Level Set Segmentation: Active Contours Without Edges

Release 0.00

Kishore Mosaliganti[1], Benjamin Smith[2], Arnaud Gelas[1], Alexandre Gouaillard[1] and Sean Megason[1]

March 13, 2009

[1]Systems Biology, Harvard Medical School, Boston, MA-02139, USA
[2]Simon Fraser University, Vancouver, Canada

Abstract

An Insight Toolkit (ITK) processing framework for segmentation using active contours without edges is presented in this paper. Our algorithm is based on the work of Chan and Vese [1] that uses level-sets to accomplish region segmentation in images with poor or no gradient information. The basic idea is to partion the image into two piecewise constant intensity regions. This work is in contrast to the level-set methods currently available in ITK which necessarily require gradient information. Similar to those methods, the methods presented in this paper are also made efficient using a sparse implementation strategy that solves the contour evolution PDE at the level-set boundary. The framework consists of 6 new ITK filters that inherit in succession from `itk::SegmentationFilter`. We include 2D/3D example code, parameter settings and show the results generated on a 2D cardiac image.

Latest version available at the Insight Journal [http://hdl.handle.net/1926/1]

Contents

1 Introduction

The levelset methodology involves numerically evolving a contour or surface according to a rate change partial differential equation (PDE). The contour or surface is embedded as the zero level-set of a higher dimensional implicit function, also called the level-set function $\phi(x,t)$. Other techniques in the literature with the same objective include snakes and parametric surfaces. The main advantages of using level-set techniques over other methods is that it can handle complex shapes, topological changes such as merging and splitting with ease. Different constraints on the contour smoothness, speed, size and shape are easily specified. An overview of the field has been made by Sethian [3].

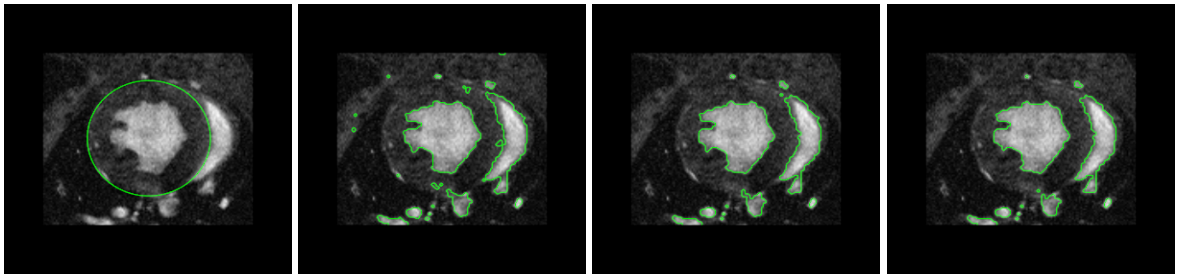
Level-sets are very popular in the image analysis community for use in image segmentation, smoothing and tracking problems. There are two main classes of the level-set methodology - explicit and implicit methods. The explicit methods have a generic PDE to determine the temporal rate of change in the level-set function. It is expressed as a sum of speed functions that contain propagation, smoothing and advection terms. Each of these terms are constructed from the image intensity function and involves the usage of image gradients. Therefore, segmentation is accomplished by detecting edges in the object of interest. Hence, these techniques are also referred as edge-based level-sets. Implicit level-sets are driven by PDE that is derived by using a variational formulation. Here, an energy functional is first constructed and then minimized. An example of such a variant is that proposed by Chan and Vese [1]. In this variant, objects are segmented in images where edge information may be poor or entirely absent. Hence, these techniques are also terms as region-based or active contours without edges. The current level-set framework in ITK is completely explicit with no support for the implicit techniques. This paper describes our implementation of the Chan and Vese algorithm.

2 Description: Active Contour without Edges

The basic idea of active contour models relies on detecting salient objects by evolving a curve or a surface subject to image-based constraints. Let $\Omega \subset \mathbb{R}^d$ be the image domain and $u_0 : \Omega \rightarrow \mathbb{R}$ be a given image function. In [2], Mumford and Shah made a seminal contribution to the literature on image segmentation problem by proposing an energy functional to be minimized by a suitable contour C in Ω.

$$F^{MS}(u,C) = \int_{\Omega} (u-u_0)^2 dx + \int_{\Omega \setminus C} |\nabla u|^2 dx + \nu |C| \tag{1}$$

where $|C|$ is the length of the contour C and ν is a non-negative constant weight. The minimization of the above functional results in an optimal contour that segments the image and provides a piecewise smooth image u that approximates the orignal image u_0. In general, this problem is ill-posed since there are no bounds on shape/topology of the unknown contour C of lower dimension and the non-convexity of the functional. i



(a) (b) (c) (d)

Figure 1: Parameters: $\varepsilon = 1$, $\mu = 0$, $\nu = 0$, $\lambda_1 = 1$, $\lambda_2 = 1$. Iterations: (a) 10 (b) 20 (c) 30 (d) 40.

Chan and Vese [1] proposed an active contour model for segmenting images containing objects that have poor boundaries. They proposed an energy that is an piece-wise constant approximation to the Mumford and Shah functional:

$$F^{CV}(C,c_1,c_2) = \lambda_1 \int_{\Omega_{in}} |I(x)-c_1|^2 dx + \lambda_2 \int_{\Omega_{out}} |I(x)-c_2|^2 dx + \mu.Area(C) + \nu.Volume(C) \tag{2}$$

where Ω_{in} and Ω_{out} represent the region inside and outside the contour C, respectively, and c_1 and c_2 are two scalar constants that approximate the image intensities. The first two terms are often referred as global binary fitting energy terms that seek to separate an image into two regions of constant image intensities. By using a level-set formulation, the minimization problem can be converted to a level-set evolution equation.

In the above model, the constants c_1 and c_2 are used to fit the image intensities in Ω_{in} and Ω_{out} respectively. Therefore, the global nature of the fit requires that the intensity histograms follow a Gaussian distribution within the two regions.

In level-set methods, a contour $C \subset \Omega$ is represented by the zero level-set of a Lipschitz continuous function $\phi : \Omega \to \mathbb{R}$. With the level-set representation, the energy functional $F^{CV}(u,c_1,c_2)$ can be rewritten as

$$\begin{aligned} F^{CV}(\phi,c_0,c_1) = {} & \lambda_1 \int_{\Omega_{in}} |I(x)-c_1|^2 H(\phi) dx \\ & + \lambda_2 \int_{\Omega_{out}} |I(x)-c_2|^2 (1-H(\phi)) dx \\ & + \mu \int_{\Omega} \delta(\phi)|\nabla\phi| dx + \nu \int_{\Omega} H(\phi) dx \end{aligned} \tag{3}$$

where H is the Heaviside function. In practice, the Heaviside function and the Dirac function in Equation 4 are approximated by smooth function H_ε and δ_ε defined as

$$H_\varepsilon(x) = \frac{1}{2}\left(1 + \frac{2}{\pi} arctan\left(\frac{x}{\varepsilon}\right)\right); \quad \delta_\varepsilon(x) = \frac{1}{\pi}\frac{\varepsilon}{\varepsilon^2 + x^2} \tag{4}$$

We then minimize the energy functional to obtain tangible segmentations. For a fixed level-set function ϕ, we minimize the functional F^{CV} with respect to the remaining parameters. By calculus of variations, it can

be shown that the following Euler-Lagrange equations need to be satisfied.

$$c_1 = \frac{\int_\Omega u(x)\big(1-H(\phi(x))\big)dx}{\int_\Omega \big(1-H(\phi(x))\big)dx} \tag{5}$$

$$c_2 = \frac{\int_\Omega u(x)\big(H(\phi(x))\big)dx}{\int_\Omega \big(H(\phi(x))\big)dx} \tag{6}$$

By minimizing the energy functional with respect to ϕ and using H_ε and δ_ε, we derive the gradient descent flow:

$$\begin{aligned} \frac{\partial \phi}{\partial t} &= \delta_\varepsilon(\phi)\Big[\mu div\big(\frac{\nabla\phi}{|\nabla\phi|}\big) - \nu - \lambda_1\big(I-c_0\big)^2 \\ &+ \lambda_2\big(I-c_1\big)^2\Big] = 0 \quad \text{in } (0,\infty)\times\Omega, \end{aligned} \tag{7}$$

$$\phi(0,x) = \phi_0(x) \qquad \text{in } \Omega \tag{8}$$

$$\frac{\delta_\varepsilon}{|\nabla\phi|}\frac{\partial\phi}{\partial\vec{n}} = 0 \qquad \text{in } \partial\phi \tag{9}$$

The above Equation 7 is the implicit active contour model implemented in this paper.

3 Implementation

In our implementation, we design the filters that follow a similar pattern of inheritance as in the other level-set filters available in ITK. The inheritance for the dense, sparse solvers and level-set function is shown in Figures 2, 3 and 4.

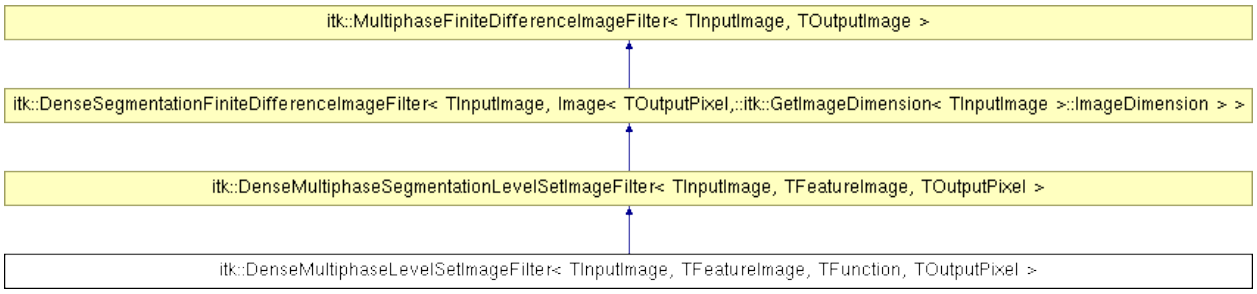


Figure 2: Inheritance model of the dense filter

The dense and sparse filter options affect the computational performance of the method. However, they both solve the same underlying equation. Hence, the parameter settings remain the same. We now describe each of the parameters, their range and typical values. There is no typical limit that can be set on most parameters but depends on experimentation. Note that except for the first two, the remaining constitute weights to the different energy terms. Depending on their contribution to the overall energy, these weights need to be modified so that all the terms have an influence.

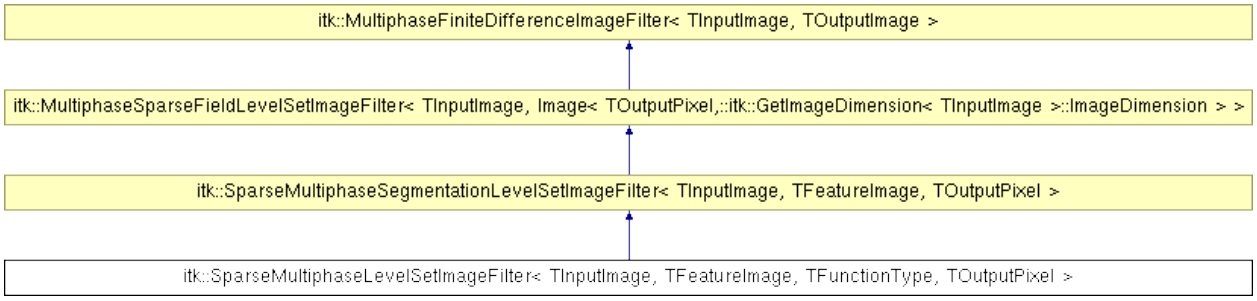


Figure 3: Inheritance model of the sparse filter

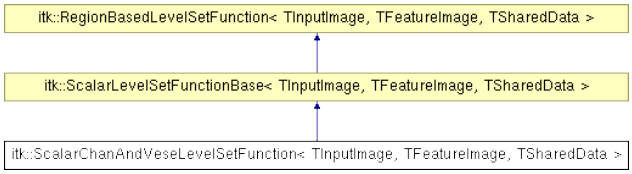


Figure 4: Inheritance model of the level-set function

`m_Iterations` - Maximum permitted iterations of the evolution in the range $[0,\infty]$. Typical value depends on the initialization. The initialization must not be too different from the final output for best results. It is usually set my trial and error and 100 iterations are usually sufficient.
`m_MaxRMSChange` - Maximum change in the level-set function averaged over all the pixels in the range $[0,\infty]$. During convergence, a low RMS change indicates that the level-set function does not change position anymore. Usually a value between 0.1 is sufficient.
`m_Epsilon` - Defines the smoothness of the Heaviside and Delta functions defined in Equation 4. Usually in the range $[1,\infty]$. We set the value to 1 in our images and obtain good results. For very high resolution images, this value can be changed to 2 or 3.
`m_Mu` - The weight of the length regularization in the energy function and lies in the range [0,∞]. The exact specification depends on the images and the expected length of the segmentation boundary. It can be as high as 10000.
`m_Nu` - The weight of the area regularization in the energy function and lies in the range [0,∞]. The exact specification depends on the images and the expected length of the segmentation boundary. It can be as high as 10000.
`m_Lambda1` - Weights of the sum of squares of the zero mean intensities inside the contour and lies in the range [0,inf]. Usually set as 1 and other parameters are decided based on this normalized value.
`m_Lambda2` - Weights of the sum of squares of the zero mean intensities outside the contour and lies in the range [0,inf]. Usually set as 1 and other parameters are decided based on this normalized value.

4 Usage

We begin by including the appropriate header files for the solver (both dense and sparse) and function. Depending on the choice, either dense or sparse is required. We then define level-set image type and the feature image type. Note that internally, these image types are cast into float type.

```
#include "itkScalarChanAndVeseLevelSetFunction.h"
#include "itkDenseMultiphaseLevelSetImageFilter.h
#include "itkSparseMultiphaseLevelSetImageFilter.h"

...
int main(int argc, char *argv[])
{
...

unsigned int Dimension = 2;
typedef float ScalarPixelType;
typedef itk::Image< ScalarPixelType, Dimension > LevelSetImageType;
typedef itk::Image< ScalarPixelType, Dimension > FeatureImageType;
```

4.1 Dense Option

If the dense option is selected, the following typedef are required.

```
typedef itk::ScalarChanAndVeseLevelSetFunction< LevelSetImageType,
  FeatureImageType > LevelSetFunctionType;
typedef itk::DenseMultiphaseLevelSetImageFilter< LevelSetImageType,
  FeatureImageType, LevelSetFunctionType, float > MultiLevelSetType;
```

4.2 Sparse Option

```
typedef itk::ScalarChanAndVeseLevelSetFunction< LevelSetImageType,
  FeatureImageType > LevelSetFunctionType;
typedef itk::SparseMultiphaseLevelSetImageFilter< LevelSetImageType,
  FeatureImageType, LevelSetFunctionType, float > MultiLevelSetType;
```

Note that the filter can support multiphase implementations. However, we choose to report that work with suitable optimizations and enhancements in a later submission. For now, we recommend the user to always set the parameters by calling the 0th difference function (`levelSetFilter->GetTypedDifferenceFunction(0)`).

```
MultiLevelSetType::Pointer levelSetFilter = MultiLevelSetType::New();
levelSetFilter->SetFunctionCount( 1 );
levelSetFilter->SetFeatureImage( featureImage );
levelSetFilter->SetLevelSet( 0, contourImage );
levelSetFilter->SetNumberOfIterations( atoi( argv[2] ) );
levelSetFilter->SetMaximumRMSError( atof( argv[3] ) );
levelSetFilter->SetUseImageSpacing( 0 );

levelSetFilter->GetTypedDifferenceFunction(0)->SetEpsilon( atof( argv[4] ) );
levelSetFilter->GetTypedDifferenceFunction(0)->SetMu( atoi( argv[5] ) );
levelSetFilter->GetTypedDifferenceFunction(0)->SetNu( atoi( argv[6] ) );
levelSetFilter->GetTypedDifferenceFunction(0)->SetLambda1( atoi( argv[7] ) );
levelSetFilter->GetTypedDifferenceFunction(0)->SetLambda2( atoi( argv[8] ) );
levelSetFilter->GetTypedDifferenceFunction(0)->SetGamma( atoi( argv[9] ) );
levelSetFilter->GetTypedDifferenceFunction(0)->SetNeta( atoi( argv[10] ) );
levelSetFilter->GetTypedDifferenceFunction(0)->SetTau( atoi( argv[11] ) );
```

```
levelSetFilter->GetTypedDifferenceFunction(0)->SetVolume( atoi( argv[12] ) );
levelSetFilter->Update();
```

5 Results

The results in this example can be obtained by using `ScalarSinglePhaseDense2DTest.cxx` on the input image `example2D.png` and initial level-set image `phi.mha`. In this example, an initial contour (a circle), is evolved to segment the anatomic structure. The image can be assumed to consist of two regions of constant intensities and hence the Chan and Vese method can be applied. The circle is embedded in a distance map (*phi.mha*) and evolved. Since there are several parameters and implementation choices to make, we illustrate their use separately. The output is written out to the image `seg.mha`. The parameters to the filter are set at the command line to facilitate easy modification and exploration by the user. The command to run this particular executable is as follows:

```
./ScalarSinglePhaseDense2DTest FeatureImage (input image)
 ContourImage (level set initialization) OutputImage (output image)
 -h   --help                  Prints this help
 -i   --iter    10  (default)  Number of Iterations
 -r   --rms     0   (default)  rms
      --epsilon 1   (default)  Epsilon parameter in the Heaviside and dirac
      --mu      0   (default)  Length Regularization weight
      --nu      0   (default)  Area Regularization weight
      --l1      1    (default) Inside parameter
      --l2      1    (default) Outside parameter
```

5.1 Dense

In the dense filter category, we use the `itk::DenseMultiphaseLevelSetImageFilter` solver that inherits from `itk::DenseMultiPhaseSegmentationLevelSetImageFilter`. Potential users of this code can execute the example with the following command line:

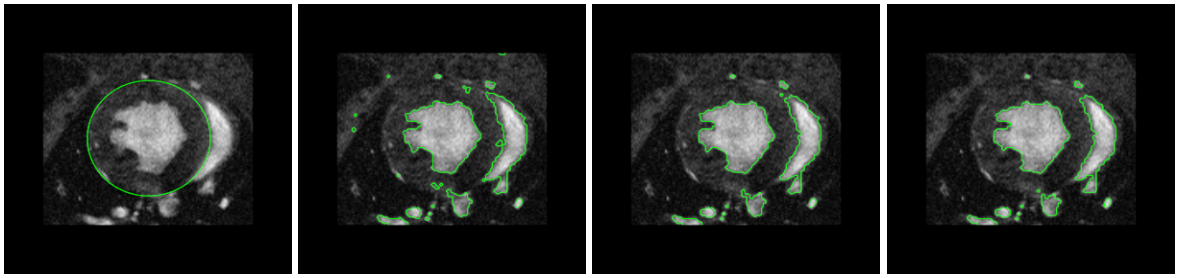
```
./ScalarSinglePhaseDense2DTest example2D.png phi.mha seg.mha -i 50 --l1 1 --l2 3
```

The dense filter iteratively updates every single pixel in the image irrespective of its position with respect to the zero level-set. As a result, the convergence is much quicker and also structures far off from the zero level-set spawn contours to segment them. Note that a very good segmentation is obtained by 10 iterations alone and very little refinement happens later on.

5.2 Sparse

In the sparse filter category, we use the `itk::SparseMultiphaseLevelSetImageFilter` solver that inherits from `itk::SparseMultiPhaseSegmentationLevelSetImageFilter`. Potential users of this code can execute the example with the following command line:

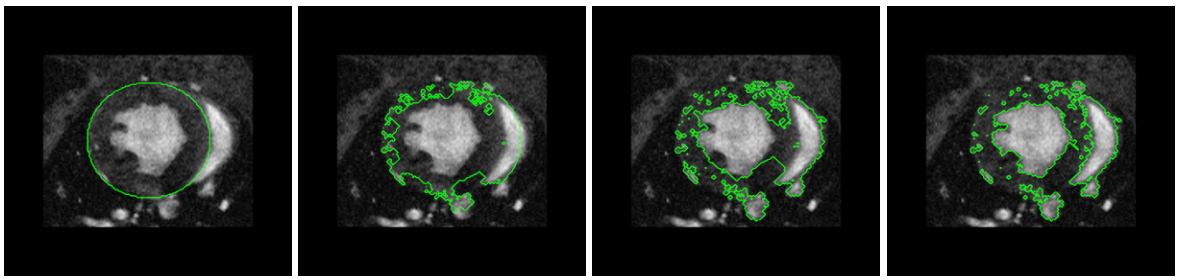
```
./ScalarSinglePhaseSparse2DTest example2D.png phi.mha seg.mha -i 50 --l1 1 --l2 1
```



(a) (b) (c) (d)

Figure 5: Parameters: $\varepsilon = 1$, $\mu = 0$, $\nu = 0$, $\lambda_1 = 1$, $\lambda_2 = 1$. Iterations: (a) 10 (b) 20 (c) 30 (d) 40.

The sparse implementation of the level-set solver maintains a band of pixels around the zero level-set which are iteratively updated. The PDE is solved exactly on those pixels that are on or immediate neighbors of the zero level-set. A user-defined band width of pixels are used in maintaining the level structure and for calculating the derivatives etc. Pixels in the image beyond this band around the zero level-set are not considered. As a result, the evolution gradually proceeds from the initialization and converges on local structures. Structures far off from the zero level-set may remain unaffected.



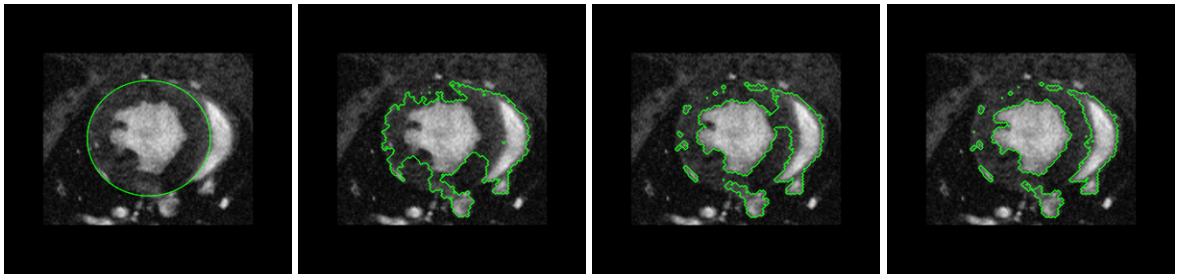
(a) (b) (c) (d)

Figure 6: Parameters: $\varepsilon = 1$, $\mu = 0$, $\nu = 0$, $\lambda_1 = 1$, $\lambda_2 = 1$. Iterations: (a) 10 (b) 20 (c) 30 (d) 40.

5.3 Perimeter regularization

In order to explore the perimeter regularization parameter μ, we stick to using the sparse implementation. Note that a segmentation of a given region with a highly fractal contour and a smooth contour may contain the same energy if there were regularization on the contour length. In order to have smooth contours, it is important to have a energy term that penalizes the length of the contour. We set $\mu = 1500$ and observe the change in the previous result. Note that the contour now is much more smooth and sharp edges are reduced.

```
./ScalarSinglePhaseSparse2DTest example2D.png phi.mha seg.mha -i 50 --l1 1 --l2 1 --mu 1500
```



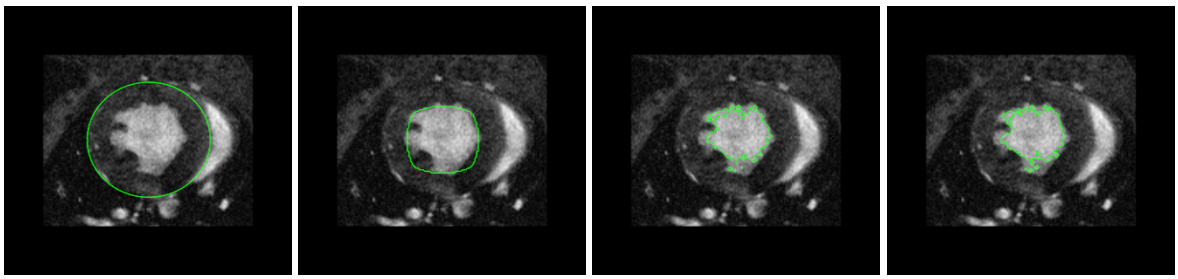
(a) (b) (c) (d)

Figure 7: Parameters: $\varepsilon = 1$, $\mu = 1500$, $\nu = 0$, $\lambda_1 = 1$, $\lambda_2 = 1$. Iterations: (a) 10 (b) 20 (c) 30 (d) 40.

5.4 Area regularization

In order to explore the area regularization parameter ν, we once again resort to using the sparse implementation. Note that a segmentation of a given region with holes and broken fragments may contain the same energy if there were no regularization on the area. In order to have consistent foreground, it is important to have a energy term that penalizes the area. We set $\nu = 10000$ and observe the change in the previous result. Note that the contour now segments the main structure alone.

```
./ScalarSinglePhaseSparse2DTest example2D.png phi.mha seg.mha -i 50 --l1 1 --l2 1 --nu 10000
```



(a) (b) (c) (d)

Figure 8: Parameters: $\varepsilon = 1$, $\mu = 0$, $\nu = 10000$, $\lambda_1 = 1$, $\lambda_2 = 1$. Iterations: (a) 10 (b) 20 (c) 30 (d) 40.

References

[1] T. Chan and L. Vese. An active contour model without edges. In *Scale-Space Theories in Computer Vision*, pages 141–151, 1999.

[2] 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.

[3] J.A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1996.