The Level Set Method

Lecture Notes, MIT 16.920J / 2.097J / 6.339J
Numerical Methods for Partial Differential Equations

Per-Olof Persson

March 7, 2005

1 Evolving Curves and Surfaces

Evolving boundaries or interfaces are part of many problems in science and engineering. In 1988, James A. Sethian and Stanley Osher proposed to represent these boundaries implicitly and model their propagation using appropriate partial differential equations. The boundary is given by level sets of a function $\phi(\boldsymbol{x})$, and they named their technique the *Level Set Method*. These notes give a short introduction to the method, and for more details we refer to the books by Sethian [4] and Osher [1], as well as their original paper [2].

Consider a boundary curve in two dimensions or a surface in three dimensions, see Figure 1. We are given a velocity field $\boldsymbol{v}$, which in general depends on space, time, properties of the boundary (such as the normal direction and the curvature), as well as an indirect dependence from physical simulations using the boundary shape. To goal is to accurately model the evolution of the boundary under the velocities $\boldsymbol{v}$. In many cases we are only interested in the motion normal to the interface. We can then use a (scalar) speed function F, and let the velocities be given by $\boldsymbol{v} = F\boldsymbol{n}$, where $\boldsymbol{n}$ is the normal direction.

1.1 Explicit Techniques

A simple approach to model the boundary motion is to represent it explicitly, for example (in two dimensions) by nodes along the curve that are connected by line segments. We then move these nodes according to the velocities $\boldsymbol{v}$ by solving the system of ODEs:

$$\frac{d\boldsymbol{x}^{(i)}}{dt} = \boldsymbol{v}(\boldsymbol{x}^{(i)}, t), \quad \boldsymbol{x}^{(i)}(0) = \boldsymbol{x}_0^{(i)}, \tag{1}$$

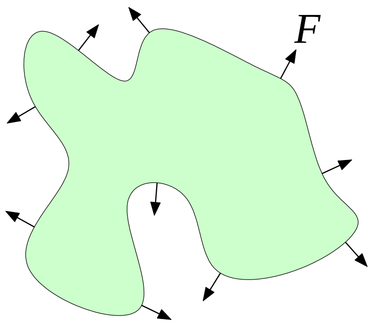

Figure 1: An interface evolving according to the speed function F.

where $\boldsymbol{x}^{(i)}$ contains the coordinates of node i, $\boldsymbol{v}$ is the velocity function, and $\boldsymbol{x}_0^{(i)}$ is the initial node location. This system can be solved accurately using explicit time integration schemes, e.g. forward Euler or Runge-Kutta methods.

When the velocity is given by $F\boldsymbol{n}$, we have to evaluate $\boldsymbol{n}$ at each node. We can approximate the tangent vector by central difference approximations involving the neighboring nodes, e.g.

$$\frac{d\boldsymbol{x}^{(i)}}{ds} \approx \frac{\boldsymbol{x}^{(i+1)} - \boldsymbol{x}^{(i-1)}}{2\Delta s} \tag{2}$$

(assuming equal distance Δs between the nodes), and expressing the normal in terms of the tangent and normalizing. Similar approximations can be used to compute other geometric variables such as the curvature.

While this explicit scheme might be sufficient for small deformations of the initial interface, it has several drawbacks for general motions:

- During the evolution, the nodes become unevenly distributed and the numerical representation of the curve gets inaccurate (see Figure 2, left). By inserting and removing nodes we can improve the representation, but this introduces errors.
- Around sharp corners, the scheme will not produce the desired “entropy solution”, but will generate an incorrect “shadowtail solution” (Figure 2, center).
- Topological changes, such as merging curves, require special treatment (Figure 2, right).
- For curvature dependent speed functions $F(\kappa)$, the scheme is sensitive to small perturbation which leads to instabilities unless the time steps are very small.

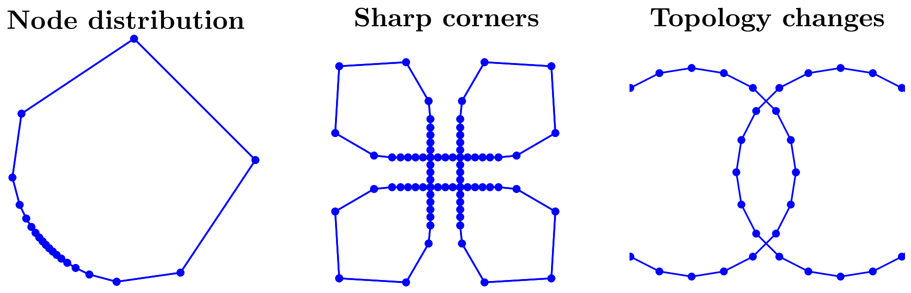

Figure 2: Problems with explicit techniques.

1.2 Implicit Geometries

In the level set method, the interface is represented implicitly by the zero level set of a function, $\phi(\boldsymbol{x}) = 0$. Note that ϕ is defined for all $\boldsymbol{x}$, not just the ones on the boundary. To represent ϕ in a finite form on a computer, we discretize using a background mesh. A common choice is a simple Cartesian grid, but quadtrees or octrees can be used for higher efficiency. An example of a discretized implicit function is shown in Figure 3.

The actual interface can be extracted from bilinear interpolation in each grid cell, but the main philosophy of the level set method is that the interface should only be accessed implicitly by operating on ϕ. Exception to this are for plotting purposes, and possibly for the reinitialization (see below).

A special case of implicit representation is the *signed distance function*. It has the property that $|\nabla\phi| = 1$, with different signs at the two sides of the interface. Also, $|\phi(\boldsymbol{x})|$ gives the (shortest) distance from $\boldsymbol{x}$ to the boundary $\phi = 0$. The level set method does not require ϕ to be a distance function, but the numerical approximations are inaccurate if ϕ has large variations in the gradient. We therefore try to keep ϕ close to a signed distance function, by frequent reinitializations (see below).

Geometric variables can be computed from ϕ without extraction of the interface. The normal vector is given by

$$\boldsymbol{n} = \frac{\nabla\phi}{|\nabla\phi|} \tag{3}$$

(since ϕ is constant at a level set, $\nabla\phi$ points in the normal direction). The curvature of a curve in two dimensions is

$$\kappa = \nabla \cdot \frac{\nabla\phi}{|\nabla\phi|} = \frac{\phi_{xx}\phi_y^2 - 2\phi_y\phi_x\phi_{xy} + \phi_{yy}\phi_x^2}{(\phi_x^2 + \phi_y^2)^{3/2}}. \tag{4}$$

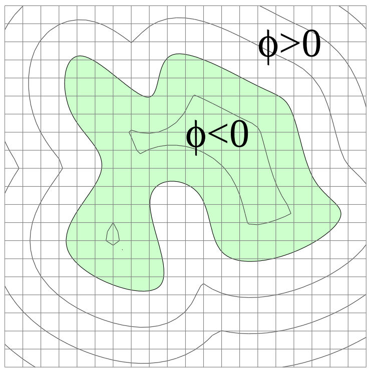

Figure 3: A signed distance function ϕ discretized on a Cartesian grid.

In three dimensions, we compute the mean curvature

$$\kappa_M = \nabla \cdot \frac{\nabla \phi}{|\nabla \phi|} = \frac{\left\{ \begin{array}{c} (\phi_{yy} + \phi_{zz})\phi_x^2 + (\phi_{xx} + \phi_{zz})\phi_y^2 + (\phi_x x + \phi_y y)\phi_z^2 \\ -2\phi_x \phi_y \phi_{xy} - 2\phi_x \phi_z \phi_{xz} - 2\phi_y \phi_z \phi_{yz} \end{array} \right\}}{(\phi_x^2 + \phi_y^2 + \phi_z^2)^{3/2}} \tag{5}$$

and the Gaussian curvature

$$\kappa_G = \frac{\left\{ \begin{array}{c} \phi_x^2(\phi_{yy}\phi_{zz} - \phi_{yz}^2) + \phi_y^2(\phi_{xx}\phi_{zz} - \phi_{xz}^2) + \phi_z^2(\phi_{xx}\phi_{yy} - \phi_{xy}^2) \\ +2[\phi_x\phi_y(\phi_{xz}\phi_{yz} - \phi_{xy}\phi_{zz}) + \phi_y\phi_z(\phi_{xy}\phi_{xz} - \phi_{yz}\phi_{xx}) \\ +\phi_x\phi_z(\phi_{xy}\phi_{yz} - \phi_{xz}\phi_{yy})] \end{array} \right\}}{(\phi_x^2 + \phi_y^2 + \phi_z^2)^2}. \tag{6}$$

From these the principal curvatures can be obtained as $\kappa_M \pm \sqrt{\kappa_M^2 - \kappa_G}$.

Using ϕ, we can also write different expressions for the two regions $\phi < 0$, and $\phi > 0$. For example, the density

$$\rho(\boldsymbol{x}) = \rho_1 + (\rho_2 - \rho_1)\theta(\phi(\boldsymbol{x})) \tag{7}$$

takes the value ρ_1 for $\phi < 0$ and ρ_2 for $\phi > 0$. On a discrete grid, the Heaviside function $\theta(\boldsymbol{x})$ has to be approximated numerically for example by smoothing.

1.3 The Level Set Equation

Using the implicit representation ϕ, we can propagate the zero level set by the velocity field $\boldsymbol{v}$ by solving the convection equation (we actually propagate all the level sets, not just $\phi = 0$):

$$\phi_t + \boldsymbol{v} \cdot \nabla \phi = 0. \tag{8}$$

For motion in the normal direction, we use (3) and write $\boldsymbol{v} = F\boldsymbol{n} = F\,\nabla\phi/|\nabla\phi|$. Insert in (8) and use $\nabla\phi \cdot \nabla\phi = |\nabla\phi|^2$ to obtain the *Level Set Equation*

$$\phi_t + F|\nabla\phi| = 0. \tag{9}$$

1.4 Discretization

The convection equation (8) can be solved numerically using first order upwinding. For the level set equation (9), with use the following first order finite difference approximation:

$$\phi_{ijk}^{n+1} = \phi_{ijk}^{n} + \Delta t_1 \left(\max(F,0)\nabla_{ijk}^{+} + \min(F,0)\nabla_{ijk}^{-} \right), \tag{10}$$

where

$$\begin{aligned}\nabla_{ijk}^{+} = \big[& \max(D^{-x}\phi_{ijk}^{n},0)^2 + \min(D^{+x}\phi_{ijk}^{n},0)^2 + \\ & \max(D^{-y}\phi_{ijk}^{n},0)^2 + \min(D^{+y}\phi_{ijk}^{n},0)^2 + \\ & \max(D^{-z}\phi_{ijk}^{n},0)^2 + \min(D^{+z}\phi_{ijk}^{n},0)^2 \big]^{1/2},\end{aligned} \tag{11}$$

$$\begin{aligned}\nabla_{ijk}^{-} = \big[& \min(D^{-x}\phi_{ijk}^{n},0)^2 + \max(D^{+x}\phi_{ijk}^{n},0)^2 + \\ & \min(D^{-y}\phi_{ijk}^{n},0)^2 + \max(D^{+y}\phi_{ijk}^{n},0)^2 + \\ & \min(D^{-z}\phi_{ijk}^{n},0)^2 + \max(D^{+z}\phi_{ijk}^{n},0)^2 \big]^{1/2}.\end{aligned} \tag{12}$$

Here, D^{-x} is the backward difference operator in the x-direction, D^{+x} the forward difference operator, etc. For the curvature dependent part of F, we use central difference approximations instead of (10). For further details and higher order schemes, see [4].

1.5 Reinitialization

After evolving ϕ under a general speed function F, it generally does not remain a signed distance function. We can *reinitialize* ϕ by finding (a new) ϕ with the same zero level set but with $|\nabla\phi| = 1$. Sussman et al [5] proposed integrating the reinitialization equation

$$\phi_t + \operatorname{sign}(\phi)(|\nabla\phi| - 1) = 0 \tag{13}$$

for a short period of time. This equation can be discretized in a similar way as the level set equation, and the discontinuous sign function is smoothed over a few grid cells.

Another option is to explicitly update the nodes close the boundary, by for example extracting the curve segments and computing the distances to the grid nodes. For the remaining nodes, we can use the efficient fast marching method (see below).

1.6 The Boundary Value Formulation

The level set equation (9) is an initial value problem, where we track the zero level set $\phi = 0$ in time and ignore ϕ away from the interface. If the speed function $F > 0$, we can alternatively formulate the evolution by an arrival function T,

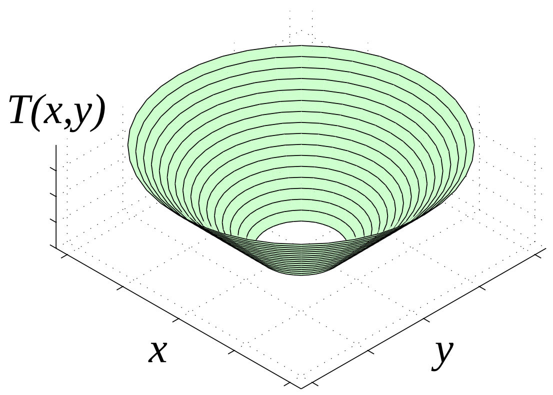

Figure 4: The boundary value formulation for interface evolution. $T(x,y)$ gives the time for the interface to reach (x,y) from the initial location.

with $T(\boldsymbol{x})$ giving the time for the interface to reach $\boldsymbol{x}$ from its initial location Γ (Figure 4). From the fact that time * rate = distance we can derive the boundary value problem known as the Eikonal equation:

$$|\nabla T|F = 1, \quad T = 0 \text{ on } \Gamma. \tag{14}$$

An important special case is $F = 1$, when (14) can be used to compute distance functions for the boundary Γ.

1.7 The Fast Marching Methods

We can discretize the Eikonal equation (14) using the same scheme as for the level set equation, to obtain the nonlinear algebraic system of equations

$$\begin{bmatrix} \max(D^{-x}_{ijk}T, 0)^2 + \min(D^{+x}_{ijk}T, 0)^2 \\ +\max(D^{-y}_{ijk}T, 0)^2 + \min(D^{+y}_{ijk}T, 0)^2 \\ +\max(D^{-z}_{ijk}T, 0)^2 + \min(D^{+z}_{ijk}T, 0)^2 \end{bmatrix}^{1/2} = \frac{1}{F_{ijk}}. \tag{15}$$

Another possibility is the scheme

$$\begin{bmatrix} \max(D^{-x}_{ijk}T, -D^{+x}_{ijk}T, 0)^2 \\ +\max(D^{-y}_{ijk}T, -D^{+y}_{ijk}T, 0)^2 \\ +\max(D^{-z}_{ijk}T, -D^{+z}_{ijk}T, 0)^2 \end{bmatrix}^{1/2} = \frac{1}{F_{ijk}} \tag{16}$$

which is slightly simpler because we choose either the forward or the backward difference along each dimensions (never both).

For efficient solution of (16), we observe that front propagates outward from Γ and that nodes with higher value of T will never affect nodes with smaller values. This is the idea behind the *Fast Marching Method* (Sethian [3], see also Tsitsiklis [6]). The given boundary values are considered “known values”, and the nodes neighboring these can be updated by (16) and inserted into a priority queue. The node with smallest unknown value can then by removed and marked as “known”, since it will not be affected by the other unknown values. Its neighbors are then updated by (16) and inserted into the queue. This is repeated until all node values are “known”, and with a priority queue based on heaps the total computation requires $\mathcal{O}(n \log n)$ operations for n nodes.

References

[1] Stanley Osher and Ronald Fedkiw. *Level set methods and dynamic implicit surfaces*, volume 153 of *Applied Mathematical Sciences.* Springer-Verlag, New York, 2003.

[2] Stanley Osher and James A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.*, 79(1):12–49, 1988.

[3] James A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proc. Nat. Acad. Sci. U.S.A.*, 93(4):1591–1595, 1996.

[4] James A. Sethian. *Level set methods and fast marching methods*, volume 3 of *Cambridge Monographs on Applied and Computational Mathematics.* Cambridge University Press, Cambridge, second edition, 1999.

[5] Mark Sussman, Peter Smereka, and Stanley Osher. A level set approach for computing solutions to incompressible two-phase flow. *J. Comput. Phys.*, 114(1):146–159, 1994.

[6] John N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Trans. Automat. Control*, 40(9):1528–1538, 1995.