

Preview Directed Control of Complex Fluid Simulations

Zeki Melek John Keyser

Department of Computer Science
Texas A&M University
College Station, TX 77843-3112, USA
E-mail: z0m8905@cs.tamu.edu
Phone: (979) 862-1792
Fax: (979) 847-8578

August 1, 2007

Abstract

In this work, we describe an automated method for directing the control of a high resolution fluid simulation based on the results of a lower resolution preview simulation. In complex simulations with events triggered by specific criteria, small variations in accuracy between low and high resolution grids can lead to divergent simulations. Our goal is to provide a simple method for ensuring that the high resolution simulation matches key properties from the lower resolution simulation. We allow a user to interact with a fast, coarse simulation, until a desired behavior is achieved. Our automated method samples the data to be matched at various positions, scales, and times in the simulation, while also recording user interaction. During the high resolution simulation, a matching process ensures that the values sampled are maintained. This matching process keeps the different resolution simulations aligned even for complex systems, such as fire simulation, where different properties can trigger chains of reactions and events. Because the final simulation is naturally similar to the preview simulation, the amount of control required is relatively low, allowing a simpler control method than that used in prior keyframing approaches.

Key Words: simulation control, physically based modeling, fire simulation, fluid dynamics

1 Introduction

Advances in the fluid simulations underlying many natural processes have made visual simulation of such phenomena quite popular. Despite recent hardware and algorithmic improvements, however, these simulations of water, smoke, clouds, fire, etc. can take a long time (many minutes to hours) to compute.

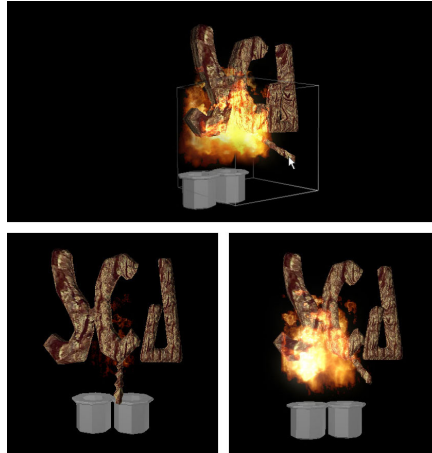


Figure 1. Uncontrolled simulation on the left, where the logo does not catch fire, and controlled simulation on the right, with the logo catching fire as seen in the preview (top).

There are two particular challenges faced in controlling simulations in many applications. One of these is making the simulation match some artificial, user-defined keyframe (e.g. smoke forming a shape). This desire has driven recent research in which the fluid system is forced or “encouraged” to match those keyframes as closely and smoothly as possible. A second challenge is the long time taken to produce a production-quality simulation. Only after a long period of simulation might one discover that a simulation is not producing the right effect, and evaluating the effect of any change is also time consuming. Our work presented here is designed to address the latter issue.

A reasonable approach is to first run a fast, low-resolution simulation before the “full scale” one, but these very changes in the grid and time resolution can result in different simulation behavior. Beyond the increases in complexity of the events and changes in the initial conditions, the numerical simulation itself will behave differently when the fluid grid is resized. Numerical dissipation in the vector field, which causes an artificial viscosity inherent to the semi-lagrangian Stable Fluids, will decrease as the cell size decreases. Also, better sampling of boundary cells for objects in the grid changes the shape of the fluid volume (this has been combatted by adaptive subdivision [18] and tetrahedral meshes [7, 16]).

In this paper, we present a simple method for matching a fine grid fluid simulation to a coarse resolution “preview” simulation. This enables one to first use a simple interactive simulation (e.g. a GPU based fluid solver [11]) to produce a plausible, but rough, simulation. With the interactive simulation, a user can fine-tune and choreograph initial conditions and parameters quickly while getting visual feedback. We sample the data from this preview simulation in order to capture the gross properties of the simulation that we are most interested in maintaining. Our matching process makes the final simulation more closely match these samples, and thus maintain the properties of interest.

The obvious concern is that by matching to a coarse simulation, the benefits of the high-resolution simulation could be lost. The key observation here is that we do not require “exact” matching at the single grid cell level, but rather ensure that the gross (and specific smaller) properties of the simulation at user-defined scales, are maintained. The controls placed on the high-resolution simulation will, in general, have only a minor effect on the simulation, and will have little effect on the fine-scale detail present.

Note that in contrast to some earlier methods that deal only with simple non-reactive gases or liquids, our method is better suited to match multiple properties at once. As a case study that confirms this, we demonstrate our implementation on a reactive gas flame model.

In summary, the main aspects of our work are that we:

- present an automated method for obtaining guidance data from coarse interactive simulations,
- present a method for adjusting a complex simulation from our guidance data, and
- can handle complex, reactive systems with multiple densities.

This is all tied together in an efficient method that is easy to incorporate into existing systems. Together, these aspects give a designer of a complex fluid simulation a valuable tool to control the details of a simulation, reducing the time costs of repeated reruns and restarts.

2 Background

Fluid simulation has gained popularity in computer graphics in recent years, particularly due to the development of stable methods for simulating incompressible flow. Simulating fluid dynamics equations in computer graphics dates back to 1984 [14]. Later, Foster and Metaxas [9, 10] used explicit integration to simulate liquids and smoke. The Stable Fluids approach introduced by Stam [33] uses an implicit solver, which makes it unconditionally stable and allows simulations to use coarse grid resolutions. Later work has offered improvements on this simulation approach, particularly in adding a term for vorticity confinement [35, 5] to keep the fluid “alive” by injecting back the velocity lost due to the implicit integration scheme.

The topic of this paper is related to simulation control. We also provide a detailed example of our approach on a fluid-based fire simulator. We give an overview of prior work in these areas here.

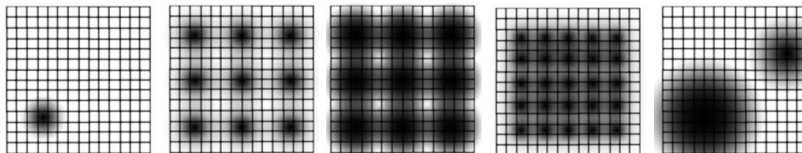


Figure 2. Data collection points could be placed in various ways. The examples show cases of changing the spacing of match points, and the size of the Gaussian filters.

2.1 Simulation Control

The goal of control in a physically-based system is to modify the “true” simulation in a way that is visually *plausible* [2]. Control of general rigid body simulations has been one of the more explored topics [3, 27, 26]. Control of fluid systems was first investigated by Foster and Metaxas [10] and later by Foster and Fedkiw [8]. Rasmussen et al. [28] proposed a production model using particles to control liquid simulations. Shi and Yu have proposed methods for matching smoke [31] and liquids [32] to changing target shapes. Pighin et al. [25] have combined Eulerian and Lagrangian representations to create a system for interactively manipulating fluid flows. Their method parameterizes a fluid simulation from advected particle paths, allowing fine-grained control of the system by manipulating these paths. Hong and Kim [12] uses a geometric potential field to generate forces to match target shapes. Schpok et al. [29] control the simulation by manipulation of automatically extracted simulation features. Recently, Angelidis et al. [1] and Kim et al. [15] provide path control for smoke simulations. Though not controlling simulation, the work of Treuille et al. [36] also allows a user to generate high resolution simulations interactively. However, this requires massive and time-consuming preprocessing of scenes similar to the one the user wants.

Closer to our approach, several algorithms have focused on the keyframe concept. Treuille et al. [37] proposed a fluid control system based on user-defined smoke density keyframes. The fluid system is controlled by parametric wind and vortex fields that are manipulated to make the simulation achieve the specified keyframes. The amount of adjustment to the vector field is minimized via a non-linear optimization. This optimization is expensive, and dimensionality grows with the length of the simulation. The approach was practical only for 2D simulation, but McNamara et al. [21] improved on it by using a discrete adjoint method for gradient calculation. They achieve much faster convergence for control parameters, enabling fine-grain keyframe control even on large 3D fluid animations.

Instead of optimization, Fattal and Lischinski [4] introduce new terms into the standard flow equations, offering a closed form solution for the control parameters. Matching target densities using only advection is usually not possible, and so they propose smoke gathering, diffusing the error field. This allows complex smoke animations to be controlled with little additional expense. Although our system was developed to match more general simulations, it resembles the keyframe matching in this work. We use different sampling, matching, and error computations. Also, while they use a global optimization to hit an arbitrary keyframe, we use local optimization to hit keyframes that are closely related to those computed without keyframing.

In these prior methods, control of fluids is usually achieved by controlling external forces f [37, 21, 4], and external sources S [21]. McNamara et al. [21] use sources and sinks only for the liquid simulation, similar to Foster and Metaxas [9], and Fattal and Lischinski [4] use smoke gathering instead of sources and sinks. Except for that approach, previous fluid control systems focus on single density fields only. When multiple densities are advected with the flow, controlling only the flow field can not guarantee simultaneous matching of all density distributions. Although there may be potential for previous methods [37, 21] to be extended in this way, there is no such published demonstration.

2.2 Fluid Based Simulation of Fire

As is described below, we have tested our method on a simulation of a reactive system, namely fire. Fire and explosion simulation has a long history in computer graphics, with a wide variety of methods

having been explored. Many approaches, including ours, are based on fluid simulation.

Nguyen et al. [24] define a flame front as a moving boundary between two fluids. A level set is used to capture very complex motion of the fire, however the gases themselves are not reactive. Feldman et al. [6] use incompressible fluid equations to model suspended particle explosions, and expansive flow by enabling $\nabla \cdot u > 0$. Ihm et al. [13] use this result to model expansion of chemical reactions. Melek and Keyser [22] couple a fluid-based reactive fire simulator with a levelset method to model burning and decomposition of volumetric solids. Losasso et al. [19] present a novel technique for simulating phase change of solid objects into fluids, such as melting volumetric solids and burning thin sheets. Also, Losasso et al. [20] extend the particle level set method [18] to as many regions as desired, and propose techniques for simulating interactions between them, such as surface tension forces, complex chemical reactions, or one or two materials converting into another one. They demonstrate liquids mixing, reacting, and releasing combustible gas that then burns in one single simulation framework.

Different techniques have been proposed to increase fine scale detail in fire simulation. Stam and Fiume [34] use a map to define the amount of fuel and temperature on every point on the object, and use turbulent wind fields and warped blobs. Neyret [23] propose adding fine detail using advected textures. Losasso et al. [18] use an octree based simulation structure for liquid and smoke simulation; such a method could presumably be used for fire. Selle et al. [30] incorporate vortex particles to reintroduce small scale detail for simulating smoke, fire and explosions, addressing the limitations of vorticity confinement [35, 5] on low resolution grids.

3 Preview Based Control

As described earlier, we begin with an initial (low resolution) simulation we call the *preview*. We will refer to the preview data at a time t , and a position \vec{x} as $v(\vec{x}, t)$. Note that v is actually a vector function, which may include various scalar and vector data from the simulation. Similarly, we have a second (high resolution) simulation that we refer to as the *final*, $\bar{v}(\vec{x}, t)$. Our goal is to produce an “adjusted” final version, $\bar{v}'(\vec{x}, t)$ that more closely matches the preview. Note that the final may differ from the preview in both spatial and temporal resolution.

Generally, we would like the ability to control how closely we want the final to match to the preview. At some regions/times we might wish to match exactly, at others we might not care about matching at all, and at others we might want something in between. We define a function, $m(\vec{x}, t) \in [0, 1]$ to describe how closely we wish to match.

The rough intuition, then, is that our matching process attempts to adjust the final such that

$$\bar{v}'(\vec{x}, t) = \bar{v}(\vec{x}, t) + m(\vec{x}, t)(v(\vec{x}, t) - \bar{v}(\vec{x}, t)) \tag{1}$$

We now describe the way we specify the matching function (section 3.1), how all the information is sampled (section 3.2), and how it is used to match our final to the preview (section 3.3).

3.1 Describing the Matching Function

The matching function m can be defined continuously over time and space. However, such a continuous representation tends to be impractical, first because the simulation data itself tends not to be continuous (but rather sampled at a grid), and second because defining this matching function continuously could use excessive amounts of memory.

Instead, we choose to specify the matching function as a collection of delta functions. That is, we choose a number of points, in time and space, at which we want to match the preview. For each of these points, we will have a single scalar value ($m_{i,j} \in (0, 1]$) that specifies the level of matching we desire, i.e. $m(\vec{x}_i, t_j)$.

There are several potential methods for choosing the positions (\vec{x}_i and t_j) at which to specify the matching function (see figure 2). The most straightforward method is to place the sample points on a regular grid across the entire simulation domain and evenly through time. However, there are other options for choosing these points: e.g. probabilistically in either a completely random fashion or weighted to capture some feature/simulation event, or interactively based on user-identified regions of high importance.

3.2 Sampling Simulation Data

In order to perform matching, we must be able to determine the data values for the simulation at the \vec{x}_i . Note that simulation data is provided over a grid. Analogous to sampling that occurs in computer vision contexts, we consider this simulation data as representing only the lowest available layer in scale space¹[17]. Any sample of simulation data, then, must be taken at a particular scale, r . Therefore, we can extend our match points to also take into account scale, i.e. $m_{i,j}^r$, allowing for collection (and match weighting) of multiple scales at a single spatial point.

While we could treat temporal data in a scale sense, there are significant problems that arise with such assumptions. Thus, we will assume that our sample is taken at a fixed point in time, t_j .

3.2.1 Scalar and Vector Data Sampling

For a scalar property, d , we obtain a sample at time t_j , at position \vec{x}_i , and at scale r as follows:

$$d_{i,j}^r = \frac{1}{\sum_x G_r} \sum_x G_r(\vec{x}_i, \vec{x}) d(\vec{x}, t_j) \tag{2}$$

G is the Gaussian weight function with radius r :

$$G_r(\vec{x}_i, \vec{x}) = e^{-r|\vec{x}_i - \vec{x}|^2} \tag{3}$$

That is, we use Gaussian weights to sample the data. Note that our samples will occur only at the i, j, r defined by the match points.

Vector data \vec{D} is sampled similarly, but we collect two pieces of information: direction \vec{D}_1 and angular momentum around the sample point with unit mass \vec{D}_2 , each stored as vectors (figure 3). Storing this angular momentum allows us to maintain angular motion in the vector field at a scale relative to the size of the filter kernel, and does not directly affect the finer-grained vortices simulated in the higher-resolution simulation.

¹Scale space can be thought of as adding an additional dimension, scale, to the data. The scale dimension is effectively the width of a blurring kernel that is applied to the data.

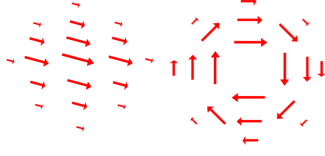


Figure 3. Two types of wind forces, directional and angular. The diagram at left has a high directional (but no angular) component, while the diagram at right has a high angular (but no directional) component.

$$\vec{D}_{i,j,1}^r = \frac{1}{\sum G_r} \sum_x G_r(\vec{x}_i, \vec{x}) \vec{D}(\vec{x}, t_j) \quad (4)$$

$$\vec{D}_{i,j,2}^r = \frac{1}{\sum G_r} \sum_x G_r(\vec{x}_i, \vec{x}) \vec{D}(\vec{x}, t_j) \times (\vec{x} - \vec{x}_i) \quad (5)$$

Position and scale is always recorded in terms of the physical space being simulated, not the grid-based space (thus, it will apply equally, regardless of simulation grid size - see figure 4). We can sample all of the key simulation data, including density fields (e.g. particle concentrations, heat, gas amounts) and vector fields.

We allow r to be chosen and set arbitrarily. However, note that the effectiveness of later matching will be governed by an interplay between r and the spacing between the \vec{x}_i . In practice, we typically use a uniform grid of \vec{x}_i , and in order to obtain a good coverage of the simulation domain, we use a single scale, r , such that nearby sample points overlap in influence (see figure 2).

3.2.2 Keyframe Storage

For a given time, t_j , we refer to the collection of match points and all sampled simulation data for those match points as a keyframe. This is a slightly different notion of “keyframe” from that of the traditional sense, but conveys a similar idea—namely that it is the information that we want to match. Each keyframe records the following information:

- *Time* (t_j) for the keyframe sample.
- *Weight* ($m_{i,j}^r$) for each of the match points.
- *Position and scale* (x_i, r) of each set of sampled variables.
- *Keyframed simulation variables* ($d_i^r, \vec{D}_{i,1}^r, \vec{D}_{i,2}^r$) such as density values, heat, and flow fields.

3.3 Matching the Final

In the matching process, we sample the final (unmatched) simulation at the same points specified in the keyframe. The difference between the final and the keyframe will be used to modify (match) the final. It is important to first understand the simulation structure.

3.3.1 Flow Equations

The inviscid Euler equations for incompressible flow u are

$$\nabla \cdot u = 0 \quad (6)$$

$$\frac{\partial u}{\partial t} = f - (u \cdot \nabla)u - \nabla p \quad (7)$$

and density d advected by the flow is defined as

$$\frac{\partial d}{\partial t} = S - (u \cdot \nabla)d + k_d \nabla^2 d \quad (8)$$

Here, f accounts for external forces and S defines a density source (or sink). Instead of changing the governing equations or adding additional controls, we opt to control the equations using external forces and density sources at the data collection locations. That is, we will define f and S in order to obtain a match. Our use of wind forces and density sources prevent physical inaccuracies, such as negative pressure. Just as importantly, the basic simulation computation is unchanged from an ‘‘uncontrolled’’ simulation. This makes our approach very easy to integrate into an existing simulation framework.

3.3.2 Matching

The matching is done in two passes. In the first pass, the simulation is sampled at the x_i and scale(s) r . For scalar data we obtain the (weighted) current value \bar{d}_i^r :

$$\bar{d}_i^r = \frac{1}{\sum G_r} \sum_x G_r(\vec{x}_i, x) d(\vec{x}, t) \quad (9)$$

To provide matching, all we need to do is to specify the source S for that scalar variable:

$$S(\vec{x}, t) = \max_{i,r} (W(t_j, t) G'_r(\vec{x}_i, \vec{x}) m_{i,j}^r (d_i^r - \bar{d}_i^r)) \quad (10)$$

The \max function should be taken to mean the maximum absolute value (maintaining the sign); the reasoning is explained below. Note that we use two functions to spread data, a spatial one, G'_r , and a temporal one, W . Both of these are discussed below (sections 3.3.3 and 3.3.4 respectively).

Similar to sources controlling densities, we use external forces (f from Eq. 7) to control the flow. $\bar{D}_{i,1}^r$ and $\bar{D}_{i,2}^r$ are computed in the obvious way (see equations 4, 5, 9)

$$f(\vec{x}, t) = \max_{i,r} (W(t_j, t) G'_r(\vec{x}_i, \vec{x}) m_{i,j}^r (f_l + f_v)) \quad (11)$$

$$f_l(\vec{x}, t) = \bar{D}_{i,1}^r - \bar{D}_{i,1}^r \quad (12)$$

$$f_v(\vec{x}, t) = (\bar{D}_{i,2}^r - \bar{D}_{i,2}^r) \times (\vec{x}_i - \vec{x}) \quad (13)$$

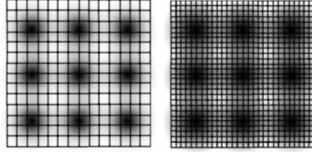


Figure 4. Sampling positions and scales are independent of the grid size resolution.

Here the \max function should be taken to mean the vector with maximum magnitude.

The \max functions that appear in equations 10 and 11 are chosen to prevent overcompensation during matching. Notice that any point can fall within the influence region (i.e. within the G'_r radius) of several match points. These separate controls may give differing or even conflicting information. For example, one match point might want to increase density, and another to decrease it. Or, two points might both want to increase the density. If we were to sum these contributions, we may add far too much density. An averaging process is feasible, but would likely overweight less important additions (e.g. a point near the center of one match point could get minimized by points from the fringes of other match points). The “ideal” would be to formulate this as a more complicated optimization problem, but this would require several iterations with no guarantee of convergence. Instead, we compromise with a method that provides convincing results, with far less work, by simply taking the maximum (magnitude) contribution from any of the sources. In this way, we are guaranteed to maintain the most important changes, while reducing chances of overcompensation.

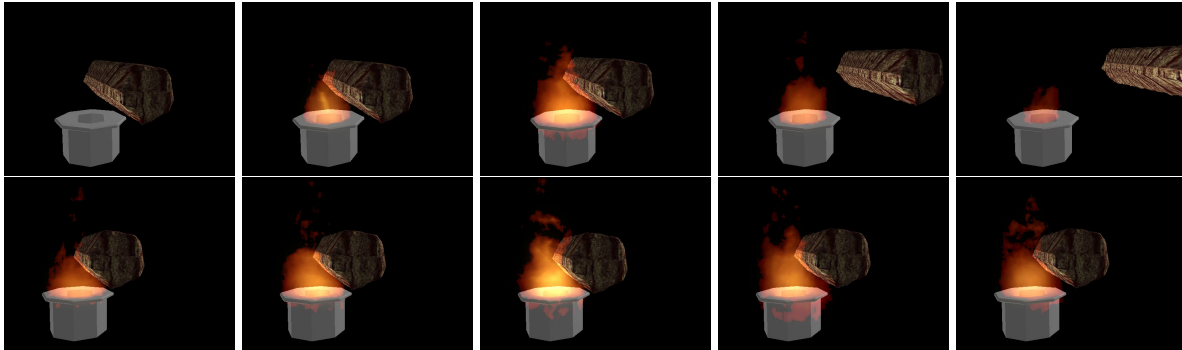


Figure 5. Simulations produced strictly from the matching process. Fire data is sampled in the preview. The burner is turned off in the final simulation, and all fire is created from the matching function. The top row shows the final with “burst” control, and the bottom row shows the final with “continuous” control.

3.3.3 Spatial blending

Spatial blending function, G'_r , spreads a factor of the difference value over a region. We use the same Gaussian weighting function as for sampling. The multiplicative factor is one over the sum of the squares of the Gaussian weights. The intuition behind this choice is described here.

Assume we have a difference Q between the sampled density of the preview and the final at some point. In order to make the final match the preview, the sampled final value must therefore increase by

Q . That is, we need to have

$$Q = \sum_{\vec{x}} G_r(\vec{x}_i, \vec{x}) Q'_{\vec{x}} \quad (14)$$

where $Q'_{\vec{x}}$ is the increase in density at point \vec{x} . One way to do this is to increase the density value of each point contributing to the sample by Q —i.e. $Q'_{\vec{x}} = Q$. However, if the sample point is of a large scale (high r), this can involve a very large number of points. Adding Q to all of them introduces a significant amount of overall density to the scene (roughly Qr^2), and also affects points far away from the sample as much as it affects the sample itself. This is undesirable.

We would prefer to instead distribute any additional density closer to the sample point (where it will have more effect), but still want to spread the additional density smoothly, to avoid artifacts. An obvious choice (but not the only one) is to use a Gaussian spread function, of radius equal to the scale we aim for. That is, we have $Q'_{\vec{x}} = G_r(\vec{x}_i, \vec{x}) Q''$. We now must find Q'' , the amount to apply via a Gaussian spread to result in the correct overall change, Q . Substituting into equation 14, we have:

$$Q = \sum_{\vec{x}} (G_r(\vec{x}_i, \vec{x}))^2 Q'' \quad (15)$$

Solving for Q'' gives us the correct weighting to use, yielding the final spatial blending function:

$$G'_r(\vec{x}_i, \vec{x}) = \frac{G_r(\vec{x}_i, \vec{x})}{\sum_{\vec{x}} (G_r(\vec{x}_i, \vec{x}))^2} \quad (16)$$

3.3.4 Temporal blending

Since keyframes are defined at only a point in time, if we perform matching only at the keyframe times, we would have a sudden introduction of density/force into the system. This would be extremely unrealistic and jarring, and therefore we choose to gradually match the simulation state incrementally, starting just before the keyframe time. This incremental matching enables us to use “dumb” sources/sinks, rather than running an optimization procedure to obtain an exact control solution, as is done in most keyframe matching approaches. As a result, we introduce a temporal spread function, W , to gradually match keyframe data.

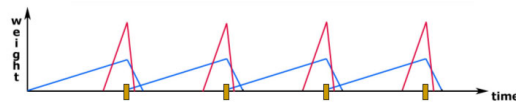


Figure 6. Two possible weight functions. The blue curve provides continuous control, while the red curve provides “burst” control. The yellow markers are keyframed points during preview.

W is defined as a function (e.g. piecewise-linear or Gaussian) that gradually increases over a number of timesteps (t_s), peaks at the actual keyframe time t_i , and falls off sharply for some time afterwards (t_e). For example, a linear temporal blending function would have the form:

$$W(t_i, t) = \begin{cases} |t - (t_i - t_s)|, & (t_i - t_s) < t < t_i \\ |(t_i + t_e) - t|, & t_i \leq t < (t_i + t_e) \\ 0, & \text{o.w.} \end{cases} \quad (17)$$

By changing t_s and t_e we can vary from “burst” like keyframe control to “continuous” control (see figure 6). A burst control minimizes control interference, but creates sharp potentially visible changes in the simulation state. Still, it is applicable to a reactive simulation, such as fire simulation.

3.4 Discussion

A key point to note about our approach, particularly in comparison to prior keyframing approaches, is that the keyframes arise from fundamentally the same simulation. Since the preview and final follow the same physically-based rules, and are matched in the previous keyframes, the two simulations should never diverge by much. Thus, the forces/densities required to match our keyframes should generally be very minor. So, while we do not guarantee mass conservation, our simple sourcing scheme should not create noticeable problems.

3.4.1 Simulation Behavior

It is possible for our additive/subtractive scheme to create negative densities (i.e. removing density when there was none to begin with). This requires correction after adding/subtracting all sources affecting a point, but is a local operation. This is only rarely observed in practice.

One could argue that our additive/subtractive scheme is blurring the fine-scale detail we are trying to generate. Additional sourced density does tend to blur the simulation results, in that it is added evenly over a region, thus reducing contrast in (though not eliminating) fine-scale detail. However, note that stronger fitting (i.e. more sourced density) only occurs when the simulations diverge significantly, which is neither typical nor expected. Also, nothing about our method prevents the use of other common methods for “faking” fine-scale detail, such as warped blobs, or advected textures. Furthermore, our spatial spread function used during matching could potentially be modified to maintain relative contrast.

An additional argument against our method could be that we use a Stable Fluids approach. This favors stability over accuracy, and coarse resolution simulations suffer from artificial viscosity and dissipation. These problems could be captured by our keyframes and thus propagated to the high resolution simulation, losing the benefit of using high-resolution at all. However, we would argue that such effects are minor and might even be desirable if they are necessary to make the final behave as one planned.

3.4.2 Comparison

Our keyframe matching draws easy comparisons with prior methods for keyframing fluid simulations. For simple simulations (e.g. smoke only), one can imagine ways to incorporate our keyframe samples as input to other keyframing approaches, then use those approaches to perform the match. Indeed, those keyframing approaches have the potential of creating more plausible motion, in that they use more sophisticated keyframe matching techniques. However, there are two main points that we feel make our approach superior for our range of applications.

First, and perhaps most importantly, the prior keyframing methods can make very few assumptions regarding the status of the system from which they hope to hit the keyframe. Since we can assume the keyframes come from a similar simulation, we still achieve very good results with significantly simpler implementation and computation.

Second, our system is better suited than prior approaches for fluid simulation systems with multiple densities. Other than Fattal and Lischinski’s method [4], earlier methods achieved most of their success by modifying the vector field; this is not a feasible approach for several applications, particularly with multiple density fields. Also, our method works fine for reactive systems, while prior approaches have to enforce a conservation of mass, or allow only limited sourcing/sinking.

On the other hand, systems requiring mass conservation would perform more poorly in our system. We can easily incorporate global mass conservation in our approach, but usually it is local mass conservation that is desired. Increasing the frequency of keyframes will minimize the visual distraction, yet adding too much control in this manner will tend to blur the simulation *temporally* and defeat some of the purpose of using a high resolution simulation.



Figure 7. During the coarse resolution interactive simulation (top row), the user ignited a match and burned the conference logo. During higher resolution simulation (second row), the match barely lit, hence the logo did not burn at all. The controlled simulation (last row) injected additional combustible gas and heat into the system, and thus burned in a manner similar to that in the preview simulation. Note that the simulation control matches the fluid domain only, and hence does not have direct control on how the logo burns and decomposes.

4 Case Study: Fire Simulation

As mentioned previously, part of our motivation is in the control of complex simulations. One such complex fluid based simulation is fire. We have implemented such a simulation by integrating a fluid-

based flame simulator, combustion model, heat transfer model, solid pyrolysis model, solid decomposition model, and rigid body simulator into a common framework. Using coarse resolution simulation modules we were able to achieve somewhat interactive speeds (7 fps for 30x30x30 resolution on a 1.8 GHz P4).

To briefly summarize our simulation framework, we assume there is only one moving gas (air), which is inviscid, incompressible, and constant density, and its motion is solved using Stable Fluids [33, 5]. The fluid motion (air) solution is applied to advect three scalar quantities: fuel gas, exhaust gas (including smoke), and heat. The portion of each cell not occupied by fuel gas or exhaust gas is assumed to be oxidizer. Some cells (such as those containing objects to be burned) are marked as filled. We model the combustion process by combining fuel and oxidizer in a cell, creating additional exhaust gas in its place. If the heat in the cell is beyond the combustion threshold, we decrease the level of fuel gas, increase the level of exhaust gas, and introduce additional heat into the cell.

4.1 Implementation and Results

Using our fire simulation framework, we record keyframes from a coarse simulation that runs at interactive rates. We place data points in a regular grid across the entire simulation environment, recording user interactions and taking keyframe data. The amounts of fuel gas, smoke, and heat are keyframed as density values, and air motion is keyframed as a vector field.

We achieved successful matching of the final simulation to the preview under a variety of example cases. This included cases where the behavior of the unmatched system was very different from that of the preview (and matched final).

Due to the number of competing variables, particularly within the simulation model itself, comprehensive performance analysis is not possible. However, based on our testing, we are able to make several observations regarding performance:

- The additional overhead added in the preview simulation due to keyframe generation was less than 5% of the simulation time in our experiments.
- The additional overhead for keyframe fitting during the final simulation was 10-30 % of the simulation time in our tests. Generally, for a fixed set of data collection points in a keyframe, as the resolution of the final simulation increases, the additional cost of matching decreases. This is to be expected, in that the keyframe matching time grows relatively slowly compared to the additional simulation cost on a larger grid.
- Using too sparse of a data collection point resolution tended to give poor matching results. This will vary, of course, based on the absolute resolution of the preview, and the relative increase in resolution for the final. In effect, the control is diffused over too wide a region.
- Finally, relatively lower resolution preview grids tended to yield larger matching overhead during the final simulation. This is due to the fact that as the resolutions become much more disparate, the simulations become less and less similar.

Figure 7 shows the user igniting a match in the preview, and then proceeding to burn a SCA logo. A 30x30x30 interactive simulation is used for the preview. An uncontrolled simulation running at

60x60x60 resolution causes the match to burn in a slightly different fashion. At first, the match does not burn strongly enough to ignite the logo, and although later the match bursts into flame it is not enough to ignite the logo. The controlled simulation uses half second separated keyframes using 10x10x10 uniformly placed data collection points. The controlled simulation more closely matches the preview, creating additional heat sufficient to ignite the logo.

Figure 5 shows a stress test on the method. In this scene, we ignite a burner, and the user puts a log into the flame, and the log starts to burn. For the stress test, we turn off the burner. Without the burner, the log would not burn. If we enable matching and use burst control with only a single data collection point placed at the burner, one can observe the control process in action as bursts of fireballs occur at keyframe intervals. If we use a temporally wider control, the burner appears to be burning.

5 Parameters and Control

We briefly summarize here the key parameters that govern the simulation matching behavior; many are referenced in the previous sections.

- *Scale and Density of Match Points.* The resolution of both the final simulation and the preview simulation, as well as the size of the simulation features we want to match, determines the scale and number of the match points we should use for sampling.
- *Temporal Resolution.* Controls on the temporal resolution of both the keyframe capture (t_j) and the temporal spread function (W) used during matching can have a significant effect. If the t_j are too far apart, the simulation may diverge to a point that control becomes obvious. The temporal spread function is important to adjust the speed at which matching occurs, and thus affects how obvious the control appears.
- *Threshold.* Not discussed here, but useful for a practical implementation, are thresholds that set a level for deviation from the keyframe, below which no matching will occur. This reduces unnecessary controls for nearly-matching simulation, and can reduce computation time.

6 Conclusion

We have presented a simple method to control fluid-based simulations from guidance data automatically generated during a coarse resolution preview. When interaction tools are provided during the preview simulation, the user can choreograph the simulation with interactive visual feedback. The user interactions are recorded, together with the simulation behavior. Then, the high-resolution simulation is fit to guarantee similar broad behavior to that seen/choreographed in the lower-resolution system. We have demonstrated the technique on a complex reactive system simulating fire. We do not try to exactly reproduce flame shape, but rather match the density and vector data of the simulation at the times, scales, and positions specified by the user (or automatically generated). In the case of fire, we match heat, combustible gas distribution, and major hot air motion.

A major benefit of our method is that no change in the broad simulation structure is required, and the computations, including Gaussian sampling and introducing new sources and sinks, are easy to implement. Our method is much simpler than prior keyframing approaches, and yet works very well in

environments where we can assume only minor deviations will be required. Also, no implicit knowledge is required about the simulation; it could be applied to a variety of fluid simulations, and any number of important variables could be matched. In the longer term, it would be ideal to have a mathematical model to compensate for the different behavior of the simulations using the characteristics of the underlying fluid equations. Until such intelligent methods are proposed, the simulation matching approach presented in this paper presents an easy-to-implement, efficient, and effective means of control.

There are several avenues still open for future work:

- The most relevant directions are probably exploring automatic non-uniform placement of data collection points (so as to capture key features of a simulation automatically), and extending the model to also work with non-fluids, such as burning solids.
- An extension for mass conservation, locally and globally, is desirable for *some* types of simulations.
- Additional improvements may also be obtained by adjusting the matching function to use first-order (or even higher) approximations to the fluid field, adjusting the position and amount of the sourced density/force to account for where it is predicted to be in future timesteps.
- Our current method of preventing overcompensation during matching seems to work well, but is not ideal. An alternative method that does not require the lengthy computation of a full optimization but provides better guarantees would be useful.

References

- [1] A. Angelidis, F. Neyret, K. Singh, and D. Nowrouzezahrai. A controllable, fast and stable basis for vortex based smoke simulation. pages 25–32, 2006.
- [2] R. Barzel, J. F. Hughes, and D. Wood. Plausible motion simulation for computer graphics animation. *Proceedings of the Eurographics Workshop on Computer Animation and Simulation*, pages 183–197.
- [3] S. Chenney and D. A. Forsyth. Sampling plausible solutions to multi-body constraint problems. *Computer Graphics (Siggraph 2000)*, pages 219–228, 2000.
- [4] R. Fattal and D. Lischinski. Target-driven smoke animation. *ACM Trans. Graph.*, 23(3):441–448, 2004.
- [5] R. Fedkiw, J. Stam, and H. W. Jensen. Visual simulation of smoke. *Proc. of ACM SIGGRAPH '01*, pages 15–22, 2001.
- [6] B. E. Feldman, J. F. O'Brien, and O. Arikan. Animating suspended particle explosions. In *Proc. of ACM SIGGRAPH '03*, pages 708–715, 2003.
- [7] B. E. Feldman, J. F. O'Brien, and B. M. Klingner. Animating gases with hybrid meshes. *ACM Trans. Graph.*, 24(3):904–909, July 2005.
- [8] N. Foster and R. Fedkiw. Practical animation of liquids. *Proceedings of ACM SIGGRAPH '01*, pages 23–30, 2001.
- [9] N. Foster and D. Metaxas. Realistic animation of liquids. *Graphical Models and Image Processing*, (58(5)):471–483, 1996.
- [10] N. Foster and D. Metaxas. Controlling fluid animation. *Computer Graphics International*, pages 178–188, 1997.
- [11] M. J. Harris, G. Coombe, T. Scheuermann, and A. Lastra. Physically-based visual simulation on graphics hardware. In *HWWS '02: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 109–118, 2002.

- [12] J.-M. Hong and C.-H. Kim. Controlling fluid animation with geometric potential: Research articles. *Comput. Animat. Virtual Worlds*, 15(3-4):147–157, 2004.
- [13] I. Ihm, B. Kang, and D. Cha. Animation of reactive gaseous fluids through chemical kinetics. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 203–212, 2004.
- [14] J. T. Kajiya and B. P. V. Herzen. Ray tracing volume densities. In *Computer Graphics (Proceedings of SIGGRAPH 89)*, (18(3)):165–174, 1984.
- [15] Y. Kim, R. Machiraju, and D. Thompson. Path-based control of smoke simulations. pages 33–42, 2006.
- [16] B. M. Klingner, B. E. Feldman, N. Chentanez, and J. F. O’Brien. Fluid animation with dynamic meshes. *ACM Trans. Graph.*, 25(3):820–825, July 2006.
- [17] T. Lindeberg. *Scale-Space Theory in Computer Vision*. Kluwer Academic Publishers, 1994.
- [18] F. Losasso, F. Gibou, and R. Fedkiw. Simulating water and smoke with an octree data structure. *ACM Trans. Graph.*, 23(3):457–462, 2004.
- [19] F. Losasso, G. Irving, E. Guendelman, and R. Fedkiw. Melting and burning solids into liquids and gases. *IEEE Transactions on Visualization and Computer Graphics*, 12(3):343–352, 2006.
- [20] F. Losasso, T. Shinar, A. Selle, and R. Fedkiw. Multiple interacting liquids. *ACM Trans. Graph.*, 25(3):812–819, 2006.
- [21] A. McNamara, A. Treuille, Z. Popović, and J. Stam. Fluid control using the adjoint method. *ACM Transactions on Graphics*, 23(3):449–456, 2004.
- [22] Z. Melek and J. Keyser. Multi-representation interaction for physically based modeling. In *Proc. ACM Symposium on Solid and Physical Modeling*, pages 187–196, 299, 2005.
- [23] F. Neyret. Advected textures. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 147–153, July 2003.
- [24] D. Nguyen, R. Fedkiw, and H. W. Jensen. Physically based modeling and animation of fire. *Proc. of ACM SIGGRAPH ’02*, pages 721–728, 2002.
- [25] F. Pighin, J. M. Cohen, and M. Shah. Modeling and editing flows using advected radial basis functions. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 223–232, 2004.
- [26] J. Popović. Interactive design of rigid body simulations for computer animation. *PhD Thesis*, 2001.
- [27] J. Popović, S. M. Seitz, M. Erdmann, Z. Popović, and A. Witkin. Interactive manipulation of rigid body simulations. *Computer Graphics (SIGGRAPH 2000)*, pages 209–218, 2000.
- [28] 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, 2004.
- [29] J. Schpok, W. Dwyer, and D. S. Ebert. Modeling and animating gases with simulation features. pages 97–105, 2005.
- [30] A. Selle, N. Rasmussen, and R. Fedkiw. A vortex particle method for smoke, water and explosions. *ACM Trans. Graph.*, 24(3):910–914, 2005.
- [31] L. Shi and Y. Yu. Controllable smoke animation with guiding objects. *ACM Trans. Graph.*, 24(1):140–164, 2005.
- [32] L. Shi and Y. Yu. Taming liquids for rapidly changing targets. In *SCA ’05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 229–236, 2005.
- [33] J. Stam. Stable fluids. *Proc. of ACM SIGGRAPH ’99*, pages 121–128, 1999.
- [34] J. Stam and E. Fiume. Depicting fire and other gaseous phenomena using diffusion process. *Proc. of ACM SIGGRAPH ’95*, pages 129–136, 1995.
- [35] J. Steinhoff and D. Underhill. Modification of the euler equations for vorticity confinement: Application to the computation of interacting vortex rings. *Physics of Fluids*, (6):2738–2744, 1994.
- [36] A. Treuille, A. Lewis, and Z. Popovic. Model reduction for real-time fluids. *ACM Trans. Graph.*, 25(3):826–834, 2006.
- [37] A. Treuille, A. McNamara, Z. Popović, and J. Stam. Keyframe control of smoke simulation. *ACM Transactions on Graphics*, 22(3):716–723, 2003.