ESAM 449 - Homework 4

Philip Moseley

December 4, 2009

1 Overview

A level-set based image segmentation method based on the paper "Level Set Evolution Without Re-initialization: A New Variational Formulation" has been implemented. The method eliminates the need for reinitialization by adding energy terms which force the level set to remain close to a signed distance function. The evolution equation is given by

$$\frac{\partial \phi}{\partial \tau} = \mu \left[\nabla^2 \phi - \kappa\right] + \lambda \delta_\epsilon(\phi) \nabla \cdot (g\mathbf{n}) + \nu g \delta_\epsilon(\phi) \tag{1.1}$$

where μ, λ, and ν are user-defined parameters. The μ term is an internal energy term that depends only on ϕ and penalizes the level set when it deviates from a signed distance function. The λ and ν terms are external energy terms that drive the motion of the level set towards the desired image features. Here g is an edge indicator function given by

$$g = \frac{1}{1 + |\nabla G_\sigma * I|^2} \tag{1.2}$$

where G_σ is the Gaussian kernel and I is the image data, and δ_ϵ is a smoothed dirac delta function

$$\delta_\epsilon(x) = \begin{cases} 0 & |x| > \epsilon \\ \frac{1}{2\epsilon}\left[1 + cos\left(\frac{\pi x}{\epsilon}\right)\right] & |x| \leq \epsilon \end{cases} \tag{1.3}$$

Two solution methods for this system of equations are compared. In the explicit solution time is discretized using the forward Euler method and solving for the next time step ϕ^{n+1},

$$\phi^{n+1} = \phi^n + \Delta\tau \left(\mu \left[\nabla^2 \phi - \kappa\right] + \lambda \delta_\epsilon(\phi) \nabla \cdot (g\mathbf{n}) + \nu g \delta_\epsilon(\phi)\right) \tag{1.4}$$

The semi-implicit solution method is more complicated. The discretized equation is

$$\frac{\phi^{n+1} - \phi^n}{\Delta\tau} = \mu \nabla^2 \phi^{n+1} - \mu \kappa^m + \lambda \delta_\epsilon(\phi^n) \nabla \cdot (g\mathbf{n}^n) + \nu g \delta_\epsilon(\phi^n) \tag{1.5}$$

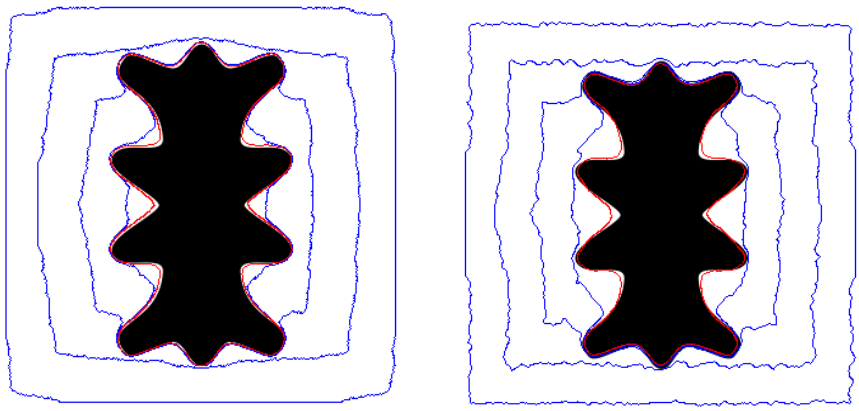
which can be rewritten as

$$\frac{1}{\Delta\tau\mu}\phi^{n+1} - \nabla^2 \phi^{n+1} = \frac{1}{\Delta\tau\mu}\phi^n - \kappa^n + \frac{1}{\mu}\left[\lambda \delta_\epsilon(\phi^n) \nabla \cdot (g\mathbf{n}^n) + \nu g \delta_\epsilon(\phi^n)\right] \tag{1.6}$$

This is the form of a Helmholtz equation. Rather than write a custom solver to solve this system of equations, we use Intel's Math Kernel library Helmholtz solver in two dimensions. Spatial derivatives for both the explicit and the implicit method are solved using central differences.

2 Solutions

A simple image with sharp edges is solved in Figure (1). The initial level set is a rectangle near the edges of the image, and in shrinks inwards with each iteration.

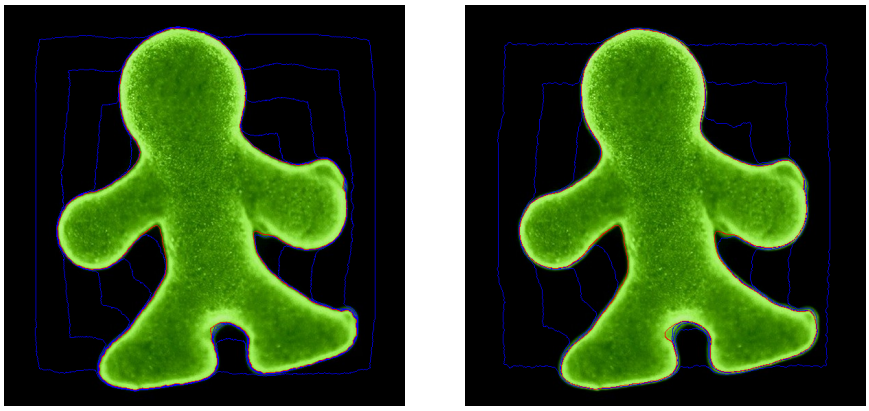


(a) 500 steps with λ=5.0, ν=7.0, $\Delta\tau$=6.0, μ=0.04

(b) 100 steps with λ=5.0, ν=9.0, $\Delta\tau$=50.0, μ=0.01

Figure 1: Partially resolved zero-contours are shown in blue, final contour shown in red. Left image is explicit solutions, right image is implicit solutions.

An image with a complex contour is shown in Figure (2). Final solutions are very similar, but the semi-implicit solution is a little less accurate because a much larger time step was used to solve the problem faster.

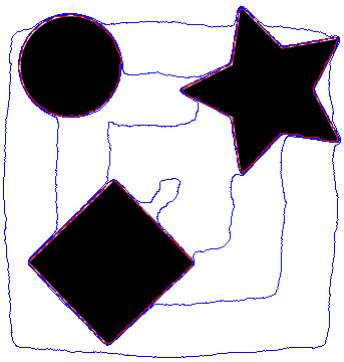


(a) 750 steps with λ=5.0, ν=5.0, $\Delta\tau$=6.0, μ=0.04

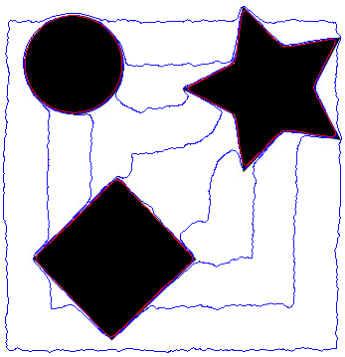
(b) 300 steps with λ=5.0, ν=4.0, $\Delta\tau$=40.0, μ=0.01

Figure 2: Partially resolved zero-contours are shown in blue, final contour shown in red. Left image is explicit solutions, right image is implicit solutions.

Solutions for a test image with 3 separate objects is shown in Figure (3). Final zero contours (shown in red) are very similar for both methods, but the semi-implicit method was stable for a much broader range of input parameters. The semi-implicit solution may be slightly less accurate, but the simulation runs nearly 5 times faster.



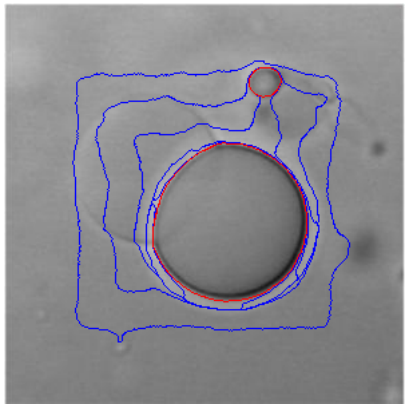
(a) 1250 steps with λ=5.0, ν=3.0, $\Delta\tau$=5.0, μ=0.04



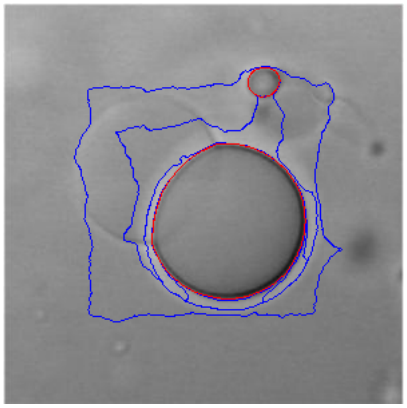
(b) 250 steps with λ=5.0, ν=3.0, $\Delta\tau$=40.0, μ=0.01

Figure 3: Partially resolved zero-contours are shown in blue, final contour shown in red. Left image is explicit solutions, right image is implicit solutions.

A more realistic application of the method is shown in Figure (4). Varying the parameters allows the method to capture different aspects of the image. The semi-implicit method still allows for a dramatically larger timestep, greatly reducing solution time.



(a) 750 steps with λ=5.0, ν=3.0, $\Delta\tau$=5.0, μ=0.04



(b) 250 steps with λ=5.0, ν=3.0, $\Delta\tau$=40.0, μ=0.01

Figure 4: Partially resolved zero-contours are shown in blue, final contour shown in red. Left image is explicit solutions, right image is implicit solutions.

3 Conclusions

The explicit and semi-implicit methods produces very similar results, but the semi-implicit scheme has a significantly broader range of stable input parameters. The semi-implicit method was able to run stably with a $\Delta\tau$ of 60.0 for some problems, while the explicit method never ran stably with $\Delta\tau$ above 10.0. Larger time steps allowed the semi-implicit simulations to complete in a fraction of the time.

Programming effort for the semi-implicit method was minimal. Intel's Math Kernel library contains an efficient Helmholtz solver which was easily integrated into my C++ code.