A Short Guide on a Fast Global Minimization Algorithm for Active Contour Models

Xavier Bresson

April 22, 2009

Abstract

This is a short guide to explain how to quickly minimize a large class of segmentation models called active contours. We believe that the proposed theory and algorithm, introduced in a series of papers [6, 4, 10, 11], produce so far one of the most efficient minimization methods for the active contour segmentation problem. For example, the well-know cameraman picture, which size is 256×256, is segmented in less than 0.1 seconds.
We also compare our algorithm with the popular and fast graph-cuts technique [2, 9]. We show that our algorithm, while being easier to code, produces results slightly faster than graph-cuts. Besides, our algorithm is more accurate than graph-cuts because it uses isotropic schemes to regularize the contour and is sub-pixel precise. Finally, the memory requirement is low.

1 General Active Contour Model

The general energy minimization problem for any (two-phase) active contour model is as follows:

$$\min_C \Big\{ F_{AC}(C) = \int_C g_b(C,s)ds + \lambda \int_{C_{in}} g_r^{in}(C_{in},x)dx + \lambda \int_{C_{out}} g_r^{out}(C_{out},x)dx \Big\}, \tag{1.1}$$

where

1. C is a closed contour (curve in 2D and surface in 3D), ds is the arclength element, C_{in}, C_{out} are respectively the inside region and outside region (see Figure 1) of the contour C, dx is the region element.

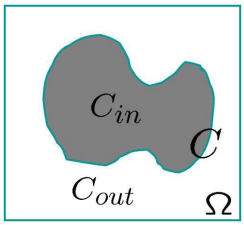


Figure 1: Illustrative figure. The contour C is the boundary between inside region C_{in} and outside region C_{out}. The image domain is Ω.

2. Fucntion g_b is the boundary function, most of time an edge detector function such as:

$$g_b(C,s) = \frac{1}{1+|\nabla I(C(s))|^2},$$

like in the Geodesic Active Contour model of Caselles, Kimmel and Sapiro [5].

3. Functions g_r^{in}, g_r^{out} are the inside and outside region functions (that detect homogeneous regions) such as:

$$g_r^{in}(C_{in}, x) = (\mu_{in}(C_{in}) - I)^2, \qquad \mu_{in} = \frac{\int_{C_{in}} I(x)dx}{\int_{C_{in}} dx}$$
$$g_r^{out}(C_{out}, x) = (\mu_{out}(C_{out}) - I)^2, \qquad \mu_{in} = \frac{\int_{C_{out}} I(x)dx}{\int_{C_{out}} dx}$$

like in the Active Contour without Edges of Chan-Vese [7]. More discriminative region detectors can be based on the variance measure or the intensity distributions [12, 11].

2 Minimizing Energy (1.1)

Energy (1.1) is minimized with the Calculus of Variations and the Shape Derivative Tool (SDT) (see [8, 12, 1] for more details), which provides the following minimizing flow:

$$\partial_t C = \Big(g_b(C,s)\kappa(C,s) + < \nabla g_b, \mathcal{N}(C,s) > + h_r^{in}(C_{in}, s) - h_r^{out}(C_{out}, s)\Big)\mathcal{N}, \tag{2.1}$$

where κ, $\mathcal{N}$ are the curvature and the normal to the curve C. The most standard approach to solve the minimizing flow (2.1) is an alternate iterative approach, defined as follows:

Repeat until convergence:

1. Fix C and update $h_r^{in}(C_{in}),\ h_r^{out}(C_{out})$.
2. Fix $h_r^{in},\ h_r^{out}$ and update C with the gradient flow:

$$\partial_t C = \Big(g_b(C,s)\kappa(C,s) + < \nabla g_b, \mathcal{N}(C,s) > + h_r^{in}(s) - h_r^{out}(s)\Big)\mathcal{N}. \tag{2.2}$$

The flow (2.2) is usually handled with the Level Set Method (LSM) introduced in [14], where C is represented by a level set function ϕ. Indeed, for $h_r^{in},\ h_r^{out}$ fixed, the flow (2.2) minimizes the energy functional:

$$\min_C \ \Big\{F_{AC}^{\dagger}(C) = \int_C g_b(C,s)ds + \lambda \int_{C_{in}} h_r^{in}(x)dx + \lambda \int_{C_{out}} h_r^{out}(x)dx\Big\}, \tag{2.3}$$

The LSM formulation of (2.3) is as follows:

$$\min_\phi \ \Big\{F_{LSM}(\phi) = \textstyle\int_\Omega g_b|\nabla H(\phi)| + \lambda \int_\Omega h_r^{in} H(\phi) + \lambda \int_\Omega h_r^{out}(1 - H(\phi))\ dx\Big\}. \tag{2.4}$$

The LSM is a successful numerical method to solve (2.4). However, the LSM only computes a local minimizer since the LSM energy is not convex. Besides, numerical minimization schemes are slow to converge to the minimizer. In the next section, we show how to convexify energy (2.4) to compute a global minimizer. Section 4 will present a fast algorithm to compute the global minimizer.

3 Convexification of the Level Set Energy (2.4)

As we said, the Level Set minimization problem (2.4) is a non-convex energy minimization problem. This means that the final solution depends on the initial contour. In other words, a bad initial position can lead to a bad solution. In [6, 4], we propose to convexify energy (1.1) to compute a global minimizer, independently of the initial contour position. We remind the general idea. The optimality condition for (2.4) is as follows:

$$0 = \Big(g_b(x) div\big(\tfrac{\nabla\phi}{|\nabla\phi|}\big) + < \nabla g_b(x), \tfrac{\nabla\phi}{|\nabla\phi|} > + h_r^{in}(x) - h_r^{out}(x)\Big)\delta_\epsilon(\phi), \tag{3.1}$$

where δ_ϵ is a regularized version the Dirac function, which is necessary for the computation of the optimality condition. Since $\delta_\epsilon(\phi) \geq 0$, we can remove this function in (3.1) without changing the optimality condition:

$$0 = g_b(x) div\big(\tfrac{\nabla\phi}{|\nabla\phi|}\big) + < \nabla g_b(x), \tfrac{\nabla\phi}{|\nabla\phi|} > + h_r^{in}(x) - h_r^{out}(x).$$

The previous optimality condition is associated with the energy (we change the notation ϕ into u to avoid any confusion with the LSM):

$$\min_{u\in[0,1]} \Big\{F_G(u) = \int_\Omega g_b(x)|\nabla u| + \lambda \int_\Omega h_r^{in}(x)u + \lambda \int_\Omega h_r^{out}(x)(1-u)\ dx\Big\}, \tag{3.2}$$

Energy functional F_G is a convex minimization problem. Besides, we have:

1. $\int_\Omega g_b(x)|\nabla u|dx$ is the weighted Total Variation of function u.
2. Function u is constrained in $[0, 1]$ because F_G is homogeneous of degree 1 and has no steady state (if $h_r^{in} - h_r^{out} > 0$ then $u \to \infty$ and if $h_r^{in} - h_r^{out} < 0$ then $u \to -\infty$).
3. The relation between the active contour energy (2.3) and the convex energy (3.2) is given by the indicator function of a set as follows:

$$F_G(u = 1_{C_{in}}) \ = \ F_{AC}^\dagger(C), \tag{3.3}$$

   where we remind that C is the boundary of C_{in}.

4. *Theorem: Call $u^\star$ any minimizer of (3.2) (which is a global minimizer by convexity), then the thresholded functions:*

$$1_{u^\star>\mu}(x) = \begin{cases} 1 & \text{if } u^\star(x) > \mu \\ 0 & \text{otherwise} \end{cases}, \ \mu \in (0, 1)$$

   are also global minimizers of (3.2), and the active contour energy (2.3) as well (using (3.3)).

In the next section, a fast and accurate numerical scheme is introduced to solve the energy minimization problem (3.2).

4 A Fast Minimization Algorithm for (3.2)

We need to compute a minimizer of (3.2), which will provide a global minimizer of the active contour energy (2.3) (by thresholding, according to the previous Theorem). A fast and accurate minimization algorithm for energy (3.2) is introduced in [10]. This algorithm is much faster and more accurate than

the one introduced in [4]. This algorithm is also slightly faster than the graph-cuts algorithm [9], while using isotropic operators and being easily extensible to 3D images. Minimizing (3.2) with respect to u is equivalent to minimize:

$$\min_{u\in[0,1]} \int_\Omega g_b|\nabla u| + \lambda h_r u \; dx,$$

where we define $h_r = h_r^{in} - h_r^{out}$. A new vectorial function d is introduced as follows:

$$\min_{u\in[0,1],d} \int_\Omega g_b|d| + \lambda h_r u \; dx, \;\; \text{such that } d = \nabla u.$$

The constraint $d = \nabla u$ is enforced using the efficient Bregman iteration approach [10, 13, 3] defined as:

$$\left\{ \begin{array}{rcl} (u^{k+1}, d^{k+1}) & = & \arg\min_{u\in[0,1],d} \int_\Omega g_b|d| + \lambda h_r u + \frac{\mu}{2}|d - \nabla u - b^k|^2 \; dx \\ b^{k+1} & = & b^k + \nabla u^{k+1} - d^{k+1} \end{array} \right. , \; k \geq 0 \qquad (4.1)$$

The minimizing solution u^{k+1} is characterized by the optimality condition:

$$\mu\Delta u = \lambda h_r + \mu div(b^k - d^k), \; u \in [0,1].$$

A fast approximated solution is provided by a Gauss-Seidel iterative scheme, i.e. for $n \geq 0$:

$$\begin{array}{rcl} \gamma_{i,j} & = & d_{i-1,j}^{x,k} - d_{i,j}^{x,k} - b_{i-1,j}^{x,k} + b_{i,j}^{x,k} + d_{i,j-1}^{y,k} - d_{i,j}^{y,k} - b_{i,j-1}^{y,k} + b_{i,j}^{y,k} \\ \mu_{i,j} & = & \frac{1}{4}(u_{i-1,j}^{k,n} + u_{i+1,j}^{k,n} + u_{i,j+1}^{k,n} + u_{i,j+1}^{k,n} - \frac{\lambda}{\mu} h_{r\,i,j} + \gamma_{i,j}) \\ u_{i,j}^{k+1,n+1} & = & \max\{\min\{\mu_{i,j}, 1\}, 0\} \end{array}$$

Finally, the minimizing solution d^{k+1} is given by soft-thresholding:

$$d_{i,j}^{k+1} = \frac{\nabla u^{k+1} + b^k}{|\nabla u^{k+1} + b^k|} \max(|\nabla u^{k+1} + b^k| - \mu^{-1} g_b, 0)$$

5 Summary

The objective is to compute a global minimizer of the (two-phase) active contour model defined as follows:

$$\min_C \; \Big\{ F_{AC}(C) = \int_C g_b(C,s)ds + \lambda \int_{C_{in}} g_r^{in}(C_{in}, x)dx + \lambda \int_{C_{out}} g_r^{out}(C_{out}, x)dx \Big\}, \qquad (5.1)$$

along with the minimizing gradient flow:

$$\partial_t C = \Big(g_b(C,s)\kappa(C,s) + < \nabla g_b, \mathcal{N}(C,s) > + \, h_r^{in}(C_{in}, s) - h_r^{out}(C_{out}, s) \Big) \mathcal{N} \qquad (5.2)$$

Unfortunately, a global minimizer of (5.1) is available only if g_r^{in}, g_r^{out} are fixed. Otherwise, the alternate iterative approach is considered (and provide satisfying segmentation results):

Repeat until convergence:

1. Fix u (representing C) and update $h_r^{in}(C_{in}),\ h_r^{in}(C_{out})$ (C_{in} is represented by $u > 0$ and C_{out} is represented by $u < 0$).

2. Fix $h_r^{in},\ h_r^{out}$ and update u with the iterative minimization scheme:

$$\begin{cases} (u^{k+1}, d^{k+1}) &= \arg\min_{u\in[0,1],d} \int_\Omega g_b|d| + \lambda h_r u + \frac{\mu}{2}|d - \nabla u - b^k|^2\ dx \\ b^{k+1} &= b^k + \nabla u^{k+1} - d^{k+1} \end{cases}$$

Then, the final active contour is given by the boundary of the set:

$$\{x \in \Omega \mid u^{\text{final}}(x) > 0.5\}.$$

6 Application #1: Chan-Vese Model (Smooth/Non-Texture Images)

The Chan-Vese model [7] is defined as:

$$\min_C \Big\{F_{CV}(C) = \int_C ds + \lambda \int_{C_{in}} (\mu_{in} - I(x))^2 dx + \lambda \int_{C_{out}} (\mu_{out} - I(x))^2 dx\Big\}, \tag{6.1}$$

where

$$\mu_{in} = \frac{\int_{C_{in}} I(x)dx}{\int_{C_{in}} dx}, \quad \mu_{out} = \frac{\int_{C_{out}} I(x)dx}{\int_{C_{out}} dx}.$$

The calculus of variations and the SDT provides the gradient flow:

$$\partial_t C = \Big(\kappa(s) + (\mu_{in} - I(s))^2 - (\mu_{out} - I(s))^2\Big)\mathcal{N} \tag{6.2}$$

The alternate iterative approach to solve the Chan-Vese model is:

Repeat until convergence:

1. Fix u and update $\mu_{in},\ \mu_{out},\ h_r^{in},\ h_r^{out}$ as follows:

$$\begin{array}{ll} \mu_{in} = \frac{\int_\Omega I(x)u(x)dx}{\int_\Omega u(x)dx}, & \mu_{out} = \frac{\int_\Omega I(x)(1-u(x))dx}{\int_\Omega 1-u(x)dx}, \\ h_r^{in}(x) = (\mu_{in} - I(x))^2, & h_r^{out}(x) = (\mu_{out} - I(x))^2. \end{array}$$

2. Fix $h_r^{in},\ h_r^{out}$ and update u with the iterative minimization scheme:

$$\begin{cases} (u^{k+1}, d^{k+1}) &= \arg\min_{u\in[0,1],d} \int_\Omega |d| + \lambda h_r u + \frac{\mu}{2}|d - \nabla u - b^k|^2\ dx \\ b^{k+1} &= b^k + \nabla u^{k+1} - d^{k+1} \end{cases},$$

   where $h_r := h_r^{in} - h_r^{out}$.

Then, the final active contour is given by the boundary of the set:

$$\{x \in \Omega \mid u^{\text{final}}(x) > 0.5\}.$$

A code is available online, see http://www.math.ucla.edu/∼xbresson.

7 Application #2: Houhou-Thiran-Bresson Model (Textures)

The Houhou-Thiran-Bresson model [11] is defined as:

$$\min_C \Big\{ F_{HTB}(C) = \int_C ds + KL(p_{in}(C_{in}), q_{out}(C_{out})) \Big\}, \tag{7.1}$$

where KL is the Kullback-Leibler distance, which measures the distnace between intensity distributions, is defined as:

$$KL(p_{in}(C_{in}), q_{out}(C_{out})) =$$
$$\int_{\mathbb{R}_+} \Big(p_{in}(C_{in}, I) \tfrac{p_{in}(C_{in}, I)}{p_{out}(C_{out}, I)} \Big) + p_{out}(C_{out}, I) \log \tfrac{p_{out}(C_{out}, I)}{p_{in}(C_{in}, I)} \Big) dI.$$

and

$$p_{in}(C_{in}, I) = \tfrac{1}{|C_{in}|} \int_{C_{in}} G_\sigma(I - I(z)) dz$$
$$p_{out}(C_{out}, I) = \tfrac{1}{|C_{out}|} \int_{C_{out}} G_\sigma(I - I(z)) dz$$

where:

$$G_\sigma(I - I(z)) = e^{-\frac{(I - I(z))^2}{2\sigma^2}}.$$

The calculus of variations and the SDT provide the gradient flow (see [11] for more details):

$$\partial_t C = \Big(\kappa(s) + h_r^{in}(s) - h_r^{out}(s) \Big) \mathcal{N}, \tag{7.2}$$

where

$$h_r^{in}(s) = \tfrac{1}{|C_{in}|} \int_{\mathbb{R}_+} \big[1 - \tfrac{p_{out}(I)}{p_{in}(I)} + \log \tfrac{p_{in}(I)}{p_{out}(I)} \big] . \big[- G_\sigma(I - I(s)) + p_{in}(I) \big] dI$$
$$h_r^{out}(s) = \tfrac{1}{|C_{out}|} \int_{\mathbb{R}_+} \big[1 - \tfrac{p_{in}(I)}{p_{out}(I)} + \log \tfrac{p_{out}(I)}{p_{in}(I)} \big] . \big[- G_\sigma(I - I(s)) + p_{out}(I) \big] dI$$

The alternate iterative approach for the Houhou-Thiran-Bresson model is thus as follows:

Repeat until convergence.:

1. Fix u and update $p_{in},\ p_{out},\ h_r^{in},\ h_r^{out}$ as follows:

$$p_{in}(I) = \tfrac{1}{\int_\Omega u(x) dx} \int_\Omega G_\sigma(I - I(z)) u(z) dz$$
$$p_{out}(I) = \tfrac{1}{\int_\Omega 1 - u(x) dx} \int_\Omega G_\sigma(I - I(z)) (1 - u(z)) dz$$

   and

$$h_r^{in}(x) = \tfrac{1}{\int_\Omega u(x) dx} \int_{\mathbb{R}_+} \big[1 - \tfrac{p_{out}(I)}{p_{in}(I)} + \log \tfrac{p_{in}(I)}{p_{out}(I)} \big] . \big[- G_\sigma(I - I(x)) + p_{in}(I) \big] dI$$
$$h_r^{out}(x) = \tfrac{1}{\int_\Omega 1 - u(x) dx} \int_{\mathbb{R}_+} \big[1 - \tfrac{p_{in}(I)}{p_{out}(I)} + \log \tfrac{p_{out}(I)}{p_{in}(I)} \big] . \big[- G_\sigma(I - I(x)) + p_{out}(I) \big] dI$$

2. Fix $h_r^{in},\ h_r^{out}$ and update u with the iterative minimization scheme:

$$\begin{cases} (u^{k+1}, d^{k+1}) &= \arg\min_{u \in [0,1], d} \int_\Omega |d| + \lambda h_r u + \frac{\mu}{2} |d - \nabla u - b^k|^2 \, dx \\ b^{k+1} &= b^k + \nabla u^{k+1} - d^{k+1} \end{cases},$$

   where $h_r := h_r^{in} - h_r^{out}$.

Then, the final active contour is given by the boundary of the set:

$$\{ x \in \Omega \mid u^{\text{final}}(x) > 0.5 \}.$$

A code is available online, see http://www.math.ucla.edu/∼xbresson.

8 Comparison with Graph-Cuts

We compare our algorithm with the popular fast graph-cuts technique [2, 9]. We show that our model produces slightly faster results tahn graph-cuts, while being easier to code. Besides, our algorithm is more accurate than graph-cuts because it uses isotropic schemes to regularize the contour (which produce smoother contours than graph-cuts, see Figure 2 and 3). Our algorithm is also sub-pixel accurate, unlike graph-cuts. Finally, the memory requirement is low (specially for 3D images).

(a) Graph-Cuts. Final Contour.

(b) Graph-Cuts. Final u.

(c) Our Algorithm. Final Contour.

(d) Our Algorithm. Final Contour.

Figure 2: Comparison Graph-Cuts vs Our Algorithm. Image size is 253×256 (The presented results are zoomed in). Computational time for Graph-Cuts is 0.2 seconds and for our Algorithm is 0.07 seconds. Our algorithm is not only faster than Graph-Cuts. It is also more accurate because it uses isotropic schemes and is sub-pixel accurate (compare the smoothness of contours). Finally, the memory allocation is low.

9 Experiments

See segmentation results with Figures 4, 5, 6, 7 and 8 for smooth/non-texture images and with Figures 9, 10, 11 and 12 for texture images.

10 Make Your Active Contour Model Fast!

We have emphasized in the code, available online at http://www.math.ucla.edu/∼xbresson where you can add your active contour model. Enjoy!

References

[1] G. Aubert, M. Barlaud, O. Faugeras, and S. Jehan-Besson. Image Segmentation Using Active Contours: Calculus of Variations or Shape Gradients? *SIAM Journal on Applied Mathematics*, 63(6):2128–2154, 2003.

[2] Y. Boykov and V. Kolmogorov. An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.

[3] L. Bregman. The Relaxation Method of Finding the Common Points of Convex Sets and Its Application to the Solution of Problems in Convex Programming. *USSR Computational Mathematics and Mathematical Physics*, 7:200–217, 1967.

[4] X. Bresson, S. Esedoglu, P. Vandergheynst, J. Thiran, and S. Osher. Fast Global Minimization of the Active Contour/Snake Models. *Journal of Mathematical Imaging and Vision*, 28(2):151–167, 2007.

[5] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic Active Contours. *International Journal of Computer Vision*, 22(1):61–79, 1997.

[6] T.F. Chan, S. Esedoglu, and M. Nikolova. Algorithms for Finding Global Minimizers of Image Segmentation and Denoising Models. *SIAM Journal on Applied Mathematics*, 66(5):1632–1648, 2006.

[7] T.F. Chan and L.A. Vese. Active Contours Without Edges. *IEEE Transactions on Image Processing*, 10(2):266–277, 2001.

[8] M.C. Delfour and J.P. Zolesio. *Shapes and Geometries: Analysis, Differential Calculus, and Optimization*. Society for Industrial and Applied Mathematics (SIAM), 2001.

[9] N. El-Zehiry, S. Xu, P. Sahoo, and A. Elmaghraby. Graph Cut Optimization for the Mumford-Shah Model. In *Visualization, Imaging, and Image Processing*, 2007.

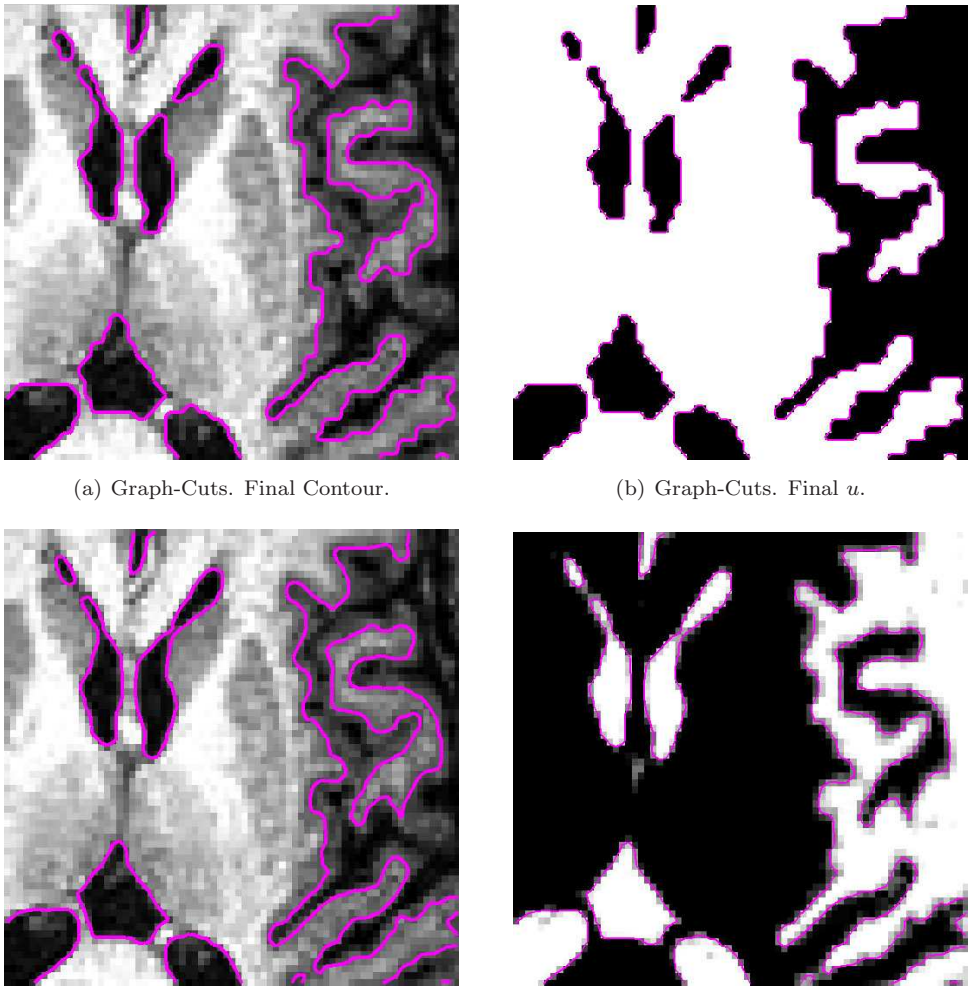
[10] T. Goldstein, X. Bresson, and S. Osher. Geometric Applications of the Split Bregman Method: Segmentation and Surface Reconstruction. *CAM Report 09-06*, 2009.

[11] N. Houhou, J-P. Thiran, and X. Bresson. Fast Texture Segmentation based on Semi-Local Region Descriptor and Active Contour. *Submitted*, 2009.

[12] S. Jehan-Besson, M. Barlaud, and G. Aubert. DREAM2S: Deformable Regions Driven by an Eulerian Accurate Minimization Method for Image and Video Segmentation. *International Journal of Computer Vision*, 53(1):45–70, 2003.

[13] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin. An Iterative Regularization Method for Total Variation-based Image Restoration. *SIAM Multiscale Modeling and Simulation*, 4:460–489, 2005.

[14] S. Osher and J.A. Sethian. Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations. *Journal of Computational Physics*, 79(1)(12–49), 1988.



(a) Graph-Cuts. Final Contour.

(b) Graph-Cuts. Final u.

(c) Our Algorithm. Final Contour.

(d) Our Algorithm. Final Contour.

Figure 3: Comparison Graph-Cuts vs Our Algorithm. Image size is 210×210 (The presented results are zoomed in). Computational time for Graph-Cuts is 0.1 seconds and for our Algorithm is 0.08 seconds. Our algorithm is not only faster than Graph-Cuts. It is also more accurate because it uses isotropic schemes and is sub-pixel accurate (compare the smoothness of contours). Finally, the memory allocation is low.

Figure 4: Chan-Vese Model (Smooth/Non-Texture Images). The image size is 253×256. Computational time is 0.07 seconds.

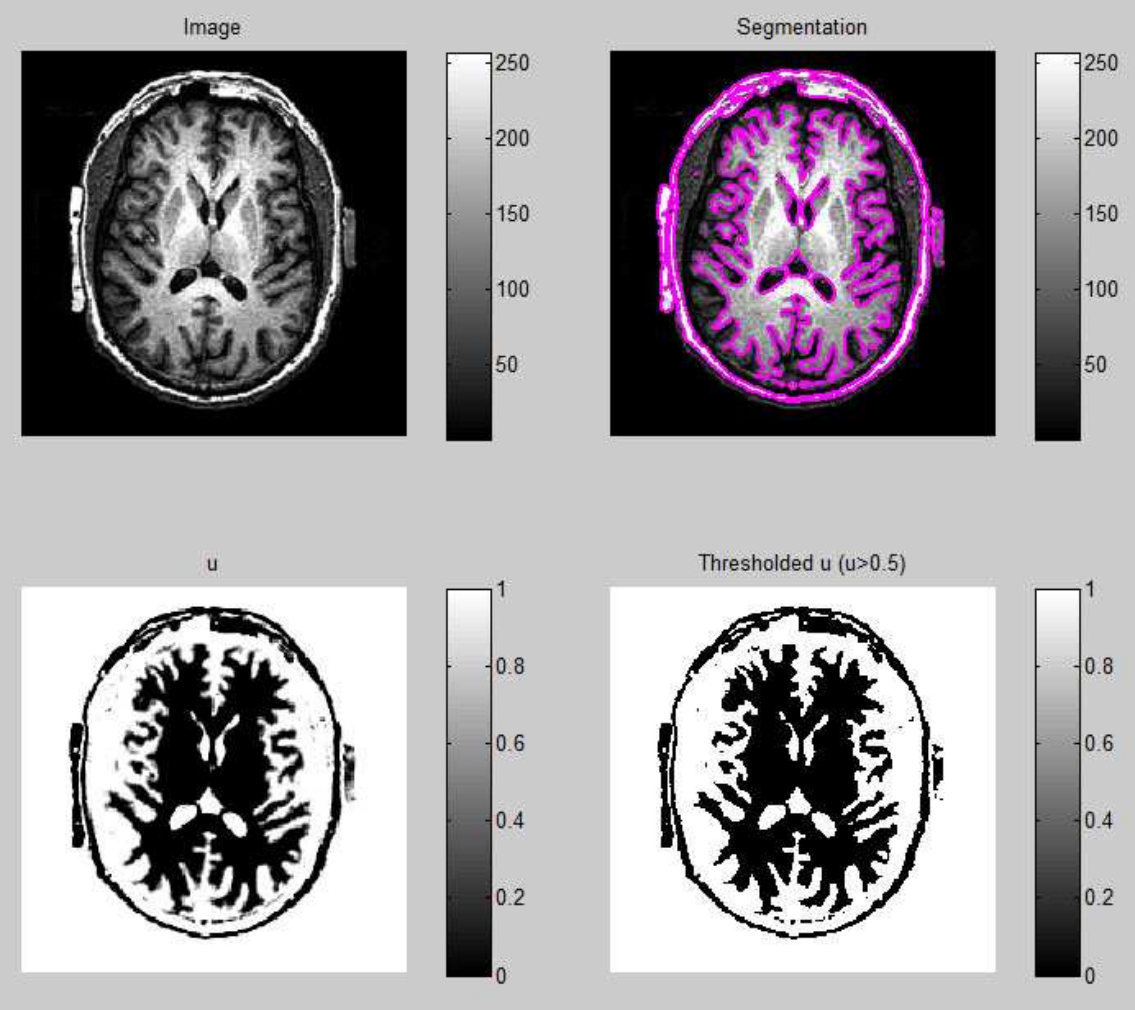


Figure 5: Chan-Vese Model (Smooth/Non-Texture Images). The image size is 210×210. Computational time is 0.08 seconds.

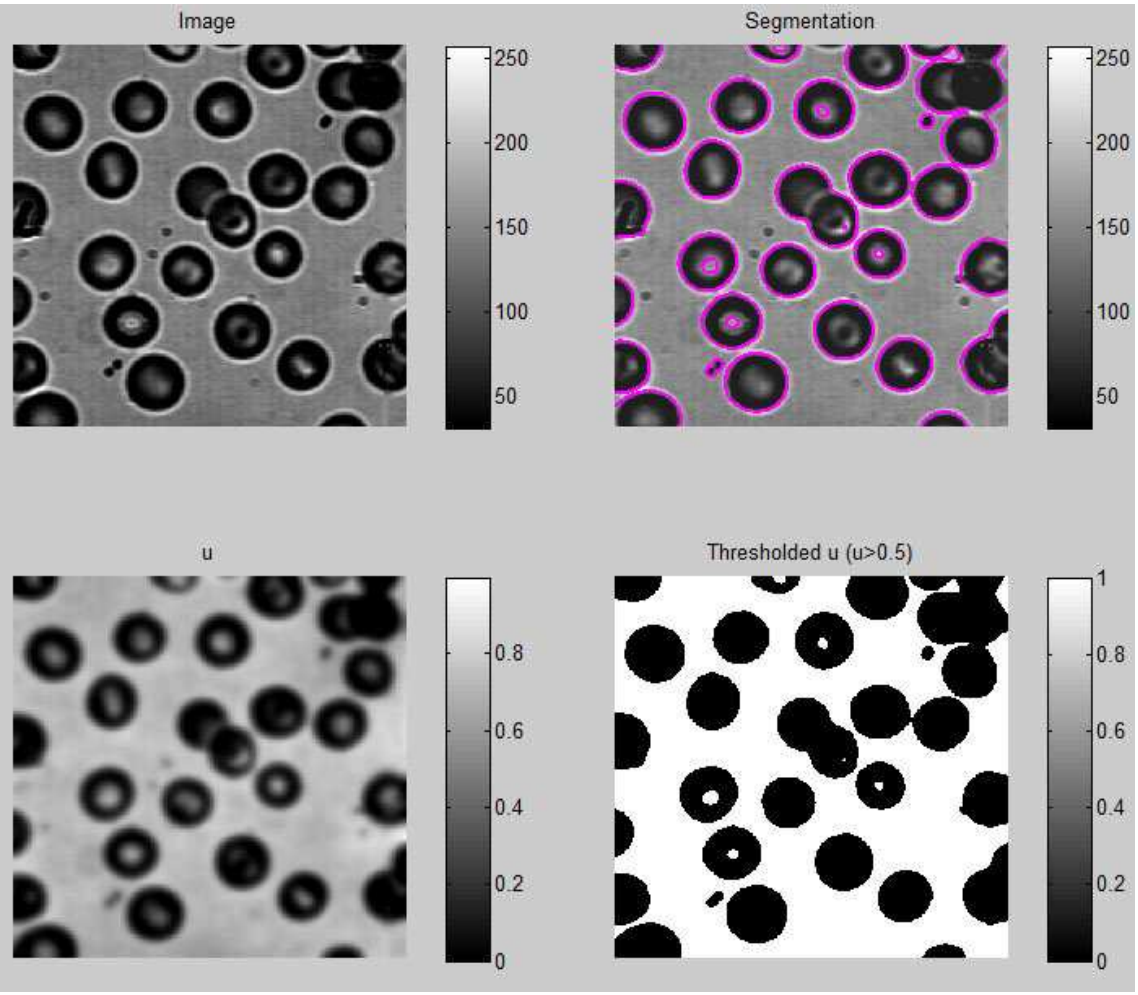


Figure 6: Chan-Vese Model (Smooth/Non-Texture Images). The image size is 380×391. Computational time is 0.9 seconds.

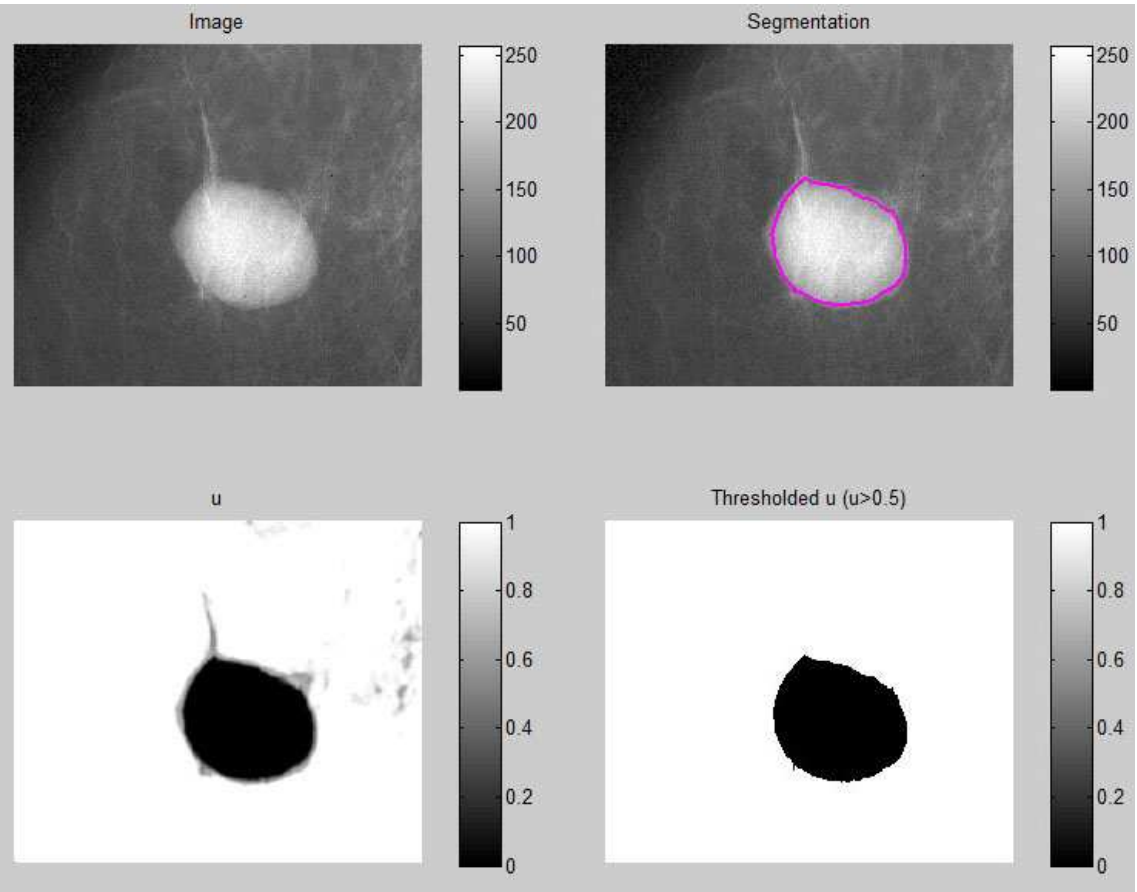


Figure 7: Chan-Vese Model (Smooth/Non-Texture Images). The image size is 403×481. Computational time is 2.9 seconds.

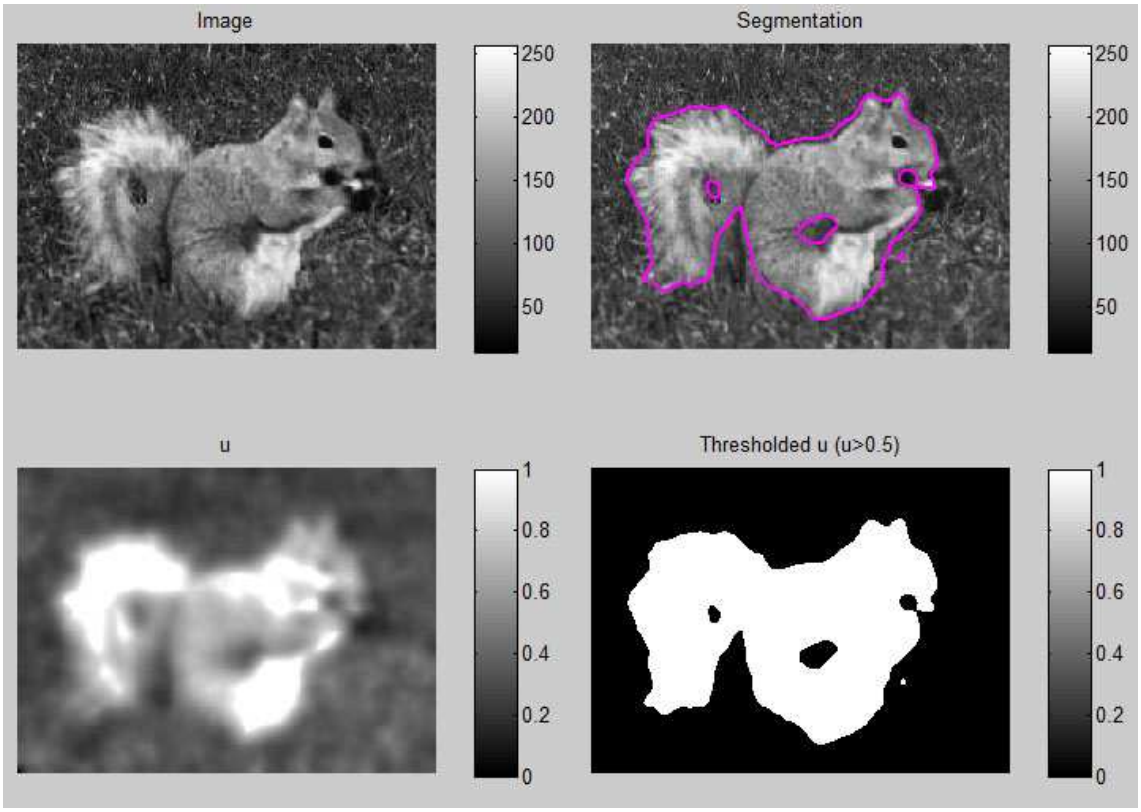


Figure 8: Chan-Vese Model (Smooth/Non-Texture Images). The image size is 209×288. Computational time is 1.1 seconds.

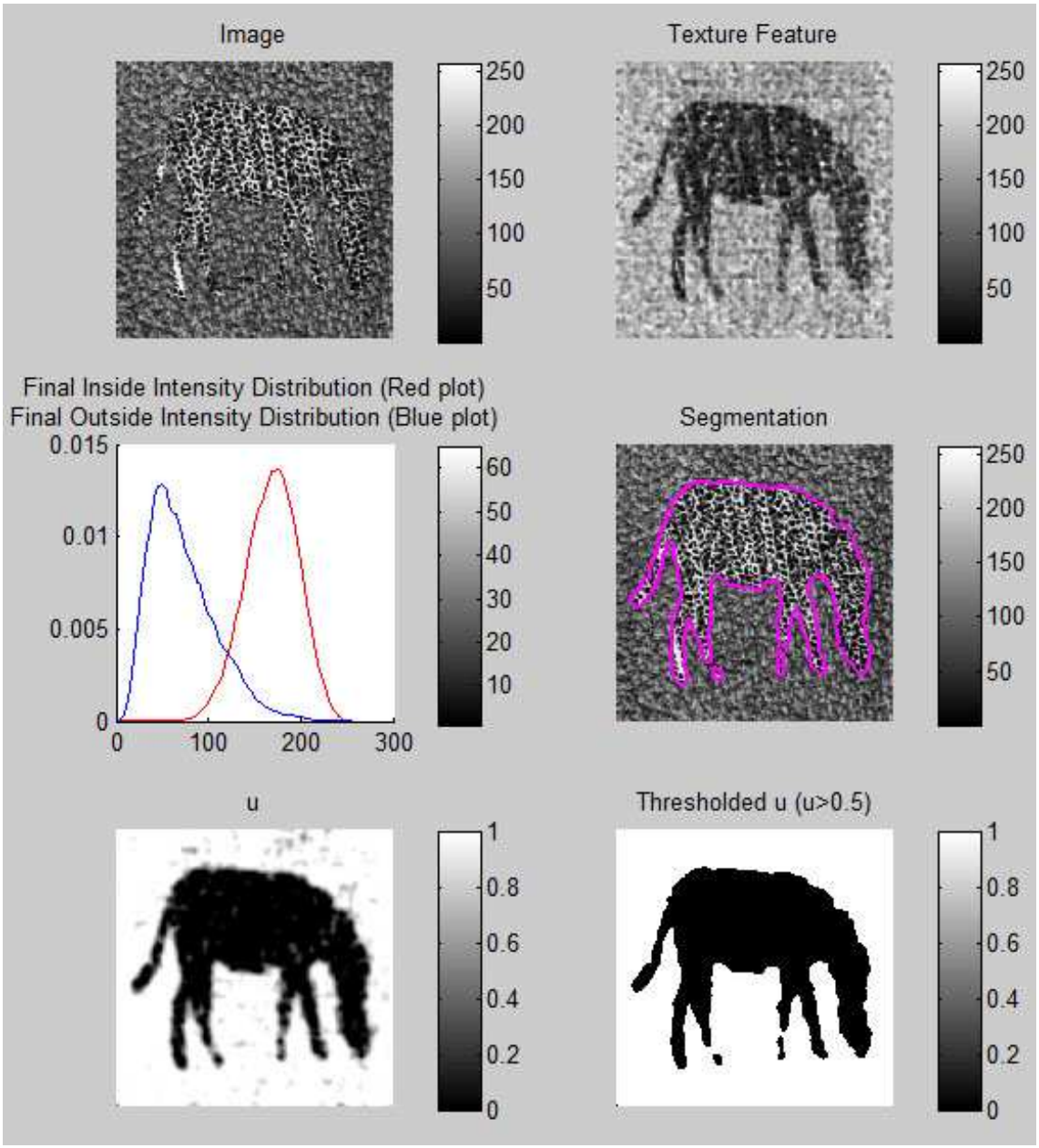


Figure 9: Houhou-Thiran-Bresson Model (Textures). The image size is 373×374. Computational time is 3.6 seconds.

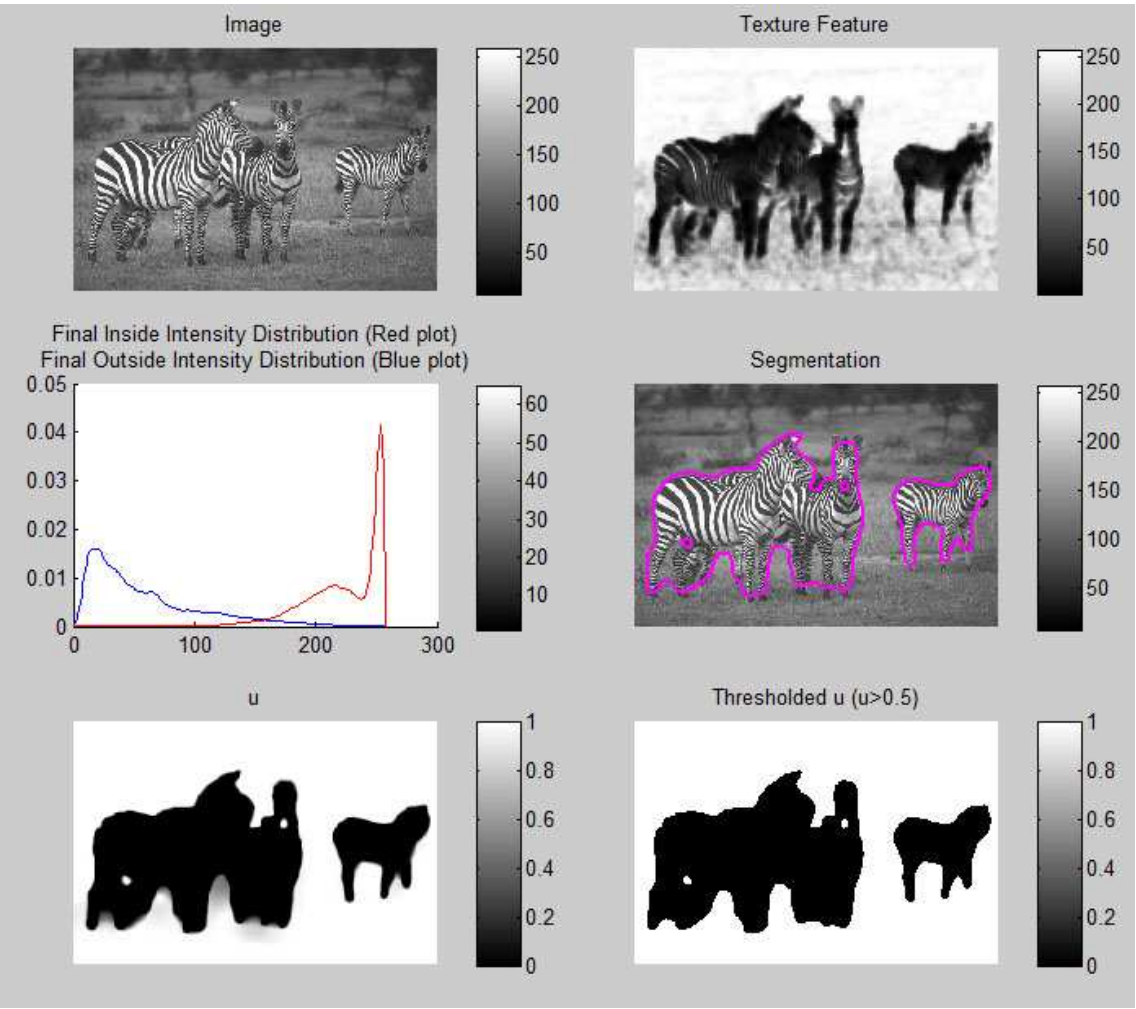


Figure 10: Houhou-Thiran-Bresson Model (Textures). The image size is 321×481. Computational time is 4.5 seconds.

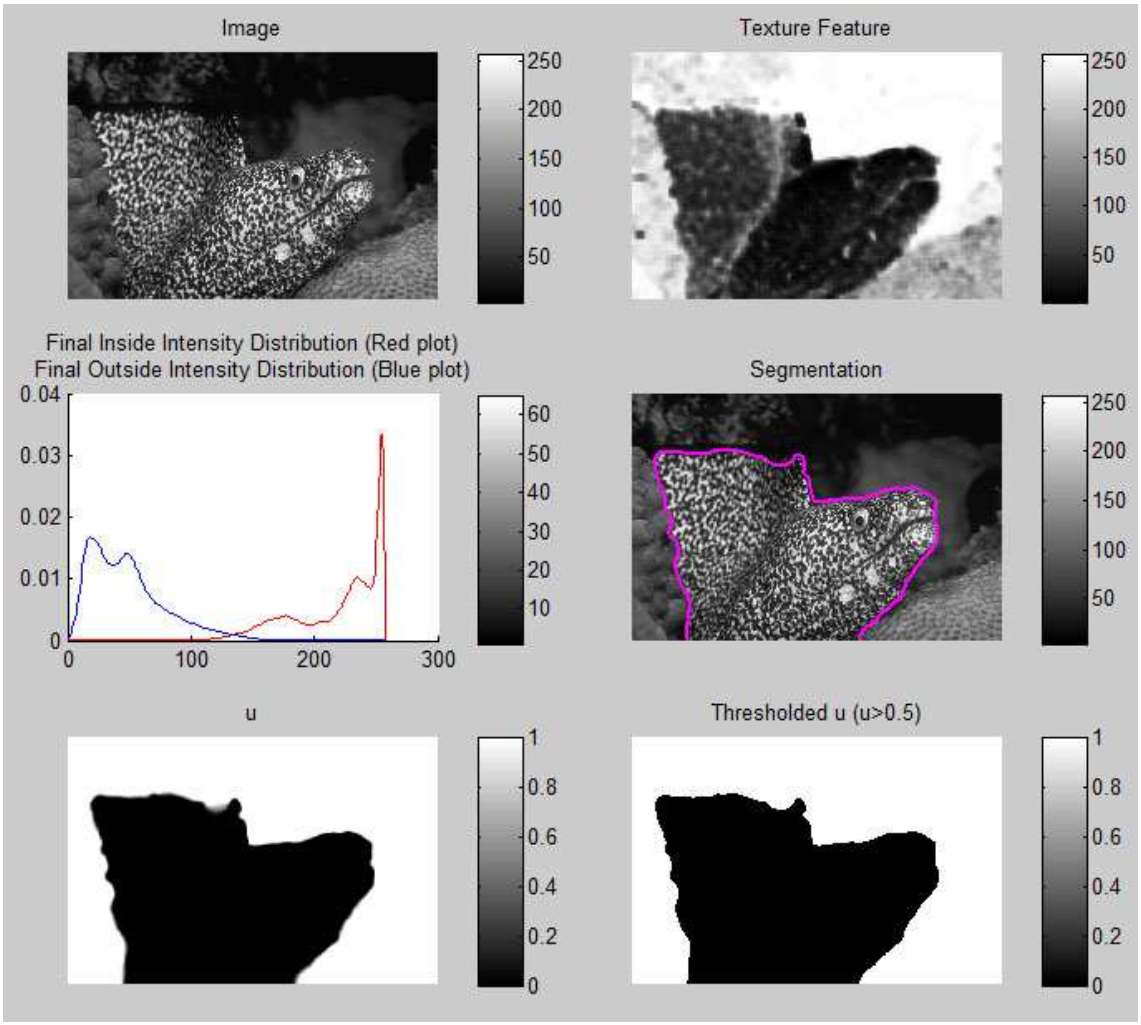


Figure 11: Houhou-Thiran-Bresson Model (Textures). The image size is 321×481. Computational time is 3.2 seconds.

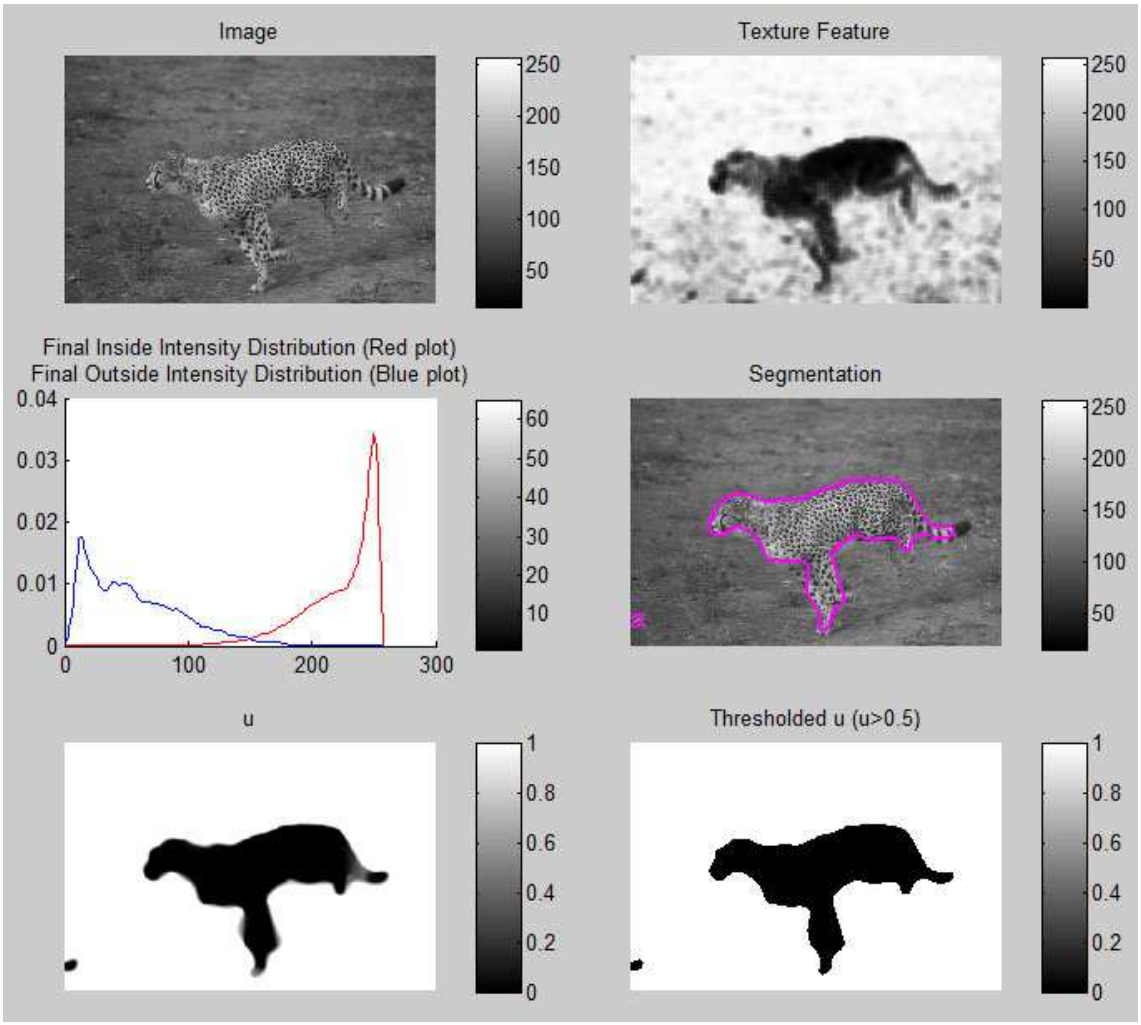


Figure 12: Houhou-Thiran-Bresson Model (Textures). The image size is 321×481. Computational time is 2.8 seconds.