Eurographics/ACM SIGGRAPH Symposium on Computer Animation (2004)
R. Boulic, D. K. Pai (Editors)

Modeling and Editing Flows Using Advected Radial Basis Functions

Frédéric Pighin Jonathan M. Cohen[†] Maurya Shah

University of Southern California, Institute for Creative Technologies
[†]Rhythm & Hues Studios

Abstract

Fluid simulations are notoriously difficult to predict and control. As a result, authoring fluid flows often involves a tedious trial and error process. There is to date no convenient way of editing a fluid after it has been simulated. In particular, the Eulerian approach to fluid simulation is not suitable for flow editing since it does not provide a convenient spatio-temporal parameterization of the simulated flows. In this research, we develop a new technique to learn such parameterization. This technique is based on a new representation, the Advected Radial Basis Function*. It is a time-varying kernel that models the local properties of the fluid. We describe this representation and demonstrate its use for interactive three-dimensional flow editing.*

1. Introduction

High-end visual effects require visually compelling simulations of natural phenomena. While still an active area of research, current computational fluid dynamics techniques produce visually realistic simulations of large scale natural phenomena such as explosions or ocean waves in reasonable computation time.

The focus for physical simulation is shifting away from computational techniques towards effective techniques for authoring, controlling, and interacting with the results of simulations. Turbulent flows are particularly difficult to understand and control due to their unpredictable nature. Small changes in the initial conditions can drastically change the resulting animation. This restricts the design of fluid flows to a tedious trial and error process, where an artist will run a "wedge" of different simulation parameters, and choose the one that most closely matches what the artist intended. To compound the difficulty, there is little an artist can do to alter a flow after it has been computed.

Fluid flow calculations are usually performed on a fixed Eulerian grid using finite differences or finite volume techniques. In an Eulerian approach, the values of the simulation are stored on a regular grid at fixed moments in time. The advantage of this representation is that approximations to differential quantities such as gradients and divergence are easy to formulate and compute, which makes solving the Navier-Stokes Equations straightforward. Unfortunately, for the purpose of fluid editing, Eulerian representations suffer from a lack of spatio-temporal relationship between components of a fluid flow. It is difficult to quantify how different parts of the flow relate to each other over time. This property is critical for fluid editing since any modification of the flow at a particular time step must be propagated both forward and backward in time. The Lagrangian approach to fluid simulation does provide a clear spatio-temporal parameterization by modeling explicitly the motion of the particles in the fluid. These particles move along characteristic curves of the Navier-Stokes Equations called *pathlines*.

Our goal is to combine the advantages of the Eulerian and Lagrangian representations to create a system for interactively manipulating fluid flows. In this system, the fluid can be manipulated as a deformable object. Our approach is to convert Eulerian simulations into Lagrangian representations by advecting particles in the fluid. These particles are used to reparameterize the fluid. The new parameterization is based on a set of radial basis functions centered at each time step around the particles. We call this new fluid representation the *Advected Radial Basis Function* (ARBF) model. The power of the ARBF model is to allow the editing of the fluid by interacting with the particles. As we change the position of the particles, we also change the fluid they represent. Dur-

ing the editing process, spatial and temporal constraints are enforced to maintain the coherency of the flow.

We limit our discussion to buoyancy-driven flows. They include explosions, rising smoke stacks, steam jets, and fires. Buoyancy-driven flows are of particular interest for the entertainment industry and often occur in physical domains that are unbounded by solid walls.

The contributions of this paper include: a formalism of the ARBF model, efficient algorithms for converting a fluid flow between an Eulerian and an ARBF representation, and an interactive method for editing fluid flows.

The rest of the paper is organized as follows. The following section presents an overview of related work. Section 3 describes how we simulate buoyancy-driven flows. Section 4 describes the ARBF representation and provides efficient algorithms for converting between the standard Eulerian and ARBF representations. Section 5 describes a system for editing the results of a fluid simulation. Finally, Section 6 presents results from our system and Section 7 concludes our paper.

2. Related Work

In computer graphics, buoyancy-driven flows are often simulated using Eulerian (grid-based) methods. Previous work has focused on simulating fire [NFJ02], large-scale explosions [RNGF03], compressible and pseudo-compressible explosions [FOA03, YOH00], and hot gases [FSJ01, FM97b, SF93].

Because of their chaotic and turbulent nature, fluid simulations are difficult to control and direct. There have been a few research efforts focussing on this issue. For instance, [FM97a] describes a number of practical interaction techniques for controlling fluid flows. [Gat94] describes an interactive system where by combining several divergence-free flow fields, a user can construct fluid-like vector fields. The system of [Col02] allows interactive manipulations of a fluid simulation by applying forces or setting specific boundary conditions. More recently, [TMPS03] describes a novel system where a user can directly specify "keyframe" density distributions, and multiple shooting algorithm solves for body forces that generate these keyframes. [FL04] and [MTPS04] follow up on this research by proposing numerical techniques adapted to more complex simulations. While these techniques allows the specification of constraints before the fluid is simulated, our system allows the manipulation of the fluid after it has been simulated. As such, our method is designed to postprocess flows.

Lagrangian (gridless) techniques based on Smoothed Particle Hydrodynamics [Mon94] have also been used in computer graphics [PTB*03, MCG03, DG96]. While promising for a number of applications, SPH techniques cannot yet match Eulerian finite-difference techniques in terms of efficiency when the simulated fluid occupies a large portion of the simulation space. However, from a user-interface design point of view, the explicit representations between fluid particles over time is natural to comprehend and manipulate. Recent work by [IH02] and [PSE*00], for example, provides direct manipulation techniques for controling Lagrangian simulations. Therefore, we wish to combine the computational efficiency of Eulerian simulation with a Lagrangian representation suitable for manipulation. The combination of Eulerian and Lagrangian representation has been used for quite some time in the digital effect community where Eulerian representations resulting from simulations are turned into a large sets of particles for the sake of rendering.

The Computational Fluid Dynamics community has been using streamlines (steady flows) and pathlines (unsteady flows) for visualizing flows. Flow visualization techniques can be divided into three groups of techniques: feature based [PVH*03], dense [LHD*04], or geometric techniques [LDPV02]. With feature based techniques, the flow is abstracted using a few features relevant to the researcher such as vortices [JMT02, vWPSP96] or shock waves [MRV96]. Dense techniques represent the flow using a texture computed from the fluid motion. For instance, Linear integral convolution [CL96] can be used to create such texture. Closer to our research, the geometric techniques use geometric objects as a basis for flow visualization. Examples include streamlines, streaklines, and pathlines [Lan94]. Of particular relevance is the work by Turk et al [TB96] related to streamline placement for effective vizualization. In our work, we are similarly interested in choosing pathlines according to some criteria. Our goal however is not to visualize flows but to learn a compact representation.

Our approach started from the realization that the SPH representation and radial basis functions are the same model used in two different fields: computational fluid dynamics and machine learning. Radial basis functions have already been used to solve complex computer graphics problems. For instance, the technique has proved very helpful in modeling implicit surfaces for applications such as shape transformation [TO99] or surface reconstruction [DTS02].

3. Eulerian Fluid Simulation

The Navier-Stokes Equations that govern incompressible buoyancy-driven fluid flows in a homogeneous medium are

$$\begin{aligned}
\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} &= \nu\nabla^2\boldsymbol{u} - \nabla p + \alpha(T - T_0)\boldsymbol{y} \\
&\quad -\beta\rho\boldsymbol{y} + \varepsilon h(\boldsymbol{N}\times\omega) \\
\nabla\cdot\boldsymbol{u} &= 0 \\
\frac{\partial T}{\partial t} + (\boldsymbol{u}\cdot\nabla)T &= k\nabla^2 T,
\end{aligned}$$

where $\boldsymbol{u}$ is the fluid velocity field, p is the pressure field, ν is the kinematic viscosity, T is the temperature field, T_0 is the reference ambient temperature, α is a scalar controlling

the amount of thermal advection, $\boldsymbol{y}$ is the unit vector pointing up ($(0,1,0)$), β is a scalar influencing gravity, and k is the coefficient of thermal diffusion. The value ε controls the amount of small scale details added to the flow through vorticity confinement.

For rendering, we also advect a scalar density field ρ, that represents dust or particles in the fluid

$$\frac{\partial \rho}{\partial t} + (\boldsymbol{u} \cdot \nabla)\rho = 0. \quad (1)$$

The set, $\{\boldsymbol{u}, T, \rho\}$, is a complete solution to the Navier-Stokes Equations. We solve these equations on a uniform grid using the semi-lagrangian method of [Sta99].

4. Lagrangian Model

In this section, we explain how we derive a Lagrangian representation from an Eulerian fluid simulation. In a nutshell, we convert the voxel based representation into a set of radial basis functions following selected pathlines in the flow. These time-varying kernels provide a functional approximation of the flow that is suitable for flow editing.

In what follows, we first present our model and then explain how it can be fitted to an Eulerian fluid simulation.

4.1. Radial Basis Functions Model

Once a fluid simulation has been computed, we convert it into a Lagrangian representation as a set of pathlines. A pathline is the spatial curve traced by a massless particle passively advected by a flow over time. The *Advected Radial Basis Function* (ARBF) representation consists of N particles that move along pathlines of a fluid simulation. Each particle is the center of a Radial Basis Function (RBF), and induces a scalar field for temperature and density in a local area centered about the particle. The total scalar fields are computed by summing over the contribution of each particle.

Let $\boldsymbol{c}_i(t)$ be the center of the i^{th} particle at time t. The RBF centered at this particle, $\phi_i(t,\boldsymbol{x})$, is defined by

$$\phi_i(t,\boldsymbol{x}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{c}_i(t)\|^2}{\sigma_i(t)^2}\right),$$

where $\sigma_i(t)$ is the time-dependent radius of the particle. We chose a gaussian kernel for its smoothness. Its disadvantage is its computational cost; it has led researchers [DG96] to use a spline approximation. In our experience, by taking advantage of the radial symetry and limiting the support of the kernel, we can sample the RBF approximation in a cube with an edge of 50 voxels in a tenth of a second on a high performance PC. This performance is adequate for the purpose of this research.

The RBF model approximates the density and temperature fields as

$$\hat{T}(t,\boldsymbol{x}) = \sum_{i=1}^{N} W_T^i(t)\phi_i(t,\boldsymbol{x}) \quad (2)$$

$$\hat{\rho}(t,\boldsymbol{x}) = \sum_{i=1}^{N} W_\rho^i(t)\phi_i(t,\boldsymbol{x}), \quad (3)$$

where N is the number of particles and $W_T^i(t)$ and $W_\rho^i(t)$ are scalar weights. Thus the ARBF model consists of a set of N time-dependent particles, $\{\boldsymbol{c}_i(t), \sigma_i(t), W_T^i(t), W_\rho^i(t)\}_{i=1,t=0}^{N,T-1}$.

In the rest of this section, we explain how the ARBF parameters are estimated from an Eulerian flow.

4.2. Converting Eulerian to Lagrangian

Given a solution set to the Navier-Stokes Equations, $\{\boldsymbol{u}, T, \rho\}$, to fit the ARBF model to this solution in the least squares sense would require solving the following minimization problem:

$$\begin{aligned}&\mathrm{Argmin}_{\{\boldsymbol{c}_i(t),\sigma_i(t),W_T^i(t),W_\rho^i(t)\}_{i=1,t=0}^{N,L-1}} \\ &\quad a\sum_t \left(\sum_{j,k,l} \left(\hat{T}(t,\boldsymbol{x}_{j,k,l}) - T(t,\boldsymbol{x}_{j,k,l})\right)^2 \right) + \\ &\quad b\sum_t \left(\sum_{j,k,l} \left(\hat{\rho}(t,\boldsymbol{x}_{j,k,l}) - \rho(t,\boldsymbol{x}_{j,k,l})\right)^2 \right),\end{aligned}$$

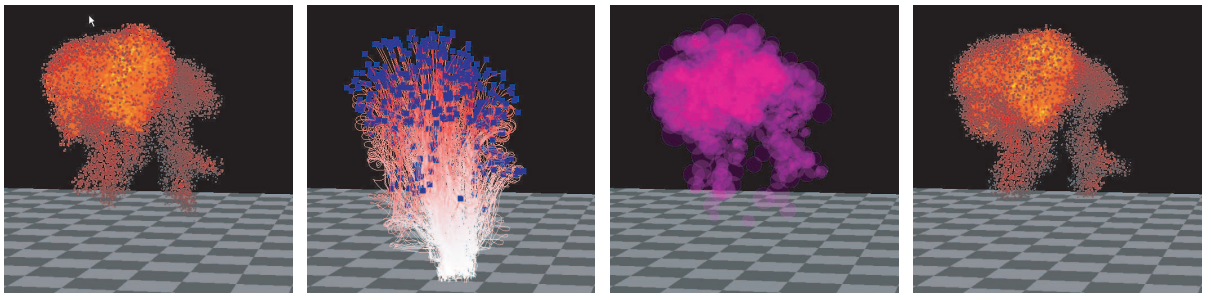
where $\boldsymbol{x}_{j,k,l}$ is the set of grid points in the computational domain, L the number of steps in the simulation, and a and b are scalar weights.

This optimization problem is highly non-linear, and the ARBF representation has an enormous number of degrees of freedom, making this minimization difficult. To further complicating matters, the optimal number of particles N is unknown, making this a combinatorial search problem as well. Our approach is to break this problem into smaller parts corresponding to different groups of variables. We solve sequentially for the following groups of variables:

1. Particle trajectories, $\{\boldsymbol{c}_i(t)\}_{i,t}$, and number of particles, N.
2. Particle radii, $\sigma_i(t)$.
3. Weights, $W_\rho^i(t)$ and $W_T^i(t)$.

The four images in Figure 1 illustrate the ARBF fitting algorithm. Figure 1 (a) shows a visualization of an Eulerian simulation at a point in time. Figure 1 (b) shows a visualization of the pathlines selected by our algorithm. Figure 1 (c) shows a visualization of the radii of the radial basis function at the same frame. Finally, figure 1 (d) shows a visualization of the sampled ARBF model.

The rest of this section explains our fitting algorithm in details, starting with a procedure for computing pathlines.



(a) (b) (c) (d)

Figure 1: *ARBF fitting: (a) fluid simulation, (b) pathlines of advected particles, (c) radial basis functions fitted to the fluid's volume, and (d) visualization after sampling the ARBF model.*

4.3. Computing Pathlines

We have developed an algorithm for selecting a small set of pathlines that best represents a simulated flow. This algoritm is designed with two goals in mind. First, the position of the particles must capture the volume of the fluid that has non-zero density at all points in time. Second, we would like to use as few particles as possible to end up with a compact representation. Our algorithm is based on a data structure, called the *Occupancy Map*, that keeps track of the particle density at all points in time. We adaptively select pathlines as long as they improve the coverage of the flow or until a user-specified mean particle density is met. In what follows, we first describe how we compute pathlines from the Eulerian simulation. We then detail the pathline selection process.

Particle Advection. A pathline is the trajectory of a massless particle in the fluid through time. Traditionnaly, pathlines are generated by integrating the velocity field forward in time [Lan94]. This method respects the definition of a pathline but, in practice, it does not faithfully capture the motion of the simulated fluid. The issue comes from the discrepancies between the physical model described by the Navier-Stokes Equations and their actual implementation in the simulator. The simulator provides an approximate solution to these equations. To advect the particles in a way that is compatible with the simulation, we compute at each time step the three-dimensional displacement field $\boldsymbol{d}$ that corresponds to the advection step. Given an initial position, $\boldsymbol{x}(0)$, the pathline can be computed by integrating forward in time

$$\frac{\partial \boldsymbol{x}(t)}{\partial t} = \boldsymbol{d}(t, \boldsymbol{x}(t)). \tag{4}$$

The displacement field is computed by seeding each voxel with its position in space and advecting these values through the simulator. By substracting at each voxel the resulting values from the seed values, we obtain our displacement field.

The Occupancy Map. In order to adaptively sample the fluid's volume with particles, we maintain a data structure that indicates the sample density around each voxel. We call this data structure the *Occupancy Map*. It stores at each voxel and each point in time the volume of the largest axis aligned cube centered at this voxel that does not contain any particles. Let us call $O_{i,j,k}(t)$ the value of the occupancy map at voxel i, j, k and time t. We define the "score" of a particle as

$$Score(\boldsymbol{c}_i) = \sum_{t=0}^{L-1} O_{Idx(\boldsymbol{c}_i(t))}(t),$$

where the function Idx maps spatial coordinates onto voxel indexes. This score will be high when the pathline $\{\boldsymbol{c}_i(t)\}_t$ passes through a region of the domain that has not been covered by previous pathlines. After a particle is traced, we conservatively update the occupancy map. At time t, we update the cells in the occupancy map that are in a cube centered at the position of the particle and whose volume is $O_{i,j,k}(t)$.

With this data structure in hand, we can now describe our adaptive pathline selection algorithm.

Adaptive pathline selection. The goal of this algorithm is to select pathlines sequentially so that each pathline has the largest possible score given the previously selected pathlines. It is a greedy algorithm. We select pathlines by choosing a voxel and assigning it an initial position that is uniformly distributed within the volume of the voxel. Only voxels that have a non-zero density at $t = 0$ are selected. To evaluate if a particular voxel has a high probability of producing a high-score particle, we maintain two statistics: the variance of the particle scores at each voxel and the mean of all the particles scores. The idea is to then choose voxels that have high variance but to accept pathlines only if their score is greater than the overall mean. When a voxel is selected, we evaluate the score of a pathline starting from a random location within the voxel. Whether we accept or reject the particle, we use the score to update the statistics. The pseudo-code given in Algorithm 1 describes this in detail.

Because the mean score is updated on Line 6, even if the candidate pathline is rejected, this algorithm will always converge. Eventually, either the mean will decrease below *thresh* on Line 13 and the algorithm will terminate, or

Algorithm 1 Pathline selection.

Input: *thresh*

repeat
  Choose grid cell i, j, k with highest $var_{i,j,k}$.
  Choose random point in cell i, j, k.
  Trace new pathline $\boldsymbol{c}(t)$ starting from this point at t_0.
  Compute $Score(\boldsymbol{c})$.
  if $Score(\boldsymbol{c}) > mean$ **then**
    Accept pathline $\boldsymbol{c}$.
    Update $O_{i,j,k}(t)$ for all grid cells at all times.
  else
    Reject pathline $\boldsymbol{c}$.
  end if
  Update *mean* and $var_{i,j,k}$.
until $mean < thresh$

the mean will decrease enough that a new pathline's score will be accepted. In this way, the algorithm will select as many pathlines as necessary to cover the simulation domain with the given quality level indicated by the user parameter *thresh*. In our experiments, we use a threshold equal to $27 \times L$. This corresponds to a density of one particle for each cube of length 3.

Since the mean and variances of the data are not known at the beginning, we bootstrap the algorithm by tracing a few particles in each voxel (10 in our experiments) without keeping their pathlines.

4.4. Radius and Weights Estimation

Once a set of pathlines has been selected, the set of radii, $\{\sigma_i(t)\}_i$, for all particles at time t, is estimated based on the distance from each particles to its neighboring particles at each time step. For each particle, we compute the distance to its closest neightbor, and we then assign to the particle a radius proportional to this distance. In our experiments, we found that multiplying this distance by a factor in the range $[1.5, 2]$ yields good results. A variant of this algorithm is to clamp the radius of the particles that are near the fluid's boundaries to avoid poor RBF reconstruction at the boundaries. This is particularly important if a small number of particles is used.

At each time step, we now have a set of radial basis functions defined by their centers and radii. To complete the approximation defined by Equations 2 and 3, we need to estimate the weights $W_\rho^i(t)$ and $W_T^i(t)$. It is straightforward to estimate these values at each time step by solving

$$\begin{aligned} \text{Argmin}_{\{W_\rho^i(t)\}_{i=1}^N} \quad & \sum_{j,k,l} \left(\hat{\rho}(t, \boldsymbol{x}_{j,k,l}) - \rho(t, \boldsymbol{x}_{j,k,l})\right)^2 \\ & + \mu \sum_i W_\rho^i(t)^2 \\ \text{with} \quad & W_\rho^i(t) \geq 0 \end{aligned}$$

and

$$\begin{aligned} \text{Argmin}_{\{W_T^i(t)\}_{i=1}^N} \quad & \sum_{j,k,l} \left(\hat{T}(t, \boldsymbol{x}_{j,k,l}) - T(t, \boldsymbol{x}_{j,k,l})\right)^2 \\ & + \mu \sum_i W_T^i(t)^2 \\ \text{with} \quad & W_T^i(t) \geq 0, \end{aligned}$$

where μ is a regularization parameter set to .1 in our experiments. This parameter could be estimated automatically using a criteria such as generalized cross valivation [Orr99]. We require the weights to be positive since they are associated with positive physical quantities in our system (see Section 5). These minimization problems fall in the category of quadratic programming problems. Note that these systems are quite sparse because each kernel has a small support. Our solver is based on a sparse interior point method [NW99].

Figure 2 illustrates the quality of the fit. More examples are provided in the video. Some of the discrepancies visible in the examples are due to the random perturbations used to texture the fluids. Section 6 provides a quantitative evaluation of the fitting error.

5. Interactive Pathline Editing

The ARBF representation provides a framework that makes interactive flow editing possible. The main idea is to use the particles as a set of handles through which the flow can be manipulated as a deformable object. We consider the set of pathlines as a pliable object that can be bent by moving particles. Bending the pathlines allows the smooth propagation in time of editing operations. To maintain continuity in the flow, we enforce two types of constraints. The first type tries to keep the set of particles coherent at a specific time step; these are spatial constraints. The second type are temporal constraints: a modification at a specific point in time needs to be propagated both forward and backward in time. We enforce spatial constraints by using a Smooth Particle Hydrodynamic (SPH) system. This system operates by identifying the RBF kernels in the ARBF model as SPH particles. We enforce weak temporal constraints by propagating smoothly in time changes in the particles positions. This propagation is performed using hierarchical B-Spline interpolation. Each editing operation includes 4 steps:

1. Selection of a group of particles, G.
2. Displacement of G.
3. Spatial constraint enforcement through an SPH simulation.
4. Temporal propagation through B-Spline interpolation.

Steps 1 and 2 are performed by the animator; steps 3 and 4 are automatically handled by the system.

Our editing algorithm is related to the hierarchical full-body motion algorithm system proposed by [LS99]. The difference is, where Lee et al alternate spatial and temporal constraint enforcements (or, using the terminology of [LS99],

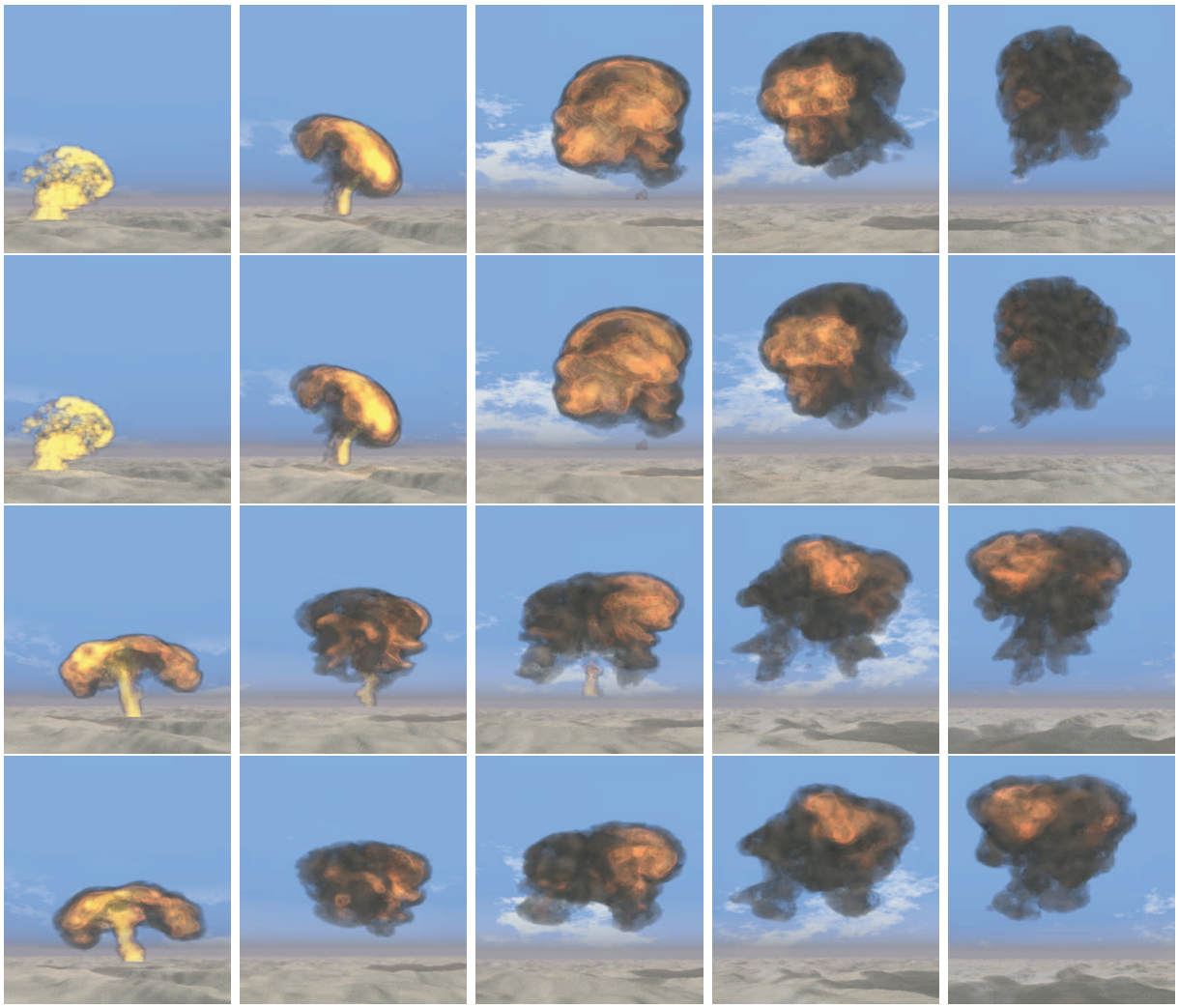


Figure 2: *Comparison between data and model. Rows 1 and 3 show two raytraced simulated fluids. Rows 2 and 4 show the respective sampled ARBF models.*

intraframe and interframe constraints), our algorithm enforces the constraints sequentially. This choice stems from fundamental differences between the SPH system, used to enforce flow constraints, and the inverse kinematics system, used to enforce body constraints. First, the SPH simulator is significantly slower than the fast inverse kinematic system used in [LS99]. Iterating the SPH system yields poor performances. Second, the changes in the fluid introduced by the SPH simulation is less intuitive than the ones introduced by the IK system. In particular, running an SPH simulation at a time step that is not the one currently displayed limits the control that the animator has over the edited fluid. The rest of this section gives more details on the SPH system and the B-Spline interpolation scheme.

5.1. Smooth Particles Hydrodynamics

We use an SPH simulator to enforce spatial constraints. In SPH, particle density ρ is a quantity that describes how many particles occupy a region of space. Incompressible fluids are difficult to simulate within an SPH framework since they give rise to very stiff systems. However, we do not use SPH to simulate the fluid but rather to create a system that reacts to perturbations of the particles. To do this, we identify the dust density ρ as the density of a compressible fluid. When the positions of the particles are changed, the change in density creates a pressure differential that sets the particles in motion. The ARBF representation translates into the SPH framework by using $\boldsymbol{c}$ as the center, σ as the radius, $W_\rho^i(t)$ as the mass, and ϕ as the kernel. Particle density, a scalar field

denoted $\rho(x)$, is then defined as

$$\rho(x) = \sum_{i=1}^{N} \boldsymbol{W}_{\rho}^{i}(t)\phi_i(t,x).$$

This equation is the same as the RBF model approximating the density field as described in Equation 2. We assimilate a machine learning concept, radial basis functions, to a fluid dynamics representation, smooth hydrodynamics particles. Note that the variable t representing time in the original simulation is constant during the SPH simulations since these are used to enforce spatial constraints. We will use τ to represent time in the SPH formulation. τ represents a fictitious time and allows the integration of the SPH equations for a fixed value of t. We now give the equation of motion for the SPH framework. The acceleration of the particles follow the pressure gradient:

$$\begin{aligned}\frac{\partial^2 \boldsymbol{c}_i(\tau)}{\partial \tau^2} &= \frac{-\nabla P_i(\tau)}{\rho_i(\tau)} \\ &= -\sum_{j=1}^{N} \boldsymbol{W}_{\rho}^{i}(t)\left(\frac{P_i}{\rho_i^2}+\frac{P_j}{\rho_j^2}\right)\nabla\phi_i(\boldsymbol{c}_j(\tau)) \quad (5)\end{aligned}$$

and fluid pressure is defined via an Equation of State as

$$P(\tau,x) = \rho(\tau,x)\left(\left(\frac{\rho(\tau,x)}{\rho_0(x)}\right)^{A} - 1\right),$$

where ρ_0 is a reference density, and A is an exponent chosen to make the system more or less stiff (we use $A = 7$). For notational convenience, we define $P_i = P(\tau, \boldsymbol{c}_i(\tau))$ and $\rho_i = \rho(\tau, \boldsymbol{c}_i(\tau))$. This formulation is due to [Mon94].

We evaluate $\rho(\tau,x)$ by resampling the SPH formulation using the new position of the particles $\{\boldsymbol{c}_i(\tau)\}_i$:

$$\rho(\tau,x) = \sum_{i=1}^{N} W_{\rho}^{i}(t)\phi_i(\tau,x).$$

Note that $W(t)_{\rho}^{i}(t)$ is not a function of τ.

If we integrate each particle forward in fictitious time using this acceleration, the particles will eventually reach an equlibrium where the particle density matches the distribution given by ρ_0. To aid convergence, we also add to each particle's acceleration a viscosity term, $-v\partial(\boldsymbol{c}_i)/\partial\tau$. For the implementation of the SPH system, we follow the recommendations of [Roy95].

The SPH system is used as follows. The original particle positions at time t are given, $\{c_i(t)\}_i$. Some particles are then perturbed by the user to new positions, $\boldsymbol{c}_i(\tau_0)$. We want to adjust all particles to resatistfy the continuity equation at time t. This is accomplished by setting ρ_0 based on the original positions $\{\boldsymbol{c}_i(t)\}_i$ of all particles, and integrate the particles in fictitious time using Equation 5.

5.2. Constraint Propagation Through Hierarchical B-Spline Interpolation

Given a set of positional contraints $\{(\boldsymbol{p}_i^j, t_j)\}$, where $\boldsymbol{p}_i^j$ is the new position of particle i at time t_j, we would like to propagate this constraint backward and forward in time to temporaly filter the modifications of the flow. This propagation is done by fitting a smooth function, $\boldsymbol{s}(t)$, to the sparse set of displacements $\{(\boldsymbol{p}_i^j - \boldsymbol{c}_i(t_j), t_j)\}$, where $\boldsymbol{c}_i(t_j)$ is the original position of the particle. We solve this scattered data interpolation problem using hierarchical B-Spline interpolation. We have found this technique to be very effective for smoothly interpolating a set of constraints that are non-uniformly sampled in time. For the implementation of hierarchical B-Spline interpolation, we were inspired by [LS99]. In this framework, $\boldsymbol{s}$ is decomposed using a sequence of functions, $\boldsymbol{d}_0, \ldots, \boldsymbol{d}_n$, modeling increasing levels of details, such that $\boldsymbol{s}$ is approximated at level n by $\boldsymbol{s}_n = \boldsymbol{d}_0 + \ldots + \boldsymbol{d}_n$. $\boldsymbol{s}_n$ is recursively derived from $\boldsymbol{s}_{n-1}$ using the constraints

$$\boldsymbol{d}_n(t_i) = \boldsymbol{p}_i^j - \boldsymbol{c}_i(t_j) - \boldsymbol{s}_{n-1}(t_j).$$

In other words, we estimate $\boldsymbol{d}_n$ so that it interpolates at the constraint points the error made by the approximation at level $n-1$. In our implementation, we use the robust cubic B-Spline fitting technique described in [LS99].

6. Results

Using the algorithm described in this paper, we fit the ARBF model to Eulerian simulations. The supporting video illustrates some of these results whereas table 1 gives a quantitative assessment.

To vizualize our results, we used two rendering algorithms. The first one uses a set of hardware rendered particles (*GL_POINTS* primitives) whose size is proportional to the local density. For each voxel, we render 20 particles whose locations are uniformely jiterred within the voxel. The second one is a modified version of the POVray public domain raytracer. Both renderers map temperatures onto colors using hand picked color maps. These color maps where determined by studying images of explosions. We tried a black body radiation framework but we were disappointed by the results.

The Eulerian simulations were generated using a modified version of an implementation of Jos Stam's algorithm [Sta03]. We found that the amount of turbulence in these simulations strongly depends on two parameters: the intial conditions and the scale of the vorticity confinement force. To determine the inital conditions, we use a fractal function for temperature, velocities, and densities. The user specifies maximum values for all three field and the extent of the initial volume (cube or sphere). The system then jitters randomly these values at each voxel according to a simple fractal pattern.

Table 1 describes numerically a few fitting experiments.

Grid size	Steps	Particles	Simulation (m)	Advection (s)	RBF (m)	RMS-D	RMS-T	Compression
$50 \times 80 \times 50$	60	608	10.4	19.1	11.1	.026	2.65	285
$50 \times 60 \times 50$	60	428	7.4	9.2	4.5	.040	3.03	303
$50 \times 80 \times 50$	100	473	18.2	35.9	5.3	.023	1.86	366
$50 \times 80 \times 50$	100	473	18.6	23.9	3.8	.021	1.72	352
$70 \times 80 \times 50$	90	605	21.0	25.4	12.4	.025	1.90	401

Table 1: *Fitting results. Timings for the ARBF fitting algorithm is split into two stages: an advection stage, corresponding to the selection and calculation of the pathlines, $\{c_i(t)\}$, and an RBF stage, corresponding to the calculations of the radial basis function parameters $\{\sigma_i(t), W_\rho^i(t), W_T^i(t)\}$. The advection timings are given in seconds; the RBF timings in minutes.*

All experiments were performed on a 2.4 GHz PC. We have split the timings into an advection stage, corresponding to the selection of the pathlines, and a fitting stage, corresponding to the calculation of the RBF model. Note that RBF calculations take much longer than the selection of the pathlines. This is not surprising since advecting and evaluating a particle can be done extremely quickly. However, solving for the RBF weights is time-consuming. Surprisingly, the bulk of the computation is not spent by the quadratic programming routine; rather it is spent during set up. Setting up the system of equations requires many evaluations of the basis functions. We think a lot of these evaluations are actually redundant and that a more careful implementation could speed up this stage considerably. Column 6 and 7 of the results table display the root mean square fitting error for the velocity and the temperature field. To avoid amortizing the error over empty voxels, we did not include in the rms summation voxels that were empty both in the simulation and the reconstruction. Densities in the fluid are within the interval $[0,1]$ and the temperatures are within $[293,4000]$. The reconstruction errors are quite small in all the examples. Since in all 5 experiments we use the same density threshold to determine the number of particles, the reconstruction errors are also quite similar.

We also evaluate the compression ratio obtained with the ARBF model. At a given point in time, in an Eurelian representation each voxel is described by 5 floating points (a 3D velocity, density, and temperature), whereas for the ARBF representation each particle is crescribed by 6 floating point values ($\boldsymbol{c}(t)$, $\sigma(t)$, $W_\rho^i(t)$, and $W_T^i(t)$). The compression amounts to factors of several hundreds in most cases. Even though it is a lossy compression, it could be an interesting way of storing fluid simulations. This is an application we intend to explore more thoroughly.

Table 2 provides timing results for a few editing operations. We report timings for experiments on two different flows. For each set, we performed editing operations involving varying numbers of particles. The second column shows how many particles were selected during the manipulation. The timings are all in seconds. The 3rd column gives the running time of the SPH simulator, the 4th covers the B-Spline propagation, the 5th column gives the time it took to resample the fluid, and the last one gives the total time. When all particles are selected, there is no SPH computations. Also, note that since we evaluate all kernels, the sampling time does not depend on how many particles were selected. A more efficient approach would be to only update the part of the fluid that has been affected. The total computations time are between [1.11, 2.30] seconds. This causes a noticeable delay during editing but it does not hinder significantly the interactivity of the process.

Total part.	Edited part.	SPH	B-Spline	Sampling	Total
700	700	0.00	0.97	0.14	1.11
700	166	1.33	0.83	0.14	2.30
700	5	1.01	0.50	0.14	1.65
346	700	0.00	0.66	0.72	1.38
346	70	0.22	0.53	0.72	1.47
346	9	0.18	0.45	0.72	1.35

Table 2: *Editing results. This table shows timings during editing operations. The timing has been split into three stages. First, the SPH calculations, then the computation related to the Hierarchical B-Spline interpolation, and finally the resampling of the ARBF to visualize the new flow. All timing results are given in seconds.*

The video presents two editing sessions. In the first one, the fluid is manipulated interactively and selected regions of the fluids are moved around. In the second one, we show how the same technique can be used to deform a flow to take into account a newly introduced obstacle. We use a simple procedure to bend the pathlines around the sphere. For each pathline, we compute the point that comes closest to the center of the sphere. If the RBF at this point intersects with the sphere, we move it outside of the sphere so that the RBF and the sphere become tangent. We then propagate these displacements using the technique described in Section 5. The

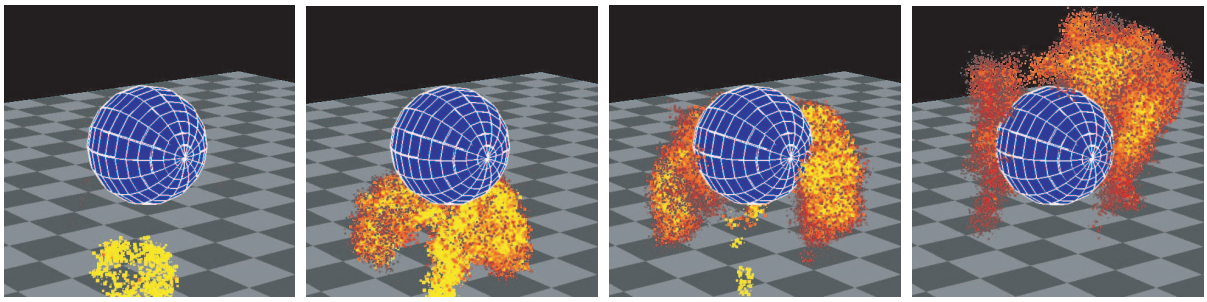


Figure 3: *Fluid fitted around a sphere using the ARBF model.*

image sequence in Figure 3 shows a few frames of the final fluid, wrapping around the sphere after colision avoidance.

7. Conclusion

In this paper, we have presented a novel way to look at fluid simulations. The representation we introduced is not completely new since both Eulerian fluid simulation and particle systems are well known to the computer graphics community. Our contribution lies elsewhere: by identifying a machine learning algorithm (radial basis function) to a computational fluid mechanics technique (smoothed particles hydrodynamics), we where able to manipulate Eulerian flows in ways that would have been very difficult with a voxel-based structure. We do not believe that the flow editing tool we presented in this research is the ultimate solution for the interactive manipulations of flows. However, we think it could find its way in a palette of tools since it addresses some interesting issues. In particular, it is useful to introduce very quickly slight modifications in a flow. The example of the collision with a sphere in the video is particularly representative.

Our approach is not without limitations. In particular, the use of particles intrinsically limits the granularity of the representation. We fit a finite number of particles to the fluid. This number determines how much details we can model within the fluid. It is particularly important for the editing application: no editing operations in the current system can affect the fluid at a scale smaller than the particles. One way of addressing this issue is to use a multi-resolution particle representation that would adaptively split particles to the resolution needed [DC99]. Also, we would like to find a more physical way of enforcing interframe constraints. The interpolation of the constraints through hierarchical B-Spline interpolation is oblivious of the physics of fluid particles. One possibility would be to use the Bernoulli equation for unsteady fluids as a model to relate the geometry of a pathline to the forces acting on the particle.

The results of large turbulent Eulerian fluid simulations can require a lot of disk space. The ARBF could be used to compress these simulations.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments. We thank Penne Lee for helping us with the raytracer, Wen C. Tien for his assistance during the paper submission, and Jerry Tessendorf for helpful discussions. This paper was developed with funds of the Department of the Army under contract number DAAD 19-99-D-0046. Any opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the Department of the Army.

References

[CL96] CABRAL B., LEEDOM L.: Imaging vector fields using line integral convolution. In *Proceedings of SIGGRAPH 96* (Aug. 1996), Computer Graphics Proceedings, Annual Conference Series, pp. 263–270.

[Col02] COLEMAN P.: *Motion Control for Fluid Animation: Flow Along a Control Path*. Tech. rep., Ohio State University, Undergraduate thesis, 2002.

[DC99] DESBRUN M., CANI M. P.: *Space-time adaptive simulation of highly deformable substances*. Tech. Rep. 3829, INRIA, December 1999.

[DG96] DESBRUN M., GASCUEL M. P.: Smoothed particles: a new paradigm for animating highly deformable bodies. In *EGCAS '96: Seventh International Workshop on Computer Animation and Simulation* (1996).

[DTS02] DINH H., TURK G., SLABAUGH G.: Reconstructing surfaces by volumetric regularization using radial basis functions. In *IEEE Transactions on Pattern Analysis and Machine Intelligence* (Oct. 2002), IEEE, pp. 1358–1371.

[FL04] FATTAL R., LISCHINSKI D.: Target-driven smoke animation. In *SIGGRAPH 2004 Conference Proceedings* (August 2004).

[FM97a] FOSTER N., METAXAS D.: Controlling fluid animation. In *Proceedings Graphics Interfaces 1997* (1997), pp. 178–188.

[FM97b] FOSTER N., METAXAS D.: Modeling the motion of a hot, turbulent gas. In *Proceedings of SIGGRAPH 97* (Aug. 1997), pp. 181–188.

[FOA03] FELDMAN B. E., O'BRIEN J. F., ARIKAN O.: Animating suspended particle explosions. *ACM Transactions on Graphics 22*, 3 (July 2003), 708–715.

[FSJ01] FEDKIW R., STAM J., JENSEN H. W.: Visual simulation of smoke. In *SIGGRAPH 2001 Conference Proceedings* (August 2001), pp. 15–22.

[Gat94] GATES W. F.: *Interactive Flow Field Modeling for the Design and Control of Fluid Motion in Computer Animation*. Master's thesis, University of California at Davis, 1994.

[IH02] IGARASHI T., HUGHES J. F.: Clothing manipulation. In *Proceedings of User Interface Software and Technology 2002* (October 2002), pp. 91–100.

[JMT02] JIANG M., MACHIRAJU R., THOMPSON D.: A novel approach to vortex core region detection. In *Proceedings SYmposium on data Visualization* (2002), pp. 178–188.

[Lan94] LANE D.: Ufat - a particle tracer for for time-dependent flow fields. In *Proceedings of IEEE Visualization* (Aug. 1994), pp. 257–264.

[LDPV02] LARAMEE H. H. R. S., DOLEISCH H., POST F. H., VROLIJK B.: *The state of the art in flow visualization, part 1: direct, texture-based, and geometric techniques*. Tech. Rep. TR-VRVis-2002-046, VRVis Research Center, 2002.

[LHD*04] LARAMEE R. S., HAUSER H., DOLEISCH H., VROLIJK B., POST F. H., WEISKOPF D.: The state of the art in flow visualization: dense and texture-based techniques. *Computer Graphics Forum 23*, 3 (2004).

[LS99] LEE J., SHIN S. Y.: A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of SIGGRAPH 1999* (Aug. 1999), pp. 39–48.

[MCG03] MULLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proceedings of the Symposium on Computer Animation 2003* (2003), pp. 154–159.

[Mon94] MONAGHAN J.: Simulating free surface flows with SPH. *Journal of Computational Physics 110*, 2 (1994), 399–406.

[MRV96] MA K. L., ROSENDALE J. V., VERMEER W.: 3d shock wave visualization on structured grids. In *Proceedings Annual Symposium on volume visualization* (Oct. 1996), pp. 154–159.

[MTPS04] MCNAMARA A., TREUILLE A., POPOVIC Z., STAM J.: Fluid control using the adjoint method. In *SIGGRAPH 2004 Conference Proceedings* (August 2004).

[NFJ02] NGUYEN D. Q., FEDKIW R., JENSEN H. W.: Physically based modeling and animation of fire. In *SIGGRAPH 2002 Conference Proceedings* (August 2002), pp. 721–728.

[NW99] NOCEDAL J., WRIGHT S.: *Numerical Optimization*. Springer, New York, 1999.

[Orr99] ORR M. J. L.: *Recent Advances in Radial Basis Function Networks*. Tech. rep., Institutre for Adaptive and Neural Computation, Edinburgh University, 1999.

[PSE*00] POPOVIĆ J., SEITZ S. M., ERDMANN M., POPOVIĆ Z., WITKIN A. P.: Interactive manipulation of rigid body simulations. In *Proceedings of ACM SIGGRAPH 2000* (July 2000), pp. 209–218.

[PTB*03] PREMOZE S., TASDIZEN T., BIGLER J., LEFOHN A., WHITAKER R.: Particle based simulation of fluids. In *Proceedings of Eurographics 2003* (September 2003).

[PVH*03] POST F. H., VROLIJK B., HAUSER H., LARAMEE R. S., DOLEISCH H.: The state of the art in flow visualization: feature extraction and tracking. *Computer Graphics Forum 22*, 4 (2003), 775–792.

[RNGF03] RASMUSSEN N., NGUYEN D. Q., GEIGER W., FEDKIW R. P.: Smoke simulation for large-scale phenomena. *ACM Transactions on Graphics 22*, 3 (July 2003), 703–707.

[Roy95] ROY T. M.: *Physically Based Fluid Modeling Using Smoothed Particles Hydrodynamics*. Master's thesis, University of Illinois at Chicago, 1995.

[SF93] STAM J., FIUME E.: Turbulent wind fields for gaseous phenomena. In *Proceedings of SIGGRAPH 93* (Aug. 1993), pp. 369–376.

[Sta99] STAM J.: Stable fluids. In *Proceedings of SIGGRAPH 1999* (Aug. 1999), Computer Graphics Proceedings, Annual Conference Series, pp. 121–128.

[Sta03] STAM J.: Real-time fluid dynamics for games. In *Proceedings of the Game Developer Conference* (Mar. 2003).

[TB96] TURK G., BANKS D.: Image-guided streamline placement. In *SIGGRAPH 96 Conference Proceedings* (Aug. 1996), ACM SIGGRAPH, pp. 453–460.

[TMPS03] TREUILLE A., MCNAMARA A., POPOVIĆ Z., STAM J.: Keyframe control of smoke simulations. *ACM Transactions on Graphics 22*, 3 (July 2003), 716–723.

[TO99] TURK G., O'BRIEN J.: Shape transformation using variational implicit functions. In *SIGGRAPH 99 Conference Proceedings* (Aug. 1999), ACM SIGGRAPH, pp. 335–342.

[vWPSP96] VAN WALSUM T., POST F., SILVER D., POST F.: Feature extraction and iconic visualization. *IEEE Transactions on Visualization and Computer Graphics 2*, 2 (1996), 111–119.

[YOH00] YNGVE G. D., O'BRIEN J. F., HODGINS J. K.: Animating explosions. In *Proceedings of SIGGRAPH 2000* (July 2000), pp. 29–36.