Surface Reconstruction using the Level Set Method

CS 870 Project

Instructor: Justin Wan

Author: Egor Larionov (20263767)

University of Waterloo

February 14, 2014

Abstract

This project reintroduces a level set surface reconstruction method presented in [1]. This method attempts to fit a smooth surface to a set of data points using surface evolution through a level set PDE. The main idea is to evolve a surface enclosing a set of data points by minimizing its surface area and its distance to the point set. A particular implementation of this method is presented, and a few implementation issues are outlined. Finally a set of simulations are presented comparable to results found in [1].

1 Introduction

Surface reconstruction is a difficult problem, studied for as long as there existed digital representations of surfaces. The goal of this problem is to obtain a smooth representation of a 2D or 3D surface given some unorganized, noisy, or damaged data set. In many applications the data set contains a set of disconnected points approximating a surface in three dimensions. For instance, in medical imaging, it is often desired to construct a 3D model of the brain or some other organ for analysis. A set of 2D digital scans may be organized in 3D space as a set of data points representing segmented 2D regions. These points can be fed into a surface reconstruction algorithm, that may be able to approximate the desired 3D model.

There are many other applications for surface reconstruction. In data visualization, a smooth surface can be used to interpolate a discretized function. In physical modelling, a perfectly closed surface is required, so that a tessellation algorithm can discretize its interior in preparation for a physical simulation (e.g. the finite element method requires such a tessellation and is often used to model soft bodies). Even the result of a 3D scan may need to be repaired due to noisy input data or missing information using a surface reconstruction algorithm.

In this project I present a particular method for surface reconstruction introduced by Zhao et. al. in [1]. I present a walkthrough of the model and the level set formulation used in this method, followed by a detailed description of used algorithms and an outline of various implementation difficulties encountered. Finally, a few simulations of the algorithms are shown at the end.

2 Model

Let $\mathcal{S}$ denote a set of data points[1], to which we would like to fit a smooth surface, $\Gamma \subset \Omega$. Let $\Omega \subset \mathbb{R}^3$, be a compact domain, with $\mathcal{S} \subset \Omega$ closed. The distance function $d : \Omega \to \mathbb{R}$, representing the distance from each point in Ω to $\mathcal{S}$, is then defined by

$$d(\boldsymbol{x}) := dist(\boldsymbol{x}, \mathcal{S}) = \inf_{\boldsymbol{p}\in\mathcal{S}} \|\boldsymbol{x} - \boldsymbol{p}\|_2 \tag{2.1}$$

The ideal $\Gamma = \Gamma'$ would be as close as possible to the data set, while maintaining smoothness. Therefore we need to find a functional on Γ that has a minimum at Γ'. One such functional is

$$E(\Gamma) = \left(\int_\Gamma d^p(\boldsymbol{x}) ds \right)^{\frac{1}{p}},$$

where ds is a surface area element and $p \in (0, \infty)$ is a parameter. This functional can be interpreted as the surface energy where d is the potential at each point on Γ. In addition $E(\Gamma)$ is the L^p norm of d restricted to Γ, that is

$$E(\Gamma) = \| \, d|_\Gamma \, \|_{L^p}.$$

To minimize E, we need to compute the first variation of E:

$$\frac{\delta E(\Gamma)}{\delta \Gamma} = \frac{1}{p} \left(\int_\Gamma d^p(\boldsymbol{x}) ds \right)^{\frac{1}{p}-1} \left(p d^{p-1} \nabla \cdot \boldsymbol{n} + d^p \kappa \right),$$

where $\boldsymbol{n}$ is the normal to the surface, and κ is the mean curvature. Setting this to zero gives us the Euler-Lagrange equation

$$d^{p-1}(\boldsymbol{x}) \left(\nabla d(\boldsymbol{x}) \cdot \boldsymbol{n} + \frac{1}{p} d(\boldsymbol{x}) \kappa \right) = 0. \tag{2.2}$$

[1]Curves and surface patches can also be included in $\mathcal{S}$

The solution to this PDE is hidden in $\boldsymbol{n}$ and κ which essentially define the goal surface, Γ. At points where Γ doesn't touch $\mathcal{S}$, we have

$$\nabla d(\boldsymbol{x}) \cdot \boldsymbol{n} = -\frac{1}{p} d(\boldsymbol{x})\kappa, \tag{2.3}$$

an equilibrium between the potential force $\nabla d(\boldsymbol{x}) \cdot \boldsymbol{n}$ (if interpreted as the change in potential) and surface tension, $d(\boldsymbol{x})\kappa$. This is precisely the property our ideal surface needs. The surface tension maintains smoothness of the surface, while the potential force pushes the surface closer the data set. From (2.3), we can see that p also affects the flexibility of the surface, where a larger p will increase the flexibility of the surface regardless of its distance from $\mathcal{S}$.

2.1 Level Set Formulation

In order to evolve a surface towards the target minimum of $E(\Gamma)$, we can use the gradient descent of E, which gives us a continuous deformation of Γ. So given some initial surface Γ_0, we evolve it in time according to the gradient flow

$$\begin{aligned} \frac{d\Gamma}{dt} &= -\frac{1}{p}\left(\int_\Gamma d^p(\boldsymbol{x})ds\right)^{\frac{1}{p}-1} \nabla \cdot (d^p(\boldsymbol{x})\boldsymbol{n})\boldsymbol{n} \\ &= -\left(\int_\Gamma d^p(\boldsymbol{x})ds\right)^{\frac{1}{p}-1} d^{p-1}(\boldsymbol{x})(\nabla d(\boldsymbol{x}) \cdot \boldsymbol{n} + \frac{1}{p} d(\boldsymbol{x})\kappa)\boldsymbol{n} \\ &= -\left(\frac{d(\boldsymbol{x})}{E(\Gamma)}\right)^{p-1} (\nabla d(\boldsymbol{x}) \cdot \boldsymbol{n} + \frac{1}{p} d(\boldsymbol{x})\kappa)\boldsymbol{n} \end{aligned} \tag{2.4}$$

Note that for large p, the surface becomes too flexible, and takes a longer time to converge. Empirically, it is witnessed [1] that $p = 2$ provides good results, in general.

Given this gradient flow, we can finally construct the level set PDE. Fix a time $t \in \mathbb{R}^+$, and let $\Gamma(t)$ be a 2-dimensional closed surface in Ω. Let $\Gamma^+(t)$ denote the inside of the surface, and $\Gamma^-(t)$ denote the outside. Then we define an implicit function $u : \Omega \times \mathbb{R}^+ \to \mathbb{R}$ as follows

$$\begin{aligned} u(\boldsymbol{x}, t) > 0 &\quad \text{in} \quad \Gamma^+(t) \\ u(\boldsymbol{x}, t) = 0 &\quad \text{on} \quad \Gamma(t) \\ u(\boldsymbol{x}, t) < 0 &\quad \text{in} \quad \Gamma^-(t) \end{aligned}$$

Thus $\Gamma(t)$ is the zero level set of u. Differentiating $u(\Gamma(t), t)$ with respect to t gives the level set PDE:

$$u_t + \frac{d\Gamma(t)}{dt} \cdot \nabla u = 0. \tag{2.5}$$

The solution to (2.5) gives us the implicit function, whose zero level set is precisely the desired surface Γ'. We can now express the energy functional in terms of u:

$$E(u) = \left(\int_\Omega d^p(\boldsymbol{x})\delta(u(\boldsymbol{x}))|\nabla u(\boldsymbol{x})|d\boldsymbol{x}\right)^{\frac{1}{p}}, \tag{2.6}$$

where $\delta(x)$ is the one dimensional Delta function, and $\delta(u(\boldsymbol{x}))|\nabla u(\boldsymbol{x})|dx$ is a surface area element at Γ. Note that we can now express the normal $\boldsymbol{n}$ and mean curvature κ of Γ in terms of u as follows

$$\boldsymbol{n} = \tfrac{\nabla u}{|\nabla u|}, \qquad \kappa = \nabla \cdot \tfrac{\nabla u}{|\nabla u|}.$$

Thus equations (2.4) and (2.5) give

$$
\begin{aligned}
u_t &= \left(\frac{d}{E(u)}\right)^{p-1} \left(\nabla d \cdot \boldsymbol{n} + \tfrac{1}{p} d\kappa\right) \boldsymbol{n} \cdot \nabla u \\
&= \left(\frac{d}{E(u)}\right)^{p-1} \left(\nabla d \cdot \nabla u + \tfrac{1}{p} d\kappa |\nabla u|\right), \qquad (2.7)
\end{aligned}
$$

which is a level set PDE of the form

$$u_t = \boldsymbol{v} \cdot \nabla u + \alpha\kappa|\nabla u|$$

where $\boldsymbol{v}$ defines the external velocity field (in the direction of ∇d) and $\alpha\kappa|\nabla u|$ is the parabolic curvature term. The discretization and computation of such an equation is carefully treated in [2].

2.2 Narrow Band Method

Since we are usually only interested in the zero level set of u, it is wasteful to compute values of u on the whole domain Ω_d. Instead, at n^{th} time step, we solve the level set equation on a subset of $T^n \subset \Omega_d$, where $T^n := \{\boldsymbol{x} : |u^n(\boldsymbol{x})| < \gamma\}$. We chose $\gamma = 4h$ to insure that there are enough neighbouring grid points with correct values for u^n near the zero level set, so that higher order derivatives can be used[2]. Note that $\gamma = 2h$ would be enough, however, to avoid numerical oscillations at the boundary of T^n, we modify the level set equation (2.5) to

$$u_t + c(u)\tfrac{d\Gamma}{dt} \cdot \nabla u = 0 \qquad (2.8)$$

where $c(u)$ is the cut-off function:

$$
c(u) = \begin{cases} 1 & \text{if } |u| \le \beta \\ (|u| - \gamma)^2 (2|u| + \gamma - 3\beta)/(\gamma - \beta)^3 & \text{if } \beta < |u| \le \gamma \\ 0 & \text{if } |u| > \gamma \end{cases}
$$

so that β is chosen to be $2h$.

2.3 Reinitialization

Evolve the implicit function u according to (2.8), may cause steep or flat slopes to develop near the boundary of T^n. In general, u may deviate from being a distance function with the same zero level set. It has been observed that having $|\nabla u|$ close to 1 within T^n gives better results for surface evolution [2, 3, 1]. A proposed solution is to introduce an auxiliary time based PDE that will normalize $|\nabla u|$ as desired. In particular, after some time t, when $|\nabla u|$ deviates enough from 1, to solve the PDE

$$
\begin{cases} \phi_\tau + S(\phi)(|\nabla\phi| - 1) = 0 \\ \phi(\boldsymbol{x}, 0) = u(\boldsymbol{x}, t) \end{cases} \qquad (2.9)
$$

where $S(\phi)$ is the sign function, approximated by

$$S(\phi) = \frac{\phi}{\sqrt{\phi^2 + |\nabla\phi|^2 \Delta x^2}}. \qquad (2.10)$$

[2]Additionally, Hamilton-Jacobi ENO and WENO discretization schemes may use values farther from the zero level set.

As suggested in [3], this approximation prevents this PDE from moving the zero level set. This is a general problem with other approximations, such as $S(\phi) = \phi/\sqrt{\phi^2 + \epsilon^2}$ for small $\epsilon > 0$, as presented by [4]. A disadvantage of (2.10) is that it depends on $|\nabla\phi|$, which must be computed before each reinitialization iteration.

3 Implementation

First we will introduce a few assumptions and definitions to be used in this section.

For simplicity, assume that $\mathcal{S}$ contains only points. There are ways of extending $\mathcal{S}$ to contain curve and surface patches, however point sets are often sufficient in most applications to surface reconstruction.

Assume that our domain is a unit cube, $\Omega = [0,1]^3$. Define $\Omega_d \subset \Omega$ to be a grid discretization of the domain, containing N grid points, and so $N-1$ grid cells. Since the domain is a unit cube, all dimensions have equal length divisions, $h = 1/(N-1)$, so $\Delta x = \Delta y = \Delta z = h$. An implicit function $u : \Omega \to \mathbb{R}$ is discretized on Ω_d, and we denote the value of u at each grid point by $u_{i,j,k} = u(x_i, y_j, z_k)$, where $(x_i, y_j, z_k) \in \Omega_d$. Time evolution is also discretized by a time step Δt. At n^{th} iteration, where $t = n\Delta t$, we denote the value of u by $u^n = u(t)$.

3.1 Computing the Distance to the Data

Many algorithms have been developed to compute the function d as defined in (2.1), efficiently. More accurate methods often use Voronoi diagrams [5], other less accurate, but faster techniques use PDE formulations [1]. There exist, however, a few optimized, hybrid algorithms [6] to compute d. Since it is sufficient for us to know a rough approximation to the distance function, we will focus on the implementation of an iterative method for solving the eikonal PDE:

$$\begin{cases} |\nabla d(\boldsymbol{x})| = 1 \\ d(\boldsymbol{x}) = 0 \quad \forall \boldsymbol{x} \in \mathcal{S} \end{cases} \tag{3.1}$$

where $|\cdot|$ denotes the Euclidean norm. The viscosity solution for $d(\boldsymbol{x}) \geq 0$ can be solved [7] using the Godunov Hamiltonian discretization of $|\nabla d|$:

$$H_G = \sqrt{(D^x)^2 + (D^y)^2 + (D^z)^2} \tag{3.2}$$

where

$$\begin{aligned} D^x &= \max(\max(D^x_-, 0), -\min(D^x_+, 0))^2, \\ D^y &= \max(\max(D^y_-, 0), -\min(D^y_+, 0))^2, \\ D^z &= \max(\max(D^z_-, 0), -\min(D^z_+, 0))^2, \end{aligned}$$

$$D^x_\pm = \frac{d_{i,j,k} - d_{i\pm1,j,k}}{h}, \qquad D^y_\pm = \frac{d_{i,j,k} - d_{i,j\pm1,k}}{h}, \qquad \text{and} \qquad D^z_\pm = \frac{d_{i,j,k} - d_{i,j,k\pm1}}{h}$$

are forward and backward differencing partial derivative approximations. Along with a clever iterative method, solving

$$H_G = 1, \tag{3.3}$$

is usually called the fast sweeping method [6]; it has been previously used in level set schemes [8]. This method is $\mathcal{O}(h)$ accurate. To illustrate how we solve equation (3.3), we will use the following values

$$x_{min} := \min(d_{i-1,j,k},\ d_{i+1,j,k}), \quad y_{min} := \min(d_{i,j-1,k},\ d_{i,j+1,k}), \quad z_{min} := \min(d_{i,j,k-1},\ d_{i,j,k+1}).$$

First, rewrite (3.2) as

$$H_G = \frac{1}{h}\sqrt{\max(0, d_{i,j,k} - x_{min})^2 + \max(0, d_{i,j,k} - y_{min})^2 + \max(0, d_{i,j,k} - z_{min})^2}, \tag{3.4}$$

where it is clear that partial derivative approximations use the values closer to the data set. A good initial guess for the distance function can be large everywhere except at grid points $d_{i,j,k}$ near $\mathcal{S}$, where we compute the exact values for the distance function. Then we can solve (3.3) by iteratively updating the $d_{i,j,k}$ values[3] using the most recently updated neighbouring values. First, let a_1, a_2, and a_3 be the values x_{min}, y_{min}, and z_{min} in sorted order, and define $a_4 = \infty$. Then we follow a simple algorithm[4] as follows.

1. Let $i = 1$.
2. If $d' := a_i + h \le a_{i+1}$, then assign $\bar{d} = d'$. Otherwise compute

   $$d' := \frac{a_i + a_{i+1} + \sqrt{2h^2 - (a_{i+1} - a_i)^2}}{2},$$

   and if $d' \le a_{i+2}$, then assign $\bar{d} = d'$,
3. If $\bar{d}$ is still unassigned, then increment i by 1, and go to step 2.
4. Otherwise simply update our distance function, $d^{(new)}_{i,j,k} \leftarrow \min(\bar{d}, d^{(old)}_{i,j,k})$, using the previously computed, $d^{(old)}_{i,j,k}$.

Note that in our case $a_4 = \infty$ so we would increment i at most once, however this algorithm works for n-dimensions for any n. This type of non-linear Gauss-Seidel iteration is performed in 8 sweeps with the following orders

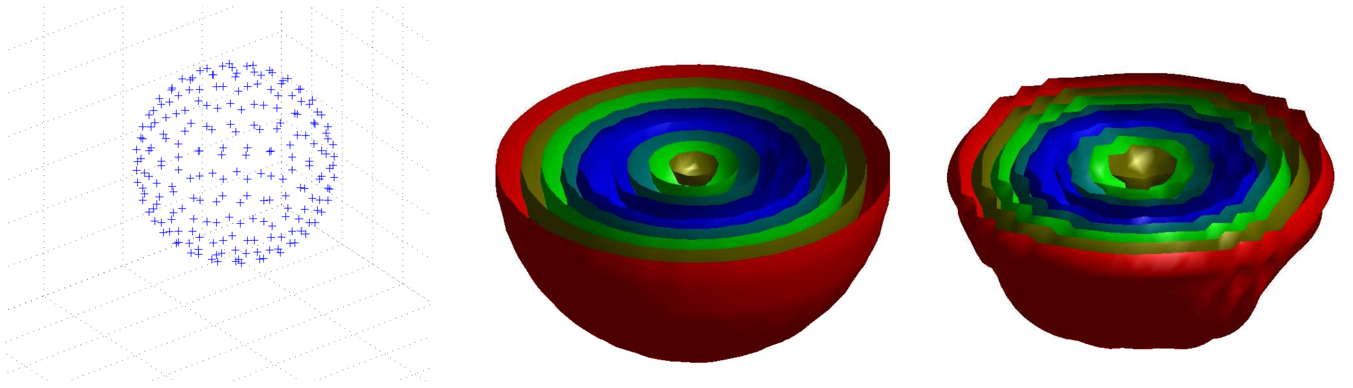
1. $i = 1:n,\ j = 1:n,\ k = 1:n$	5. $i = n:1,\ j = n:1,\ k = 1:n$
2. $i = n:1,\ j = 1:n,\ k = 1:n$	6. $i = 1:n,\ j = n:1,\ k = n:1$
3. $i = 1:n,\ j = n:1,\ k = 1:n$	7. $i = n:1,\ j = 1:n,\ k = n:1$
4. $i = 1:n,\ j = 1:n,\ k = n:1$	8. $i = n:1,\ j = n:1,\ k = n:1$

followed by a sweep with an arbitrary order.

This algorithm produces good results near the data points, and becomes increasingly more inaccurate farther from the data. This is not a significant problem if the initial surface is close to the data set, and with the narrow band method developed in [3, 10], we never compute values of the implicit function far from the data. The comparison of various PDE solutions for the distance function is shown in Figure 3.1.

[3]Of course there is no need to update grid points with exact distance values.

[4]See [9] a more detailed analysis of this method.



(a) Given point set $\mathcal{S}$.

(b) Exact distance function for comparison.

(c) 1 cycle of 8 sweeps + 1 random sweep.

Figure 3.1: The computed distance function on a 32x32x32 grid with 214 data points aligned in a sphere centred at $(0.5, 0.5, 0.5)$ with radius 0.2. The tops of the shells were clipped to expose the internal structure of the distance function. The colours of the shells represent their distance from the data set, in descending order: red $\rightarrow$ green $\rightarrow$ blue, where blue is closest to the data set. This example shows that the computed distance function closely approximates the true distance function. Furthermore, increasing the number of sweep cycles does not affect the approximation much.

3.2 Initial Surface

To perform curve evolution, we need an initial surface to start. To reduce the time for the solution to converge, and avoid room for errors, it is desirable to start with a surface close to the data set. One way to accomplish this is to construct a shell, $\epsilon > 0$ away from $\mathcal{S}$, enclosing all points in $\mathcal{S}$, and use it as the initial surface Γ_0. Note that if ϵ is too small, then the initial surface will collapse into the data set, i.e. will not resolve the desired Γ' but another global minimum of $E(\Gamma)$. So we must find an appropriate ϵ from what we know about $\mathcal{S}$. To maintain generality of the method, we can't assume anything about the set $\mathcal{S}$, however we may inquire about its minimum local feature size l and the maximum distance between two neighbouring data points r. A simple algorithm (Algorithm 1 in Appendix A) approximates l and r as the smallest and largest distance between every pair of neighbouring points in $\mathcal{S}$ respectively. Note that if $l > r/2$, then $\mathcal{S}$ is a fairly uniform sampling, and we can choose $l > \epsilon > r/2$. This will insure that the initial surface will "resolve" small local features since $l > \epsilon$ however be large enough to maintain a connectivity between all neighbouring points since $\epsilon > r/2$. If $l < r/2$, then we may need to choose an ϵ that changes according to local feature size. In practice it seems to be favourable to choose a larger ϵ since Algorithm 1 underestimates r for complex surfaces.

Computing the initial surface is a simple tagging process followed by solving the eikonal equation (3.1). In particular, we follow the algorithm prescribed in [1]:

Starting from the corner at $(i, j, k) = (1, 1, 1)$, we perform a depth (or breadth) first search through the grid. We label each visited grid point (i, j, k) as *discovered*, and if $d_{i,j,k} > \epsilon$, then we recurse on each neighbour of (i, j, k), otherwise we do nothing. For the undiscovered points (the interior of the surface) we solve the eikonal equation for $-d$ (using the fast sweeping method mentioned above), where the values $d_{i,j,k}$ at each discovered grid point give the initial condition. Then u_0 is the negative of the resulting solution.

Excluding the computation of l and r, it takes $\mathcal{O}(N^3)$ steps to compute u_0. The complexity is dominated by the depth (or breadth) first search, since solving the eikonal equation takes a constant number iterations.

3.3 Curve Evolution

Putting everything together, we get the following equation for (2.8) and (2.7):

$$u_t(\boldsymbol{x}) = c(u)\left(\frac{d(\boldsymbol{x})}{E(\Gamma)}\right)^{p-1}\left(\underbrace{\nabla d(\boldsymbol{x})\cdot\nabla u(\boldsymbol{x})}_{(*)} + \underbrace{\tfrac{1}{p}d(\boldsymbol{x})\kappa|\nabla u(\boldsymbol{x})|}_{(**)}\right) \tag{3.5}$$

where we already have $d(\boldsymbol{x}) = d_{i,j,k}$ for each grid point. The energy is computed as in (2.6), where

$$\delta_{i,j,k} = \begin{cases} \frac{1}{2\varepsilon}\left(1+\cos\left(\pi u_{i,j,k}/\varepsilon\right)\right) & \text{if } -\varepsilon < u_{i,j,k} < \varepsilon \\ 0 & \text{otherwise} \end{cases}$$

and ∇u in (2.6) and $(**)$ is computed using central differencing, $\nabla u \approx (u_x, u_y, u_z)$, where

$$u_x = \frac{u_{i+1,j,k} - u_{i-1,j,k}}{2h}, \quad u_x = \frac{u_{i,j+1,k} - u_{i,j-1,k}}{2h}, \quad u_z = \frac{u_{i,j,k+1} - u_{i,j,k-1}}{2h}, \tag{3.6}$$

and so $|\nabla u| \approx \sqrt{u_x^2 + u_y^2 + u_z^2}$. The integral in (2.6) is approximated using trapezoid rule over each dimension. The gradient $\nabla d = (d_x, d_y, dz)$ is computed once using a central differencing (same as (3.6)) over the whole grid Ω_d. The $\nabla u(\boldsymbol{x})$ term in $(*)$ above, is computed using a second order ENO scheme as developed in [11] and described in [2, §3.3] where $\nabla d(\boldsymbol{x})$ is the external velocity field. Finally the curvature term in $(**)$ is computed as prescribed in [2, §1.4].

The time variation in (3.5) is computed using a two stage Runge-Kutta scheme (Heun's method) as follows:

$$\begin{aligned}\tilde{u}^{n+1} &= u^n + \Delta t L(u^n)\\ u^{n+1} &= u^n + \tfrac{\Delta t}{2}\left(L(u^n) + L(\tilde{u}^{n+1})\right)\end{aligned}$$

where L is simply the RHS of (3.5). The time step Δt is chosen empirically based on the resolution, such that the CFL condition, $\Delta t|\frac{d\Gamma}{dt}| < h$ is satisfied.

We update $u^n_{i,j,k}$ only if $(i,j,k) \in T^n$. The narrow band T^n is computed by tagging each grid point with a 2, and keeping an array of indices of tagged grid points. This way to update $u_{i,j,k}$, we only need to iterate through the array. This take $\mathcal{O}(N^2)$ steps given that the surface is not "dense" in Ω, i.e. it doesn't occupy most of Ω.

The reinitialization step must be performed on a region containing T^n. Since the surface advances no further than one grid point during the evolution step, it is sufficient to update u on T^n and its neighbouring grid points. The above tagging algorithm can be used to tag these values as 1 to differentiate them from the main band. The PDE (2.9) is solved with Euler time discretization, and Godunov's spatial discretization as follows:

$$\phi^{n+1}_{i,j,k} = \phi^n_{i,j,k} + \frac{\Delta\tau}{h}\left(\max(0, s_{i,j,k})\left(1-\sqrt{D_1}\right) + \min(0, s_{i,j,k})\left(1-\sqrt{D_2}\right)\right),$$

where $s_{i,j,k}$ is a straight forward approximation of (2.10), and

$$\begin{aligned} D_1 := &\max(\max(0,\phi_x^-), -\min(0,\phi_x^+))^2 \\ &+\max(\max(0,\phi_y^-), -\min(0,\phi_y^+))^2 \\ &+\max(\max(0,\phi_z^-), -\min(0,\phi_z^+))^2 \end{aligned} \quad \text{and} \quad \begin{aligned} D_2 := &\max(\max(0,\phi_x^+), -\min(0,\phi_x^-))^2 \\ &+\max(\max(0,\phi_y^+), -\min(0,\phi_y^-))^2 \\ &+\max(\max(0,\phi_z^+), -\min(0,\phi_z^-))^2 \end{aligned}$$

where ϕ_x^+ and ϕ_x^- are the one-sided differences:

$$\phi_x^+ = \frac{\phi_{i+1,j,k} - \phi_{i,j,k}}{h}, \qquad \text{and} \qquad \phi_x^- = \frac{\phi_{i,j,k} - \phi_{i-1,j,k}}{h},$$

and $\phi_y^\pm$, $\phi_z^\pm$ follow in the same fashion. Alternatively one can use an ENO [11] or WENO [12] scheme to approximate these differences.

4 Implementation Issues

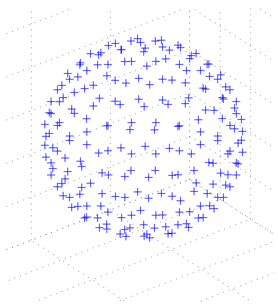
A few As Γ touches the data set $\mathcal{S}$, the curvature term in our PDE is undefined. This is easily fixed by setting $\kappa = 0$, since Γ should not move further.

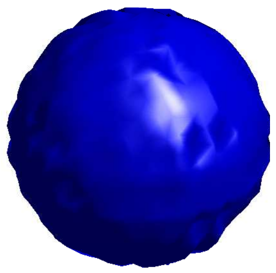
Unfortunately the ϵ used in the determining the thickness of the initial surface was constant and was chosen manually. This substantially increased the time of convergence of the level set method. In a future implementation, this is an important issue to be addressed.

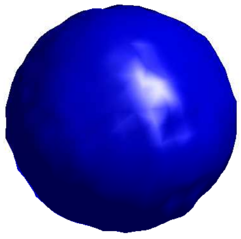
More careful investigation of the function at the boundary of the narrow band can reveal whether or not we can reduce its thickness, and thus improve the performance of this method by a constant factor. Certainly we need not maintain many values outside the propagating front since it always moves inward, towards the data set.

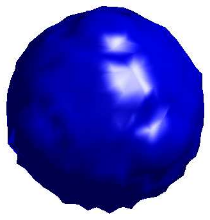
The reinitialization step has proven to be very valuable, since without it, the resulting surface is very irregular and noisy.

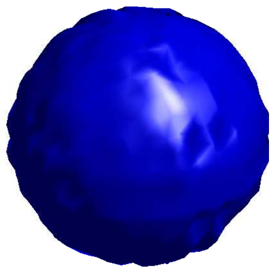
5 Results

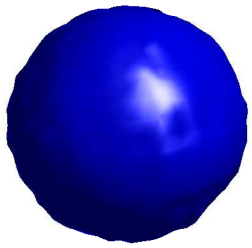

(a) Given point set $\mathcal{S}$

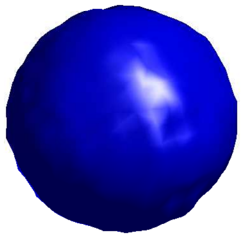

(b) Initial surface


(c) 20 iterations


(d) 40 iterations


(e) Initial surface


(f) 15 iterations


(g) 20 iterations

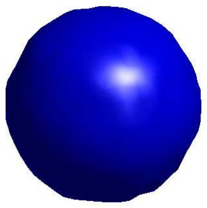
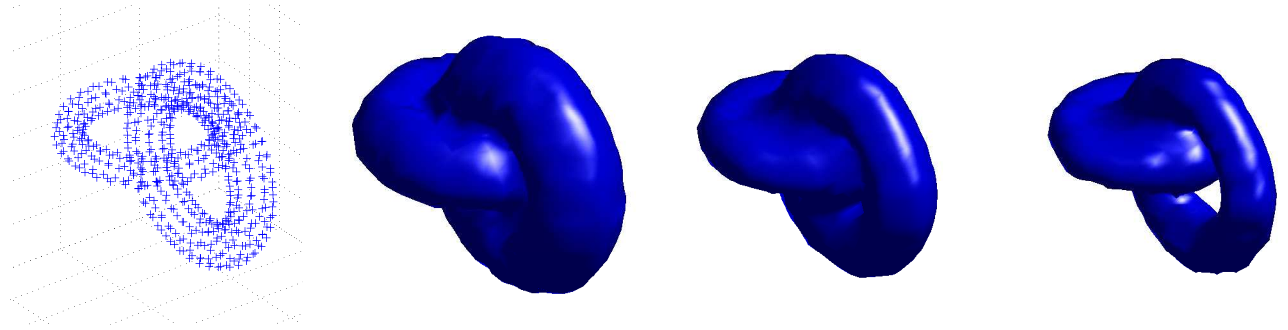

(h) 40 iterations

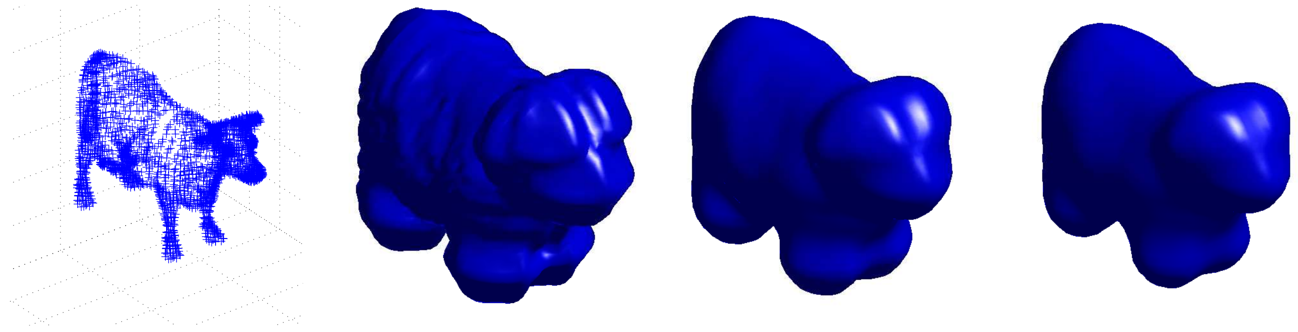
Figure 5.1: Surface reconstruction of a sphere with radius 0.2 on a 32x32x32 grid with 214 data points. The narrow band contained approximately 8000 points, as opposed to $32 \times 32 \times 32 = 32768$. In the first row, (b), (c), and (d), show evolution without reinitialization. In the second row, (e), (f), (g), and (h) show evolution with reinitialization triggered when the gradient deviated far enough from 1. Note the developing kinks in the surface of (d) as the gradient increased throughout evolution.

With this implementation, simple surfaces tend to converge to the their respective data sets fast (few iterations), and accurately even with a relatively high Δt.



(a) Given data $\mathcal{S}$ (b) Initial surface (c) 45 iterations (d) 95 iterations

Figure 5.2: Surface reconstruction of two linked tori on a 49x49x49 grid with 766 data points.



(a) Given data $\mathcal{S}$ (b) Initial surface (c) 10 iterations (d) 20 iterations

Figure 5.3: Surface reconstruction of the "cow" on a 60x60x60 grid with 2903 data points.

Unfortunately there was not enough time for the above to examples to converge to their respective data sets. As a result, a relatively large time step was used to push the surface to the data as quickly as possible, which sometimes left the CFL condition unsatisfied.

6 Conclusions

In conclusion, I was able to successfully implement the proposed (in [1]) surface reconstruction algorithm without much difficulty. However it is intractable to simulate this implementation on a grid larger than 60x60x60. New variational methods for surface reconstruction have been proposed since the conception of the paper [1]. In particular, [13] (2010) uses total variation based functionals, and [14] (2012), which uses the Multigrid PDE method while employing the narrow band idea presented here.

References

[1] H.-K. Zhao, S. Osher, B. Merriman, and M. Kang. *Implicit and nonparametric shape reconstruction from unorganized data using a variational level set method.* Computer Vision and Image Understanding, 80(3), pp. 295–314 (2000).

[2] S. Osher and R. Fedkiw. *Level set methods and dynamic implicit surfaces*, vol. 153. Springer (2003).

[3] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang. *A pde-based fast local level set method.* Journal of Computational Physics, 155(2), pp. 410 – 438 (1999). ISSN 0021-9991. doi:http://dx.doi.org/10.1006/jcph.1999.6345.

[4] M. Sussman, P. Smereka, and S. Osher. *A level set approach for computing solutions to incompressible two-phase flow.* Journal of Computational physics, 114(1), pp. 146–159 (1994).

[5] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial tessellations: concepts and applications of Voronoi diagrams*, vol. 501. Wiley. com (2009).

[6] Y.-h. R. Tsai. *Rapid and accurate computation of the distance function using grids.* Journal of Computational Physics, 178(1), pp. 175–195 (2002).

[7] E. Rouy and A. Tourin. *A viscosity solutions approach to shape-from-shading.* SIAM Journal on Numerical Analysis, 29(3), pp. 867–884 (1992). doi:10.1137/0729053.

[8] J. A. Sethian. *A fast marching level set method for monotonically advancing fronts.* Proceedings of the National Academy of Sciences, 93(4), pp. 1591–1595 (1996).

[9] H. Zhao. *A fast sweeping method for eikonal equations.* Mathematics of computation, 74(250), pp. 603–627 (2005).

[10] H.-K. Zhao, T. Chan, B. Merriman, and S. Osher. *A variational level set approach to multiphase motion.* Journal of Computational Physics, 127(1), pp. 179 – 195 (1996). ISSN 0021-9991. doi: http://dx.doi.org/10.1006/jcph.1996.0167.

[11] S. Osher and C. Shu. *High-order essentially nonoscillatory schemes for hamiltonjacobi equations.* SIAM Journal on Numerical Analysis, 28(4), pp. 907–922 (1991). doi:10.1137/0728049.

[12] G. Jiang and D. Peng. *Weighted eno schemes for hamilton–jacobi equations.* SIAM Journal on Scientific Computing, 21(6), pp. 2126–2143 (2000). doi:10.1137/S106482759732455X.

[13] J. Ye, X. Bresson, T. Goldstein, and S. Osher. *A fast variational method for surface reconstruction from sets of scattered points.* CAM Report, 10(01) (2010).

[14] J. Ye, I. Yanovsky, B. Dong, R. Gandlin, A. Brandt, and S. Osher. *Multigrid narrow band surface reconstruction via level set functions.* In *Advances in Visual Computing*, pp. 61–70. Springer (2012).

All code is written entirely by me, and is provided purely for reference. All code used to generate images appearing in this document can be found in:

http://www.egorlarionov.com/static/cs870/cs870proj.tar.gz

A Algorithms

The following algorithm was used to compute the minimum local feature size l and the maximum distance between two neighbouring data points r.

```
1:  $l \leftarrow \infty$
2:  $r \leftarrow 0$
3:  for $\boldsymbol{p} \in \mathcal{S}$ do
4:      $s \leftarrow \infty$
5:      for $\boldsymbol{q} \in \mathcal{S},\ \boldsymbol{q} \neq \boldsymbol{p}$ do
6:          if $\|\boldsymbol{p} - \boldsymbol{q}\| < s$ then
7:              $s = \|\boldsymbol{p} - \boldsymbol{q}\|$
8:          end if
9:      end for
10:     if $s < l$ then
11:         $l \leftarrow s$
12:     end if
13:     if $s > r$ then
14:         $r \leftarrow s$
15:     end if
16: end for
```

Algorithm 1: Computing local information about the input data set $\mathcal{S}$.