An Exercise in Image Processing with Geodesic Active Contours in MATLAB

Jeremy Lerner
Department of Applied Mathematics and Statistics
Stony Brook University

October 14, 2014

Contents

Part I
Background

1 Equations

An image, when represented as a matrix of pixel values I, has edges where the gradient is large, that is $\|\nabla I\| >> 1$. A naive approach to finding edges would be to simply declare the locations with a large gradient to be an edge. However, for noisy images, this will give far too many false positives, because when one pixel doesn't follow the pattern of those around it, the gradient is also large. A more accurate method of locating the edges involves using the inverse of the gradient, as in equation 1. Equation 2 shows the relationship between $\phi(i,j)$ and the proximity of the point (i,j) to an edge.

$$\phi(i,j) = \frac{1}{1+\|\nabla I(i,j)\|^2} \tag{1}$$

$$\phi(i,j) = \begin{cases} \approx 0, & (i,j) \text{ is near an edge} \\ \approx 1, & (i,j) \text{ is far from an edge} \end{cases} \tag{2}$$

Level set methods, as introduced in [1], involve operating on the image with nonlinear heat equation, equation 3, with initial condition that has a zero level set (that is, where the function value is zero) on the outer edge of the image, until the zero level set converges, is a fairly safe method of finding edges. One such initial condition is given by equation 4, where the image has m rows and n columns of pixels and $C \in \mathbb{R}$ is chosen appropriately such that the zero level set is on the outside of any edges. It is helpful to note that the image, I, is only involved in the differential equation via its relationship with ϕ, as given in equation 1.

$$\frac{\partial \Phi}{\partial t} = \phi \cdot \operatorname{div}\left(\frac{\nabla \Phi}{\|\nabla \Phi\|}\right)\|\nabla \Phi\| \tag{3}$$

$$\Phi(t=0)_{i,j} = \left(i - \frac{m}{2}\right)^2 + \left(j - \frac{n}{2}\right)^2 - C \tag{4}$$

While equation 3 will eventually find all the edges of an image, large curvature will cause the process to slow down, almost to the point of stopping. For extremely simple images, this can be tolerable, though the number of iterations necessary to find the edges of an image can be quite large. For complicated images, the number of iterations necessary to find edges is astonomically large, and the change in the zero level set between iterations is extremely small, making the numerical calculations difficult, if not impossible.

There are three helpful fixes to avoid problems with equation 3. Firstly, adding the term $\phi \cdot \|\nabla \Phi\| \nu$ to the equation pushes the shrinking zero level set faster, giving earlier convergence. Secondly, taking the divergence of $\frac{\nabla \Phi}{\sqrt{1+\|\Phi\|^2}}$,

instead of $\frac{\nabla\Phi}{\|\nabla\Phi\|}$, avoids division by small numbers when $\|\nabla\Phi\|$ is near zero. Lastly, the term $\nabla\phi \cdot \nabla\Phi$ can be added to slow the zero level set down when it is near edges and to push it back out if it goes inside a shape. With all of these fixes, the new differential equation is given by equation 5.

$$\frac{\partial\Phi}{\partial t} = \phi \cdot \operatorname{div}\left(\frac{\nabla\Phi}{\sqrt{1+\|\nabla\Phi\|^2}}\right)\|\nabla\Phi\| + \phi\,\|\nabla\Phi\|\,\nu + \nabla\phi\cdot\nabla\Phi \tag{5}$$

The simplified form of equation 5 can be obtained through the steps in derivation 6.

$$\frac{\partial\Phi}{\partial t} = \phi \cdot \operatorname{div}\left(\frac{\nabla\Phi}{\sqrt{1+\|\nabla\Phi\|^2}}\right)\|\nabla\Phi\| + \phi\,\|\nabla\Phi\|\,\nu + \nabla\phi\cdot\nabla\Phi \tag{6}$$

$$\frac{\partial\Phi}{\partial t} = \phi \cdot \sqrt{\Phi_x^2+\Phi_y^2}\left[\operatorname{div}\left(\frac{\langle\Phi_x,\Phi_y\rangle}{\sqrt{1+\Phi_x^2+\Phi_y^2}}\right)\right]$$

$$+\nu\cdot\phi\cdot\sqrt{\Phi_x^2+\Phi_y^2}+\langle\phi_x,\phi_y\rangle\cdot\langle\Phi_x,\Phi_y\rangle$$

$$\frac{\partial\Phi}{\partial t} = \phi \cdot \sqrt{\Phi_x^2+\Phi_y^2}\left[\frac{\partial}{\partial x}\left(\frac{\Phi_x}{\sqrt{1+\Phi_x^2+\Phi_y^2}}\right)+\frac{\partial}{\partial y}\left(\frac{\Phi_y}{\sqrt{1+\Phi_x^2+\Phi_y^2}}\right)\right]$$

$$+\nu\cdot\phi\cdot\sqrt{\Phi_x^2+\Phi_y^2}+\langle\phi_x,\phi_y\rangle\cdot\langle\Phi_x,\Phi_y\rangle$$

$$\frac{\partial\Phi}{\partial t} = \phi \cdot \sqrt{\Phi_x^2+\Phi_y^2}\left[\left(\frac{\Phi_{xx}\left(1+\Phi_x^2+\Phi_y^2\right)^{\frac{1}{2}}-\Phi_x\left(1+\Phi_x^2+\Phi_y^2\right)^{-\frac{1}{2}}\left(\Phi_x\Phi_{xx}+\Phi_y\Phi_{xy}\right)}{1+\Phi_x^2+\Phi_y^2}\right)\right.$$

$$\left.+\left(\frac{\Phi_{yy}\left(1+\Phi_x^2+\Phi_y^2\right)^{\frac{1}{2}}-\Phi_y\left(1+\Phi_x^2+\Phi_y^2\right)^{-\frac{1}{2}}\left(\Phi_x\Phi_{xy}+\Phi_y\Phi_{yy}\right)}{1+\Phi_x^2+\Phi_y^2}\right)\right]$$

$$+\nu\cdot\phi\cdot\sqrt{\Phi_x^2+\Phi_y^2}+\langle\phi_x,\phi_y\rangle\cdot\langle\Phi_x,\Phi_y\rangle$$

$$\frac{\partial\Phi}{\partial t} = \phi\cdot\frac{\sqrt{\Phi_x^2+\Phi_y^2}}{\left(1+\Phi_x^2+\Phi_y^2\right)^{\frac{3}{2}}}\left[\Phi_{xx}+\Phi_{yy}+\Phi_y^2\Phi_{xx}\right.$$

$$\left.+\Phi_x^2\Phi_{yy}-2\Phi_x\Phi_y\Phi_{xy}\right]+\nu\cdot\phi\cdot\sqrt{\Phi_x^2+\Phi_y^2}+\phi_x\cdot\Phi_x+\phi_y\cdot\Phi_y$$

To find the edges, equation 5 is applied successively to the initial condition, $\Phi_0 = \Phi\,(t=0)$, as seen in figure 1, until $\frac{\partial\Phi}{\partial t} = 0$. Each successive iteration moves the zero level set, where $\Phi = 0$, inward towards the edges. Once the zero level set reaches an edge, it stops moving, because for an edge located at the row index i and column index j, $\phi_{i,j} \approx 0$, so equation 7 holds. Thus,

at each step, the program checks if the surface has stopped moving and stops iterating once it stops moving everywhere.

$$\Phi_{i,j}^{n+1} = \Phi_{i,j}^{n} + \Delta t \cdot \phi_{i,j} \cdot \operatorname{div}\left(\frac{\nabla \Phi_{i,j}}{\sqrt{1 + \|\nabla \Phi_{i,j}\|^2}}\right) \|\nabla \Phi_{i,j}\| \tag{7}$$
$$+ \phi_{i,j} \|\nabla \Phi_{i,j}\| \nu + \Delta t \cdot (\nabla \phi \cdot \nabla \Phi)_{i,j} = \Phi_{i,j}^{n} + \epsilon$$
$$\epsilon \sim 10^{-5}$$

The MATLAB command C = contourc(I, [0,0]) gives the coordinates of the zero level set as a matrix, where the first row contains row values and the second row contains the corresponding column values. As shown in figure 1, the initial condition has a zero level set (highlighted in red) around the edges of the image, equation 5 then moves the zero level set inwards towards the edges of the image. Figure 2 shows the same zero level set on the $z = 0$ plane, as the original snake. This snake contracts towards, and stops at, the edges. As a side note, this is only one possible initial condition; many possibilities exist. An initial condition should work as long as the zero level set is outside the edges of the image and the surface is continuous.

Figure 1: The Initial Condition
Zero level set in red

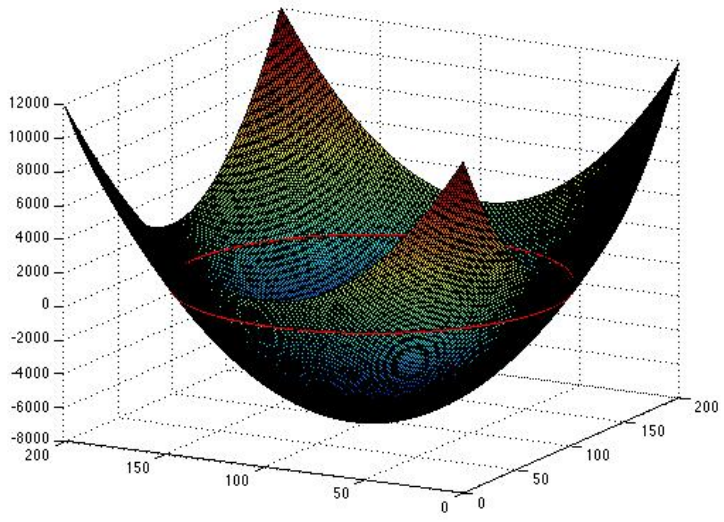
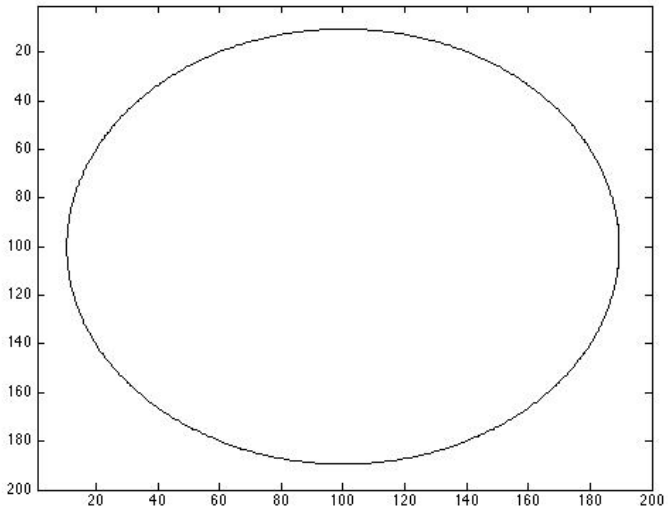


Figure 2: The Initial Snake/Zero Level Set



2 Numerics

2.1 Redistancing and Rebuilding

After several iterations, $\|\nabla\Phi\|$ can become too large or too small, causing inaccurate numerical approximations. Therefore, this algorithm requires redistancing with a signed distance function every few iterations to keep $\|\nabla\Phi\|$ under control. Redistancing keeps the zero level set in the same position and adjusts the magnitude of the elements in Φ. One redistancing function is given by equation 8.

$$\begin{aligned}
&\Phi_{i,j}^{redist} = \operatorname{sign}(\Phi_{i,j}) \cdot \operatorname{bwdist}(\Phi_{i,j}) \\
&\operatorname{sign}(\Phi_{i,j}) = \begin{cases} -1 & \text{if } \Phi_{i,j} < 0 \\ 0 & \text{if } \Phi_{i,j} = 0 \\ 1 & \text{if } \Phi_{i,j} > 0 \end{cases} \\
&\operatorname{bwdist}(\Phi_{i,j}) = \sqrt{(i-i_0)^2 + (j-j_0)^2} \\
&\Phi_{i_0,j_0} = \text{nearest nonzero pixel}
\end{aligned} \tag{8}$$

However, to keep the number of iterations small, Φ was rebuilt, using the following code, every five iterations. Rebuilding better guaranteed $\|\nabla\Phi\| = 1$, despite the larger computational cost.

```
mask_in = (Phi <= 0);
mask_out = 1-mask_in;
u = double(double(bwdist(mask_in)) +
    (1-double(bwdist(mask_out))).*mask_in));
```

2.2 Upwinding

To numerically solve equation 5, we use finite differences. Specifically, for the parabolic term, $\phi \cdot \operatorname{div}\left(\frac{\nabla\Phi}{\sqrt{1+\|\nabla\Phi\|^2}}\right)\|\nabla\Phi\|$, we use standard second order centered differences in the interior of the image and second order accurate one sided differences on the boundaries. However, the hyperbolic terms, $\phi\,\|\nabla\Phi\|\,\nu$ and $\nabla\phi\cdot\nabla\Phi$, must be upwinded in order to conserve entropy and obtain a physically relevant solution, as given in equations 9 and 10 respectively (where some terms are defined in equation 11), as discussed in [2] and [3].

$$\begin{aligned}\phi\,\|\nabla\Phi\|\,\nu \approx \nu\cdot\phi\Big[&\max\left(D_x^+\left\{\Phi_{i,j}\right\},0\right)^2+\min\left(D_x^-\left\{\Phi_{i,j}\right\},0\right)^2\\&+\max\left(D_y^+\left\{\Phi_{i,j}\right\},0\right)^2+\min\left(D_y^-\left\{\Phi_{i,j}\right\},0\right)^2\Big]^{\frac{1}{2}}\end{aligned} \tag{9}$$

$$\begin{aligned}\nabla\phi\cdot\nabla\Phi \approx -\big[&\max\left(-\phi_x,0\right)\cdot D_x^-\left\{\Phi_{i,j}\right\}+\min\left(-\phi_x,0\right)D_x^+\left\{\Phi_{i,j}\right\}\\&+\max\left(-\phi_y,0\right)D_y^-\left\{\Phi_{i,j}\right\}+\min\left(-\phi_y,0\right)D_y^+\left\{\Phi_{i,j}\right\}\big]\end{aligned} \tag{10}$$

$$\begin{aligned}D_x^+\left\{\Phi_{i,j}\right\}&=\Phi_{i,j+1}-\Phi_{i,j}\\D_x^-\left\{\Phi_{i,j}\right\}&=\Phi_{i,j}-\Phi_{i,j-1}\\D_y^+\left\{\Phi_{i,j}\right\}&=\Phi_{i+1,j}-\Phi_{i,j}\\D_y^-\left\{\Phi_{i,j}\right\}&=\Phi_{i,j}-\Phi_{i-1,j}\end{aligned} \tag{11}$$

Using a forward difference for time and $\Delta t = 0.05$, the image at time step $n+1$, Φ^{n+1}, can be estimated by equation 12, where Φ_x indicates a centered differences for Φ in the x direction, similarly for Φ_y, and where Φ_{xx} is the second derivative in the x direction, calculated using a centered difference in the x direction, similarly for Φ_{xy} and Φ_{yy}.

$$\begin{aligned}\Phi^{n+1}=\Phi^n&+\Delta t\cdot\phi\cdot\sqrt{\Phi_x^2+\Phi_y^2}\left[\frac{1}{\left(1+\Phi_x^2+\Phi_y^2\right)^{\frac{3}{2}}}\left(\Phi_{xx}+\Phi_{yy}+\Phi_y^2\Phi_{xx}\right.\right.\\&\left.\left.+\Phi_x^2\Phi_{yy}-2\Phi_x\Phi_y\Phi_{xy}\right)\right]\\&+\Delta t\cdot\nu\cdot\phi\Big[\max\left(D_x^+\left\{\Phi_{i,j}\right\},0\right)^2+\min\left(D_x^-\left\{\Phi_{i,j}\right\},0\right)^2\\&+\max\left(D_y^+\left\{\Phi_{i,j}\right\},0\right)^2+\min\left(D_y^-\left\{\Phi_{i,j}\right\},0\right)^2\Big]^{\frac{1}{2}}\\&-\Delta t\cdot\big[\max\left(-\phi_x,0\right)\cdot D_x^-\left\{\Phi_{i,j}\right\}+\min\left(-\phi_x,0\right)D_x^+\left\{\Phi_{i,j}\right\}\\&+\max\left(-\phi_y,0\right)D_y^-\left\{\Phi_{i,j}\right\}+\min\left(-\phi_y,0\right)D_y^+\left\{\Phi_{i,j}\right\}\big]\end{aligned} \tag{12}$$

Part II
Binary Images

3 Simple Geometric Shapes

For simple shapes, the zero level set evolves rather quickly, and when it converges, the values inside the shape are negative, values outside are positive, leaving the zero level set exactly at the edges. For the simple shapes in figures 3 and 5, the green lines around the edges represent the zero level set after the surface stops evolving, as do the red lines on the surfaces, in figures 4 and 6. For all the results below, $\nu = 10$.

Figure 3: Zero Level Set, 114 iterations

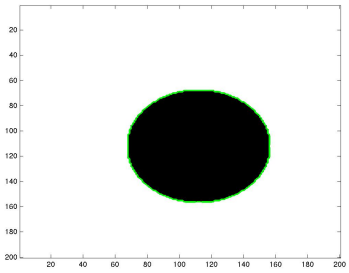


Figure 4: Entire Contour, 114 iterations

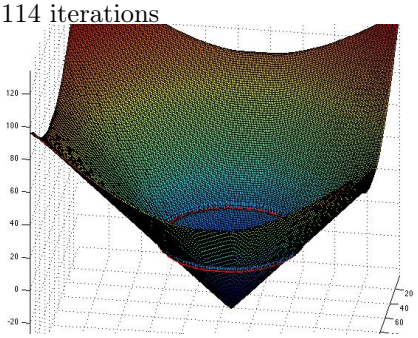


Figure 5: Zero Level Set, 43 iterations

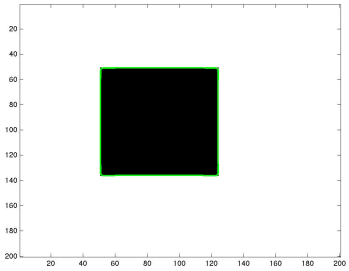
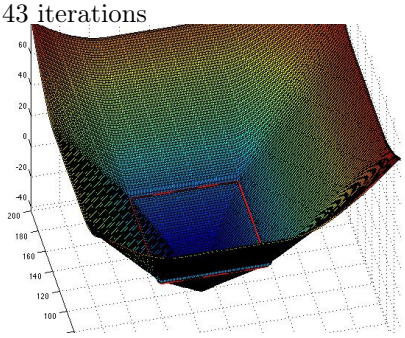


Figure 6: Entire Contour, 43 iterations



4 Complicated Shapes

For complicated shapes, the algorithm requires more iterations to converge, generally because the large curvature slows down the zero level set's evolution. Figures 7 and 8 show the final zero level set when the image has two shapes, and figures 9, 10, 11 and 12 show the final zero level set for a few complicated shapes.

Figure 7: Zero Level Set, 144 iterations

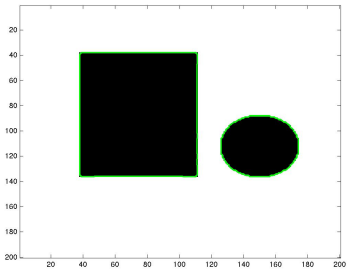


Figure 8: Zero Level Set, 149 iterations

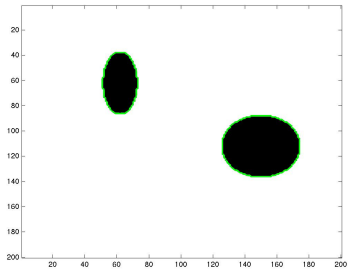


Figure 9: Zero Level Set, 139 iterations

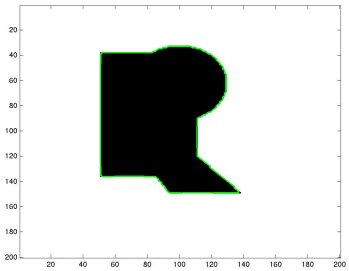


Figure 10: Zero Level Set, 134 iterations

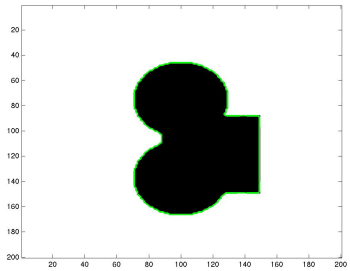


Figure 11: Zero Level Set, 164 iterations

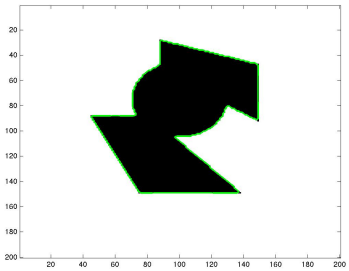
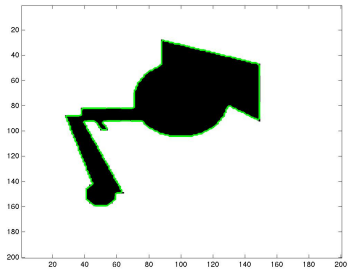


Figure 12: Zero Level Set, 174 iterations



5 Shells

Figures 13 and 14 show the zero level set after the algorithm converges for two binary images of shells, one thin and one thick. Note that the thin shell has edges two pixels wide and the thick shell has edges five pixels wide. The zero level set ends just inside the outermost edge for both shells.

Figure 13: Zero Level Set,
154 iterations
Thin shell

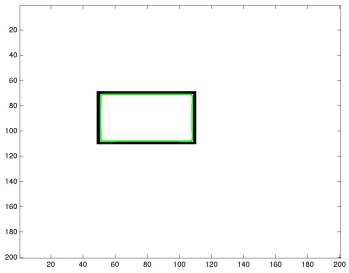
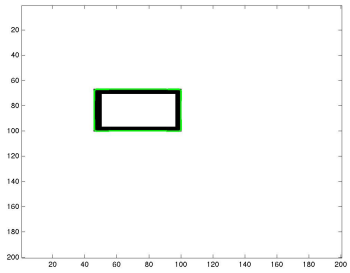


Figure 14: Zero Level Set,
159 iterations
Thick shell



Part III
Grayscale Images

6 Simple Geometric Shapes

It is more complicated to find edges of grayscale images than it is for binary images. Because the change from white to gray pixels causes a nonzero gradient, some pixels can cause the zero level set to slow down or stop entirely. Because of this behavior, the zero level set takes more iterations to converge to the true edges. A binary image with a circle takes 114 iterations to converge to the edges, but when gray pixels are added around those edges, it takes about 324 iterations to converge to the sharp edges. As figures 15 and 16 show, the algorithm converges to the sharp edges of the image, the solid black shapes.

Figure 15: Zero Level Set, 334 iterations

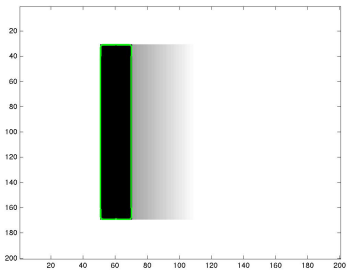
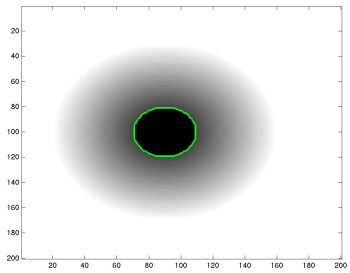


Figure 16: Zero Level Set, 324 iterations



7 Noisy Images

It is worth noting that this algorithm is often used with smoothing. That is, instead of ϕ simply calculated as in equation 1 above, the original image is first smoothed using a Guassian filter or heat equation, giving $\hat{I}$, as used in equation 13. This can reduce the gradient near gray pixels, causing faster convergence. However, smoothing can also blur the true edges, causing the zero level set converge somewhere near the edge, not exactly at the edge.

$$\phi(i,j) = \frac{1}{1 + \left\|\nabla \hat{I}\right\|^2} \tag{13}$$

7.1 Unsmoothed Noisy Images

Without first smoothing a noisy image, it is still possible to achieve positive results. Adding noise causes deceptively small values for ϕ far away from true edges, as seen in figures 19 and 20. Even non noisy grayscale images can have deceptively small values of ϕ, as seen in figure 18. Figures 21, 22, 23 and 24 show the algorithm operating on slightly noisy images.

Figure 17: ϕ, for a non noisy image, as seen in figure 7

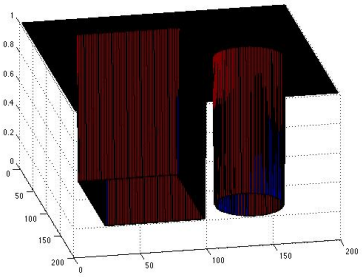


Figure 18: ϕ for a non noisy image, as seen in figure 16

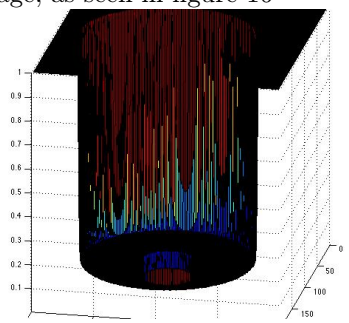


Figure 19: ϕ, for a noisy image, as seen in figure 22

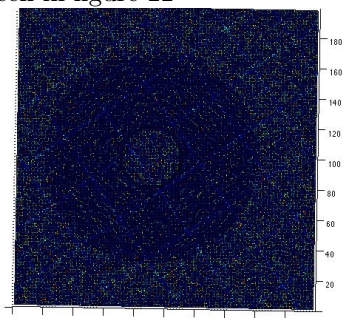


Figure 20: ϕ, for a noisy image, as seen in figure 22

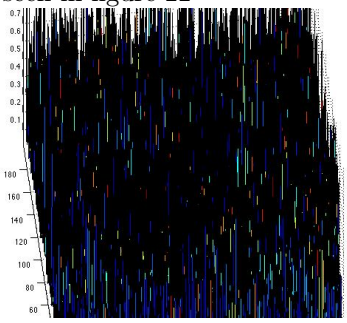


Figure 21: Zero Level Set, 439 iterations

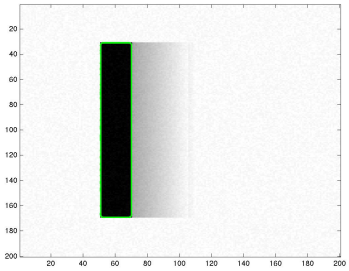


Figure 22: Zero Level Set, 389 iterations

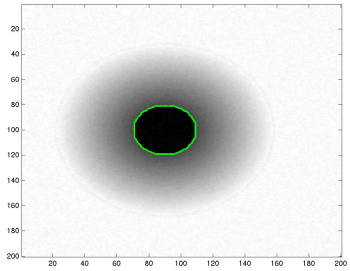


Figure 23: Zero Level Set, 122 iterations

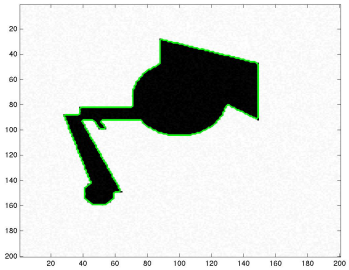
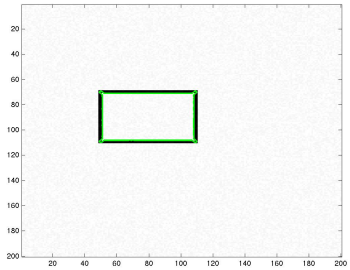


Figure 24: Zero Level Set, 96 iterations



7.2 Smoothed Noisy Images

When first smoothing an image, the edges become more pronounced. Figure 25 was smoothed using a nonlinear heat equation, equation 14, running to time $t_{end} = 2$ with $\Delta t = 0.1$, resulting in figure 26, with ϕ, as defined in equations 1 and 2, given by figure 27. The detected edges given by figure 28.

$$\frac{\partial U}{\partial t} = \operatorname{div}\left(\frac{\nabla U}{\|\nabla U\|}\right) \cdot \|\nabla U\| \tag{14}$$

Figure 25: Zero Level Set, 439 iterations

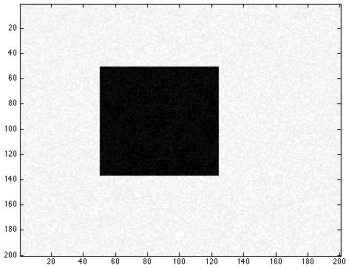


Figure 26: Zero Level Set, 389 iterations

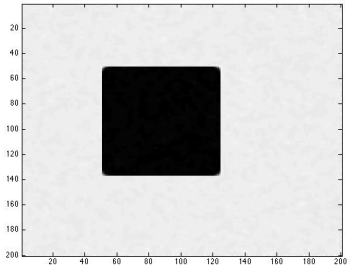


Figure 27: Zero Level Set, 122 iterations

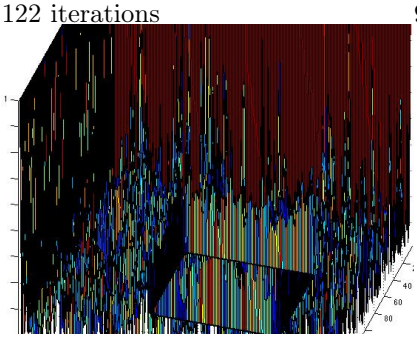
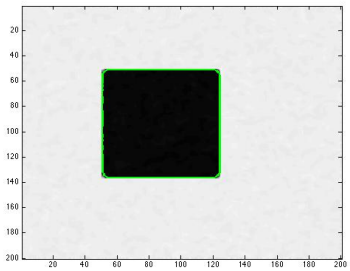


Figure 28: Zero Level Set, 96 iterations



References

[1] S. Osher and J. Sethian, "Fronts propogating with curvature dependent speed: Algorithms based on hamilton-jacobi formulations," *Journal of Computational Physics*, vol. 79, pp. 12–49, 1988.

[2] J. A. Sethian, "Adaptive fast marching and level set methods for propagating interfaces," *Acta Math. Univ. Comenianae*, vol. LXVII, pp. 3–15, 1998.

[3] ——, "Evolution, implementation, and application of level set and fast marching methods for advancing fronts," *Journal of Computational Physics*, vol. 169, pp. 503–555, 2001.

[4] D. L. Chopp, "Computing minimal surfaces via level set curvature flow," *Journal of Computational Physics*, vol. 106, pp. 77–91, 1993.

[5] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," *Interational Journal of Computer Vision*, vol. 22(1), pp. 61–79, 1997.

[6] R. Kimmel, N. Kiryati, and A. M. Bruckstein, "Analyzing and synthesizing images by evolving curves with the osher-sethian method," *Interational Journal of Computer Vision*, vol. 24(1), pp. 37–55, 1997.