Visualizing Neurons: A Comparative Analysis of PDE-Based Image Reconstruction

James Rosado

March 9, 2020

Midterm Project Report

I have currently implemented a numerical method that utilizes active contours (or snakes) [5], which is constrained by a given raw image $u_0(x)$, where $x \in \Omega \subset \mathbb{R}^2$ in 2D as an example. The snake models that are used [5] are based on

$$F_1(C) = \alpha \int_0^1 |C'(s)|^2 \, ds + \beta \int_0^1 |C''(s)| \, ds - \lambda \int_0^1 |\nabla u_0(C(s))|^2 \, ds$$

where the contour C that minimizes the above functional is the contour that realizes the image and this is analogous to a non-linear PDE. A more compact version [5] is given by

$$F_2(C) = \int_0^1 g(\nabla(u_0(s))) \, ds$$

which is analogous to

$$\phi_t = |\nabla \phi| g(\nabla u_0) \kappa + \nabla g(\nabla u_0) \cdot \nabla \phi \tag{1}$$

where ϕ are the isocontours, level sets, of our image u_0, this formulation is given by [6]. This is also a variation of the formulation for Geometric active contour [2], or geodesic active contour [4]. This method employs Euclidean curvature shortening evolution and the isocontours split and merge depending on the detection of objects in the image, for the detection I use Gaussian blurring. Below is the corresponding formulation:

$$\frac{\partial C}{\partial t} = g(I)(c + \kappa)\vec{N} - \langle \nabla g, \vec{N} \rangle \vec{N},$$

where κ is the curvature, g is the edge detector also called the halting function, c is a Lagrange multiplier, and $\vec{N}$ is the inward unit normal. For my initial attempt at the level set method I implemented the update rule initially with only the advection term and then I did an implementation with the advection and curvature term. Below is the update rule I utilize:

$$\frac{\phi_{i,j}^{n+1} - \phi_{i,j}^n}{k}$$
$$= \sqrt{\left(\frac{\phi_{i,j}^n - \phi_{i,j-1}^n}{h}\right)^2 + \left(\frac{\phi_{i,j}^n - \phi_{i-1,j}^n}{h}\right)^2} [g(\nabla u_0)]_{i,j} \kappa_{i,j}(u_0) + ([g_x(\nabla u_0)]_{i,j}, [g_y(\nabla u_0)]_{i,j}) \cdot \left(\frac{\phi_{i,j}^n - \phi_{i-1,j}^n}{h}, \frac{\phi_{i,j}^n - \phi_{i,j-1}^n}{h}\right)$$

The formulation given by [6] is based on level sets and follows from the theorem:

Theorem 1 *The solutions to the minimization problem: minimize $E_1 + E_2$ in a fixed domain D subject to integral constraint*

$$\iint \frac{(\sum H(\phi_i(x,y,t)) - 1)^2}{2} \, dx \, dy = \epsilon$$

satisfying for $i = 1, \ldots, n$

$$\delta(\phi_i) \left(\gamma_i \nabla \cdot \left(\frac{\nabla \phi_i}{|\nabla \phi_i|} \right) - e_i - \lambda \left(\sum_{i=1}^n H(\phi_i) - 1 \right) \right) = 0$$

with boundary conditions

$$\frac{\delta(\phi_i)}{|\nabla \phi_i|} \frac{\partial \phi_i}{\partial n} = 0$$

on ∂D, where λ is a Lagrange multiplier.

For reference $E_1 + E_2$ is given by

$$E_1 = \sum_{i=1}^{n} \gamma_i \iint \delta(\phi_i(x,y,t)) |\nabla \phi_i(x,y,t)| \; dx \; dy$$
$$E_2 = \sum_{i=1}^{n} e_i \iint H(\phi_i(x,y,t)) \; dx \; dy.$$

Below is the initial image and the corresponding Gaussian blurred image for edge detection:

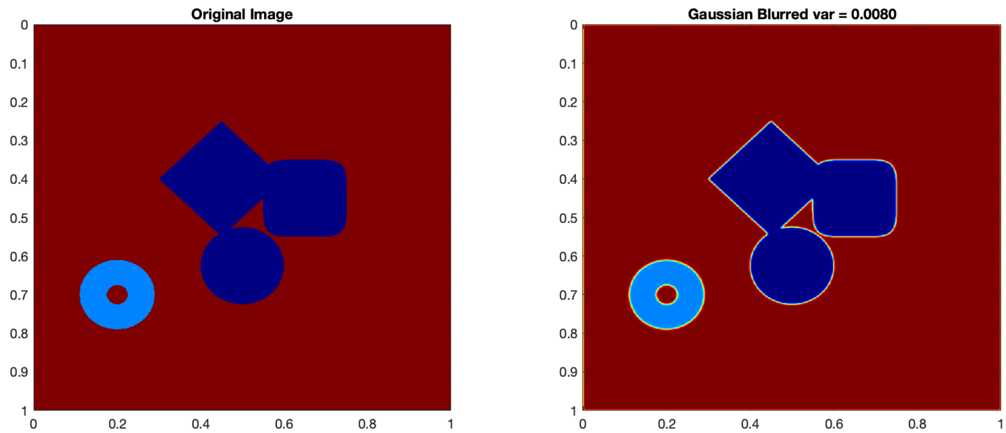


Figure 1: The figure on the left is the original image, on the right we have blurring with variance $\sigma = 0.008$.

Below are the time evolutions: I did two sets of plots to observe the difference between including and not

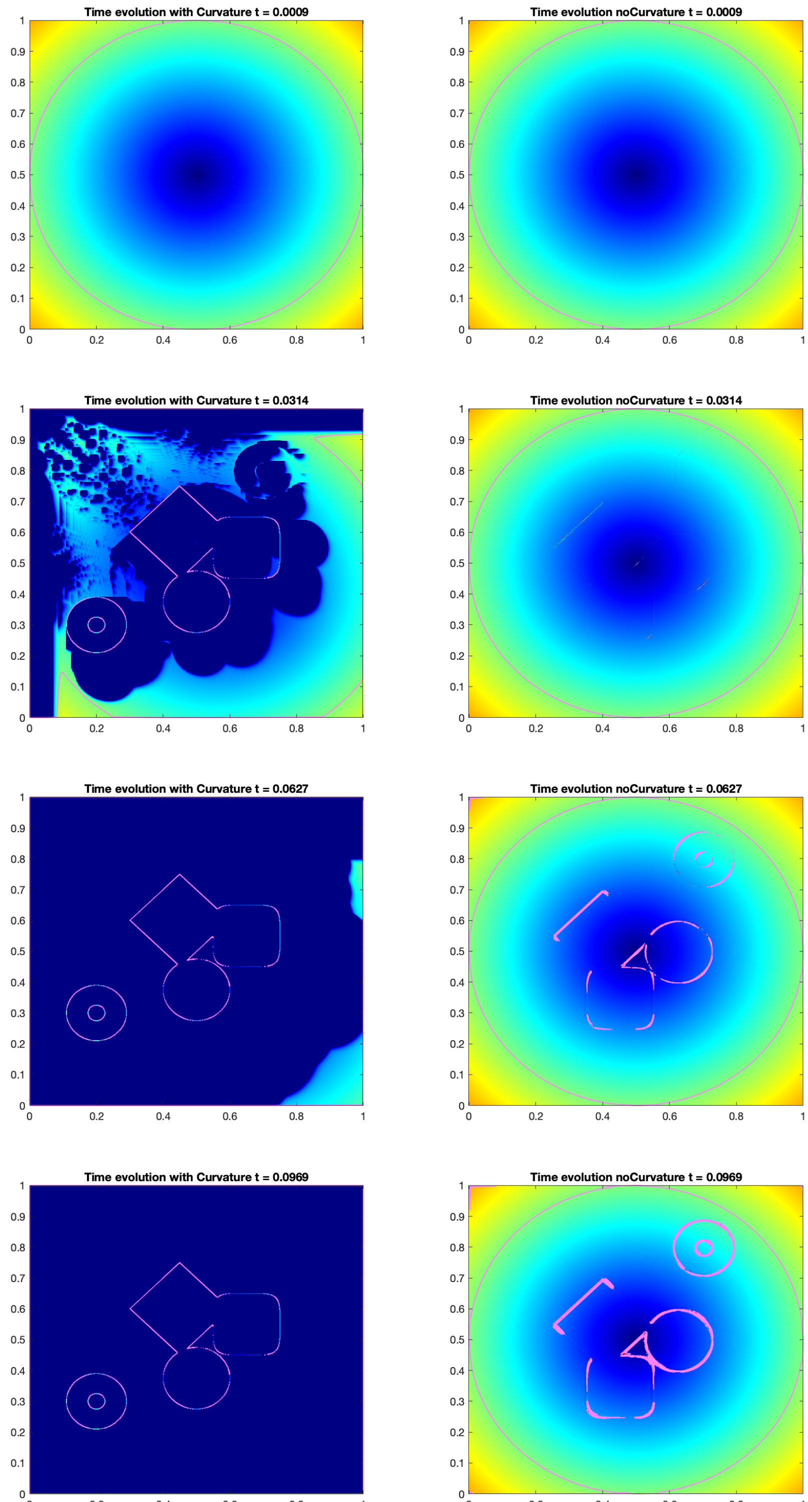


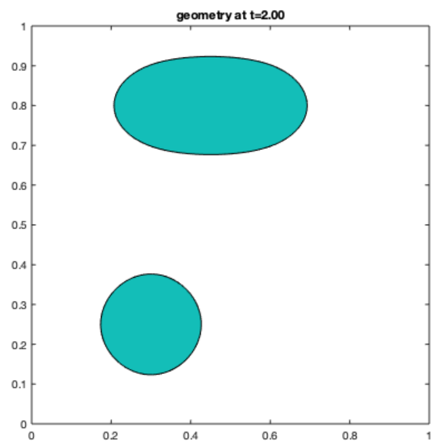
Table 1: The left plots have curvature, the right plots do not have curvature

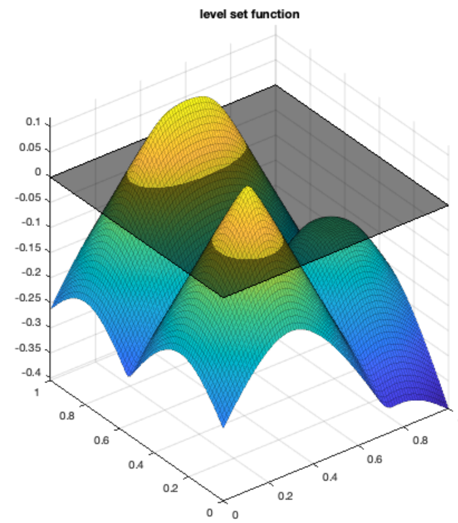
including the curvature term. Below is a snippet of the code of the update rule:

```
for n=1:nT
    for i=2:npts-1
        for j=2:npts-1
            phi(i,j)=cfl*sqrt((phi(i,j)-phi(i,j-1))^2+(phi(i,j)-phi(i-1,j))^2)*g_DU(i,j)*...
            kurv(i,j)+(1+cfl*(Dxg_DU(i,j)+Dyg_DU(i,j)))*phi(i,j)-cfl*(Dxg_DU(i,j)*...
            phi(i-1,j)+Dyg_DU(i,j)*phi(i,j-1));

            phi2(i,j)=(1+cfl*(Dxg_DU(i,j)+Dyg_DU(i,j)))*phi2(i,j)-cfl*(Dxg_DU(i,j)*...
            phi2(i-1,j)+Dyg_DU(i,j)*phi2(i,j-1));
        end
    end
```

This implementation did not work in that you do not see the isocontour "move" towards the shape of the raster image; instead, the contours develop separately along with other numerical artifacts, such as jumps and holes. For my second implementation I will modify the levelset font code, initially it computes the movement of fronts under a given velocity field:





First, our original formulation (1) can be re-written as

$$\phi_t = |\nabla\phi| \underbrace{\nabla \cdot \left(g(\nabla u_0) \left(\frac{\nabla\phi}{|\nabla\phi|} \right) \right)}_{F}$$

where F is the generalized curvature. The code for levelset front solves the problem $\phi_t + F|\nabla\phi| = 0$. Therefore, I will modify the definition of F in the code to accept the generalized curvature for F.

Another methodology I have to investigate involves gradient vector flows [3]

$$E_{\text{GVF}} = \iint \mu(u_x^2 + u_y^2 + v_x^2 + v_y^2) + |\nabla f|^2 |\mathbf{v} - \nabla f|^2 \, dx \, dy$$

where μ is a controllable smoothing term, which is solved by the Euler equations

$$\mu \nabla^2 u - \left(u - \frac{\partial}{\partial x} F_{\text{ext}} \right) \left(\frac{\partial}{\partial x} F_{\text{ext}}(x,y)^2 + \frac{\partial}{\partial y} F_{\text{ext}}(x,y)^2 \right) = 0$$

$$\mu \nabla^2 v - \left(v - \frac{\partial}{\partial y} F_{\text{ext}} \right) \left(\frac{\partial}{\partial x} F_{\text{ext}}(x,y)^2 + \frac{\partial}{\partial y} F_{\text{ext}}(x,y)^2 \right) = 0$$

A third methodology is to use the Mumford-Shah functional [1]

$$E[J,B] = C\int_D (I(\vec{x}) - J(\vec{x}))^2 \,\mathrm{d}\vec{x} + A\int_{D/B} \vec{\nabla} J(\vec{x}) \cdot \vec{\nabla} J(\vec{x}) \,\mathrm{d}\vec{x} + B\int_B \, ds$$

optimizing this functional leads to a criteria for segmenting an image into sub-image regions and Ambrosio-Tortorelli give an algorithm for acheiving the minimum and also show the minimum is well-defined. To summarize:

- Continue investigating the level set method described by [6]
- Implement the gradient vector flow methodology
- Test the two implementations above on 2d neuron geometries e.g. and 2d graph essentially
- Then implement on 3d geometries of neurons
- Implement an algorithm for minimizing the Mumford-Shah functional.

References

[1] Luigi Ambrosio and Vincenzo Maria Tortorelli. Approximation of functional depending on jumps by elliptic functional via t-convergence. *Communications on Pure and Applied Mathematics*, 43(8):999–1036, 1990.

[2] Vicent Caselles, Ron Kimmel, and Guillermo Sapiro. Geodesic active contours, 1997.

[3] Chenyang Xu and J. L. Prince. Gradient vector flow: a new external force for snakes. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 66–71, June 1997.

[4] Satyanad Kichenassamy, Arun Kumar, Peter Olver, Allen Tannenbaum, and Anthony Yezzi. Conformal curvature flows: From phase transitions to active vision. *Archive for Rational Mechanics and Analysis*, 134:275–301, 01 1996.

[5] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Applied Mathematical Sciences. Springer New York, 2006.

[6] Hong-Kai Zhao, T. Chan, B. Merriman, and S. Osher. A variational level set approach to multiphase motion. *Journal of Computational Physics*, 127(1), 8 1996.