Keyframe Animation of Implicit Models

by

David I. White

B.Sc., The University of Western Ontario, 2004

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

The Faculty of Graduate Studies

(Computer Science)

The University Of British Columbia

August 2006

Abstract

We present an approach that automatically constructs physically plausible in-between frames, given keyframes of arbitrary implicit surface geometry and feature points registered between adjacent keyframes. This extends to usable keyframe control of computer animated fluid-like materials. Most current implicit surface morphs do not allow feature point tracking and none guarantee physically plausible in-between frames of arbitrary motion. Standard triangle surface mesh morphing techniques do not guarantee physically plausible in-betweens either, nor can they handle topological changes. Current fluid control approaches do not respect keyframes nor track feature points.

Our variational approach finds a volume mapping between keyframes which minimizes a physics-based objective function using Gauss-Newton modified to handle linear constraints. We then create as-rigid-as-possible trajectories of the volume respecting this map, which we use to create physically plausible in-between frames.

Contents

List of Tables

List of Figures

Acknowledgements

This thesis would not have been possible without the insight and guidance of my supervisor, Robert Bridson. First and foremost I am grateful to him. Secondly, thanks to my second reader Michiel van de Panne for his helpful comments. He and Kevin Loken were incredible research collaborators as well.

Thanks also to Scott Singer, who provided motivation and advice during SIGGRAPH this summer.

For a fun, yet productive work environment I thank the crew from Imager; there are too many individuals to mention by name, but how can I forget Biff?

Dima, Scott, Dustin, Dan, Greg, Thomas, Bram and Christopher, thanks for making the Orphanage a great place to live.

Thanks to my family especially my sister and parents, without whom I might still think dogs run on batteries – or worse yet, have never questioned it.

DAVID I. WHITE

The University of British Columbia

August 2006

Chapter 1

Introduction

"What the animator does on each frame of film is not as important as what he or she does in between."

- Norman McLaren

Keyframe-based animation of general shapes is difficult on the computer. We provide a solution that can automatically construct physically plausible in-between frames given implicit surface keyframes of arbitrary geometry. Implicit surfaces are an attractive representation for general shapes that may undergo extreme deformation or topological changes during animation. There are many approaches to modeling objects with implicit surfaces, but there is little work on animating them. Techniques for animating fixed topology triangle meshes, which do not undergo large deformation, based on morphing are becoming mature. The current state of morphing for implicit surfaces is comparatively more primitive.

Our strategy is to find a mapping between Cartesian grid points of adjacent implicit surface keyframes (Chapter 2), then estimate as-rigid-as-possible trajectories of mapped points (Section 3.1), which are used to create implicit surfaces for the in-between frames (Section 3.2).

1.1 Motivation

The inspiration for this work comes from the tradeoff between physical realism and directability in computer animated fluids. Fluid simulations based on the Navier-Stokes equations accurately represent the physical properties of fluids, but are computationally time consuming, difficult to create from scratch and most importantly they offer no user-control of the fluid. In practice, computer animation studios generally use fluid systems that are set up to be highly user-intensive so that artists have control over the motion of the fluid. However, this can be time consuming and may lead to fluid that does not move in a physical manner. Our system allows full control of the fluid through user-defined keyframes and a minimal number of feature points, which we use to create in-between frames that are physically plausible.

1.2 Previous Work

Considerable work has been done on morphing, which is analogous to creating in-betweens for a given pair of keyframes. This work began with image morphing techniques in the early 1990's.

1.2.1 Image Morphing

Image morphing techniques began with [3], which uses feature mapping to control the morph between two images. While this works well for its intended purpose of simply blending two images, the created morph is non-physical. The work of [37] produces a three-dimensional "view morph" from two-dimensional images. It can produce physically consistent morphs, but not for arbitrary objects nor arbitrary motion. Image representations tend to lose or obscure detail. For a more accurate representation of objects, we need to consider morphing of

geometric models.

1.2.2 Morphing with Tetrahedral Meshes

Standard morphing techniques based on triangle surface meshes [19, 29, 48] generally preserve fine details well, but do not provide any guarantees of volume conservation nor do they handle topological changes. Least-distorting volume morphing techniques [1, 24] preserve volume but do not handle topological changes. Topological changes are trivial if we consider implicit surfaces.

1.2.3 Implicit Surface Morphing

Current implicit surface morphing techniques [11, 47] successfully morph between given surfaces; however in-between surfaces are generally not physically realistic. They do not guarantee preservation of volume, nor do they accurately provide rigid motion. For static model morphing this is reasonable, but physical in-between frames are crucial for keyframed animation. For example, [47] cannot match a simple rigid rotation. The approach of [7] works well for globally rigid movement, but will not hold for articulated motion. The volume morph technique in [28] is still non-physical because it merely blends the two models, which will not work for rigid motions. A novel, but non-physical, technique of morphing by expansion and contraction of shapes is presented in [4]. A keyframe technique is presented in [2], but it is contingent on an underlying skeletal structure and also is unproven for topological changes. In most of the aforementioned approaches, it is not guaranteed that feature points will map as expected. For example, the nose in an initial model may inadvertently map to an ear in the final model. With our user-defined feature points, we can guarantee a mapping to the user's specifications.

1.2.4 Fluid Simulation for Computer Animation

Fluid simulation techniques for computer animation were first presented in [34], where particles were used to represent clouds, smoke, water and fire. Later on, [25] provided a model for animating propagations along water surfaces that relied on solving the wave equation. The introduction of Navier-Stokes to computer graphics, for modeling the full body of fluid, came in [15]. A higher-level look at this same method is [17], which also explains how it was used to animated fluids in the film Antz – the first computer animated film containing fluids. A few years later, this approach was extended to be unconditionally stable in [42], which also introduced the semi-Lagrangian method. This was later refined for inviscid fluids, such as smoke, in [13] and liquids in [14]. The latter also introduced level sets to the computer graphics community. Photorealistic computer animated fluids were first shown in [10] by introducing, among other things, the particle level set. More recent advances include: using the octree data structure [30], introducing vortices for more interesting motion [38], animating fluids on meshes [26], and multiple fluids of different densities interacting [31]. For a detailed introduction to fluid simulation, see [5].

1.2.5 Controllable Fluid Simulations

Controllable fluids for computer animation were introduced in [16], where the user could change parameters to alter the fluid motion. More recent work in fluid control can be split into two categories: target matching [12, 20, 39, 40, 41, 45, 46] and user-defined control points [32, 33, 43, 51]. The former have difficulty perfectly matching fluid to the specified targets, and the latter use highly user-intensive control methods. Our approach guarantees that we exactly match the specified keyframes while keeping the number of user-defined control points reasonable.

1.3 Algorithm Overview

Our system takes as input a series of keyframes and feature points. We compute in-between frames for each pair of keyframes, Φ and Ψ, and their corresponding feature points, $\mathbf{x}'$ and $\mathbf{y}'$(shown in Figure 1.1). Using this information we create

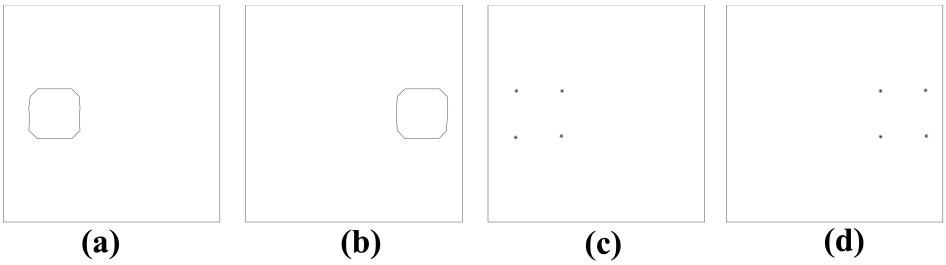

Figure 1.1: An example of input to our system: initial (a) and final (b) implicit surfaces, and their respective feature points (c) and (d).

a map, $\mathbf{y}$, of each object point in our initial keyframe to a corresponding point in the final keyframe (shown in Figure 1.2). Initially we do this using Radial Basis Functions built on the feature points, and then we use optimization to modify the map to fit several physical properties, such as volume conservation. From the map we create trajectories for each point (shown in Figure 1.3). In

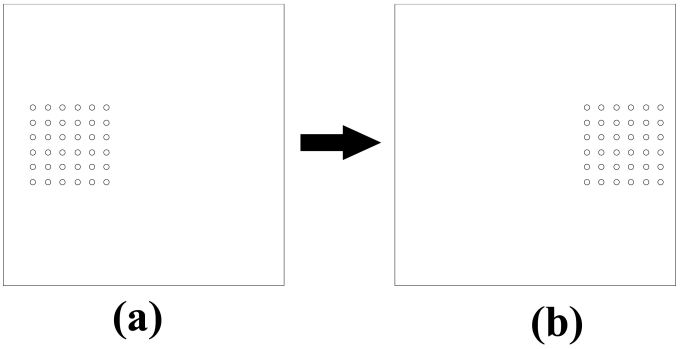

Figure 1.2: Our computed mapping from the initial keyframe (a) to the final keyframe (b).

order to capture rigid motions, such as rotations, these trajectories are as-rigid-as-possible, and are computed locally to match articulated motion.

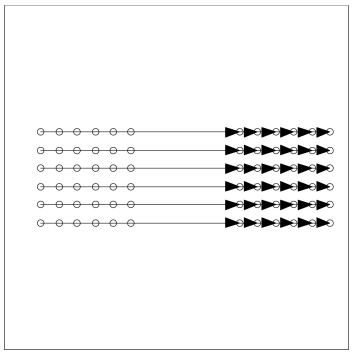

Figure 1.3: The as-rigid-as-possible trajectories from our initial keyframe to the final keyframe.

By sampling these trajectories, we create new implicit surfaces for in-between frames (shown in Figure 1.4).

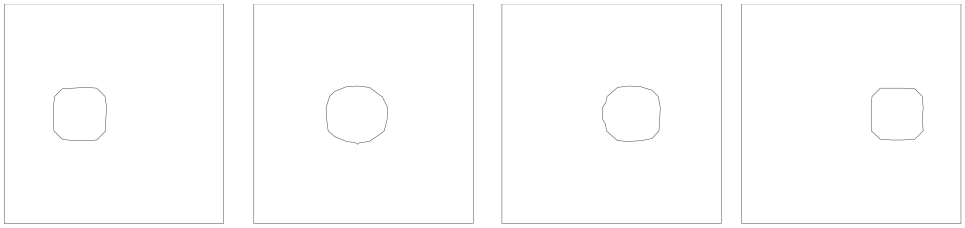

Figure 1.4: The output of our system: in-between frames that correspond to the input keyframes.

Our approach is outlined in Algorithm 1.

Algorithm 1 System Overview

```
1: Input: keyframes (as implicit surfaces) and feature points mapped between
   adjacent keyframes
2: for each pair of keyframes do
3:    map grid points from the initial keyframe to the final keyframe using
      Radial Basis Functions based on the feature points
4:    optimize this map using our physically based objective function and
      Gauss-Newton modified to handle linear constraints
5:    for each mapped point do
6:       create a trajectory between the keyframes that is as-rigid-as-possible
7:    end for
8:    for each in-between frame do
9:       create a level set based on the keyframe(s) and the trajectories
10:   end for
11:   Output: in-between frames
12: end for
```

Chapter 2

Mapping Adjacent Keyframes

Given a series of keyframes represented as implicit surfaces and selected feature points on adjacent keyframes, we find a mapping of all the material points in the initial frame to the material points in the final frame. Using Radial Basis Functions and the given feature points we obtain an initial guess for this map. Then we optimize this map for physically realistic movement by minimizing an objective function of physical properties. For this optimization we use the Gauss-Newton method modified to include linear constraints. These constraints ensure the feature points are mapped correctly, whether they lie on grid points, or not.

2.1 Keyframe Input

The inputs to our system are keyframes, represented as implicit surfaces sampled on regular Cartesian grids, and sets of selected feature points mapped between adjacent keyframes. We compute intermediate frames between each pair of adjacent keyframe surfaces, $\Phi(\mathbf{x})$ and $\Psi(\mathbf{y})$, and their corresponding feature points, $\mathbf{x}'$ and $\mathbf{y}'$. Feature points can be sparse, and can differ for each pairing of keyframes. It is important to note that the feature points are not required to align with the grid points as our optimizer is more general. Also they are not

confined to the material boundary - they can enter the interior of our material as specified by the user.

2.2 Radial Basis Functions

Using the initial feature points, $\mathbf{x}'$, and their final positions, $\mathbf{y}'$, we map the final positions of all the material points in our initial keyframe. This map, $\mathbf{y}(\mathbf{x})$, is one-to-one and we present scenarios in $\Re^2$ and $\Re^3$. It is used as an initial guess for the optimization outlined in Section 2.3 and Section 2.5. Using the Radial Basis Function formulation outlined in [6], we obtain our initial map in each dimension from:

$$\mathbf{y}(\mathbf{x}) = \mathbf{x} + \mathbf{p}(\mathbf{x}) + \sum_{i=1}^{N} \lambda_i R_B(\|\mathbf{x} - (y_i' - x_i')\|_2), \tag{2.1}$$

$$\text{ensuring that } \mathbf{y}(x_i') = y_i' \tag{2.2}$$

where R_B is the Radial Basis Function, N is the number of feature points, $\mathbf{p}(\mathbf{x})$ is a (dimension-1)-degree polynomial with M coefficients $\mathbf{c}$, and λ_i are the coefficients of the Radial Basis Functions. Notice that we use the RBF to get a mapping of displacements, so we convert it to final values by adding the initial locations of each point, $\mathbf{x}$. This is different than the standard RBF formulation. For two dimensions we use the thin-plate spline:

$$R_B(r) = r^2 \log r, \tag{2.3}$$

and in three dimensions we use the triharmonic spline:

$$R_B(r) = r^3. \tag{2.4}$$

To obtain λ and $\mathbf{c}$, we solve:

$$\begin{bmatrix} A & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ \mathbf{c} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ 0 \end{bmatrix}, \tag{2.5}$$

where

$$\begin{aligned} A_{i,j} &= R_B(\|x_i - (y'_j - x'_j)\|_2), \qquad i,j = 1,...,N \\ P_{i,j} &= p_j(x_i), \qquad i = 1,...,N, j = 1,...,M \\ f_i &= \mathbf{y}(x'_i), \qquad i,j = 1,...,N. \end{aligned}$$

Given initial keyframes containing multiple objects, we perform the Radial Basis Function step separately on each object.

2.3 Objective Function

At the heart of our system lies an objective function, f, that ensures the mapping from initial to final grid points, $\mathbf{y}(\mathbf{x})$, preserves certain physical properties. This objective function is:

$$\begin{aligned} f(\mathbf{y}) = \quad & \int |\det(\tfrac{\partial \mathbf{y}}{\partial \mathbf{x}}) - 1|^2 \quad + \quad \int \|\tfrac{\partial \mathbf{y}}{\partial \mathbf{x}}^T \tfrac{\partial \mathbf{y}}{\partial \mathbf{x}} - I\|_F^2 \\ & + \quad \int \|H(\Phi(\mathbf{x})) - H(\Psi(\mathbf{y}(\mathbf{x})))\|_2^2, \end{aligned} \tag{2.6}$$

where Φ and Ψ are the initial and final keyframes, and H is the Heaviside function:

$$H(\Phi(x_{ij})) = \begin{cases} 0 & \text{if } \Phi(x_{ij}) < -\epsilon, \\ \frac{1}{2} + \frac{\Phi(x_{ij})}{2\epsilon} + \frac{1}{2\pi}\sin\left(\frac{\pi\Phi(x_{ij})}{\epsilon}\right) & \text{if } \epsilon \leq \Phi(x_{ij}) \leq \epsilon, \\ 1 & \text{if } \epsilon < \Phi(x_{ij}), \end{cases} \tag{2.7}$$

with $\epsilon = \frac{3\Delta x}{2}$ to numerically smear over several grid cells to permit gradient estimates.

It is crucial that our feature points map properly, so this objective function is subject to the constraints:

$$\mathbf{y}(\mathbf{x}') = \mathbf{y}'. \tag{2.8}$$

In our objective function, the first term resists volume change in the material by ensuring that $\det\left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right)$, the ratio of the approximated final volume to the actual initial volume of a grid cell, is close to one. The second term avoids unnecessary deformation within the material, so for example we match rigid body motions exactly. This is done by enforcing that the partial derivative with respect to each point is dependent only on its final position, and not that of any of the other points. Meanwhile the third term matches the initial level set surface to the mapped image on the final level set surface by comparing the Heaviside function at each initial grid point with the Heaviside function of the corresponding final mapped position. To account for additional criteria, more terms could be added to this objective function. Prioritizing certain terms could also be accomplished by adding scaling factors to the terms. We minimize this objective function using the approach outlined in Section 2.5.

2.4 Discretization

Since our map is sampled on a Cartesian grid, $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$ is computed in two dimensions using the naïve discretization:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} u_1 & u_2 \\ v_1 & v_2 \end{bmatrix}, \tag{2.9}$$

where

$$u_1 = \frac{(\mathbf{x}^{(1)}_{i+1,j+1} + \mathbf{x}^{(1)}_{i+1,j}) - (\mathbf{x}^{(1)}_{i,j+1} + \mathbf{x}^{(1)}_{i,j})}{2\Delta u},$$

$$u_2 = \frac{(\mathbf{x}^{(1)}_{i+1,j+1} + \mathbf{x}^{(1)}_{i,j+1}) - (\mathbf{x}^{(1)}_{i+1,j} + \mathbf{x}^{(1)}_{i,j})}{2\Delta u},$$

$$v_1 = \frac{(\mathbf{x}^{(2)}_{i+1,j+1} + \mathbf{x}^{(2)}_{i+1,j}) - (\mathbf{x}^{(2)}_{i,j+1} + \mathbf{x}^{(2)}_{i,j})}{2\Delta v},$$

$$v_2 = \frac{(\mathbf{x}^{(2)}_{i+1,j+1} + \mathbf{x}^{(2)}_{i,j+1}) - (\mathbf{x}^{(2)}_{i+1,j} + \mathbf{x}^{(2)}_{i,j})}{2\Delta v}.$$

Note: the superscript represents the dimension; it is not an exponent. Unfortunately this discretization (shown in Figure 2.1a) allows the emergence of "hour-glassing" – unwanted null-space modes where cells badly deform into equal area trapezoids. So we further subdivide each grid cell into four parts (shown in Figure 2.1b) and compute $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$ for each of them.

The extension of this subdivision to three dimensions is shown in Figure 2.2,

and the discretization is:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ w_1 & w_2 & w_3 \end{bmatrix}, \tag{2.10}$$

where

$$u_1 = \frac{(\mathbf{x}^{(1)}_{i+1,j+1,k}+\mathbf{x}^{(1)}_{i+1,j,k})-(\mathbf{x}^{(1)}_{i,j+1,k}+\mathbf{x}^{(1)}_{i,j,k})+(\mathbf{x}^{(1)}_{i+1,j+1,k+1}+\mathbf{x}^{(1)}_{i+1,j,k+1})-(\mathbf{x}^{(1)}_{i,j+1,k+1}+\mathbf{x}^{(1)}_{i,j,k+1})}{4\Delta u},$$

$$u_2 = \frac{(\mathbf{x}^{(1)}_{i+1,j+1,k}+\mathbf{x}^{(1)}_{i+1,j,k})-(\mathbf{x}^{(1)}_{i+1,j+1,k+1}+\mathbf{x}^{(1)}_{i+1,j,k+1})+(\mathbf{x}^{(1)}_{i,j+1,k}+\mathbf{x}^{(1)}_{i,j,k})-(\mathbf{x}^{(1)}_{i,j+1,k+1}+\mathbf{x}^{(1)}_{i,j,k+1})}{4\Delta u},$$

$$u_3 = \frac{(\mathbf{x}^{(1)}_{i+1,j,k}+\mathbf{x}^{(1)}_{i,j,k})-(\mathbf{x}^{(1)}_{i+1,j,k+1}+\mathbf{x}^{(1)}_{i,j,k+1})+(\mathbf{x}^{(1)}_{i+1,j+1,k}+\mathbf{x}^{(1)}_{i,j+1,k})-(\mathbf{x}^{(1)}_{i+1,j+1,k+1}+\mathbf{x}^{(1)}_{i+1,j,k+1})}{4\Delta u},$$

$$v_1 = \frac{(\mathbf{x}^{(2)}_{i+1,j+1,k}+\mathbf{x}^{(2)}_{i+1,j,k})-(\mathbf{x}^{(2)}_{i,j+1,k}+\mathbf{x}^{(2)}_{i,j,k})+(\mathbf{x}^{(2)}_{i+1,j+1,k+1}+\mathbf{x}^{(2)}_{i+1,j,k+1})-(\mathbf{x}^{(2)}_{i,j+1,k+1}+\mathbf{x}^{(2)}_{i,j,k+1})}{4\Delta v},$$

$$v_2 = \frac{(\mathbf{x}^{(2)}_{i+1,j+1,k}+\mathbf{x}^{(2)}_{i+1,j,k})-(\mathbf{x}^{(2)}_{i+1,j+1,k+1}+\mathbf{x}^{(2)}_{i+1,j,k+1})+(\mathbf{x}^{(2)}_{i,j+1,k}+\mathbf{x}^{(2)}_{i,j,k})-(\mathbf{x}^{(2)}_{i,j+1,k+1}+\mathbf{x}^{(2)}_{i,j,k+1})}{4\Delta v},$$

$$v_3 = \frac{(\mathbf{x}^{(2)}_{i+1,j,k}+\mathbf{x}^{(2)}_{i,j,k})-(\mathbf{x}^{(2)}_{i+1,j,k+1}+\mathbf{x}^{(2)}_{i,j,k+1})+(\mathbf{x}^{(2)}_{i+1,j+1,k}+\mathbf{x}^{(2)}_{i,j+1,k})-(\mathbf{x}^{(2)}_{i+1,j+1,k+1}+\mathbf{x}^{(2)}_{i+1,j,k+1})}{4\Delta v},$$

$$w_1 = \frac{(\mathbf{x}^{(3)}_{i+1,j+1,k}+\mathbf{x}^{(3)}_{i+1,j,k})-(\mathbf{x}^{(3)}_{i,j+1,k}+\mathbf{x}^{(3)}_{i,j,k})+(\mathbf{x}^{(3)}_{i+1,j+1,k+1}+\mathbf{x}^{(3)}_{i+1,j,k+1})-(\mathbf{x}^{(3)}_{i,j+1,k+1}+\mathbf{x}^{(3)}_{i,j,k+1})}{4\Delta w},$$

$$w_2 = \frac{(\mathbf{x}^{(3)}_{i+1,j+1,k}+\mathbf{x}^{(3)}_{i+1,j,k})-(\mathbf{x}^{(3)}_{i+1,j+1,k+1}+\mathbf{x}^{(3)}_{i+1,j,k+1})+(\mathbf{x}^{(3)}_{i,j+1,k}+\mathbf{x}^{(3)}_{i,j,k})-(\mathbf{x}^{(3)}_{i,j+1,k+1}+\mathbf{x}^{(3)}_{i,j,k+1})}{4\Delta w},$$

$$w_3 = \frac{(\mathbf{x}^{(3)}_{i+1,j,k}+\mathbf{x}^{(3)}_{i,j,k})-(\mathbf{x}^{(3)}_{i+1,j,k+1}+\mathbf{x}^{(3)}_{i,j,k+1})+(\mathbf{x}^{(3)}_{i+1,j+1,k}+\mathbf{x}^{(3)}_{i,j+1,k})-(\mathbf{x}^{(3)}_{i+1,j+1,k+1}+\mathbf{x}^{(3)}_{i+1,j,k+1})}{4\Delta w}.$$

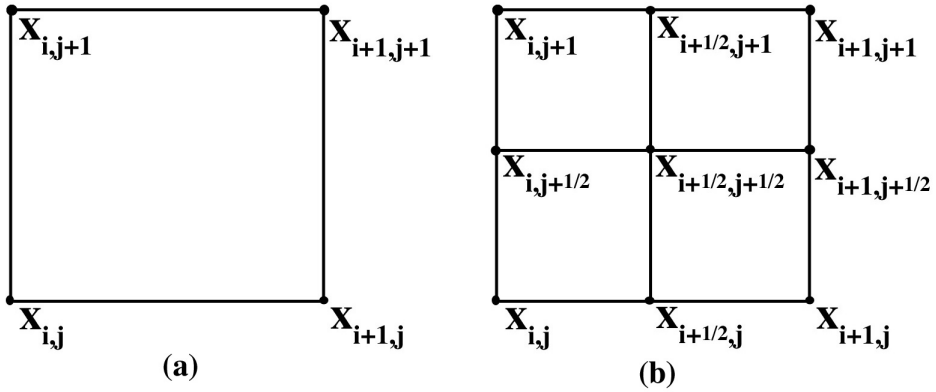

Figure 2.1: Naïve (a) and subdivided (b) discretization of a two-dimensional grid cell.

2.5 Modified Gauss-Newton Optimization

To solve the objective function we use Gauss-Newton optimization. However, since the optimal map must match the selected feature points, we have linear constraints. To perform this optimization we modify the standard Gauss-Newton approach of:

$$\min_{\mathbf{y}(\mathbf{x})} \|p(\mathbf{y})\|^2, \tag{2.11}$$

which updates each step, $\Delta\mathbf{y}_k$, by

$$J_k^T J_k \Delta\mathbf{y}_k = -J_k^T p(\mathbf{y}_k), \tag{2.12}$$

where

$$J_k = \frac{\partial p}{\partial \mathbf{y}}\Big|_{\mathbf{y}_k}. \tag{2.13}$$

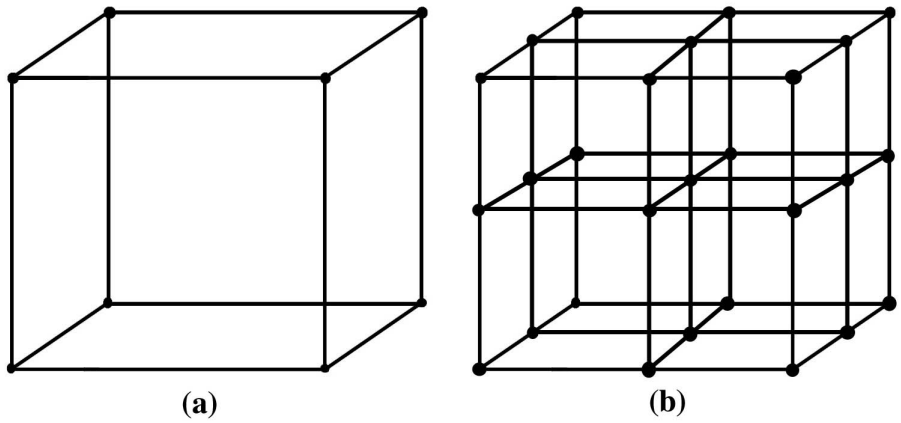

Figure 2.2: Naïve (a) and subdivided (b) discretization of a three-dimensional grid cell.

Our modified approach that satisfies linear constraints is:

$$\min_{\mathbf{y}(\mathbf{x})} \|p(\mathbf{y})\|^2, \tag{2.14}$$

$$\text{subject to } C\mathbf{y}' = \mathbf{d}, \tag{2.15}$$

and $\Delta\mathbf{y}_k$ is obtained from

$$\begin{bmatrix} J_k^T J_k & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} \Delta\mathbf{y}_k \\ \lambda \end{bmatrix} = \begin{bmatrix} -J_k^T p(\mathbf{y}_k) \\ 0 \end{bmatrix}. \tag{2.16}$$

Our initial map from the Radial Basis Functions matches the feature points exactly, so the constraints are satisfied when the optimization begins. The update steps, $\Delta\mathbf{y}_k$, then just have to maintain these constraints.

When the feature points lie along Cartesian grid points, our constraints are

simply:

$$C_{ij} = \begin{cases} 1 & \text{if } y'_i \text{ is on grid point } j, \\ 0 & \text{otherwise,} \end{cases} \tag{2.17}$$

$$d_i = x'_i. \tag{2.18}$$

For feature points that are not aligned with the grid points we use barycentric weights, t_i, to guarantee that the optimal map meets these constraints (Figure 2.3). Our constraints then become:

$$C_{ij} = \begin{cases} t_i & \text{if } j \text{ is a grid point on the grid cell of } y'_i, \\ 0 & \text{otherwise,} \end{cases} \tag{2.19}$$

$$d_i = (t_1 + t_2 + t_3 + t_4)x'_i. \tag{2.20}$$

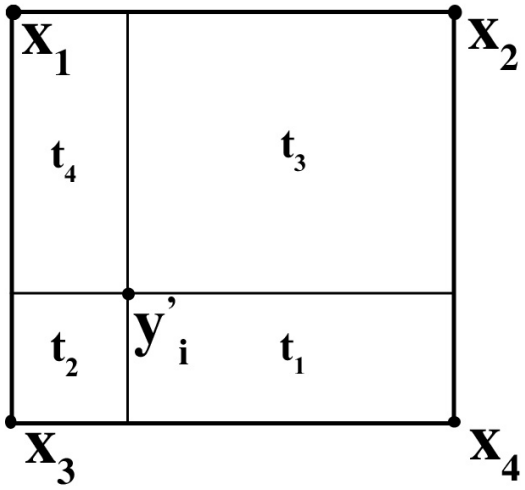

Figure 2.3: Barycentric weights, t_i, of y_i' in the grid defined by x_1, x_2, x_3, and x_4.

Chapter 3

Computing In-between Frames

Once we have obtained the optimal mapping from the initial keyframe to the final keyframe, we still need to compute the corresponding in-between frames. In the traditional morphing sense, this is done simply by blending between the initial and final keyframes, with perhaps a small warp to help them match up. We also take a warp and blend approach, though our warp does everything up to truncation error in matching the implicit surfaces of the keyframes. The third term of our objective function (Equation 2.3) corrects this (tiny) truncation error. More sophisticated methods, [4], could be used if desired.

Our warping is carried out by creating as-rigid-as-possible trajectories for each mapped point. We then create a level set at each intermediate time step by blending the initial and final implicit surfaces advected along these trajectories.

3.1 Rigid Trajectories

After we have computed the optimal map, we then create trajectories which are the basis for the in-between frames. An obvious choice would be to move each point along a straight line towards it's final location, however this fails to match rigid rotations. Since we want the local neighbourhood of a point to be mapped in a rigid manner, the trajectory should be the source of this rigidity. This

gives a natural arc, instead of a straight line, for rotations. We generalize this to as-rigid-as-possible by finding the rigid body motion as close as possible to the deformation of the point's neighbourhood. We use the following formulation to compute the trajectory of an initial grid point, $\mathbf{x}_{ij}$, at the t^{th} intermediate frame:

$$\mathbf{x}_{ij}^{(t)} = \mathbf{x}_{ij} + (\mathbf{x}_{ij} - \bar{x})R^*(\mathbf{x}_{ij}, t) + \left(\tfrac{t-1}{n-1}\right)(\bar{y} - \bar{x}) + \left(\tfrac{t-1}{n-1}\right)(1 - \delta_{tn})\left(\mathbf{y}_{ij} - \mathbf{x}_{ij}^{(n)}\right), \tag{3.1}$$

where $\bar{x}$ is the mean of $\mathbf{x}$, $\bar{y}$ is the mean of $\mathbf{y}$, n is the total number of frames, δ is the Kronecker Delta, $\mathbf{x}_{ij}^{(n)}$ is the final position of $\mathbf{x}_{ij}$, and $R^*(\mathbf{x}_{ij}, t)$ is the optimal rotation of $\mathbf{x}_{ij}$ and its grid neighbours at the t^{th} frame.

For each mapped point we compute the translation and rotation component of it and its nearest grid neighbours using the closed-form solution presented in [21, 22]. For two dimensions we reduce this approach, shown in Appendix A. We linearly interpolate the residual deformation to ensure our trajectories match the final keyframe. For components of the model that move rigidly this approach yields the exact solution. Since we compute this locally, if different parts of a body move with different rigid motions, we still get the correct trajectories.

3.2 Level Sets

Given our initial and final implicit surfaces, Φ and Ψ respectively, we compute intermediate surfaces based on the as-rigid-as-possible trajectories. To generate intermediate implicit surfaces we advect the initial level set along these trajec-

tories using weighted averages:

$$\Phi(x_{ij}) = \left[\sum_p \Phi_p w(x_{ij} - x_p)\right] / \left[\sum_p w(x_{ij} - x_p)\right] \tag{3.2}$$

where

$$w(x_{ij}) = \frac{1}{||x_{ij}||^2 + \epsilon^2}. \tag{3.3}$$

For higher accuracy, we could blend it with the final level set advected back along the same trajectories, computing the intermediate level set for the t^{th} frame, $\Phi^{(t)}$, from:

$$\Phi^{(t)}(x_{ij}^{(t)}) = (n-t)\,\Phi(x_{ij}^{(t)}) + t\,\Psi(\mathbf{y}(x_{ij}^{(t)})). \tag{3.4}$$

Since we have an explicit map, we can not only blend the level set values, but also colour and texture coordinate information, for example. If these are only defined on the surface of the initial and final frames, we extend them into the volume by taking the same value as the closest point on the surface.

To resample the level set or other values onto regular grids at intermediate frames, if desired, we use scattered data interpolation. Currently our implementation simply uses weighted averages, but this is easily extended to more accurate Moving-Least Squares estimates [8].

Chapter 4

Results

Our system strives to animate materials in a physically reasonable manner given a set of keyframes. We also automatically allow for topological changes, and allow the user to guide the motion by identifying a sparse set of feature points. The number of feature points must be large enough for the Radial Basis Function to build an initial map. Similar methods, [32, 33] use hundreds of thousands of control points, but as shown in Table 4.1 the number of feature points in our examples are reasonable for manual manipulation.

Simulation	**Number of Feature Points**
Merge	8
Bend	6
Splash	10,14,12

Table 4.1: Number of feature points in presented simulations.

4.1 Rigid Motion

We begin with a simple example that demonstrates the rigid-motion-preservation of our technique. As mentioned in Section 3.1, rigid motion, such as rotation or translation, is matched exactly by our system. Since the second term of our objective function (Equation 2.6) preserves rigid-body motions, as does the initial guess obtained from the Radial Basis Functions, the optimal map from our optimization is completely rigid. This leaves no residual deformation for our as-

rigid-as-possible-trajectories, thus providing rigid in-between trajectories. Some examples of rigid motion are shown in Figure 4.1.

4.2 Topological Changes

Our system automatically handles the merging of components, as shown in Figure 4.2. In the future we would like to extend this to topological splitting of components, based on a fracture-like criterion.

4.3 Articulated Motion

To show that our system works for movement that is rigid in its components, but not as a whole, we provide the case of an articulated rod bending in Figure 4.3. There are currently bugs in the level set code that cause unexpected surface artifacts to appear in some of the in-between frames here and in the following example.

4.4 Splash

In Figure 4.4 we present a simulation of a droplet splashing into a pool of fluid. Note, we are using our current system that does not include the additional inertial term for fluid-like movement.

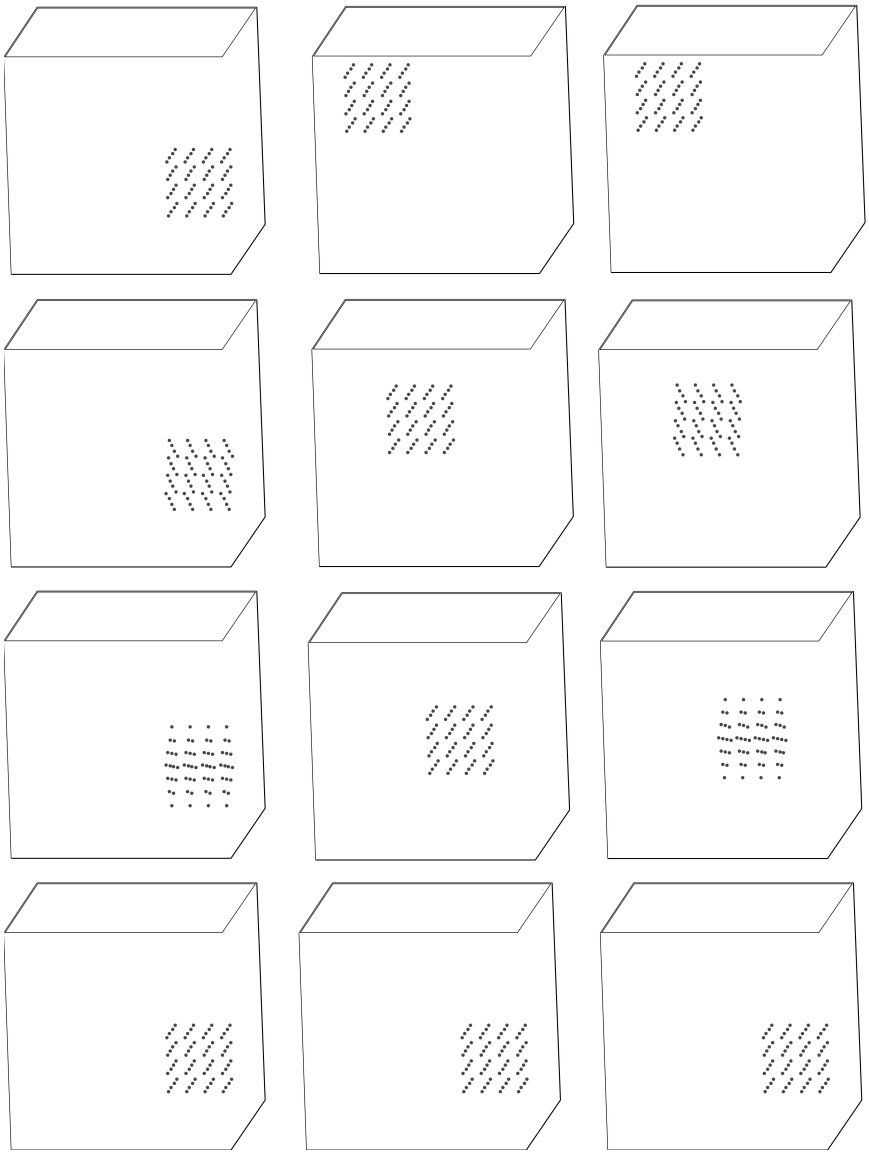

Figure 4.1: A Cartesian grid representation of three-dimensional rigid examples: rotation (top row), translation (middle row), and both translation and rotation (bottom row).

Figure 4.2: Two-dimensional merging example, with the first and last images as keyframes. The level sets are displayed, as are the feature points.

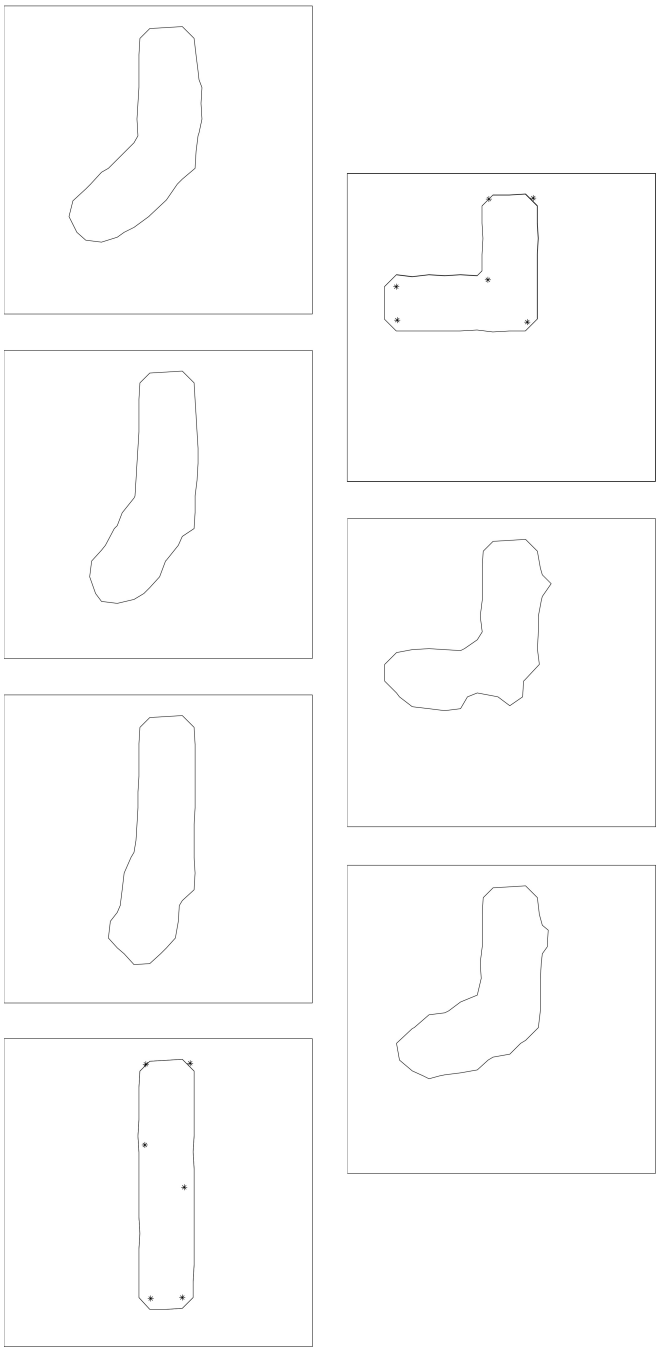

Figure 4.3: Two-dimensional bending example, with the first and last images as keyframes. The level sets are displayed, as are the feature points.

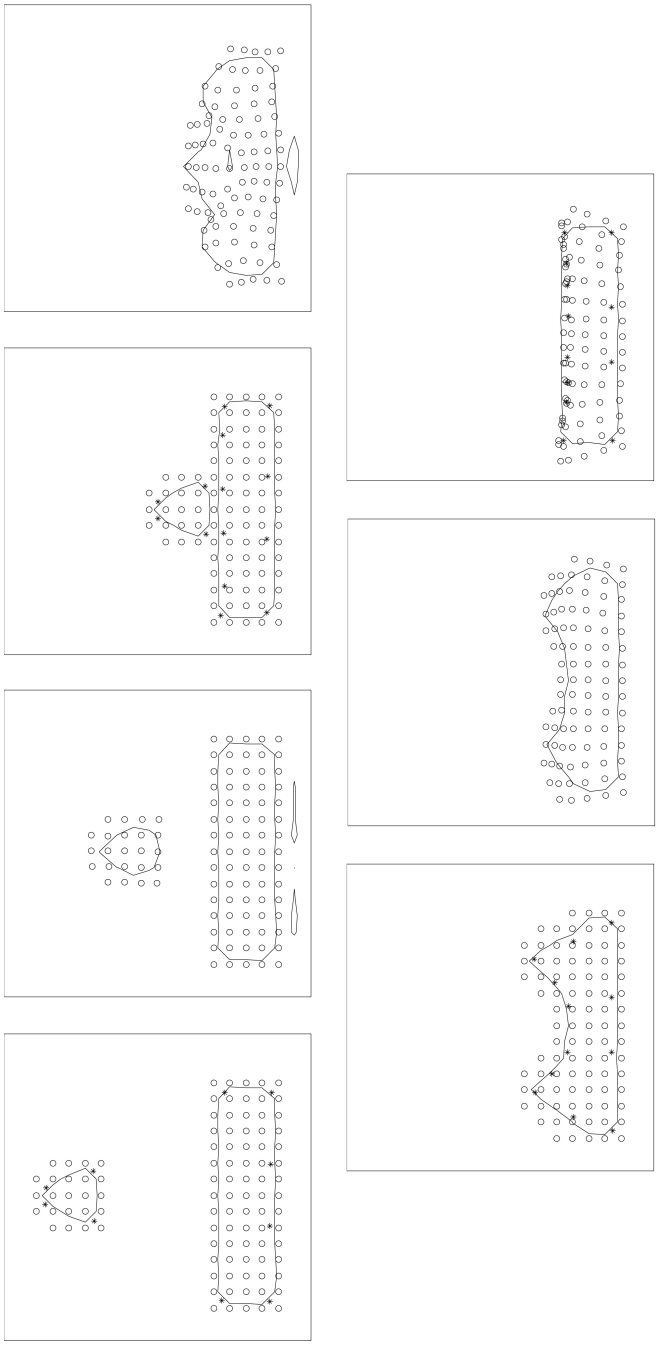

Figure 4.4: Two-dimensional splashing example with the level set and feature points shown. The first, third, fifth and seventh images represent keyframes. The initial grid points, warped to their intermediate or final positions are also shown.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

We present a method to create physically plausible in-between frames, given arbitrary implicit models as keyframes. Our variational approach finds a volume mapping between keyframes which minimizes a physics-based objective function, and then creates as-rigid-as-possible trajectories of the volume respecting this map, which we use to create physically plausible in-between frames.

5.2 Future Work

Currently we animate objects in a rigid manner, however this method could be extended to animate fluid-like materials by adding a parameter that controls sloppy liquid-like motion of the fluid. One application of this would be for animating liquid characters, which are currently animated in an *ad hoc* and highly artist-intensive manner to ensure absolute control of the fluid and fluid-like motion [33, 51].

We believe the addition of an inertial term to our objective function would provide an automated solution to this problem. This term would penalize accelerations, by accounting for the difference between predicted position, based

on given initial velocities, and the corresponding mapped final position. We in essence would provide a variational form for continuum dynamics, with additional control terms or constraints for feature point and level set matching. For trajectory estimation we also need to take into account initial velocities, as the final velocity dictated by the computed trajectory should match the velocity at the start of the following keyframe. We also plan to add the capability to split topologies, by providing a fracture-like criterion.

5.2.1 Cartoony Fluids

Inspired by the work of [9], which creates cartoony fluids in an artist-intensive approach, we would like to extend our system to provide a more automated solution. Our approach would be to combine our system, including the inertial term for fluid-like motion, with cartoon animation filters, [49, 50], that automatically apply well-known animation principles [27, 44].

5.2.2 Hands-on Control

We would also like to extend this work to include a hands-on approach to modeling keyframes for animation. The idea is to have a user create the keyframes using modeling clay, and then scan the three-dimensional model with a handheld camera. Relevant research include the areas of mosaicing, [18, 23] and model acquisition [35, 36].

Bibliography

[1] Marc Alexa, Daniel Cohen-Or, and David Levin. As-rigid-as-possible shape interpolation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 157–164, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[2] Aurélien Barbier, Eric Galin, and Samir Akkouche. A framework for modeling, animating, and morphing textured implicit models. *Graph. Models*, 67(3):166–188, 2005.

[3] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 35–42, New York, NY, USA, 1992. ACM Press.

[4] David E. Breen and Ross T. Whitaker. A level-set approach for the metamorphosis of solid models. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):173–192, 2001.

[5] Robert Bridson, Matthias Müller-Fischer, Eran Guendelman, and Ronald Fedkiw. Fluid simulation. In *SIGGRAPH 2006 Course Notes*, 2006.

[6] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *SIGGRAPH '01: Proceedings of the*

28th annual conference on Computer graphics and interactive techniques, pages 67–76, New York, NY, USA, 2001. ACM Press.

[7] Daniel Cohen-Or, Amira Solomovic, and David Levin. Three-dimensional distance field metamorphosis. *ACM Trans. Graph.*, 17(2):116–141, 1998.

[8] Richard Corbett. Point-based level sets and progress towards unorganized particle based fluids. Master's thesis, University of British Columbia, 2005.

[9] Peter DeMund. Cartoony fluid animation. In *Proceedings of SIGGRAPH 2005, Sketches & Applications*. ACM Press, 2005.

[10] Douglas Enright, Stephen Marschner, and Ronald Fedkiw. Animation and rendering of complex water surfaces. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 736–744, New York, NY, USA, 2002. ACM Press.

[11] Xiang Fang, Hujun Bao, Pheng Ann Heng, TienTsin Wong, and Qunsheng Peng. Continuous field based free-form surface modeling and morphing. *Computers and Graphics*, 25(2):235–243, April 2001.

[12] Raanan Fattal and Dani Lischinski. Target-driven smoke animation. *ACM Trans. Graph.*, 23(3):441–448, 2004.

[13] 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.

[14] Nick Foster and Ronald Fedkiw. Practical animation of liquids. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 23–30, New York, NY, USA, 2001. ACM Press.

[15] Nick Foster and Dimitri Metaxas. Realistic animation of liquids. *Graphical models and image processing: GMIP*, 58(5):471–483, 1996.

[16] Nick Foster and Dimitris Metaxas. Controlling fluid animation. In *CGI '97: Proceedings of the 1997 Conference on Computer Graphics International*, page 178, Washington, DC, USA, 1997. IEEE Computer Society.

[17] Nick Foster and Dimitris Metaxas. Modeling water for computer animation. *Commun. ACM*, 43(7):60–67, 2000.

[18] Paolo Grattoni and Massimiliano Spertino. A mosaicing approach for the acquisition and representation of 3d painted surfaces for conservation and restoration purposes. *Mach. Vision Appl.*, 15(1):1–10, 2003.

[19] Arthur D. Gregory, Andrei State, Ming C. Lin, Dinesh Manocha, and Mark A. Livingston. Feature-based surface decomposition for correspondence and morphing between polyhedra. In *Computer Animation '98*, pages 64–71, 1998.

[20] Jeong-Mo Hong and Chang-Hun Kim. Controlling fluid animation with geometric potential: Research articles. *Comput. Animat. Virtual Worlds*, 15(3-4):147–157, 2004.

[21] Berthold K.P. Horn. Closed form solution of absolute orientation using unit quaternions. *Journal of the Optical Society A*, 4:629–642, 1987.

[22] Berthold K.P. Horn, Hugh M. Hilden, and Shahriar Negahdaripour. Closed form solution of absolute orientation using orthonormal matrices. *Journal of the Optical Society A*, 5:1127–1135, 1988.

[23] Chiou-Ting Hsu, Tzu-Hung Cheng, Rob A. Beuker, and Jyh-Kuen Horng. Feature-based video mosaic. In *ICIP*, 2000.

[24] Takeo Igarashi, Tomer Moscovich, and John F. Hughes. As-rigid-as-possible shape manipulation. *ACM Trans. Graph.*, 24(3):1134–1141, 2005.

[25] Michael Kass and Gavin Miller. Rapid, stable fluid dynamics for computer graphics. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 49–57, New York, NY, USA, 1990. ACM Press.

[26] Bryan M. Klingner, Bryan E. Feldman, Nuttapong Chentanez, and James F. O'Brien. Fluid animation with dynamic meshes. *ACM Trans. Graph.*, 25(3):820–825, 2006.

[27] John Lasseter. Principles of traditional animation applied to 3d computer animation. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 35–44, New York, NY, USA, 1987. ACM Press.

[28] Apostolos Lerios, Chase D. Garfinkle, and Marc Levoy. Feature-based volume metamorphosis. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 449–456, New York, NY, USA, 1995. ACM Press.

[29] Yaron Lipman, Olga Sorkine, David Levin, and Daniel Cohen-Or. Linear rotation-invariant coordinates for meshes. *ACM Trans. Graph.*, 24(3):479–487, 2005.

[30] Frank Losasso, Frédéric Gibou, and Ron Fedkiw. Simulating water and smoke with an octree data structure. *ACM Trans. Graph.*, 23(3):457–462, 2004.

[31] Frank Losasso, Tamar Shinar, Andrew Selle, and Ronald Fedkiw. Multiple interacting liquids. In *SIGGRAPH '06: Proceedings of the 33rd annual*

conference on Computer graphics and interactive techniques, New York, NY, USA, 2006. ACM Press.

[32] Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. Fluid control using the adjoint method. *ACM Trans. Graph.*, 23(3):449–456, 2004.

[33] N. Rasmussen, D. Enright, D. Nguyen, S. Marino, N. Sumner, W. Geiger, S. Hoon, and R. Fedkiw. Directable photorealistic liquids. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 193–202, New York, NY, USA, 2004. ACM Press.

[34] William T. Reeves. Particle systems - a technique for modeling a class of fuzzy objects. In *SIGGRAPH '83: Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, pages 359–375, New York, NY, USA, 1983. ACM Press.

[35] Szymon Rusinkiewicz, Olaf Hall-Holt, and Marc Levoy. Real-time 3d model acquisition. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 438–446, New York, NY, USA, 2002. ACM Press.

[36] Tomokazu Sato, Masayuki Kanbara, Naokazu Yokoya, and Haruo Takemura. Dense 3-d reconstruction of an outdoor scene by hundreds-baseline stereo using a hand-held video camera. *Int. J. Comput. Vision*, 47(1-3):119–129, 2002.

[37] Steven M. Seitz and Charles R. Dyer. View morphing. *Computer Graphics*, 30(Annual Conference Series):21–30, 1996.

[38] Andrew Selle, Nick Rasmussen, and Ronald Fedkiw. A vortex particle method for smoke, water and explosions. *ACM Trans. Graph.*, 24(3):910–914, 2005.

[39] Lin Shi and Yizhou Yu. Controllable smoke animation with guiding objects. *ACM Trans. Graph.*, 24(1):140–164, 2005.

[40] Lin Shi and Yizhou Yu. Taming liquids for rapidly changing targets. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 229–236, New York, NY, USA, 2005. ACM Press.

[41] Seung-Ho Shin, Jung Lee, Sun-Jeong Kim, and Chang-Hun Kim. Controlling liquids using pressure jump. In *Proceedings of SIGGRAPH 2006, Sketches & Applications*. ACM Press, 2006.

[42] Jos Stam. Stable fluids. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 121–128, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[43] Nigel Sumner, Samir Hoon, Willi Geiger, Sebastien Marino, Nick Rasmussen, and Ron Fedkiw. Melting a terminatrix. In *Proceedings of SIGGRAPH 2003, Sketches & Applications*. ACM Press, 2003.

[44] Frank Thomas and Ollie Johnston. *The Illusion of Life*. Disney Editions, pp. 62., 1981.

[45] Nils Thürey, Richard Keiser, Ulrich Rüde, and Mark Pauly. Detail-preserving fluid control. In *SCA '06: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, New York, NY, USA, 2006. ACM Press.

[46] Adrien Treuille, Antoine McNamara, Zoran Popović, and Jos Stam. Keyframe control of smoke simulations. *ACM Trans. Graph.*, 22(3):716–723, 2003.

[47] Greg Turk and James F. O'Brien. Shape transformation using variational implicit functions. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 335–342, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[48] Wolfram von Funck, Holger Theisel, and Hans-Peter Seidel. Vector field based shape deformations. In *SIGGRAPH '06: Proceedings of the 33rd annual conference on Computer graphics and interactive techniques*, New York, NY, USA, 2006. ACM Press.

[49] Jue Wang, Stephen M. Drucker, Maneesh Agrawala, and Michael F. Cohen. Cartoon animation filter. In *In SIGGRAPH 06: Proceedings of the 33rd annual conference on Computer graphics and interactive techniques*, New York, NY, USA, 2006. ACM Press.

[50] David White, Kevin Loken, and Michiel van de Panne. Slow in and slow out cartoon animation filter. In *Proceedings of SIGGRAPH 2006, Research Posters*. ACM Press, 2006.

[51] Mark Wiebe and Ben Houston. The tar monster: Creating a character with fluid simulation. In *Proceedings of SIGGRAPH 2004, Sketches & Applications*. ACM Press, 2004.

Appendix A

Optimal Rotation in Two Dimensions

In [21] and [22], a closed form solution for computing the optimal rotation of a set of points is derived for three dimensions. We use this for three-dimensional simulations and our two-dimensional examples use a reduced formulation outlined below. Note that translation, regardless of the dimensionality, is simply $\bar{y} - \bar{x}$, the difference between final and initial center points.

We begin by stating that we seek a rotation, R, such that

$$\tilde{y}_i = R\tilde{x}_i, \tag{A.1}$$

where

$$\tilde{y}_i = (y_i - \bar{y}),$$
$$\tilde{x}_i = (x_i - \bar{x}),$$
$$i = 1, ..., N.$$

Given that we are not necessarily dealing with absolute rigidity, we use a least-

squares error metric, given by:

$$\min_{R} \quad \sum_{i} \|\tilde{y}_i - R\tilde{x}_i\|^2, \tag{A.2}$$

$$\text{subject to } \det(R) = 1. \tag{A.3}$$

This expands to

$$\sum_{i}(|\tilde{y}|^2 + |\tilde{x}|^2 - 2\tilde{y}_i^T R\tilde{x}_i), \tag{A.4}$$

and since $|\tilde{y}|^2$ and $|\tilde{x}|^2$ are independent of R, we reduce the error term to

$$\textstyle\sum_i(-\tilde{y}_i^T R\tilde{x}_i) \tag{A.5}$$

$$= \textstyle\sum_i(\tilde{y}_i^{(1)}\tilde{y}_i^{(2)}) \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} \tilde{x}_i^{(1)} \\ \tilde{x}_i^{(1)} \end{pmatrix}$$

$$= \textstyle\sum_i -(\tilde{y}_i^{(1)}\tilde{y}_i^{(2)}) \begin{pmatrix} \tilde{x}_i^{(1)}\cos(\theta) + \tilde{x}_i^{(2)}\sin(\theta) \\ -\tilde{x}_i^{(1)}\sin(\theta) + \tilde{x}_i^{(2)}\cos(\theta) \end{pmatrix}$$

$$= \textstyle\sum_i - \left[\tilde{y}_i^{(1)}\tilde{x}_i^{(1)}\cos(\theta) + \tilde{y}_i^{(1)}\tilde{x}_i^{(2)}\sin(\theta) - \tilde{y}_i^{(2)}\tilde{x}_i^{(1)}\sin(\theta) + \tilde{y}_i^{(2)}\tilde{x}_i^{(2)}\cos(\theta)\right]$$

$$= \cos(\theta)\left[-\textstyle\sum_i\left(\tilde{y}_i^{(1)}\tilde{x}_i^{(1)} + \tilde{y}_i^{(2)}\tilde{x}_i^{(2)}\right)\right]$$

$$+ \sin(\theta)\left[\textstyle\sum_i(\tilde{y}_i^{(2)}\tilde{x}_i^{(1)} - \tilde{y}_i^{(1)}\tilde{x}_i^{(2)})\right]$$

$$= A\cos(\theta) + B\sin(\theta). \tag{A.6}$$

where

$$A = -\sum_i \left(\tilde{y}_i^{(1)} \tilde{x}_i^{(1)} + \tilde{y}_i^{(2)} \tilde{x}_i^{(2)} \right), \tag{A.7}$$

$$B = \sum_i (\tilde{y}_i^{(2)} \tilde{x}_i^{(1)} - \tilde{y}_i^{(1)} \tilde{x}_i^{(2)}). \tag{A.8}$$

Our constraint (Equation A.3) is now $\cos^2(\theta) + \sin^2(\theta) = 1$, which combines with A.6 to yield:

$$\cos(\theta) = \frac{\pm A}{\sqrt{A^2 + B^2}}, \tag{A.9}$$

$$\sin(\theta) = \frac{\mp B}{\sqrt{A^2 + B^2}}, \tag{A.10}$$