COMP 558: Final Project Report
Surface reconstruction using the level set method

Alexandre Vassalotti Éric Renaud-Houde

27 April 2012

1 Introduction

Modelling surfaces from unorganized set of points, or point clouds, is a long standing problem in the computer vision community. Indeed, problem is known to be very challenging in three and higher dimensions. Furthermore, the problem is ill-posed which means there is not a unique solution. When the point cloud is dense enough and the topology of the surface is not complicated, a simple solution could be to perform a Denaulay triangulation of points, as described by Boissonnat [1]. However, even in this context ambiguities can arises and lead to non desirable surface reconstructions.

A desirable reconstruction method should be able to deal with irregularities caused by noise and non-uniformity of the data collected. It should also be able to deal with complex surface topologies as well. In addition, the reconstructed surface should be representative of the point cloud. It should be reasonably smooth yet maintain discontinuities.

Many approaches have been proposed to solve the surface reconstruction problem. In general, the solutions can be categorized by the surface representation they use—i.e, explicit or implicit. In an explicit representation, the location of all the points on the surface is described precisely. Triangulations methods yield such representations. Conversely, in an implicit representation, we describe the surface as a constraint in a higher dimension, or 3D space in our case. Implicit representations are often advantageous because they lead to solutions capablable of handling complex topologies. Their main drawback is they tend to be more expensive in time and space.

Level set methods are a class of techniques for the deformation of implicit surfaces according to arbitrary rules. These methods were developed conjointly by Osher and Sethian [9] and are widely applicable to a variety of problems. In the context of surface reconstruction, Zhao showed [10] how to apply these methods to minimize a given energy functional which is analogous to a least-square fitting on a point cloud. This is the approach we will explore in this report.

Our final project was motivated by the arrival of low-cost depth sensors, such as the Microsoft's Kinect. The opportunities for object and scene recon-

Figure 1: In this depth image taken using Microsoft's Kinect sensor, we see holes, represented as black regions, where the sensor couldn't take measurements.

struction using these sensors are obvious. These depth sensors provide accurate and dense measurements from structured light. A caveat however is the depth data from such sensors have large holes due to the shadows casted by the objects being measured. This poses extras challenges for surfaces reconstruction.

2 Level set method

The level set method involves the evolution of a function, called ϕ, using an iterative scheme for numerical integration. The surface we wish to recover is represented as a constraint on ϕ. Specifically, we define this surface as the level set $\Gamma = \{\mathbf{x} : \phi(\mathbf{x}, t) = 0\}$.

The central problem of the level set method is to propagate the surface Γ like a firefront. If a velocity field $\vec{v}$ gives the direction and speed of each point for movement, then the evolution of the surface Γ over time t is described by the equation

$$\frac{\partial \Gamma}{\partial t} = \vec{v} \tag{1}$$

This can be extended to all level sets of ϕ to yield the fundamental level set equation

$$\frac{\partial \phi}{\partial t} + \vec{v} \cdot \nabla \phi = 0 \tag{2}$$

Moreover, we can observe that it is only the normal component F of the velocity field to level sets that moves the surface. So by reformulating

$$\vec{v} \cdot \nabla\phi = \vec{v} \cdot \frac{\nabla\phi}{|\nabla\phi|}|\nabla\phi| = F|\nabla\phi| \tag{3}$$

we find another form of the level set equation (2) more suitable for computation

$$\frac{\partial\phi}{\partial t} + F|\nabla\phi| = 0 \tag{4}$$

We have yet to define the level set function ϕ. One particularly attractive definition is the signed distance function $d(\mathbf{x})$, which gives the distance of a point to the surface Γ and the sign: generally $d > 0$ if the point $\mathbf{x}$ is outside and $d < 0$ if it is inside of the surface (assuming it is a closed surface). Although all definitions of ϕ are equally good theoretically, we prefer the signed distance function to avoid numerical instabilities and inaccuracies during computations. But even with this definition, ϕ will not remain a signed distance function and we may need a reinitialization procedure to keep the level set intact [7].

2.1 Discrete time and space formulation

Given that the motion of ϕ is formulated as a differential equation, we can solve it using an iterative scheme for numerical integration. As most differential equations which might appear fairly innocuous at first, solving them numerically can present quite a challenge.

Assuming we have an initial value for ϕ^0 at time $t = 0$, we can use the the finite difference formula for the derivative to estimate

$$\frac{\partial\phi}{\partial t} \approx \frac{\phi^{n+1} - \phi^n}{\Delta t} \tag{5}$$

and to produce the first-order approximation of the solution

$$\phi^{n+1} = \phi^n - \Delta t\, F|\nabla\phi| \tag{6}$$

However, we cannot use this scheme to differentiate reliably in space to get $\nabla\phi$ as this approximation will suffer from stability problems in the presence of topological changes. Therefore, a different numerical procedure is needed.

Upwind Scheme One approach for discretizing PDEs, originally defined by Couran, Isaacson and Rees [3] is called the upwind scheme. Instead of using central differences for approximating the first derivative, the scheme uses both the forward or the backward difference formulæ. So if we assume ϕ is discretized with respect to space on a uniform 3D grid where the entries are

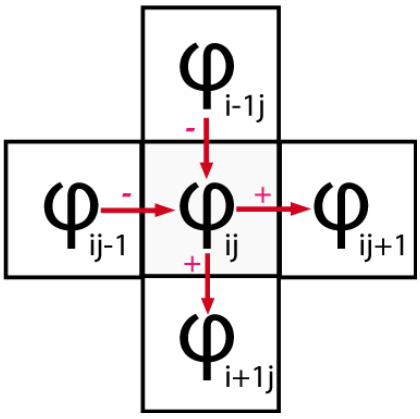


Figure 2: Illustration of the forward and backward differences over a 2D grid.

indexed by $\phi_{i,j,k}$ in the x, y and z directions respectively, then the first-order forward and backward difference formulæin the x direction are

$$D^{+x} = \frac{\phi_{i+1,j,k} - \phi_{i,j,k}}{\Delta x} \tag{7}$$

$$D^{-x} = \frac{\phi_{i,j,k} - \phi_{i-1,j,k}}{\Delta x} \tag{8}$$

We have analoguous formulæfor other axes y and z. Figure 2 illustrate the formulæin the 2D case. These lead to a simple first-order scheme proposed by Osher and Sethian [6] to estimate the level set evolution

$$\phi_{ijk}^{n+1} = \phi_{ijk}^{n} - \Delta t[\max(F_{ijk}, 0)\nabla^{+}\phi + \min(F_{ijk}, 0)\nabla^{-}\phi]$$

where

$$\nabla^{+} = \begin{bmatrix} \max(\phi^{-x}, 0)^2 + \min(\phi^{+x}, 0)^2 + \\ \max(\phi^{-y}, 0)^2 + \min(\phi^{+y}, 0)^2 + \\ \max(\phi^{-z}, 0)^2 + \min(\phi^{+z}, 0)^2 \end{bmatrix}^{1/2} \tag{9}$$

$$\nabla^{-} = \begin{bmatrix} \max(\phi^{+x}, 0)^2 + \min(\phi^{-x}, 0)^2 + \\ \max(\phi^{+y}, 0)^2 + \min(\phi^{-y}, 0)^2 + \\ \max(\phi^{+z}, 0)^2 + \min(\phi^{-z}, 0)^2 \end{bmatrix}^{1/2} \tag{10}$$

As opposed to computing the magnitude of the gradient using the central differences functions, we found the evolution of ϕ under the upwind scheme to be much more stable. We show in figure 4 and 5 the result of applying different schemes on the initial level set function shown in figure 3.

Note, other higher-order schemes are also available when more stability and accuracy are required.

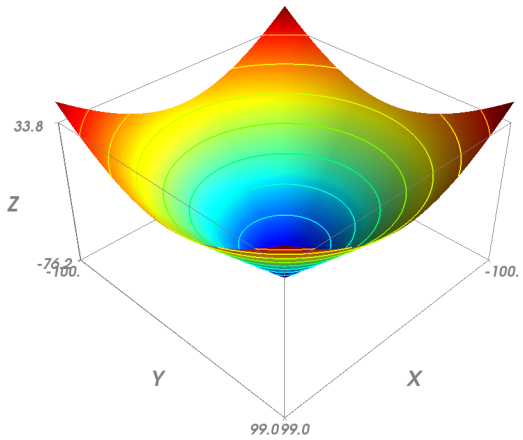


Figure 3: Initial level set function ϕ defined as the signed distance function to a 2D curve.

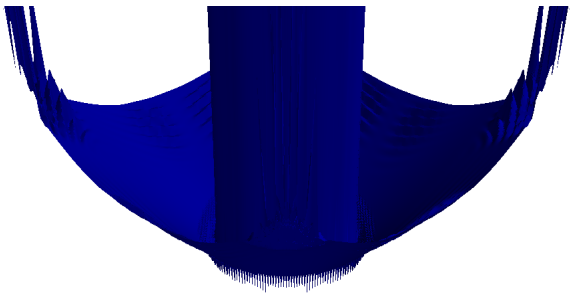


Figure 4: Result of 140 iterations using $\Delta t = 0.5$ and a constant force $F = 1$ using the central differences approximations. The resulting ϕ has exploded in the center and near the borders.

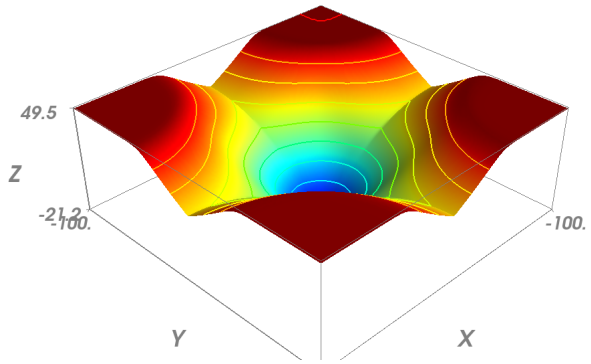


Figure 5: Result of 140 iterations using $\Delta t = 0.5$ and a constant force $F = 1$ using the upwind scheme. The level set evolution is very stable in case. It ultimately reaches a steady state.

Implementation In our implementation, we used a few strategies to implement the simple upwind scheme. We used NumPy [5] dense matrix representation to store the discretized level set function. Also note that only one matrix was used to store the results because the squared differences can be added on top of each other. Finally for the boundaries of the grid, we mirrored the inward row/column when the difference formulærequired entries outside of the matrix.

Adaptive time steps Based on the work on level set stability and convergence by Chaudhury and Ramakrishnan [2], we were able to set the size of time steps Δt adaptively during the evolution to reach steady state as fast as possible. Simply stated, we use the convergence condition for the level set evolution

$$\Delta t \leq \frac{\min(\Delta \mathbf{x})}{F_{max}}$$

where F_{max} is the maximum absolute value of the entries of F and $\Delta \mathbf{x}$ is the grid spacing. Then, the optimal time step is given by

$$\Delta t_{opt} = c\frac{\min(\Delta \mathbf{x})}{F_{max}}$$

where c is a reasonable safety factor. We found value of c between 0.8 and 0.9 to work well in our implementation.

Reinitialization As stated in section 3.1, a reinitialization step might be needed, especially for forces using mean curvatures. In essence, the reinitialization will reset the level set function to be a signed distance function to the surface, such that it satisfies the following Eikonal equation

$$|\nabla d(\mathbf{x})| = 1, d(\mathbf{x}) = 0, \mathbf{x} \in \Gamma$$

We solved this using an implementation of the Fast Marching Method [**?**] provided by the Scikit library.

2.2 Forces governing surface evolution

Up until now, we have left one variable undefined, namely the force matrix F. The definition of this term very much depends on the application of the level set method. For example, a popular approach used in 2D segmentation of medical images employs the magnitude of gradients of an image to limit the evolution of the level set within a uniform region. However, the force must be derived differently to reconstruct a surface from a point cloud. A formulation was specifically developed by Zhao [11] [10] for reconstruction from unorganized datasets.

Stating his formulation upfront, we have

$$F = \nabla d(\mathbf{x}) \cdot \frac{\nabla \phi}{|\nabla \phi|} + d(\mathbf{x}) \left(\nabla \cdot \frac{\nabla \phi}{|\nabla \phi|} \right) \tag{11}$$

$$= \nabla d(\mathbf{x}) \cdot \vec{n} + d(\mathbf{x})\kappa \tag{12}$$

where $d(\mathbf{x})$ is the distance to the closest point in our dataset S to any point $\mathbf{x}$, is the unit normal vector to the level sets and κ is the mean curvature. The effect of this force is twofold: it is simultaneously attracting the surface to the dataset while maintaining its smoothness.

Energy functional To understand where this force formulation comes from, we have to understand how the desired level set surface is derived from an energy functional. Without restating all the derivations, this functional essentially maps a surface to a value which assesses its quality in terms of its distance to S. More precisely, Zhao [11] defined it as follows

$$E(\Gamma) = \left[\int_{\Gamma} d^p(\mathbf{x}) ds \right]^{1/p} \tag{13}$$

where ds is the surface area, and p is a weighting parameter for the distance function—which works analogously as for a p-norm. Assuming $p = 1$, it can be shown that the deformed surface Γ under F will minimize (13) when it reaches the steady state

$$\nabla d(\mathbf{x}) \cdot \vec{n} + d(\mathbf{x})\kappa = 0$$

For more detailed explanations, read [11], [10] and [8].

Unsatisfactory local minimum Note that the level set can reach an unsatisfactory local minimum if the initial value was not close enough to the final surface. As such, it is expected that this methods results in some loss of details.

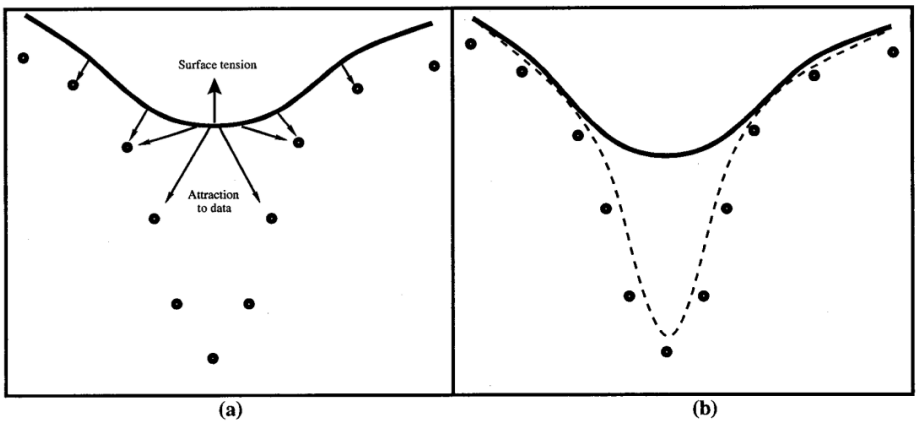


Figure 6: This figure taken from [8] illustrates how the level set might be unable to reach the bottom of a crevasse. The local minumum (b) is reached from (a). The surface tension prevents the curve from bending and lower points cannot attract the surface further down because they are not the closest.

2.3 Source of errors

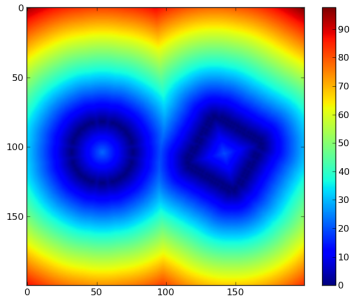
As stated earlier, numerical solutions for solving the level set equation can very easily become unstable. We found the sources of errors noted Sethian [**?**] to be representative of issues we found in our implementation

- **Reinitialization**: If reinitialization is used, the interface can be displaced to incoherent positions, affecting numerical accuracy in an undesirable way. This is why it is normally avoided as much as possible. Methods have been developped to avoid the need for reinitialization entirely, such as [4] by *Li, Xu, Gui and Fox*.
- **Incorrect gradient**: As stated earlier, central differences should be avoided.
- **Order of approximation**: First order approximations can lead to numerical diffusion, higher order methods are prefered.
- **Force (or speed) function**: The force function might be well-defined around the level set but could distort ϕ further out, requiring reinitialization procedures.

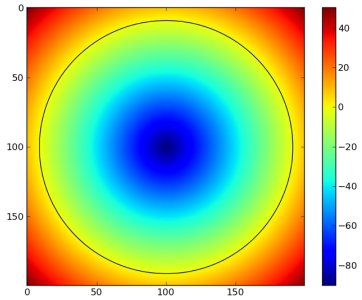
3 Results

In order to implement of the level set evolution as described above, we first favored a 2D version implementation in Python (using a combination of NumPy, SciPy, Scikit and Matplotlib). Here we present our final 2D results.

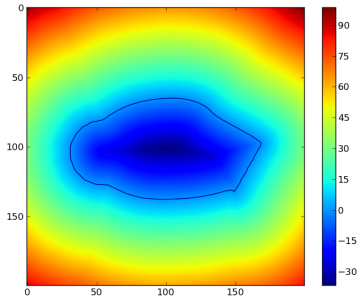
(a) Initial data set of points.



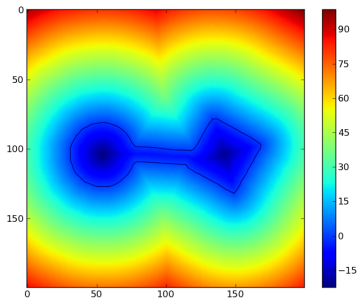
(b) Unsigned distance function to the data set of points.



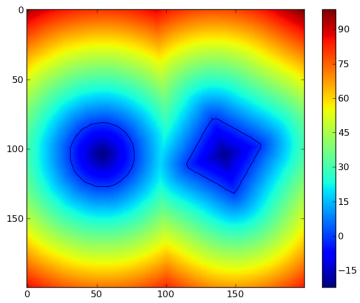
(c) Initial ϕ with the embedded level set curve Γ.



(d) ϕ and Γ, some frames later.



(e) ϕ and Γ before the topology change.

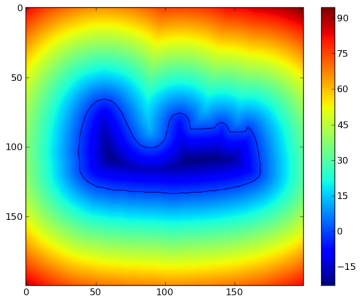


(f) Final stable state of ϕ and its level set Γ.

As stated in section 3.3, since our implementation is based on Zhang's force formulation it might sometimes suffer from loss of surface details. We con-

firmed this expected behavior using a shape with pronounced concavities.

(g) Shape whose outline acts as S.



(h) The level set reaches an unsatisfactory local minimum.

4 Conclusion

In order to reconstruct surface so that it conforms to a data set of points, we have seen how to represent a surface implicity through a level set function ϕ and how to evolve it according to a given velocity field. We have detailed numerical discretization methods used to solve the level set equation. Furthermore using Zhao's force formulation, we have detailed how the level set surface acts as an elastic membrane trying to minimize its area, effectly "shrink wrapping" the data points.

While this method is powerful in dealing with complex topologies and can certainly help smooth noisy input, it proved to be very challenging in regard to the numerical stability and convergence. The speed of convergence proved to be notably slow both in terms its step size (even with the adaptive technique) and in terms of raw computational complexity.

We have attempted to transfer our 2D implementation to a 3D grid, but the results were either incorrect due to the grid size being too small or impossibly slow to converge due to cubic growth. To deal with this lack of efficiency, the optimization method developed by Adalsteinsson and Sethian [**?**] called the narrow band method is almost obligatory for the level set method to be reallistically usable. Future work for applications using the Kinect data should therefore make it their priority. One might even attempt to harness the power of the GPU for real-time applications.

References

[1] D. Adalsteinsson, *A fast level set method for propagating interfaces*, Ph.D. thesis, University of California, 1994.

[2] J.D. Boissonnat, *Geometric structures for three-dimensional shape representation*, ACM Transactions on Graphics (TOG) **3** (1984), no. 4, 266–286.

[3] K.N. Chaudhury and KR Ramakrishnan, *Stability and convergence of the level set method in computer vision*, Pattern recognition letters **28** (2007), no. 7, 884–893.

[4] R. Courant, E. Isaacson, and M. Rees, *On the solution of nonlinear hyperbolic differential equations by finite differences*, Communications on Pure and Applied Mathematics **5** (1952), no. 3, 243–255.

[5] C. Li, C. Xu, C. Gui, and M.D. Fox, *Distance regularized level set evolution and its application to image segmentation*, Image Processing, IEEE Transactions on **19** (2010), no. 12, 3243–3254.

[6] Travis E. Oliphant, *Guide to NumPy*, Provo, UT, March 2006.

[7] S. Osher and J.A. Sethian, *Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations*, Journal of computational physics **79** (1988), no. 1, 12–49.

[8] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang, *A PDE-based fast local level set method*, Journal of Computational Physics **155** (1999), no. 2, 410–438.

[9] P. Savadjiev, *Surface recovery from three-dimensional point data*, Ph.D. thesis, McGill University, 2003.

[10] J. A. Sethian, *Evolution, implementation, and application of level set and fast marching methods for advancing fronts*, J. Comput. Phys. **169** (2001), no. 2, 503–555.

[11] JA Sethian, *Advancing interfaces: level set and fast marching methods*, ICIAM, 1999.

[12] J.A. Sethian, *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, no. 3, Cambridge Univ Pr, 1999.

[13] H.K. Zhao, S. Osher, and R. Fedkiw, *Fast surface reconstruction using the level set method*, Variational and Level Set Methods in Computer Vision, 2001. Proceedings. IEEE Workshop on, IEEE, 2001, pp. 194–201.

[14] 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** (2000), no. 3, 295–314.