

$$\nabla \hat{E} = \left[\frac{\partial \vec{F}}{\partial X}\right]^T N + J\left[\frac{\partial \vec{F}}{\partial X}\right] T \tag{11}$$

The tangential component in the above equation is assumed to be cancelled out after expanding as the energy functional is a geometric function. So now this can be further simplified as:

$$\nabla \hat{E} = \left(T^T \left[\frac{\partial \vec{F}}{\partial X}\right]^T N \right) T + \left(N^T \left[\frac{\partial \vec{F}}{\partial X}\right]^T N \right) N + \left(T^T J \left[\frac{\partial \vec{F}}{\partial X}\right] T \right) T + \left(N^T J \left[\frac{\partial \vec{F}}{\partial X}\right] T \right) N$$

$$= \left(T^T \left[\frac{\partial \vec{F}}{\partial X}\right]^T N \right) T + \left(N^T \left[\frac{\partial \vec{F}}{\partial X}\right]^T N \right) N - \left(N^T \left[\frac{\partial \vec{F}}{\partial X}\right] T \right) T + \left(T^T \left[\frac{\partial \vec{F}}{\partial X}\right] T \right) N$$

$$= \left(N^T \left[\frac{\partial \vec{F}}{\partial X}\right]^T N \right) N + \left(T^T \left[\frac{\partial \vec{F}}{\partial X}\right] T \right) N$$

$$= \left(N^T \left[\frac{\partial \vec{F}}{\partial X}\right] N \right) N + \left(T^T \left[\frac{\partial \vec{F}}{\partial X}\right] T \right) N \tag{12}$$

The above equation is of the form $a^T Aa + b^T Ab$ (Matrix Equation), where a and b form an orthonormal basis and $A = \left[\frac{\partial \vec{F}}{\partial X}\right]$. Here in (12) similarly both N and T form an orthonormal basis, and A is diagonalizable. Thus, it can be shown as $a^T Aa + b^T Ab = trace(A)$.

$$\nabla \hat{E} = trace\left(\left[\frac{\partial \vec{F}}{\partial X}\right]\right) N$$

$$= trace\left(\begin{bmatrix} \frac{\partial \overrightarrow{F_1}}{\partial x} & \frac{\partial \vec{F}_1}{\partial y} \\ \frac{\partial \overrightarrow{F_2}}{\partial x} & \frac{\partial \overrightarrow{F_2}}{\partial y} \end{bmatrix}\right) N$$

$$= \left(\nabla \cdot \vec{F}\right) N = f_{in} N \tag{13}$$

Hence now C_t can be obtained:

$$C_t = -\nabla \hat{E} = -f_{in} N \tag{14}$$

Now we can obtain the level set equation as:

$$\Psi_t = f_{in} \|\nabla \Psi\| \tag{15}$$

This equation involves just involves the function being integrated over the area inside the contour. We have an additional term that involves the integrating over the area outside the contour. We can say that the integration outside of the contour is nothing but the difference between the integration over the entire image and the integration over the region inside the contour:

$$\iint_{R^C} f_{out} dxdy = \iint_{\Omega} f_{out} dxdy - \iint_{\mathrm{R}} f_{out} dxdy \tag{16}$$

Note that the integration over the region of the entire image has no time dependence and hence we need to take just the second term into consideration:

$$\Psi_t = -f_{out} \|\nabla \Psi\| \tag{17}$$

Now we apply the same method over the final term left of the energy functional in (1) i.e. the part of the function that defines the length of the contour (By switching the time dependent variables with the time independent ones and then moving the derivative inside the integrand then applying integration by parts):

$$\frac{d}{dt} \hat{E}(C) = \frac{d}{dt} \int_C ds$$

$$= \frac{d}{dt}\int_0^1 \|C_p\|\, dp$$

$$= \int_0^1 \|C_p\|_t \tag{18}$$

Now, we can use substitute $\|C_p\|_t = \left(\sqrt{C_p \cdot C_p}\right)_t = \frac{C_{pt} \cdot C_p}{\|C_p\|}$:

$$\frac{d}{dt}\hat{E}(C) = \int_0^1 \left(\frac{C_{pt} \cdot C_p}{\|C_p\|}\right) dp$$

$$= \int_0^1 (C_{pt} \cdot T) dp$$

$$= \int_0^1 (C_{tp} \cdot T) dp \tag{19}$$

Switching back to time dependent variables s and then applying integration by parts:

$$\frac{d}{dt}\hat{E}(C) = -\int_0^1 (C_t \cdot T_p) dp$$

$$= -\int_0^1 (C_t \cdot T_s \|C_p\|)\frac{ds}{\|C_p\|} - \int_0^1 (C_t \cdot T_s) ds \tag{20}$$

Isolating C_t, extracting $\nabla\hat{E}$ as before (as it is of the form in (3)), and using the relationship $T_s = kN$, where k is the curvature of the contour:

$$C_t = -\nabla\hat{E} = kN \tag{21}$$

Defining the curvature $k = -\nabla \cdot \left(\frac{\nabla\Psi}{\|\nabla\Psi\|}\right)$, the level-set function is found to be:

$$\Psi_t = -k\|\nabla\Psi\|$$

$$= \nabla \cdot \left(\frac{\nabla\Psi}{\|\nabla\Psi\|}\right)\|\nabla\Psi\| \tag{22}$$

Finally, substituting the level-set implementations derived in (15)

(22) in the energy functional defined in *(1)* i.e. the level-set gradient descent equation for the Chan-Vese algorithm is computed as:

$$\Psi_t = \lambda_1(I-u)^2\|\nabla\Psi\| \ -\lambda_2(I-v)^2\|\nabla\Psi\| + \ \mu\nabla\cdot\left(\frac{\nabla\Psi}{\|\nabla\Psi\|}\right)\|\nabla\Psi\| + \nu\|\nabla\Psi\| \tag{23}$$

3. Discretization and Implementation

This process is usually carried out as a first step toward making them suitable for numerical evaluation and implementation on digital computers and to apply the level-set equation derived to an image. From the equation (23), we can see that there is one diffusion term and three non-linear terms. Hence, we need to derive two discretization schemes.

The first discretization scheme can be applied for the diffusion term:

$$\nabla\cdot\left(\frac{\nabla\Psi}{\|\nabla\Psi\|}\right)\|\nabla\Psi\| = \frac{\Psi_x^2\Psi_{yy} - 2\Psi_x\Psi_y\Psi_{xy} + \Psi_y^2\Psi_{xx}}{\Psi_x^2+\Psi_y^2} \tag{24}$$

Using the central difference, as it captures the information from the border pixels on both side of a given point:

$$\begin{aligned}
\Psi_x(x,y,t) &= \frac{\Psi(\mathrm{x}+\Delta x,y,t) - \Psi(\mathrm{x}-\Delta x,y,t)}{2\Delta x} \\
\Psi_{yy}(x,y,t) &= \frac{\Psi(\mathrm{x},y+\Delta y,t) - 2\Psi(\mathrm{x},y,t) + \Psi(\mathrm{x},y-\Delta y,t)}{\Delta y^2} \\
\Psi_{xy}(x,y,t) &= \frac{\Psi_x(\mathrm{x},y+\Delta y,t) - 2\Psi_x(\mathrm{x},y,t) + \Psi_x(\mathrm{x},y-\Delta y,t)}{\Delta y^2}
\end{aligned} \tag{25}$$

As the values of Δx, Δy, and Δt maintain the the stability of this discretization scheme, we now determine the CFL condition:

$$CFL \text{ for diffusion term: } \Delta t \le \frac{\Delta x^2}{2}, \Delta t \le \frac{\Delta y^2}{2} \tag{26}$$

For the non-linear terms in (23), we cannot use central differences nor forward or backward difference with no modifications. The signs of its coefficient and the signs of the approximation methods used determine the nature of the non-linear terms. The upwind entropy accounts for such difference in the signs.

$$\alpha\|\nabla\Psi\| = \begin{cases} \alpha\sqrt{\max^2(0,\Psi_x^+) + \min^2(0,\Psi_x^-) + \max^2\left(0,\Psi_y^+\right) + \min^2(0,\Psi_y^-)}, & \alpha > 0 \\ \alpha\sqrt{\min^2(0,\Psi_x^+) + \max^2(0,\Psi_x^-) + \min^2\left(0,\Psi_y^+\right) + \max^2(0,\Psi_y^-)}, & \alpha < 0 \end{cases} \tag{27}$$

We can see in the upwind entropy equation that both the backward and forward difference are used in the calculation of the norm of the level-set gradient.

$$\Psi_x^+(x,y,t) = \frac{\Psi(\mathrm{x}+\Delta x,y,t)-\Psi(\mathrm{x},y,t)}{\Delta x}$$
$$\Psi_y^-(x,y,t) = \frac{\Psi(\mathrm{x},y,t)-\Psi(\mathrm{x},y-\Delta y,t)}{\Delta y} \quad (28)$$

The CFL condition of this discretization scheme is:

$$CFL\ for\ non\ linear\ terms: \Delta t \leq \frac{\Delta x^2}{\sqrt{2}}, \Delta t \leq \frac{\Delta y^2}{\sqrt{2}} \quad (29)$$

As there are two discretization schemes here, we have two different CFL conditions that need to be satisfied for the overall stability.

Algorithm used in MATLAB

We now have everything to perform image segmentation. We need to take a note that we select the num_iter value (number of iterations) to be 1 and 5 for noisy images with the same parameters.

The basic algorithm used:

Given an image I, an initial level set function and parameters are initialized

- Compute u and v for the current level set function
- Determine Ψ_t using the discretization method
- Update $\Psi^{n+1} = \Psi^n + \Delta t * \Psi_t$ using Chan Vese iteration
- Reinitialize the level set function and check whether the solution is stationary. If not, continue else stop.

We will now examine the performance of this algorithm when applied to several types of images. The following set of parameters will be taken as a baseline:

λ1 = λ2 = 1

u = 0.5 v = 0

Δt = 0.1 ($\Delta t \leq 0.5$ by using CFL conditions)

The code used for this implementation of the Chan-Vese algorithm is available on this link as well as the images used here for the report:

https://drive.google.com/open?id=0B9AINM5fJmVxdDNBZFNXLVNycWs

3. Experimental Results

Finally, we present some segmentation results from various image types to showcase the utility of this algorithm.

Segmenting A Synthetic Bilevel Image

We start with a trivial example; a smooth, synthetically generated image consisting of only two distinct gray levels. The correct segmentation is immediately obvious in this case, and could be obtained. This image is segmented in a few iterations with the default parameters of our algorithm.

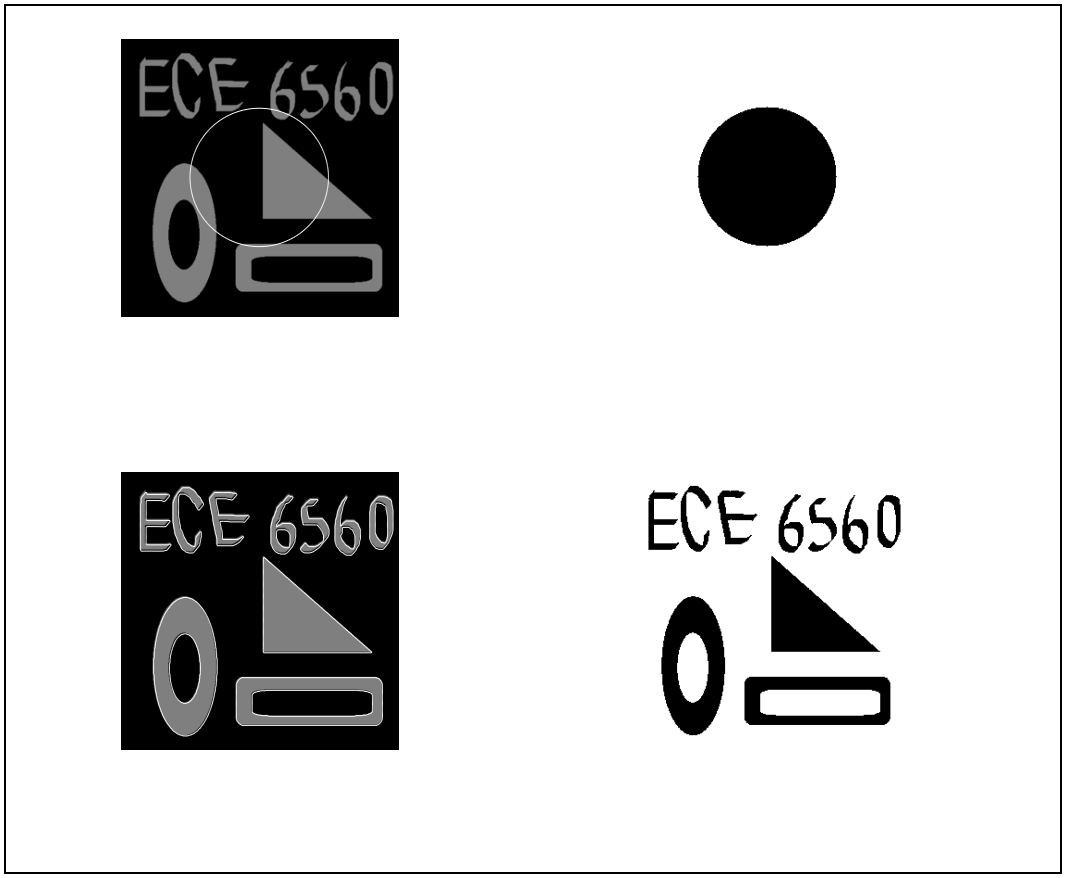


Figure 1. Application of the Chan-Vese Algorithm on a Synthetic Binary Image.

The segmentation is achieved in a few iterations and is nearly perfect. Now let us try by adding some noise to these synthetic binary images.

We can examine this algorithm by understanding the relationship between the energy and the number of iterations. The graph bellow shows how the energy varies for every iteration when applied over the Synthetic Binary Image.

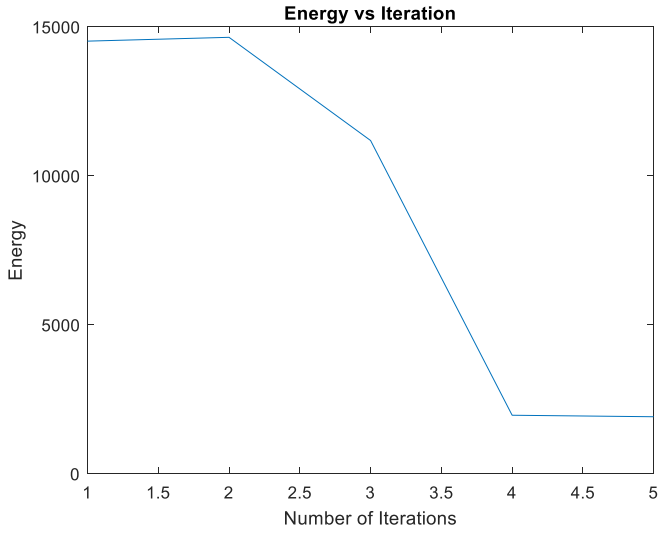
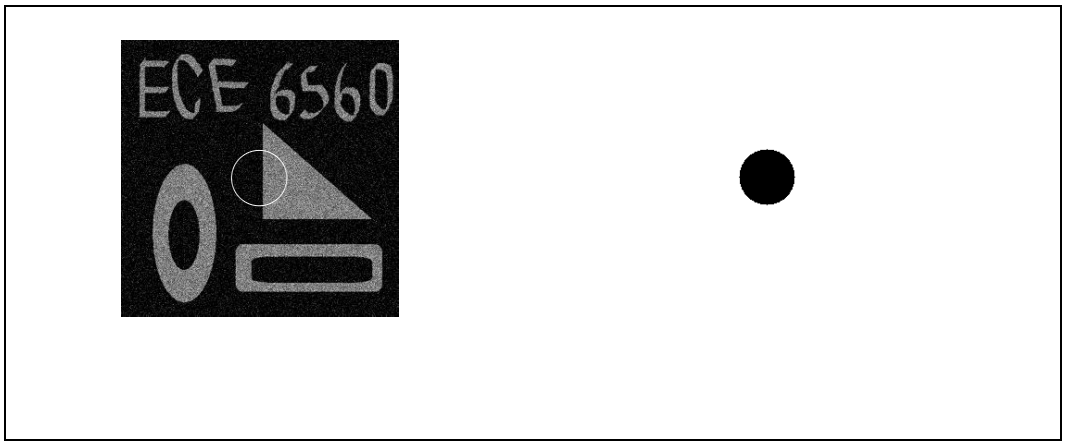


Figure 2. Energy vs the number of iterations.

The plot clearly shows that the steady state is obtained after the 4th iteration where the energy reaches its minimum value. This corresponds to a proper segmentation. The gradient descent usually obtains a steady state once the segmentation is achieved.

A Synthetic Binary Image with Noise:

The previous image was easily segmented by our algorithm. Now we add a significant amount of noise to get an image that is less trivial to segment correctly. Figure below shows the results after it is applied to an image with "Gaussian" noise.



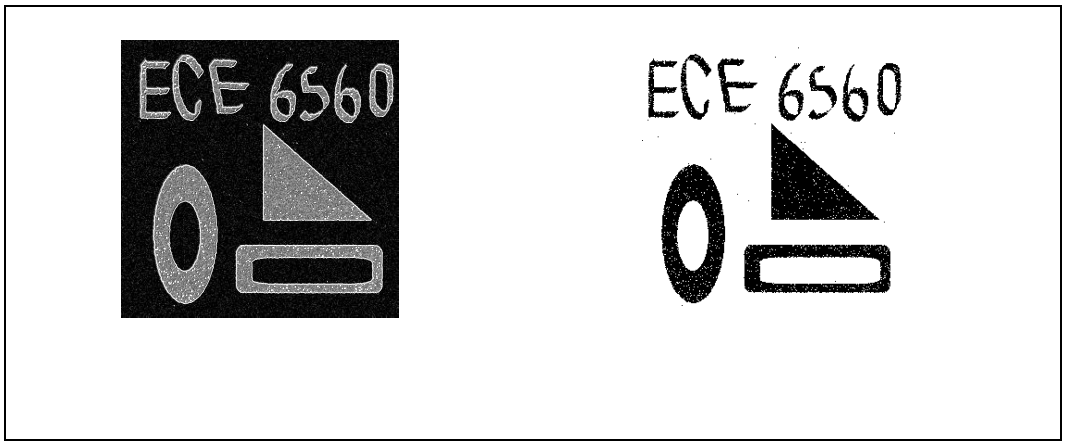


Figure 3. Application of the Chan-Vese Algorithm on a Noisy Synthetic Binary Image.

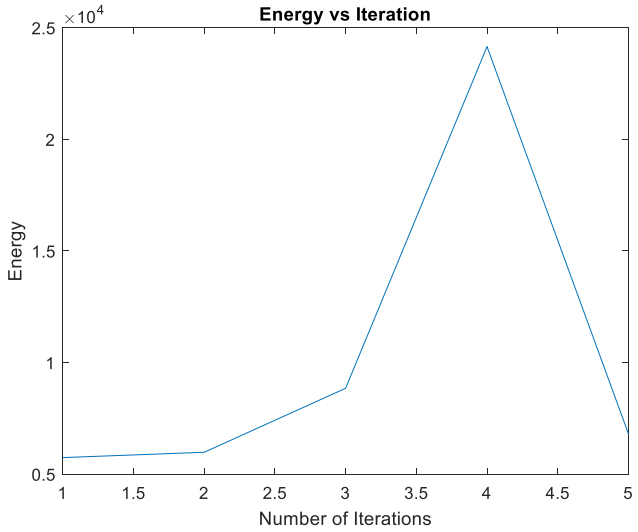


Figure 4. Energy vs Number of iterations for a Noisy Synthetic Binary Image.

We can clearly see the Chan-Vese algorithm can give a good segmentation even of very noisy images. Overall the algorithm extracts the segments in under 4 iterations. But despite that, the algorithm has picked up tiny segments of pixels appearing in the lower background of the image.

In this plot, we see a rise and then a fall in the energy curve. Here as the contour evolves, it is normal for the energy to increase greatly because the contour must spread to segment a majority of the image. This causes the energy to rise before it can fall and settle on a minimum value.

Miscellaneous Scenarios:

The algorithm is more suited to applications where speed and processing power are not of paramount importance, such as in medical image analysis where processing can be done offline after the images are acquired.

Next we apply this algorithm over different types of images to understand the utility of this algorithm.The scaling parameters are set and the number of iterations are varied to achieve the desired segmentation.

RESULTS:

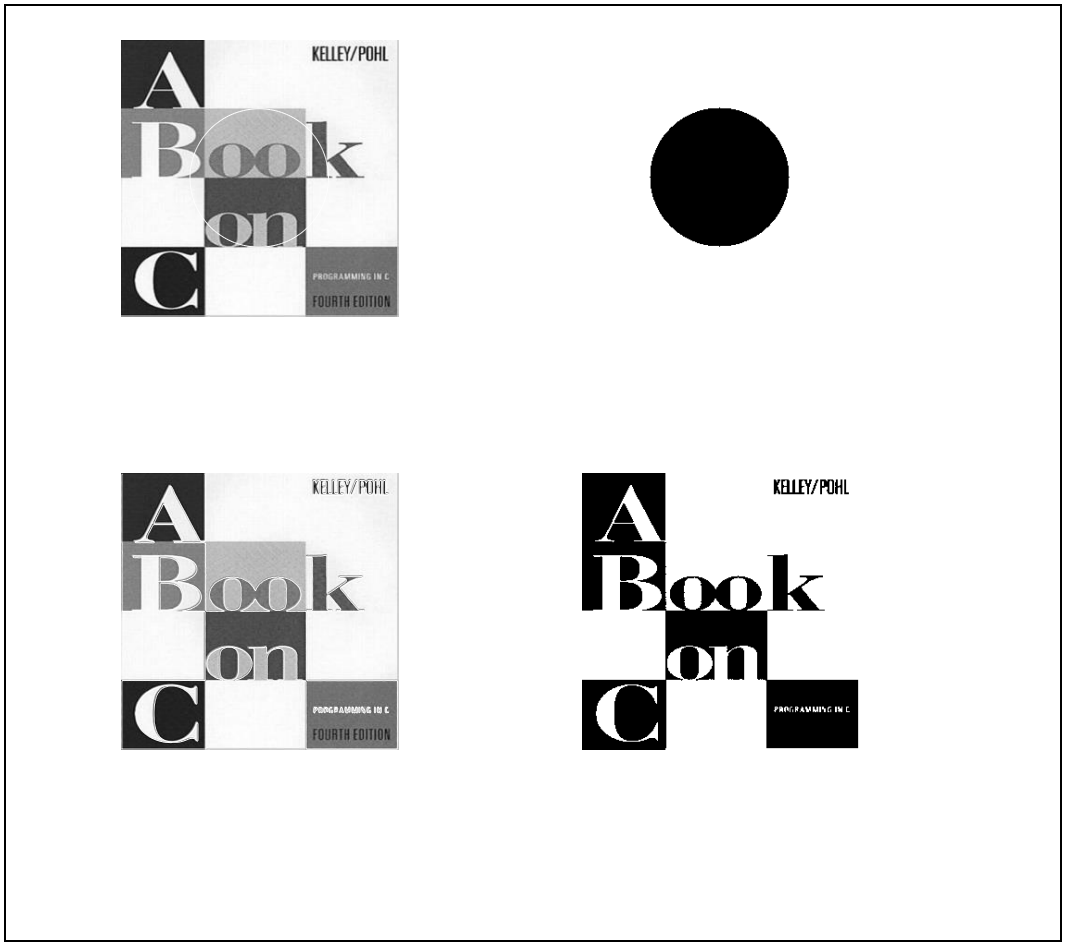


Figure 5. Application of the Chan-Vese Algorithm on Image "A Book on C".

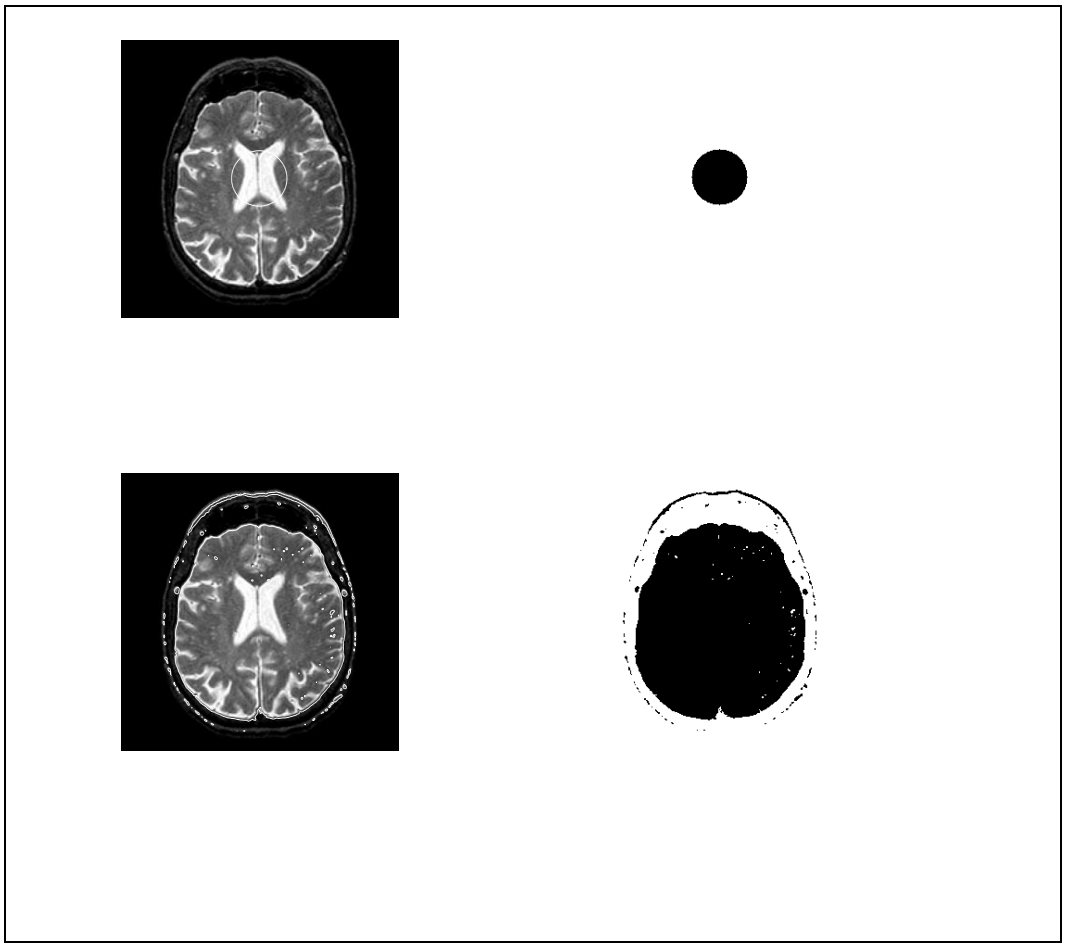


Figure 6. Application of the Chan-Vese Algorithm on "Cortex" Image

4. Conclusions

We have introduced the Chan-Vese algorithm for image segmentation and shown that it is effective on a wide variety of images. It is especially useful in cases where an edge-based segmentation algorithm will not suffice, since it relies on global properties (gray level intensities, contour lengths, region areas) rather than local properties such as gradients. This means that it can deal gracefully with noisy images and blurry images.

Although all the test images that we experimented on were under a minute to segment, the Chan-Vese algorithm is slow for some applications. Depending on the type and size of the image and the number of iterations needed, the segmentation can take several seconds, which is too slow to keep up with typical video framerates. A survey of the literature reveals that the predominant uses of this method have been in medical image analysis, particularly for segmentation problems where a diagnostically relevant feature such as a lesion or tumor shows up as a dark or light spot as in figure 15.

Finally, we mention that this algorithm represents an exciting trend in the "modernization" of image processing and analysis. Mathematical subjects and methodologies which have traditionally found little use in image processing, such as partial differential equations and variational calculus, have over the past decade or so found a multitude of applications. From segmentation to image inpainting and denoising

problems and beyond, such methods will no doubt play an important role in future image analysis research.

5. References

[1] T. Chan and L. Vese, “Active contours without edge”, IEEE Transactions on image processing, 10(2):266– 277, 2001.

[2] M. Kass, A. Witkin, and D. Terzopoulos. “Snakes: Active contour models”, International journal of computer vision, 1(4):321–331, 1988.

[3] Pascal Getreuer, “Chan Vese Segmentation”, Image Processing On Line, ISSN 2105–1232 c 2012.

[4] Chan, T.F., & Sandberg Y. B(2000). Active contours without edges for Vector-valued Image. Journal of Visual Communication and Image Representation 11, 130–141 (2000)

[5] Kass, M., Witkin, A., & Terzopoulos, D. (1988). Snakes: Active contour models. International Journal of Computer Vision, 1(4), 321-331.

[6] Chan, T. F., & Vese, L. A. (2002). A Multiphase level set framework for image segmentation using the Mumford and Shah model. International Journal of Computer Vision 50(3), 271–293, (2002).