Point–Based Level Sets and Progress Towards Unorganised Particle Based Fluids

by

Richard D. Corbett

B.Eng., Memorial Univerity of Newfoundland 2003

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES

Computer Science

The University of British Columbia

June 2005

Abstract

We combine the advantages of level sets, particle level sets, and point-set surfaces in a simple new framework, where we store signed distance values on a volume of scattered points. This permits easy and effective surface operations of appropriate quality for graphics and animation. We present techniques such as accurate deformation by advection, constructive solid geometry, mean curvature flow and more general surface flows as well as a strategy to convert triangulated meshes to this point representation. Finally, it is shown that by harnessing the information generated during implicit surface evolution, ray-traced renderings are close at hand.

An investigation into employing the point-based level set techniques for fluid animation is presented. Here, strategies for a particle based Poisson solves and potential boundary conditions are discussed. Although some positive results are achieved, more research is required to correctly handle the necessary boundary conditions and discontinuities for fluid animation.

Contents

Abstract ii

Contents iii

Acknowledgements vi

1 Introduction 1

1.1 Level sets and point-set surfaces 1

1.2 Comparison 2

1.2.1 Merging and pinching 3

1.2.2 Volume operations 3

1.2.3 Adaptivity 3

1.2.4 Numerical diffusion 4

1.3 Other alternatives 4

1.3.1 Particle level set 4

1.3.2 Radial basis functions 5

1.4 Point-based level set 5

2 Definition 6

2.1 MLS interpolation 6

2.2 Dolbow interpolation 8
2.3 Adaptivity 10
2.4 Acceleration 10

3 Signed Distance Computation **12**
3.1 Reinitializing 12
3.2 Marching 13

4 Geometric Operations **17**
4.1 Deformation by advection 17
4.2 Boolean operations 18
4.3 Motion by mean curvature and general flows 19

5 Mesh Conversion **22**
5.1 Particle placement issues 23

6 Rendering **26**

7 Towards Fluid Animation **29**
7.1 Point based animation 29
7.2 Previous fluid animations 30
7.3 Motivation 32
7.4 Navier-Stokes solution 32
7.5 Particle advection 33
7.6 Particle fluids 34
7.7 Approximate pressure solve 35
7.7.1 Projection 35
7.7.2 Poisson system 36

7.7.3 Filtering 37

7.7.4 Discontinuity handling and boundary conditions 38

7.7.5 Approximate projection 41

8 Conclusions 45

8.1 Future work 46

Bibliography 47

Acknowledgements

First and foremost I'd like to thank Robert Bridson for offering so many hours of his time to my educational cause. His patience and enthusiasm for teaching have made pursuing this degree an incredibly enjoyable academic experience.

I would like to thank Ian Mitchell for his helpful comments in the preparation of this thesis.

Finally, I want to thank all my friends and family who have held me together for these past two years. Of course, Payal gets the largest share of this credit (shukriya!).

RICHARD D. CORBETT

The University of British Columbia
June 2005

Chapter 1

Introduction

The range of geometric operations required in graphics has driven the development of a variety of different geometric representations. This thesis introduces one such representation, the point-based level set, where signed distance values are sampled on a volume of scattered points. Our aim is to combine the advantages of two existing complementary approaches, point-set surfaces and grid-based level sets in a new tool appropriate for graphics applications.

The challenges of maintaining mesh connectivity and good parameterization in the face of deformation and more extreme geometric operations have encouraged researchers to look at unparameterized representations. In this section we review a number of such representations.

1.1 Level sets and point-set surfaces

An implicit surface can be defined in 3D as the set of points (x,y,z) satisfying

$$f(x,y,z) = c \tag{1.1}$$

where f is some function, and the constant c is often chosen to be zero. One common choice for f is signed distance: the magnitude of $f(x, y, z)$ is equal to the distance from (x, y, z) to the surface and the sign of f indicates whether (x, y, z) is inside or outside the surface. For example,

$$\sqrt{x^2 + y^2 + z^2} - 5 = 0 \tag{1.2}$$

defines a sphere centered at the origin, and of radius 5. For any point, the signed distance is also computed through this equation. Plugging point $p = [0, 0, 0]$ into the left side of Equation 1.2 will give a right side of -5, showing that p is inside the surface (negative) and 5 units from the surface.

Level sets, introduced by Osher and Sethian [26], are an Eulerian representation for dynamic implicit surface simulation. In general, the implicit surface is defined by a grid of signed distance samples. The implicit surface function (i.e. approximate signed distance to the surface) can be readily evaluated by interpolation of the values at the grid nodes. The regular sampling allows accurate and efficient finite difference approximations when derivatives are needed.

Point-set surfaces [1] [17] [2], are representations of geometry through an arbitrary point sampling of the surface. Often, this geometric representation is the result of 3D range scanning. A surface can be easily reconstructed from the point sampling using the *moving-least-squares* (MLS) construct.

1.2 Comparison

In this section, we iterate some of the advantages and disadvantages of the above representations, motivating a solution that can combine the attractive features of both.

1.2.1 Merging and pinching

One of the most attractive strengths of the level set approach is its capability for numerical regularization. This property allows for merging and pinching of the implicit surface as it evolves through the grid during simulations such as mean curvature flow. Operations such as Constructive Solid Geometry (CSG) [24] are also possible with this representation.

Point-set surfaces are not suitable for general surface evolution. Operations such as mean curvature flow involving topological changes have not yet been addressed and CSG operations are also difficult.

1.2.2 Volume operations

Level sets are capable of fast volume operations such as approximating distance and direction to the surface from anywhere in space. Maintenance of the signed distance field throughout the volume is possible through signed distance marching and reinitialization [27].

Point-set surfaces have very little, if any, off-surface representation, making them a limited volumetric approach. For example, it can be difficult to determine whether an arbitrary position in space is inside or outside a point-set surface.

1.2.3 Adaptivity

It is often desirable to have spatially varying resolution in a geometric representation. Point-set surfaces afford such capability by varying each interpolation radius and the density of the point locations [1].

Although spatially varying resolution is possible in the level set representation, it comes at a significant cost in implementation complexity. Such an approach is presented in [18], where an octree with higher resolution near the implicit surface is used for high resolution fluid animation.

1.2.4 Numerical diffusion

Level sets are notorious for numerical diffusion where even simple motion such as 3D translation and rotation will cause smoothing of the surface, despite the use of sophisticated high-resolution 5th-order HJ-WENO methods [6]. Because of the fixed grid representation, surface features can never be exactly preserved during general translations or rotations.

Point-set surfaces can easily be translated and rotated. Since this is a Lagrangian representation, diffusion is not a problem and surface features are guaranteed to be preserved: the surface is exactly invariant under rigid body motion.

1.3 Other alternatives

The following representations attempt to combine the positive features of the above-mentioned representations.

1.3.1 Particle level set

Enright *et al* [6] finalized the particle level set method in an effort to combat the diffusion in the original level set representation. Here, particles were used in conjuction with a grid. The particles are moved at each time step with the grid velocities and are used to fix numerical diffusion errors in the grid-based level set after the implicit surface is evolved. This strategy presents substantial improvements over the basic level set, but still does not perfectly preserve shape for even basic flows.

1.3.2 Radial basis functions

There has been recent work on the use of *radial basis function* (RBF) representations for geometric operations [3]. This is almost exactly the level set representation, using unmoving RBFs instead of a fixed grid. This work is focused on solving Hamilton Jacobi PDEs in higher dimensions, and it remains to be tested for the lower order, but fast, approximations required for graphics. Furthermore, there is no discussion of what can be done by moving the points, or whether this would be beneficial. One attractive feature of this representation is the ease of incorporating adaptivity.

1.4 Point-based level set

The following chapters describe the point-based level set, which is our attempt to combine the attractive features of grid-based level sets and point-set surfaces into one simple framework. By leveraging the *moving-least-squares* (MLS) construct, a volumetric level set representation is created through a set of arbitrarily placed signed distance samples. This representation is capable of simple adaptivity, and geometric operations. Techniques for triangulated mesh conversion and signed distance computation are presented. Finally, efforts towards two-phase fluid animation using this representation are described.

Chapter 2

Definition

We represent our geometry with an arbitrarily organized set of signed distance sample points. At each point, interpolation of gradient, curvature, and signed distance can be achieved through construction of a *moving-least-squares* (MLS) solve. In situations where MLS solves are carried out at all sample points in the signed distance field, further interpolation can be sped up at the cost of accuracy using Dolbow interpolation.

An example of our representation is shown in Figure 2.1. With this set of unordered points we are able to achieve what previously required a point-set representation, coupled with either an embedded regular grid [13] [14], or an embedded octree [25].

2.1 MLS interpolation

At sample point j we fit a quadratic polynomial, $S(x)$, to the data in its local neighborhood:

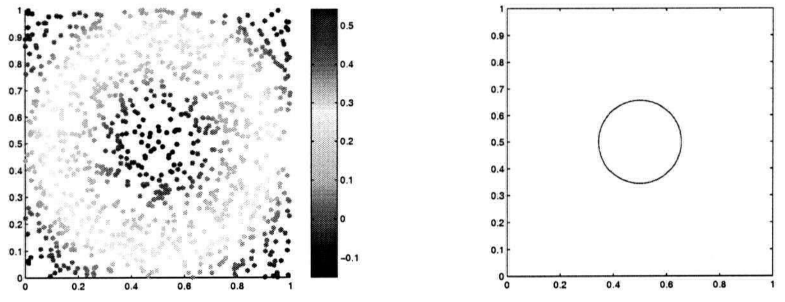
Figure 2.1: These images are an example of a point cloud representing a geometric shape. The colored points in the left image are samples of the signed distance field about a circle. Arbitrary organization of the points is possible, as well as the use of extra points in areas where more resolution is desired (See Figure 2.3).

$$\min \sum_{i=1}^{N} (S(x_{ij}) - s_i)^2 w(x_{ij}) \tag{2.1}$$

where we are minimizing over the space of quadratics, s_i are the signed distance values in the local neighborhood, and $x_{ij} = x_i - x_j$. The function $w(x_{ij})$ is the weighting, or kernel, function, centered at j,

$$w(x_{ij}) = (1 - |x_{ij}|/R)^4(4|x_{ij}|/R + 1): \quad |x_{ij}| < R \tag{2.2a}$$

$$w(x_{ij}) = 0: \quad |x_{ij}| \geq R \tag{2.2b}$$

R is the kernel radius, which defines the size of the local neighborhood to consider, which in turn, controls the amount of smoothing (large R, lots of smoothing). If enough points are present, a small kernel radius can be used, thereby preserving smaller features.

Equation 2.1 can be solved by the usual minimizing procedure of taking the derivative and setting it to zero. In 1D, after some rearranging, this results in the linear system

$$\sum_{i=1}^{n} w(x_{ij}) \begin{bmatrix} 1 & x_{ij} & x_{ij}^2 \\ x_{ij} & x_{ij}^2 & x_{ij}^3 \\ x_{ij}^2 & x_{ij}^3 & x_{ij}^4 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \sum_{i=1}^{n} w(x_{ij}) \begin{bmatrix} 1 \\ x_{ij} \\ x_{ij}^2 \end{bmatrix} s_i \tag{2.3}$$

which we use LAPACK to efficiently solve.

Once the 10 quadratic coefficients of S for a 3D simulation are known at a point, we can treat operations on the signed distance at that point as operations on S. Explicity, each locally defined trivariate quadratic is:

$$S = A + Bx + Cy + Dz + Exy + Fyz + Gxz + Hx^2 + Iy^2 + Jz^2 \tag{2.4}$$

From which a local gradient can be computed as

$$\nabla S = (B, C, D) \tag{2.5}$$

and the local mean curvature as

$$\nabla \cdot \frac{\nabla S}{|\nabla S|} = 2\left(\frac{B^2I - BCE + C^2H + B^2J - BDF + D^2F + C^2J - CDG + D^2I}{(B^2 + C^2 + D^2)^{1.5}}\right). \tag{2.6}$$

2.2 Dolbow interpolation

Dolbow and Belytchko [5] suggest leveraging the above solutions by calculating the weighted average of the local S functions evaluated at an arbitrary point of interest. This interpolation of quadratics can be done using zeroth order approximation to achieve similar results to linear-fit MLS interpolation of the original points (see below). With this approach, we can approximate solutions at arbitrary points without more MLS solves. Although the result is similar to a linear fit to the original

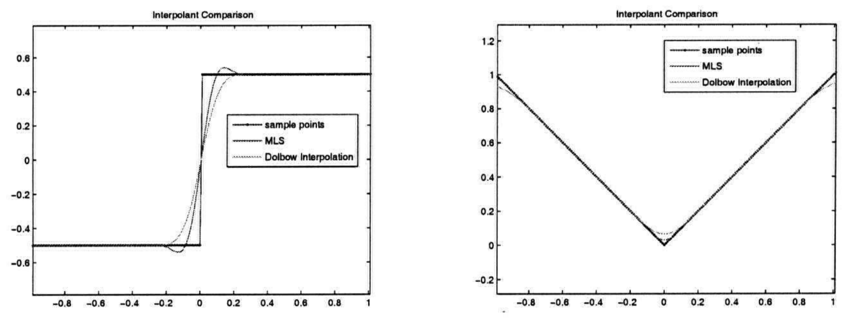

Figure 2.2: The above graphs display results of both MLS interpolation and Dolbow interpolation on discontinuous data. The MLS interpolation gives a truer approximation at discontinuities, however, the Dolbow interpolation can be speedier in situations where MLS solves have been done and further evaluations are required. The Dolbow interpolation gives similar results to linear-fit MLS interpolation of the original data. The benefits of the Dolbow technique are that interpolations at new locations are easily done at a lower computational cost, and that interpolation of higher order operations such as curvature is possible.

data, we can now evaluate higher order operators (such as curvature) that cannot be achieved with linear-fit approximation. This operation is formally described below.

Given

$$S(x) = P(x)^T a(x) \tag{2.7}$$

where

$$P(x)^T = [1 \quad x \quad y \quad z \quad xy \quad xz \quad yz \quad x^2 \quad y^2 \quad z^2] \tag{2.8}$$

is the 3D quadratic basis and $a(x)$ is a vector of coefficients solved for in the MLS solve, field interpolation is attainable at point x_j through a weighted average of the local $S(x_i)$ functions evaluated at x_j:

$$\phi(x_j) = \frac{\sum_{i=1}^{N} S_j(x_{ij})w(x_{ij})}{\sum_{i=1}^{N} w(x_{ij})} \tag{2.9}$$

where x_{ij} is $x_i - x_j$. It follows that:

$$\begin{aligned} \nabla \phi(x_j) &= \nabla \left(\frac{\sum_{i=1}^{N} S_j(x_{ij}) w(x_{ij})}{\sum_{i=1}^{N} w(x_{ij})} \right) \qquad (2.10) \\ &= \frac{\sum_{i=1}^{N} \nabla S_j(x_{ij}) w(x_{ij})}{\sum_{i=1}^{N} w(x_{ij})}. \qquad (2.11) \end{aligned}$$

As shown in Figure 2.2, Dolbow interpolation, although allowing for faster interpolation once MLS solves have been done, does introduce more smoothing into the system. It also requires more memory to retain each of the solution vectors $a(x)$ for future processing.

2.3 Adaptivity

By placing extra points where higher resolution is desired, it is possible to compute MLS estimations of the geometry with a smaller kernel radius, reducing the spatial smoothing. This can be done adaptively during each MLS solve by noting the number of points accumulated within the kernel radius and recursing with a larger radius, if necessary. This allows for a small default radius to be chosen, and for solves in sparsely populated areas, the radius can be increased without user interaction. An example of a sharp corner being interpolated with MLS is shown in Figure 2.3.

2.4 Acceleration

To accelerate the search for adjacent points in a simulation we partition the simulation space into an acceleration grid. Since all operations are local, we usually only need to check grid cells adjacent to the point of interest during an MLS solution. If the minimum number of required points cannot be found in the set of adjacent grid cells, it is easy to query the grid cells as far away as needed. However, we note that

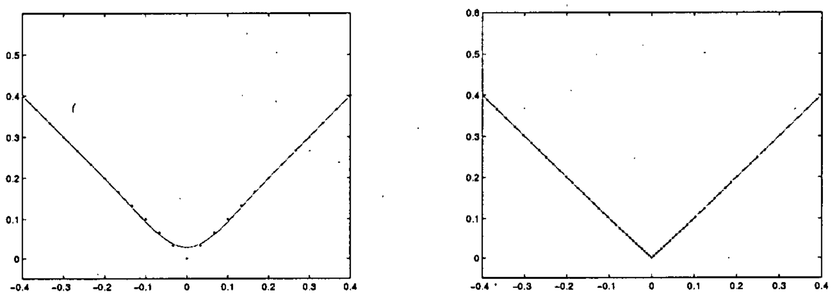

Figure 2.3: The figure on the left shows MLS sharp feature interpolation without adaptivity. The figure on the right shows the interpolation resulting from the adaptive approach.

for very large variations in kernel radius, a uniform acceleration grid is inefficient. We recommend a grid cell size equal to the default MLS kernel radius.

Chapter 3

Signed Distance Computation

Given the movement of particles, or signed distance evolution over time, a proper signed distance field will eventually require upkeep of the signed distance values. Two major strategies can be used for maintaining a signed distance field:

- Reinitializing
- Marching

This chapter visits both of these strategies, describing how they can be implemented with the point-based level set representation.

3.1 Reinitializing

Reinitialization can be achieved by solving the following PDE to steady state in pseudo-time t (i.e. not simulation time):

$$\phi_t + sign(\phi_0)(|\nabla\phi| - 1) = 0. \tag{3.1}$$

We can adopt a semi-Lagrangian approach in the solution of this equation, which involves computing the gradient of ϕ at all the sample points, looking towards the

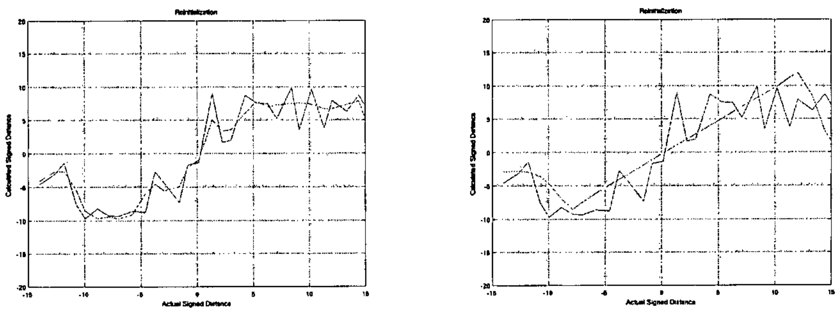

Figure 3.1: Here, two frames of a 1D reinitialization scheme are shown. The blue lines show the extremely noisy signed distance data to be reinitialized. The red lines show the current signed distance values after one iteration (left image) and 18 iterations (right image). Note that the signed distance domain may never become monotonic if the original data contains erroneously signed gradients.

interface along the gradient, and interpolating to find the new value (a more accurate Eulerian ENO alternative to this can be found in [3]).

Although these operations are possible within the point-based level set framework, it is not a solid strategy because there are situations where it fails to correct errors in the signed distance field. Most notably, in areas where the gradient erroneously switches sign, signed distance values will flow towards the interface, instead of out from it. Even if care is taken to avoid switching signs at sample points during this process, some unwanted kinks in signed distance apparently will never be fixed. A 1D example of this scenario is shown in Figure 3.1.

3.2 Marching

In the absence of a robust reinitialization technique, marching values away from the implicitly defined interface was investigated. This technique can be done in less time than reinitialization and is less sensitive to existing errors in the data.

We can reconstruct a signed distance field from approximate initial signed distance values on the points. We only require approximate signed distance values for the points near the interface, and the others can be initialized during the marching process, which is an adaptation of the grid-based strategy found in [34].

Our strategy involves the following sets of particles:

- P is the set of projected surface particles that sample the zero level set. Once this set is constructed it does not change during marching.
- I is the set of initialized particles. Each of these particles knows which particle in P it is closest to, and its distance to it.
- U is the set of uninitialized particles that are farther than one kernel radius away from any member of I.
- H is a heap containing the uninitialized particles that are within one kernel radius of particles in I. Each of these elements keeps track of which element in P it believes it is closest to, and its distance to that point (candidate distance). The element with the smallest candidate distance is at the top of the heap.

We first create set P by selecting the points that are within a kernel radius of points on the opposite side of the interface (i.e. have opposite sign), copying them, and projecting them onto the interface as suggested in [2] to create a temporary point-set surface representation of the zero level set. This is done by calculating the gradient of the signed distance in the local neighborhood of the new point's starting position and then moving the point closer to the interface along this direction. At the new location, we again calculate the gradient, and again move the point closer to the interface. This process is repeated until the point has been projected to a location with an interpolated signed distance magnitude lower than some threshold. Once this is done for each near-interface particle, we save them in P.

I is first populated by adding to it the points that were projected to the interface. These points can easily calculate their signed distance as the distance between their starting location and the final projected location. The closest element in R for each of these points is where it was projected onto the interface.

H is first created by adding all the points outside of I that are within one kernel radius of one of I's elements. This set is maintained as a heap, whose top element will always be the point that has the smallest known distance to any of the elements in P.

U simply contains the points that are both uninitialized and farther than 1 kernel radius away from any member of I.

Marching is the process of taking the point off the top of the heap, adding it to I and updating the nearby points in the heap. The point from the top of the heap will likely be within one kernel radius of some points in U. These points should be moved to H, and have their candidate distances set to their distances to the point in P that the top element was closest to. Also, any current elements of H that are within one kernel radius of the top element's position should check to see if the surface particle from P that the top element was closest to is closer than the element's current closest point.

Continuing the above migration of particles from U to I until there are no more particles in U will march the signed distance throughout the domain in $O(n \log n)$ time.

Of course, the marching algorithm described above requires that all points in the domain be connected in the network of points. This means that no points are allowed to be separated by more than a kernel radius from all other points. In the case of such an isolated point, it is possible to attempt to look for adjacent points within a larger radius, but this may affect both the speed and the accuracy of the signed distance marching.

Chapter 4

Geometric Operations

In graphical applications ranging from modeling/sculpting to simulation there are a number of geometric operations that a surface representation should easily support. We present here a variety of such operations where point-based level sets work well.

4.1 Deformation by advection

Deformation by advection in a continuous velocity field is handled simply by moving the points. Grid-based level sets cannot do this without introducing unwanted smoothing, overcome by the somewhat complex particle level set method [6]. While point-set surfaces are sometimes just as effective (and can, in fact, be viewed as a particular subcase of our method), applications such as fluid animation require a degree of regularization—to handle merging and breaking—that is not found in point-set surfaces. We do not need to include particles exactly on the surface, and thus can attain the same attractive regularization as grid-based level sets without the attendant numerical diffusion.

To increase accuracy during an advection step, we copy and project the near-

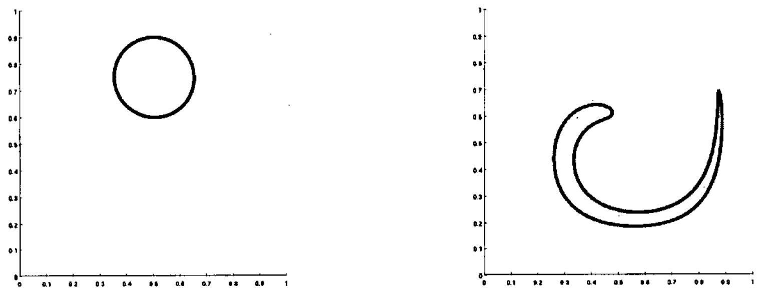

Figure 4.1: By placing points densely around the interface, we are able to maintain a smoothed closed surface during deformation by advection.

interface particles onto the surface, and move them with the flow. Then, at the end of the time step we can use these temporary points as our zero level set to march out from as described in Chapter 3. This solution still maintains the attractive regularization properties of level sets.

In some applications, permanently carrying around points on the zero level set can be useful (as seen in Figure 4.1). Other times, when surfaces are merging, keeping points on the interface will eliminate the desired regularized solution. Thus, for applications such as fluid animation, we advocate not keeping points on the interface, and rebuilding the interface after each time step.

4.2 Boolean operations

Operations from CSG are almost as simple in this representation as they are in level sets: we simply union the point sets of the two objects, and perform the appropriate min/max operation at each point (using either the value sampled there or an MLS estimate from local points as appropriate). Figure 4.2 shows simple examples of CSG operations on two spheres. The evident smoothing at the interface of the two shapes would also be an artifact of grid-based level sets. The smoothing can be controlled to a degree by the level of sampling and the size of the interpolation radius, but even sharper features would be possible using the strategy described in [10].

4.3 Motion by mean curvature and general flows

Unlike the particle level set and point-set approaches, point-based level sets can handle motion by mean curvature,

$$\phi_t = \nabla \cdot \frac{\nabla \phi}{|\nabla \phi|} \tag{4.1}$$

and more general motions in the normal direction such as,

$$\phi_t + c|\nabla \phi| = 0 \tag{4.2}$$

where parts of the surface may be destroyed or created (e.g.[24]). We do this by keeping our points fixed, and evolving their signed distance values with derivative quantities estimated from the coefficients of the MLS solves. A more computationally intensive but more accurate alternative for this one case is described in [3].

With the point-based level set representation, it is possible to combine all of the above operations as required. The particles can be easily moved to deal with advective motion, while the sample values can be updated to deal with mean-curvature flow.

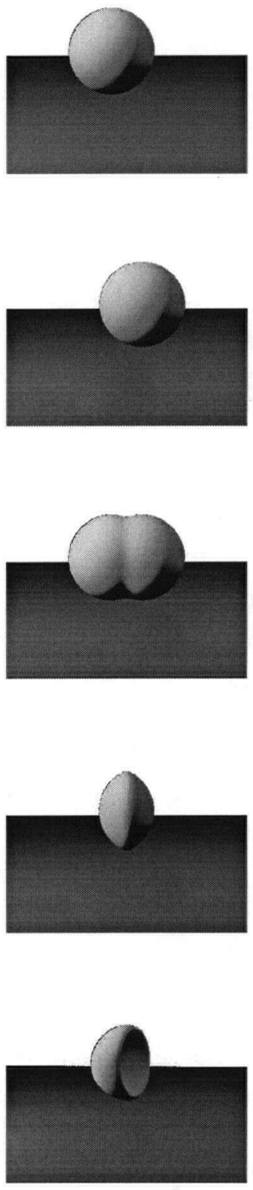

Figure 4.2: Examples of Constructive Solid Geometry (CSG) operation is shown here. The union and intersection of the two spheres is simply done by considering the minimum and maximum of the signed distances at the sample points. The degree of smoothing at the interface can be controlled by the number of points used, as well as the size of the interpolation kernel.

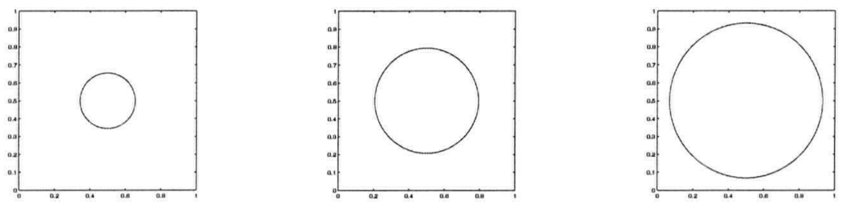

Figure 4.3: These 3 images from a simulation of motion in the normal direction show that the interface expands smoothly. This is achieved by keeping the points in place and changing their signed distance values.

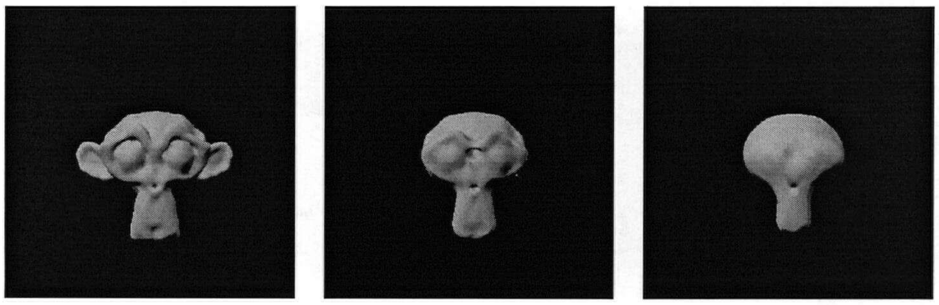

Figure 4.4: Motion is simulated by 3D mean curvature flow in these 3 images. The images were rendered using the strategy described in Chapter 6

Chapter 5

Mesh Conversion

For our point representation to be useful, it is necessary to be able to convert existing geometry stored in standard triangle mesh format. This is straightforward once consistent face normals are available or have been computed.

To create a field of signed distance sample points from a polygonized mesh we follow these steps:

- Load the mesh information retaining only face vertices and face normals.
- Add all the triangle vertices and assign them signed distances of zero.
- Partition any non-triangle convex faces into triangles. This is easily done for any polygon by selecting a starting vertex and walking around the outside of the polygon. By traversing the vertices in either clockwise or counter-clockwise order, triangles can be created by selecting pairs of adjacent vertices and completing the triangle with the starting vertex.
- For each triangle that is longer than the desired kernel radius, we must subdivide it, adding new points spaced less than the kernel radius apart. There are

many ways to do this; our current algorithm introduces a grid of points based on the longest edge. We walk along each of the shorter edges in steps that are a fraction of the kernel radius, and at each step march a vector of points normal to the longest edge. This step is visualized in Figure 5.1.

- At each point that is visited while marching across the face of a triangle add a new point on the surface, as well as one slightly offset to each side of the face. Assign these new points signed distances equal to their offsets, taking care to ensure the signs are correct.

- Once these interface points have been added, and their signed distances assigned, it is possible to scatter points throughout the volume (wherever they are be desired). These new points can be assigned signed distance values using the marching strategy described in Chapter 3.

Of course, the resolution at which points are sampled both on the triangulated surface, as well as anywhere else in the domain, will determine how well sharp features can be resolved.

5.1 Particle placement issues

At a sharp edge between two faces, care must be taken to avoid placing particles in areas that could cause errors. For instance, placing offset particles at sharp edges could result in placing particles of a particular sign on the wrong side of the interface. An example of such a situation is shown in Figure 5.2. This type of problem can be avoided by limiting the distance to offset particles from the interface, as well as taking care not to blindly place particles very close to edges. This has been done

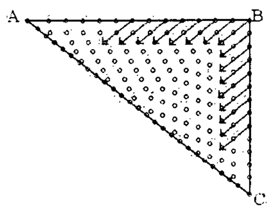

Figure 5.1: Particles have been placed throughout this triangle using the strategy in this section. At each step along edge AB a vector of particles is marched normal to AC until AC is encountered. The green arrows indicate the direction that the marching across the face follows. Similarly, edge BC is traversed, adding vectors of particles towards AC.

for each individual example by tuning the parameters that determine how close to edges points may be placed, as well as the distance to offset points from faces.

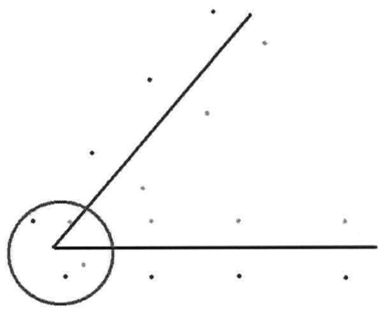

Figure 5.2: This figure shows the importance of taking care when inserting offset points at a sharp edge. The red circle shows where two green particles, which are intended to be on the inside of the shape, have been erroneously placed on the outside. This will cause complications when marching signed distance values away from the interface.

Chapter 6

Rendering

Since many of the simulation techniques that can be performed on points require solution of MLS interpolants, Dolbow interpolation is recommended for use in a ray tracer for fast isosurface rendering. We are able to ray trace a set of signed distance sample points quickly by using the signed distance to help predict how much farther a ray can be traversed before reaching the isosurface.

A ray is marched through the geometry by completing a Dolbow approximation of the surrounding signed distance data at each desired ray position. This estimated distance to the interface is then used to determine how large the next step along the ray should be. In practice, the step size is set to 75% of the estimated distance, plus a minimum step size to ensure marching continues non-asymptotically near the interface. The crossing of the interface is easily identified by a change in sign of the signed distance estimate at consecutive locations along the ray.

Once two consecutive estimates with opposite signs are found along a ray, the two corresponding locations are used as the initial points for a secant solve. When the surface has been located within a maximum threshold (the secant solve has converged), the normal can be identified through another Dolbow estimate, this time blending together gradients.

Figure 6.1 shows examples of images created by ray tracing the point-based level set representation.

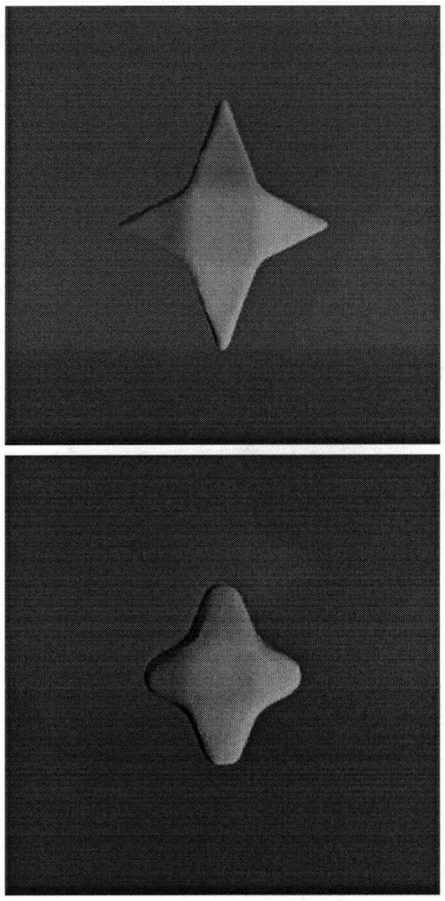

Figure 6.1: Theses ray-traced images have been created through Dolbow interpolation. The two frames show an evolution through motion by mean curvature.

Chapter 7

Towards Fluid Animation

In this chapter we investigate the use of the point-based level set for two-phase fluid animation. A review of recent publications is presented, then a short discussion of the Navier–Stokes equations and their solution is introduced. Next, a new method for solving Poisson problems is described, followed by a discussion of different boundary condition implementations. Finally, a short section describes the results that were achieved while attempting to apply the new particle based Poisson solve in the solution of the Navier-Stokes equations.

7.1 Point based animation

Simple advection, execution speed, and easy implementation have qualified point representations for a variety of animations. Reeves [31] introduced particle techniques for animating what he referred to as "fuzzy objects". Results for fire, explosions, grass, and fireworks were presented.

Recently, more advanced and physically based techniques have been used to model deformable objects. A real-time system for animating elastic, plastic, and melting objects through the use of SPH and MLS approximations was created by Müller*et al* [21] and has since been expanded to accommodate fracture [28]. Another strategy for animating deformable objects, this time through shape matching, was presented in [23].

7.2 Previous fluid animations

Fluid animations to date generally fall into one of three categories. Grid based simulation, the first category, allows for solution of the Navier-Stokes equations into an approximate divergence-free domain at the cost of smoothing and damping. The alternative, point based approaches have, to date, most often been based on *Smoothed Particle Hydrodynamics* (SPH), the accuracy of which is known to be dependant upon the particle distribution (there has also been some work on advecting radial basis functions [29]). The third category, a combination of grid based and particle based animations, has been used to couple the undamped freedom of particles, with the simple divergence-free projection of grids.

Foster and Metaxas [12] introduced the three dimensional solution of the Navier Stokes equations for animation using a *Marker and Cell* (MAC) grid. They described the treatment of boundary conditions for stationary obstacles, inflow and outflow, and free surfaces. Stam [32] introduced unconditionally stable semi-Lagrangian advection to the graphics community. However, the stable smoke simulation introduced a large degree of velocity dissipation. Fedkiw and Stam [9] attempted to limit the dissipation of Stam's earlier work with the introduction of vorticity confinement and clamped splines. Foster and Fedkiw [11] introduced to graphics the use of a level set to track the fluid interface on the grid. However, this level set representation was still plagued by numerical dissipation. Enright *et* *al* [7] improved Fedkiw and Foster's approach of combining properties of particles with those of a grid by placing particles on each side of the interface to provide a less dissipative solution. With similar motivation, Losasso *et al* [18] introduced the use of an octree to allow for sharper interface resolution. Most recently, Zhu and Bridson[35] showed that using both the*Particle-in-Cell* (PIC) method and *Fluid-Implicit-Particle* (FLIP) method, particles can be animated through a grid to simulate a variety materials.

Another branch of grid based fluid animation involves the *Volume of Fluid* (VOF) technique, created by Hirt and Nichols [15]. This technique again uses a three dimensional grid, but considers the water/air fraction in each cell. Recently, Sussman [33] showed how to couple level set techniques with the VOF approach. Two phase animations were provided through VOF by Hong and Kim [16], which provided for real time bubble simulation. Mihalef *et al* [19] stitched together a number of 2D VOF wave simulations to create realistic 3D waves.

SPH was introduced to the animation world by Desbrun and Gascuel [4]. The main advantages of this method include ease of implementation and speed of execution. These reasons have qualified its employment in a number of recent publications, including [22] and [30]. However, SPH is limited by requiring regular organization of the particles for accurate estimates [20], thus limiting its flexibility. Furthermore, reconstructing a smooth surface from this simple set of particles is very difficult; for example Premoze *et al* [30] resort to an expensive grid-based level set evaluation coupled to the particles in the face of this challenge.

7.3 Motivation

The point-based and grid-based level set techniques are capable of many of the operations required in fluid simulations. Both techniques hold the properties of regularization that are desired for fluid flow simulation. However, adaptivity is much more easily achieved with the randomly placed set of points, than with the use of an octree on a grid. Certainly, this warrants an investigation into replacing the usual grid with the point-based level set for fluid simulation.

The dependence of SPH on a regular particle spacing can limit its utility in more interesting flows. With the MLS approach in the point-based level set, there is little dependency on particle organization, making this new technique an attractive alternative. Furthermore, the difficulty of constructing a smooth interface around SPH particles can be bypassed with the point-based level set approach.

7.4 Navier-Stokes solution

The Navier-Stokes equations, which govern the flow of an incompressible inviscid fluid are as follows:

$$u_t + u \cdot \nabla u + (1/\rho)\nabla p = f \tag{7.1}$$

$$\nabla \cdot u = 0 \tag{7.2}$$

Where u_t is the time derivative of the flow and f is any external force, which in most cases is a gravitational force, g. The variable u denotes the three-dimensional velocity and ρ is the fluid density. Equation 7.1 ensures that momentum is conserved, while Equation 7.2 ensures the flow is incompressible. Solution of these equations is generally carried out as described, for example, by Stam [32]. This involves rearranging Equation 7.1 as

$$u_t = -u \cdot \nabla u - (1/\rho)\nabla p + f \tag{7.3}$$

and easing computation by splitting into:

$$u_t = f \tag{7.4}$$

$$u_t + u \cdot \nabla u = 0 \tag{7.5}$$

$$u_t + (1/\rho)\nabla p = 0. \tag{7.6}$$

With equations setup as above, time can be evolved by solving Equations 7.4-7.6 for each time step.

Solution of Equation 7.5 on a grid can introduce unwanted numerical diffusion [35], while the approximations that are presented through SPH are only valid for regular particle distributions [20]. To this end, it seems reasonable to explore the available tools presented in Chapters 2 and 3 as a framework for fluid animation. The remainder of this chapter presents efforts in this direction.

7.5 Particle advection

The solution of the advection Equation 7.5 with a Lagrangian representation, such as the point-based level set, can be greatly simplified. Since the particles move with the fluid, and encapsulate the total derivative of the flow, $u \cdot \nabla u$ can be ignored. With this assumption, particle represented inviscid fluids can be simulated in two steps: adding any outside forces to the particles (Equation 7.4), and enforcing incompressibility(Equation 7.6). This simplification is made implicitly in each of the following sections.

7.6 Particle fluids

Using the tools in Chapter 2, a strategy was devised to represent fluid flow with a set of randomly spaced sample points. Here, the flow was modeled as weakly compressible, allowing for slight deviations in density to control the pressure gradient. The following equations governed the flow:

$$x_t = u \tag{7.7}$$

$$u_t = -(1/\rho)\nabla p + g \tag{7.8}$$

$$\rho_t = -\rho\nabla \cdot u \tag{7.9}$$

$$p = c^2(\rho - \rho_o) \tag{7.10}$$

where c is the artificial incompressibility factor, ρ is the present density at a point, and ρ_o is the known scientific density of the fluid. As before, a subscripted t denotes the time derivative.

When using this strategy, the divergence and gradient can be computed with an MLS solution at each point. Thus, tracking changes in density becomes a matter of computing a number of MLS solves.

Enforcing boundary conditions of a closed container was attempted both by reflecting particles to the outside of the domain and by simple particle collision analysis. Neither of these strategies managed to create a convincing fluid simulation. The main problem was the lack of a steady solution for incompressibility, which can be exposed by considering a standing column of water. In a column, after the first time step, the particles at the bottom will experience a convergence. The effect of this convergence will only affect the nearby particles, pushing them upwards, while the particles above continue to gather downward momentum caused by the unbalanced gravitational pull. After a number of timesteps, the entire column is in disarray. Unfortunately, a proper heuristic for determining the parameter c and its appropriate treatment for incompressibility, was never reached, and a steady column of water was never achieved.

7.7 Approximate pressure solve

Given the results of the investigation in Section 7.6 we hypothesized that projection methods would be more effective. By determining a consistent solution for the pressure at each particle, an incompressible flow could be approximated.

7.7.1 Projection

The projection of a velocity field onto a divergence-free space (i.e., the solution of Equations 7.6 and 7.2) can be achieved in the following way.

After velocities have been updated and particles have been moved, the velocity field is no longer divergence free. This velocity field will be referred to as u^*. The desired divergence-free velocity field is u_{new}. Combining Equations 7.6 and 7.2 gives

$$\nabla \cdot (u^* - (1/\rho)\nabla p) = 0 \tag{7.11}$$

which can be rearranged as

$$\nabla \cdot (1/\rho)\nabla p = \nabla \cdot u^* \tag{7.12}$$

and once this has been solved for p, the final divergence free velocity field can be reached through

$$u_{new} = u^* - \frac{\nabla p}{\rho}. \tag{7.13}$$

7.7.2 Poisson system

The ability to approximate divergence, gradient, and Laplacians of scattered data with local MLS solves motivated a search for a way to leverage the MLS construct for the Poisson solve. This was especially attractive given that MLS systems would already be set up during the divergence solves at every point. In this section, an example of the solution in 1D is presented. The expansion to multiple dimensions follows naturally.

To explain how the Poisson problem was addressed, the system generated by Equation 2.1, fitting quadratics to pressures p_i on randomly spaced points x_i, is expanded as

$$\sum_{i=1}^{n} w(x_{ij}) \begin{bmatrix} 1 & x_{ij} & x_{ij}^2 \\ x_{ij} & x_{ij}^2 & x_{ij}^3 \\ x_{ij}^2 & x_{ij}^3 & x_{ij}^4 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \sum_{i=1}^{n} w(x_{ij}) \begin{bmatrix} 1 \\ x_{ij} \\ x_{ij}^2 \end{bmatrix} p_i \tag{7.14}$$

where the solution, $\begin{bmatrix} A & B & C \end{bmatrix}^T$, are the coefficients of the locally fit least squares quadratic $Pressure(x_{ij}) = A + Bx_{ij} + Cx_{ij}^2$. w is a finite radius weighting function. Rewriting the right side gives

$$\sum_{i=1}^{n} w(x_{ij}) \begin{bmatrix} 1 & x_{ij} & x_{ij}^2 \\ x_{ij} & x_{ij}^2 & x_{ij}^3 \\ x_{ij}^2 & x_{ij}^3 & x_{ij}^4 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} w_1 & w_2 & \dots & w_n \\ w_1 x_{1j} & w_2 x_{2j} & \dots & w_n x_{nj} \\ w_1 x_{1j}^2 & w_2 x_{2j}^2 & \dots & w_n x_{nj}^2 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix} \tag{7.15}$$

where $w_i = w(x_{ij})$ is the evaluation of the local weighting function. Since the weighting function is of small finite support in comparison to the domain, many of the entries on the right hand side are zero.

Labeling the coefficients on the left hand side of $\begin{bmatrix} A & B & C \end{bmatrix}^T$ in Equation 7.15 as S and the coefficient matrix of the p vector on the right hand side of Equation 7.15 as T, we can determine the $\begin{bmatrix} A & B & C \end{bmatrix}^T$ vector with

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} = (S^{-1}T)P. \tag{7.16}$$

Noting that the Laplacian of the 1D quadratic $A + Bx_{ij} + Cx_{ij}^2$ is $2C$, we can solve for p_i by aggregating the third row of Equation 7.15 from the solve at each point, multiplying it by two and setting it equal to the Laplacian at that point. Furthermore, to find the gradient as is necessary in the projection step for incompressible fluid flow, the second rows of Equation 7.15 can be retained and applied to the solution to find the gradient with one simple matrix multiply.

7.7.3 Filtering

As seen in Chapter 2 the use of MLS introduces smoothing to higher frequency features. Of course, this is to be expected when fitting a quadratic to sharp or discontinuous data. The smoothing introduced in the MLS solves creates high frequency artifacts in the Poisson system solve described above (i.e. the smoothing in the MLS estimate becomes sharpening in the above Poisson solve, where we invert the MLS operator). Fortunately, these artifacts are easily handled by a simple filtering step.

To demonstrate these artifacts, and the effects of filtering, an example solve was carried out. In the example, the Poisson solve was used to determine the data that generates a discontinuous Laplacian. Specifically, the following data was used:

$$y(x) = x^2 + x : \quad x < 1 \tag{7.17a}$$

$$y(x) = -x^2 + 5x - 2 : \quad x >= 1 \tag{7.17b}$$

which gives the discontinuous Laplacian:

$$\nabla^2 y(x) = 2 : \quad x < 1 \tag{7.18a}$$

$$\nabla^2 y(x) = -2 : \quad x >= 1. \tag{7.18b}$$

Using Dirichlet boundary conditions from the known solution, the system presented in Section 7.7.2 was employed to solve for Equation 7.17. The function was randomly sampled, and the kernel size was set so that each particle had at least 5 valid neighbors within its neighborhood. The results are plotted in Figure 7.1.

The filtering is easily achieved by local weighted averaging at each sample point using the same weighting function w.

7.7.4 Discontinuity handling and boundary conditions

The results of the above system for solving Poisson problems warranted further investigation into this technique. This section presents discussion of a number of boundary condition scenarios, including periodic, Dirichlet, and Neumann.

Periodic boundary conditions

It is possible to solve the Poisson problem as described above when considering periodic boundary conditions. The periodic boundary conditions are easily implemented in 1D by connecting the top of a domain to its bottom, thus making the two ends

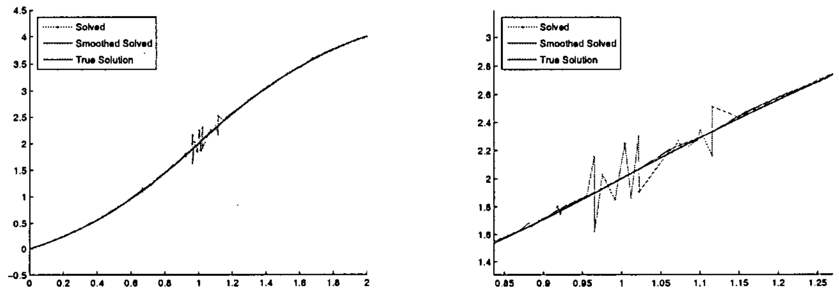

Figure 7.1: The results of the example Poisson solve are plotted to demonstrate the need for post-filtering. The blue line represents the true solution to the data, the green line is the original solution created through our method, and the red line shows the smoothed version of the green line. The figure on the left shows the general correctness of the solution while that on the right shows a close up of the difference made by the post-filtering.

adjacent. Therefore, particles that were previously at opposite ends of the domain, are now within one kernel radius of each other, and affect the divergence, etc. at each others positions. In higher dimensions this is only slightly more complicated.

Dirichlet boundary conditions

Dirichlet boundary conditions are easily implemented by placing particles outside the domain where the value is known. Dirichlet boundary conditions are used in the example shown in Figure 7.1.

Neumann boundary conditions

In a fluid simulation, Neumann boundary conditions are essential for the implementation of solid boundaries. This was implemented by copying particles across the interface and assigning them values to create the desired gradient at the interface.

In an effort to simulate the solid wall boundaries in a Poisson solve for a fluid simulation, ghost points were added outside the domain with positions and pressures assigned as:

$$x_i^G = x_i + s\psi(x_i)\hat{n}_i \tag{7.19}$$

$$p_i^G = p_i + (u_i \cdot \hat{n}_i)2\psi(x_i) \tag{7.20}$$

In Equation 7.19, x_i is the location of a near-boundary particle, x_i^G is the location of the reflected particle, and ψ is the distance function to the boundary. Equation 7.20 follows the same convention and u_i is the particle's velocity. This system attempts to approximate the ideal solid wall boundary

$$\frac{\partial p}{\partial n} \approx \frac{p_i^G - p_i}{\mid x_i^G - x_i \mid} = u_i \cdot \hat{n}_i. \tag{7.21}$$

Unfortunately, this approach, although promising, does not fully create the solid wall boundary required – a particular high-frequency velocity mode centered on the boundary is amplified by the projection step, causing an instability. We have not yet determined how to avoid this.

Discontinuities

Pressure discontinuities present in 2 phase flows motivated an investigation of Poisson solves with discontinuous coefficients. A pressure solve in a 2 phase simulation of standing water with air above, should display a piecewise linear pressure gradient, parallel to the force of gravity. However, fitting smooth quadratic MLS estimates to this non-smooth piecewise linear data can be error prone. It would be even more unrealistic to expect to arrive at a discontinuous solution during the Poisson solve, given that the second derivative of a discontinuity has infinite energy. In an effort to alleviate this problem the density factor, $1/\rho$, was introduced. So, instead of setting up the MLS fit to the non-smooth $(p_j - p_i)$ it was fit to the differentiable $\frac{1}{\rho_{ij}}(p_j - p_i)$

with the blending function

$$\rho_{ji} = \rho_{ij} = \frac{\phi_i}{\phi_i - \phi_j}\rho_i - \frac{\phi_j}{\phi_i - \phi_j}\rho_j \tag{7.22}$$

where ϕ is the signed distance. This is a smooth function that could be handled in the Poisson solve and is similar to the grid-based ghost-fluid method [8]

Results generated through this discontinuity handling strategy are shown in the next Section.

7.7.5 Approximate projection

The above-mentioned techniques were implemented in search of an approximate projection (see Section 7.7.1) to create a divergence-free solution through the Poisson solve for 2 phase fluid flow.

Normally, with a collocated centered finite-difference approach, the gradient of the pressure is subtracted from a smoothed version of the original velocity to give the divergence-free domain. One interpretation of this necessity is that the collocated centered finite-difference divergence operator cannot resolve high frequency divergence, and the projection only filters out the divergence it sees, requiring smoothing of the original velocity. Our strategy requires the same smoothing, which is easily implemented using a weighted average.

The Neumann boundary conditions were unsuccessfully demonstrated in a 2 dimensional, single phase flow. Figure 7.2 shows results after one time step of gravity being applied, and having the approximate projection successfully counteract the force to keep the particles stationary. However, after a number of iterations, the noise that is present is unattenuated and causes instability. Evidence of this can be visualized by looking at the eigenmodes of the projection matrix. The artifacts visible in Figure 7.3 can be mitigated by removing the treatment of Neumann

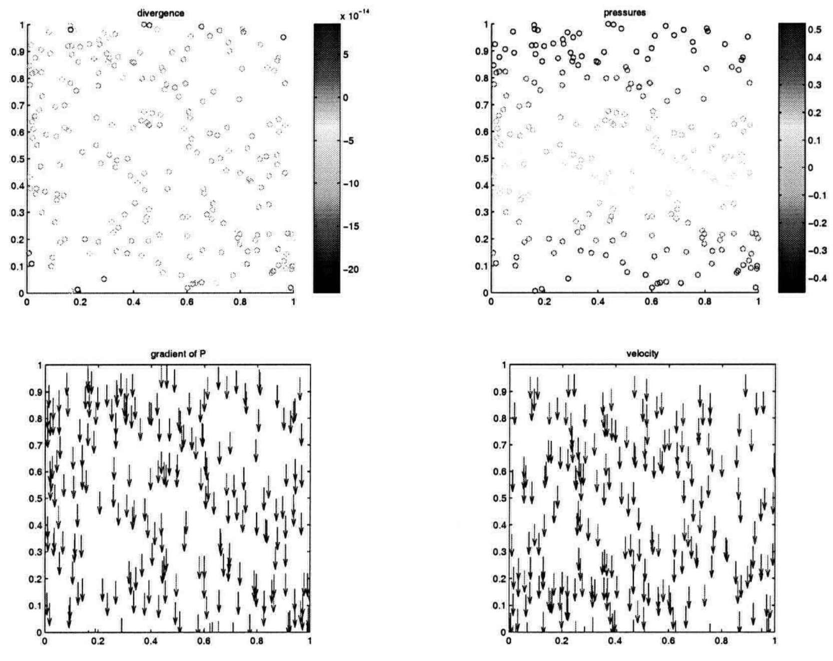

Figure 7.2: The 4 images in this figure show results after one time step of a 2 dimensional, single phase, fluid flow. The image on the top left shows that there is little or no divergence observed after gravity has been applied. The image on the bottom left shows the velocities of the particles after one time step, before the approximate projection is applied. The two images on the right show the pressure gradient. Note that the pressure gradient lines up perfectly with the particle velocities, and when it is subtracted from the velocity field, there will be no flow in the box.

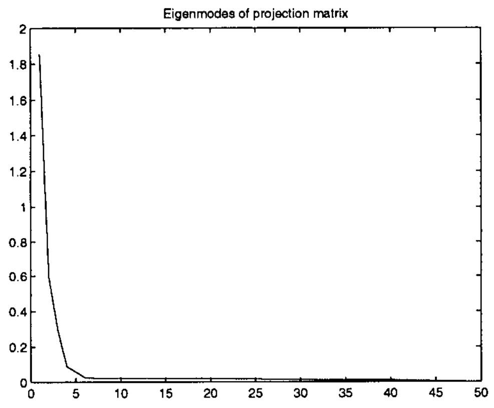

Figure 7.3: A plot of the eigenmodes of the projection matrix. As can be seen, some modes experience an amplification.

boundary conditions, and instead using periodic boundaries. The same investigation was carried out using periodic boundary conditions, with successful results. The eigenmodes of the projection matrix are plotted in Figure 7.4. Given the results of the eigenmode analysis, further investigation was focused on periodic boundaries only.

Finally, a test was done on periodic boundary conditions for 2 phase flow. The treatment of the discontinuity, as described above, was again unstable.

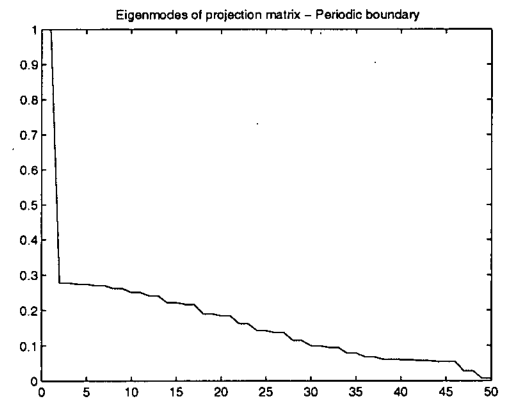

Figure 7.4: This plot of the eigenmodes of the projection matrix shows that with the periodic boundary conditions the approximate projection is stable.

Chapter 8

Conclusions

We feel that the generality available with the point-based level set representations can be useful for graphics and animation applications. Geometric deformations have been demonstrated, and strategies for signed distance upkeep and mesh conversion are described. Also, it was shown that additional resolution is always attainable through incorporation of more points, and the use of smaller interpolation radii. Finally, when tasks require repeated operations on unchanging geometry, Dolbow interpolation can make the task quicker.

We also believe that before employing the point-based level set in higher-precision simulations outside of graphics, further quantitative results may be required.

The investigation into the employment of point-based level sets for fluid animation showed that more work is required to enforce the required boundary conditions. The Poisson solve presented could be beneficial under the appropriate circumstances.

8.1 Future work

It would be interesting, and perhaps very fruitful to incorporate the recent work described in [10]. This might prove to be a better option for dealing with sharp features than the one we present here. Any other techniques that would help with the preservation of gradient discontinuities would be beneficial.

These techniques may be most useful in complex simulations such as those of fluids. Integration into such systems will provide for interesting development of these techniques as well as the development of interesting application specific strategies.

A quantification of the approximation quality as compared to grid-based level sets would be useful for formal comparison. Furthermore, detailed investigation of computational complexity might shed more light on the advantages of our representation.

It might be useful to investigate ways to speed up the use of the point-based level set techniques, perhaps with the introduction of a hash grid for accelerating particle searches.

Bibliography

[1] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Point set surfaces. In *VIS '01: Proceedings of the conference on Visualization '01*, pages 21–28, Washington, DC, USA, 2001. IEEE Computer Society.

[2] Nina Amenta and Yong Joo Kil. Defining point-set surfaces. *ACM Trans. Graph.*, 23(3):264–270, 2004.

[3] T. Cecil, J. Qian, and S. Osher. Numerical methods for high dimensional Hamilton-Jacobi equations using Radial Basis Functions. *J. Comp. Phys.*, 196:327–347, 2004.

[4] Mathieu Desbrun and Marie-Paule Cani. Smoothed particles: A new paradigm for animating highly deformable bodies. In R. Boulic and G. Hegron, editors, *Computer Animation and Simulation '96 (Proceedings of EG Workshop on Animation and Simulation)*, pages 61–76. Springer-Verlag, Aug 1996. Published under the name Marie-Paule Gascuel.

[5] John Dolbow and Ted Belytchko. An introduction to programming the meshless element free Galerkin method. *Archives of Computational Mechanics in Engineering*, 5:207–214, 1998.

[6] Douglas Enright, Ronald Fedkiw, Joel Ferziger, and Ian Mitchell. A hybrid particle level set method for improved interface capturing. *J. Comput. Phys.*, 183(1):83–116, 2002.

[7] Douglas Enright, Stephen Marschner, and Ronald Fedkiw. Animation and rendering of complex water surfaces. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 21(3):736–744, 2002.

[8] R. Fedkiw, X. D. Liu, and M. Kang. A Boundary Condition Capturing Method for Poisson's Equation on Irregular Domains. *Journal of Computational Physics*, 160:151–178, 2000.

[9] Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. Visual simulation of smoke. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 15–22, New York, NY, USA, 2001. ACM Press.

[10] Shachar Fleishman, Daniel Cohen-Or, and Cláudio T. Silva. Robust moving least-squares fitting with sharp features. *ACM Trans. Graph.*, 2005.

[11] Nick Foster and Ronald Fedkiw. Practical animation of liquids. In *Proc. of ACM SIGGRAPH 2001*, pages 23–30, 2001.

[12] Nick Foster and Dimitri Metaxas. Realistic animation of liquids. *Graph. Models Image Process.*, 58(5):471–483, 1996.

[13] Xiaohu Guo, Jing Hua, and Hong Qin. Point set surface editing techniques based on level-sets. In *Computer Graphics International*, pages 52–59, 2004.

[14] Xiaohu Guo and Hong Qin. Dynamic sculpting and deformation of point set surfaces. In *Pacific Graphics*, pages 123–130, 2003.

[15] C. W. Hirt and B. D. Nichols. Volume of fluid /VOF/ method for the dynamics of free boundaries. *Journal of Computational Physics*, 39:201–225, January 1981.

[16] Jeong-Mo Hong and Chang-Hum Kim. Animation of bubbles in liquid. *Computer Graphics Forum*, 22(3), 2003.

[17] David Levin. Mesh-independent surface interpolation. *Geometric Modeling for Scientific Visualization*, pages 37–49, 2003.

[18] F. Losasso, F. Gibou, and R. Fedkiw. Simulating water and smoke with an octree data structure. *ACM Trans. Graph. (SIGGRAPH Proc.)*, pages 457–462, 2004.

[19] Viorel Mihalef, Dimitris Metaxas, and Mark Sussman. Animation and control of breaking waves. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 315–324. ACM Press, 2004.

[20] J. J. Monaghan. Smoothed particle hydrodynamics. *Annual Reviews of Astronomy and Astrophysics*, 30:543–574, 1992.

[21] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. Point based animation of elastic, plastic, and melting objects. In *Eurographics/ACM Symposium on Computer Animation*, pages 141–151, 2004.

[22] Matthias Müller, David Charypar, and Markus Gross. Particle-based fluid simulation for interactive applications. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer animation*, pages 154–159, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[23] Matthias Müller, Bruno Heidelberger, Matthias Tescher, and Markus Gross. Meshless deformations based on shape matching. In *Proceedings of ACM SIGGRAPH 2005*. ACM Press, August 2005.

[24] Ken Museth, David E. Breen, Ross T. Whitaker, and Alan H. Barr. Level set surface editing operators. *ACM Trans. Graph.*, 21(3):330–338, 2002.

[25] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. Multi-level partition of unity implicits. *ACM Trans. Graph.*, 22(3):463–470, 2003.

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

[27] Stanley Osher and Ronald Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, 2003.

[28] Mark Pauly, Richard Keiser, Bart Adams, Philip Dutré, Markus Gross, and Leonidas J. Guibas. Meshless animation of fracturing solids. In *Proceedings of ACM SIGGRAPH 2005*. ACM Press, August 2005.

[29] Frédéric Pighin, Jonathan M. Cohen, and Maurya Shah. Modeling and editing flows using advected radial basis functions. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 223–232, New York, NY, USA, 2004. ACM Press.

[30] S. Premoze, T. Tasdizen, J. Bigler, A. Lefohn, and R. Whitaker. Particle-based simulation of fluids. In *Eurographics 2003 Proceedings*, pages 401–410. Blackwell Publishers, 2003.

[31] W. T. Reeves. Particle systems–a technique for modeling a class of fuzzy objects. *ACM Trans. Graph.*, 2(2):91–108, 1983.

[32] Jos Stam. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 121–128. ACM Press/Addison-Wesley Publishing Co., 1999.

[33] M. Sussman and E. G. Puckett. A coupled level set and volume of fluid method for computing 3d and axisymmetric incompressible two-phase flows. *Journal of Computational Physics*, 162(2):301–337, 2000.

[34] Yen-Hsi Richard Tsai. Rapid and accurate computation of the distance function using grids. *J. Comput. Phys.*, 178(1):175–195, 2002.

[35] Yongning Zhu and Robert Bridson. Animating sand as a fluid. In *Proceedings of ACM SIGGRAPH 2005.* ACM Press, August 2005.