**ECE 6560**

**Analysis of Chan-Vese Segmentation**

**Rahmaan Lodhia**

**April 25, 2015**

Contents

[1. Introduction 3](#_Toc417594813)

[2. The Chan-Vese Model 3](#_Toc417594814)

[2.1. Defining the Energy Functional 3](#_Toc417594815)

[2.2. Deriving the Gradient Descent and Level-set Functions 4](#_Toc417594816)

[3. Discretization and Implementation 9](#_Toc417594817)

[3.1. Defining the Discretization Scheme 9](#_Toc417594818)

[3.2. Implementing the Chan-Vese Algorithm in MATLAB 11](#_Toc417594819)

[4. Investigations and Results 11](#_Toc417594820)

[4.1. Performance in Regular and Noisy Images 11](#_Toc417594821)

[4.2. Energy Minimization over Iterations 12](#_Toc417594822)

[4.3. Effect of Area in Energy Functional 12](#_Toc417594823)

[5. Conclusions 12](#_Toc417594824)

[6. References 12](#_Toc417594825)

[Appendix A: MATLAB Source Code 12](#_Toc417594826)

# Introduction

Image segmentation is a thoroughly explored topic in Image Processing and Computer Vision. The goal of image segmentation is to partition an image into meaningful section, which is used in a multitude of fields from medical applications such as locating tumors in medical images to recognition applications such as determining fingerprints. Some of the more basic implementation of segmentation include thresholding, clustering and edge detection, but there are more robust methods available, some of which involves the concept of active contours.

Active contours uses the idea of energy minimizations to define the movement of a curve to enclose significant objects in an image. The first implementation of these methods known as the Snakes model proposed Kass et al. [1] was successful in segmentation but was found to be heavily influenced by the nature of the edges found in an image. Later on, a new set of contours were developed that were based on region and as result independent of edges in an image. These contours are influenced by the information found in the area enclosed by the active contour and the area outside the active contour allowing it to be unaffected by edges. These region-based contours are the subject of this paper, and the one that will be discussed and analyzed is the Chan-Vese model proposed by Chan and Vese [2].

In this paper, the definition and derivation of the mathematical model used in this algorithm will be shown in Section 2. Section 3 will demonstrate how to discretize and implement this model on a computing application. Section 4 will present and explore several quantitative and qualitative results from the implementation of this method. Finally, Section 5 will provide closing remarks discussing weaknesses found in the implementation and any potential work that could improve upon the model used in this paper.

# The Chan-Vese Model

## Defining the Energy Functional

Active contour models require defining an energy functional. The purpose of an energy functional is to create a model of the energy of the image that the active contour will attempt to minimize. At each time step, the goal of the active contour is to progress to a lower energy. The Chan-Vese model is a region-based active contour, so as a result, the energy functional should attempt to minimize the energy in the regions inside and outside the contour.

The energy functional used in the Chan-Vese model can be defined as follows:

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

In this equation (1), represents the image, is the average intensity inside the image, is the average intensity outside the image, is the domain of the inside of the contour, is the domain of the outside of the contour, is the contour, and ,, , are non-negative scaling parameters. Each term in this equation is a part of the overall defined energy in the image and are the values we wish to minimize. The first two terms pertain to the intensity found in the two regions the contour seeks to create. Whatever is inside the contour should be an object of interest and whatever is outside of the contour should be the rest of the image. The first two terms seek to minimize the energy on both sides of the contour and keep the regions as uniform as possible. The third term seeks to penalize the length of the contour, which will lead the contour to attempt to enclose with as little length as needed. The final term seeks to penalize the area of the contour, which will attempt to force it only enclose the smallest area possible. The parameters in front of each term vary the influence each energy has on the overall energy.

In its current state, this energy functional (1) does not explicitly show us how the contour changes each time step. We need to derive what is known as the gradient descent of the function. The gradient descent is a function that shows how the contour will evolve during each time step of the algorithm. After its derivations, we can redefine the gradient descent in terms of a level-set function, which helps to simplify its numerical computation. The next section details the derivation of this contour evolution model.

## Deriving the Gradient Descent and Level-set Functions

The gradient descent of a function is defined as:

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

We will use the following relationship to derive the gradient descent (2) of our energy functional defined in (1):

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

This relationship notes that the time derivative of the energy functional can be rewritten as the inner product of and . This inner product can also be redefined as the L2 continuous norm defined over the contour. As a result, the time derivative of the energy functional can be simplified down to this form, which provides the ability to extract from the integrand once we have isolated .

We will apply this method first to a generic non-linear function integrated over the region inside the contour, which resembles several terms in the Chan Vese energy functional in (1) :

|  |  |  |
| --- | --- | --- |
|  |  | (4) |

In order to use the relationship defined in (3), we need to rewrite the energy functional (4) in terms of an integral around the curve of the contour. If we set , this relationship can be found by application of the divergence theorem:

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

Before we can take the time derivative of this integral, we need to replace all instances of the time-dependent variable with the time-independent parametric variable. This can be achieved with the following relationship: .

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

For the rest of the derivation, the following numerical definition of the curve’s unit normal and unit tangent components will be applied:

|  |  |  |
| --- | --- | --- |
|  |  | (7) |

Now, the time derivative of the energy functional can be applied. Since we removed the time-dependence by switching variables, we can bring the derivative inside the integral and instead take the derivative of the integrand with respect to :

|  |  |  |
| --- | --- | --- |
|  |  | (8) |

Using the product and chain rule for derivation, we arrive at the next step. In this step, we also switch the order of derivation for and use instead and by substituting in the Jacobian for :

|  |  |  |
| --- | --- | --- |
|  |  | (9) |

We now switch back to our original variable s by the use of the relationship, :

|  |  |  |
| --- | --- | --- |
|  |  | (10) |

We can now apply integration by parts to the second term in the integrand of (10). This in turn results in the isolation and factoring of the variable in our integrand:

|  |  |  |
| --- | --- | --- |
|  |  | (11) |

The functional is now in a form similar to (3). The can now be extracted from the integrand of (11) as follows:

|  |  |  |
| --- | --- | --- |
|  |  | (12) |

Since the energy functional defined in (1) is a geometric function, the tangential component of (12) can be assumed to be cancelled out after expanding the equation. The following simplification is implemented:

|  |  |  |
| --- | --- | --- |
|  |  | (13) |

We can now see that only the normal component of the gradient influences the contour, which matches the behavior of a geometric functional. Note that the result in (13) can be viewed as a matrix equation , where and are part of an orthonormal basis and . Both and form an orthonormal basis, and A is diagonalizable. As a result, . Applying this knowledge to (13) and using the original definition of the function , we now arrive at:

|  |  |  |
| --- | --- | --- |
|  |  | (14) |

With result in (14), we can now define our gradient descent function:

|  |  |  |
| --- | --- | --- |
|  |  | (15) |

Since the level-set implementation of the gradient set is being discussed in this paper, one final simplification brings us to our level-set function:

|  |  |  |
| --- | --- | --- |
|  |  | (16) |

The result found in (16) can now be applied to any part of the Chan Vese energy functional in (1) that involves integrating over the area inside the contour. However, there is still a term that involves the integration over the area outside the contour. With a slight modification, the result in (16) can still be applied.

We first note that the integration outside of the curve is equivalent to the difference between the integration over the entire image and the integration over the region inside the curve for a given function:

|  |  |  |
| --- | --- | --- |
|  |  | (17) |

Note that the integration over the region of the entire image has no time dependence. When we take the time derivative of the functional as we did in (8), the first term of (17) equates to zero. This means we only need to consider the second term of (17), which using the same method to find (16), results in:

|  |  |  |
| --- | --- | --- |
|  |  | (18) |

With the results noted in (16) and (18), three of the four terms in the Chan Vese energy functional (1) from now have a level-set implementation defined. The final term left is the part of the function that defines the length of the contour, which acts as the diffusion term. We apply the same method of derivation to this function as we did for the other terms and begin by taking the time derivative of the length after switching out the time dependent variables:

|  |  |  |
| --- | --- | --- |
|  |  | (19) |

Now, we can use substitute , simplify, and swap derivative parameters:

|  |  |  |
| --- | --- | --- |
|  |  | (20) |

Applying integration by parts and using the relationships defined in (7), the following equation is derived:

|  |  |  |
| --- | --- | --- |
|  |  | (21) |

Factoring , extracting from our integrand as defined in (3), and noting that , where is the curvature of the contour, the gradient descent function can be defined as:

|  |  |  |
| --- | --- | --- |
|  |  | (22) |

Using the value of curvature , the level-set function is found to be:

|  |  |  |
| --- | --- | --- |
|  |  | (23) |

Finally, using the level-set implementations found in (16), (18), and (23) on the original energy functional in (1), the level-set gradient descent equation for the Chan Vese algorithm is computed as:

|  |  |  |
| --- | --- | --- |
|  |  | (24) |

# Discretization and Implementation

## Defining the Discretization Scheme

In order to apply the level-set equation derived in the previous section to an image, a discretization scheme must be defined for the derivatives needed in this equation. Looking at (24), we can note that the desired function has three non-linear terms and one diffusion term. Therefore, the level-set functions requires the usage of two discretization schemes.

The first discretization scheme can be found by expanding out the diffusion term of (24):

|  |  |  |
| --- | --- | --- |
|  |  | (25) |

This formula involves first and second order partial derivatives. Each of these derivatives are best computed by the use of a central difference scheme, as this scheme captures information from bordering pixels on both side of a given point and provides a more accurate approximation. The central difference equations used to approximate these derivatives are as follows:

|  |  |  |
| --- | --- | --- |
|  |  | (26) |

By replacing the values in (25) with the respective equations found in (26), we can approximate the diffusion component of the level-set. It is important to note that the choices of , , and influence the stability of this discretization scheme. Van Neumann analysis was used to determine the CFL condition for these terms, which demonstrate the relationship needed between the values of , , and in order for this approximation to maintain stability. The CFL conditions for the discretization scheme of (25) is:

|  |  |  |
| --- | --- | --- |
|  |  | (27) |

For the non-linear terms, a different discretization scheme is required. We cannot use central differences for this approximation, and we cannot simply use a forward or backward difference with no modifications. The partial derivatives of the level-set can come out to be positive or negative within the absolute value operator, which means the same scheme cannot be used for all values of the level-set. The nature of the non-linear terms’ approximation depends on the sign of its coefficient and the signs of the approximation schemes used in it. To properly account for differences in sign, an upwind entropy scheme is needed. The upwind entropy scheme is as follows:

|  |  |  |
| --- | --- | --- |
|  |  | (28) |

Within the upwind entropy scheme, both the backward and forward difference are used in its calculation. As a result, the discretization will now correctly be calculated depending on the sign of the coefficient and the signs of both differencing schemes. The backward and forward differences used in (28) can be approximated with following equations:

|  |  |  |
| --- | --- | --- |
|  |  | (29) |

Once again, to ensure the stability for this scheme for all approximations, the CFL condition must be fulfilled when choosing the parameters. The CFL condition of this discretization scheme is:

|  |  |  |
| --- | --- | --- |
|  |  | (30) |

Since we have two different discretization schemes being used, we have two different CFL conditions that need to be fulfilled in order for the overall scheme to be stable. Therefore, for the overall discretization scheme, we must fulfill the most restrictive of these CFL conditions, which will correspond to the diffusion term’s CFL found in (27).

## Implementing the Chan-Vese Algorithm in MATLAB

The basic algorithm used to code the Chan-Vese algorithm is as follows:

1. Initialize the parameters ,, , , and
2. Initialize a level-set function with a desired shape and location for the contour
3. Using the regions defined by the current contour, calculate values for and
4. Calculate the resulting using the discretization schemes in (25) and (28) and the values of and from step 3
5. Calculate in order to evolve the contour
6. Repeat steps 3-5 until desired segmentation

In order to implement the above algorithm in MATLAB, the parameters , , and used to define the discretization scheme must be determined. Setting aided in simplifying the code used to calculate the partial derivatives in our gradient descent function, as this selection allowed the calculations to be computed using MATLAB’s robust matrix indexing and operations. Plugging in these chosen values for and into our CFL condition from (30), we determined that . For our implementation, we set , which falls well within our CFL condition.

In this implementation, we also needed to decide on the initial contour and scaling parameters. For most of the experiments to be shown in the next section, we kept these parameters constant for most tests. In regards to the contour, we decided to use the same contour initialization for all of experiments, which was a grid of 16 equally spaced circle contours. The zero level-set of our contour used is displayed below in Figure

For the first two experiments, we used the scaling values and, as many applications of the Chan-Vese algorithm usually remove the area penalty. However, the last experiment explores the effect of the area penalty on the algorithm’s performance. In that experiment, will be varied while keeping the other variables constant.

Finally, to further simplify the implementation, all images used were converted to grayscale, scaled to a 200x200 pixel size, and normalize to a range . This lowered computation time and simplified calculations as only one channel of the image was being operated on. The code used for this implementation of the Chan-Vese algorithm is documented in Appendix A: MATLAB Source Code.

# Investigations and Results

## Performance in Regular and Noisy Images

## Energy Minimization over Iterations

## Effect of Area in Energy Functional

# Conclusions

# References

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

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

# Appendix A: MATLAB Source Code