

Preview-based Sampling for Controlling Gaseous Simulations

Ruoguan Huang^{†1}, Zeki Melek^{‡2}, John Keyser^{§1}

¹Texas A&M University

²PDI/Dreamworks

Abstract

In this work, we describe an automated method for directing the control of a high resolution gaseous fluid simulation based on the results of a lower resolution preview simulation. Small variations in accuracy between low and high resolution grids can lead to divergent simulations, which is problematic for those wanting to achieve a desired behavior. Our goal is to provide a simple method for ensuring that the high resolution simulation matches key properties from the lower resolution simulation. We first let a user specify a fast, coarse simulation that will be used for guidance. Our automated method samples the data to be matched at various positions and scales in the simulation, or allows the user to identify key portions of the simulation to maintain. During the high resolution simulation, a matching process ensures that the properties sampled from the low resolution simulation are maintained. This matching process keeps the different resolution simulations aligned even for complex systems, and can ensure consistency of not only the velocity field, but also advected scalar values. Because the final simulation is naturally similar to the preview simulation, only minor controlling adjustments are needed, allowing a simpler control method than that used in prior keyframing approaches.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—Animation; Simulation and Modeling [I.6.8]: Types of Simulation—Animation

1. Introduction

As fluid simulation technology has improved, the incorporation of fluid-based visual effects has steadily increased. However, despite hardware and algorithmic improvements, these simulations of smoke, clouds, fire, etc. can still take a long time to compute.

The length of time taken for simulations can be problematic in production situations. Initial conditions are set up, and then the simulation is run. Only after hours of running might it become apparent that the simulation itself is not producing the desired results.

A reasonable approach to address this problem is to first run a lower-resolution simulation (which will be much faster), and once an acceptable simulation has been found, then run a higher resolution simulation. However, changes in resolution

can result in different simulation behavior, with seemingly small local variations resulting in significant changes at a larger scale. Several aspects of the typical grid-based fluid simulation methods, including artificial viscosity and effects in the pressure projection stage, will vary based on the grid cell size. Further, the changes in resolution around object boundaries can cause significant differences in simulations of differing resolution.

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 fast, possibly interactive, simulation to produce a plausible but rough simulation. With the coarse simulation, a user can fine-tune and choreograph initial conditions and parameters quickly while getting visual feedback. We use a sampling approach to capture gross properties of this preview simulation that we are most interested in maintaining. Our matching process makes the final simulation more closely match these samples, and thus maintains the properties of interest present in the low resolution simulation.

[†] e-mail:huangruoguan@gmail.com

[‡] e-mail:zeki@melekzek.com

[§] e-mail:keyser@cs.tamu.edu

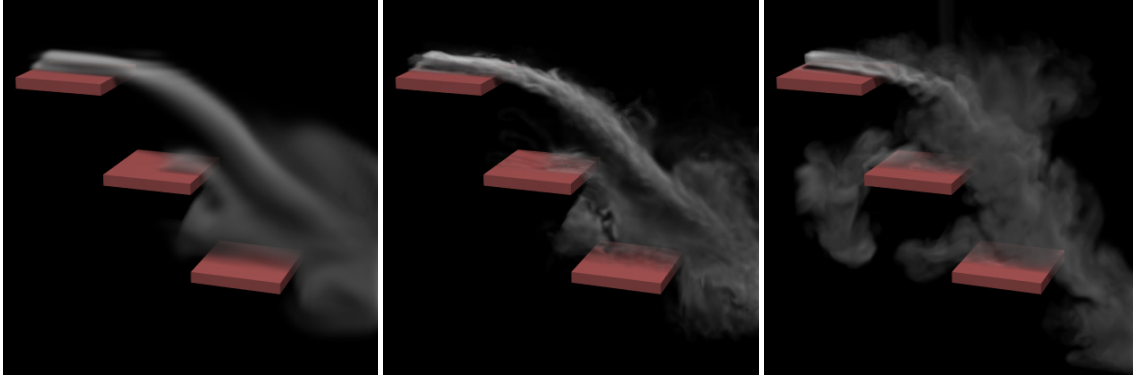


Figure 1: Example of our controlled simulation. Smoke is flowing in from left to right and falling down along the stairs. At left, a low resolution simulation provides guidance. The high resolution simulation, at right, exhibits different behavior. Our controlled high resolution simulation, middle, exhibits the detailed behavior of a high resolution simulation, while matching the gross behavior of the low resolution input.

An 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. Details of the higher-resolution simulation are maintained, and typical changes at any one frame are minor.

Among the key features of our method are the following:

- Our approach is general, and can be used to control any properties stored in the grid, including both the velocity field itself as well as densities, temperatures, or other simulation properties.
- Users can easily choose where and with what resolution to sample the coarse simulation. Sampling locations can be selected interactively and it is trivial to adjust the sampling to obtain varying resolutions. Sampling can be performed separately for different simulation properties.
- Control of the high resolution simulation is achieved through an adjustment procedure that can be easily inserted into an existing simulation framework.

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. Prior Work

The method proposed in this paper assumes a grid-based (Eulerian) fluid simulation framework. Interest in fluid dynamics simulations in computer graphics increased significantly with the work of Foster and Metaxas [FM96, FM97] and the subsequent Stable Fluids approach introduced by

Stam [Sta99]. Stam’s method used an implicit solver, making the simulation unconditionally stable, and thus allowing coarse grid resolutions and large time steps. There has been a tremendous amount of follow-on work which has presented improvements in various aspects of the simulation, and a thorough discussion of this work is beyond the scope of this paper. While particle-based (Lagrangian) simulations have also become popular [MCG03], Eulerian simulations continue to be widely used, particularly for gaseous and incompressible situations.

The topic of this paper is related to simulation control. The goal of control in a physically-based system is to modify the “true” simulation in a way that is visually *plausible* [BHW96]. Control has been explored in most areas of simulation, including rigid body simulations [CF00, PSE*00, Pop01], and thin shell simulations [BMWG07].

Control of fluid systems was first investigated by Foster and Metaxas [FM97] and later by Foster and Fedkiw [FF01], and has been addressed by a number of other researchers. Rasmussen et al. [REN*04] proposed a production model using particles to control liquid simulations. Particle based control of simulations was also used by Thürey et al. [TKPR06], who would extract the particles from coarser simulators. Shi and Yu have proposed methods for matching smoke [SY05a] and liquids [SY05b] to changing target shapes. Pighin et al. [PCS04] 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 [HK04] use a geometric potential field to generate forces to match target shapes. Schpok et al. [SDE05] control the simulation by manipulation of automatically extracted simulation features. Angelidis et al. [ANSN06] and Kim et al. [KMT06] provide path control for smoke simulations.

A few recent methods approach the problem described from a different perspective, namely that high frequency “detail” is added into a coarse simulation, thus giving the appearance of a higher-resolution simulation. Kim et al. [KTJG08] use a wavelet-based method to generate fine turbulent details within a fluid simulation. Schechter and Bridson [SB08] similarly upsample a lower resolution simulation and add turbulent detail. Narain et al. [NSCL08] also use a procedural method to add in turbulence, but in addition they identify locations for additional turbulence based on the energy computation in the fluid simulation. Finally, Pfaff et al. [PTC*10] add the procedural turbulence by generating and advecting turbulence particles based on the simulation.

Closer to our approach, several algorithms have focused on the keyframe concept. Treuille et al. [TMPS03] 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. [MTPS04] improved on it by using a discrete adjoint method for gradient calculation. They achieve much faster convergence for control parameters, thus enabling fine-grain keyframe control on larger 3D fluid animations.

Instead of optimization, Fattal and Lischinski [FL04] 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 similar to non-keyframed results.

In these prior methods, control of fluids is usually achieved by controlling external forces [TMPS03, MTPS04, FL04], and external sources [MTPS04]. McNamara et al. [MTPS04] use sources and sinks only for the liquid simulation, similar to Foster and Metaxas [FM96], and Fattal and Lischinski [FL04] 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 densities.

Finally, our work is probably most similar to the recent work of Nielsen et al. [NCZ*09, NC10]. Their work, which shares the same motivation as ours, creates a direct coupling between a low resolution preview simulation and a higher res-

olution final simulation. The high resolution simulation is created by solving for velocity values that, in effect, blend between an upsampled low-resolution simulation, and the high resolution simulation. In their earlier paper, Nielsen et al. [NCZ*09] introduce the method, and present the multi-grid framework used to solve it. In their later paper, Nielsen and Christensen [NC10] extend the earlier work to support more efficient time-varying computation, and use density to help determine the level of control to provide. Though density is used to help determine control, the actual adjustments are made only to the vector field. A detailed comparison of our method with Nielsen et al.’s work is provided below in section 5.2.

Some ideas presented here have been described previously in [Mel07, MK07].

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, $\bar{v}(\vec{x}, t)$. Our goal is to produce an “adjusted” final version, $\bar{v}'(\vec{x}, t)$ that more closely matches the preview. Note that the final version 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 matching 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 high resolution simulation such that

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

A brief overview of our approach, explained in detail below (see also Figure 2, where steps are labeled) is as follows:

1. We begin with a low resolution preview simulation
2. We sample the properties of the preview using various Gaussian kernels of fixed radius, the match points. The locations and radii of the match points are chosen to conform to certain criteria.
3. We run the simulation in high resolution. At each time step, after adding external forces/densities, we apply matching:
 - a. We sample the properties of the high resolution simulation at any match points.
 - b. We compute the (weighted) error vs. preview samples.

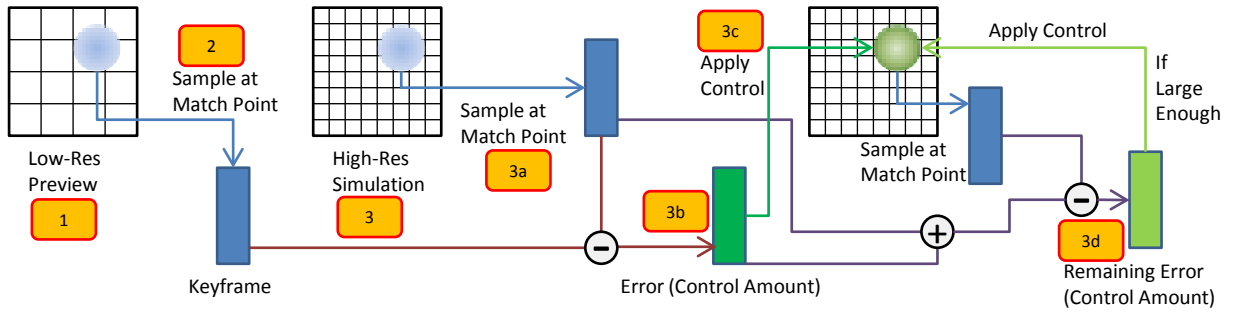


Figure 2: An overview of our simulation control procedure. Orange box numbers refer to steps described in section 3.

- c. We modify the simulation by applying a control, spread spatially at the match points, to compensate for error.
- d. We compute the change in the error vs. the preview samples. We repeat step 3.c. until this error is near 0.

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

3.1. Match Points

The matching function m could be defined continuously over time and space. However, such a continuous representation becomes problematic or redundant to specify independently of the simulation data that we are interested in. Instead, we have found it more useful in practice to associate the matching function directly with the samples that we use. Thus, we will define a set of sample points, each of which will have an associated matching value that can vary in space and time. We call these sample points (assuming the matching value is > 0) match points.

Each match point will be at a fixed position, 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 space [†] [Lin94]. When we sample simulation data, then, we do so at a particular spatial scale, r . For practical implementation, we assume that each match point has finite support, i.e. only data within a certain radius (proportional to r) around each match point will contribute to the sampling for that point. While we could also treat temporal data in a scale sense, there are significant problems that

[†] Scale 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 convolved with the data.

arise with such assumptions. Thus, we will assume that our sample is taken at a fixed point in time.

Each match point is therefore uniquely identified by an (i, r) value. When referring to match points, we will use a superscript r , and subscript i to refer to a sample taken at x_i with scale r . So, our matching function will be described by specifying $m_i^r \in [0, 1]$. Note that in practice, it is often convenient to use either $m_i^r = 0$ (i.e. ignore this sample) or $m_i^r = 1$ (i.e. match this sample exactly).

The particular x_i and r for the match points can be chosen in many ways, including interactive user specification, random distributions, or regular spacing in space and time. Figure 3 provides a conceptual demonstration of the various ways that samples might be placed. A more thorough discussion of ways to place these samples is given in Section 4.

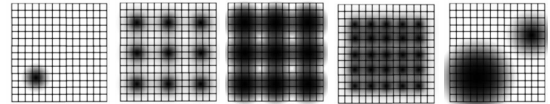


Figure 3: Match points can be placed in various ways. The examples show cases of changing the spacing of match points, and the size of the Gaussian filters.

3.2. Sampling Simulation Data

As mentioned above, each match point is at a fixed position, time, and scale. To sample in scale space, simulation data must be convolved with a Gaussian kernel of the given scale. We note that although the Gaussian filter is appropriate for capturing simulation data in the scale space sense, other filtering kernels might prove more beneficial for capturing various effects. So, although we use Gaussians in our discussion and examples, the details below can be easily adjusted should an alternative kernel be desired (e.g. a box filter might be used to capture a hard edge, or a uniform weighting to enforce total conservation properties).

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). We typically upsample low resolution data before sampling, so that the filter convolution is more accurate. We can sample all of the key simulation data, including density fields (e.g. smoke particle concentrations, heat, chemical concentrations) and vector fields.

3.2.1. Scalar Data Sampling

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

$$d_i^r = \frac{1}{\sum G_r} \sum_x G_r(\vec{x}_i, \vec{x}) d(\vec{x}) \quad (2)$$

G is the Gaussian weight function with radius r :

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

3.2.2. Vector Data Sampling

Vector data \vec{D} is sampled similarly to scalar data, 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. 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 higher-resolution.

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

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

3.2.3. Keyframe Storage

For a given time, t , 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 the traditional sense, but conveys a similar idea—namely that it is the information that we want to match at a particular time. Each keyframe records the following information:

- *Weight* (m_i^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.

We might wish to sample only the directional component, $\vec{D}_{i,1}^r$, and ignore the curl component, $\vec{D}_{i,2}^r$ (see Section 4).

3.3. Matching in High Resolution

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.

3.3.1. Flow Equations

The inviscid 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

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

These equations are typically solved by operator splitting. Notice that f accounts for external forces and S defines a density source (or sink). The controls that we apply will have the same effect as introducing additional f and S to the system, though the process for control is different. In typical implementations, such external forces/sources are handled in an individual step. Our matching process thus acts as an additional stage, extending the external force step of a typical solver. This makes our approach very easy to integrate into an existing simulation framework, and the basic simulation computation is unchanged from an “uncontrolled” simulation, with no modification needed within individual stages such as convection or pressure projection.

3.3.2. Determining Amount of Control

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

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

and likewise determine $\bar{\vec{D}}_{i,1}^r$ and $\bar{\vec{D}}_{i,2}^r$ in the obvious way (see equations 4, 5, and 9).

To provide matching, we need to determine weighted errors:

$$\delta_i^r = m_i^r (d_i^r - \bar{d}_i^r) \quad (10)$$

$$\Delta_{i,1}^r = m_i^r (\vec{D}_{i,1}^r - \bar{\vec{D}}_{i,1}^r) \quad (11)$$

$$\Delta_{i,2}^r = m_i^r (\vec{D}_{i,2}^r - \bar{\vec{D}}_{i,2}^r) \quad (12)$$

This error represents the amount of control that needs to be applied to the high resolution simulation at the match point.

The idea of matching is that we want to modify the high resolution simulation by the control amounts. To perform matching, we will basically spread the control amount out over the region around the match points.

3.3.3. Spreading Control

In order to apply the control amount computed in the first pass, we will make a second pass in which we modify the values at the individual grid cells. Before describing the iterative process for applying control, we must define a spatial blending function.

The spatial blending function, G'_r , basically describes how to spread the error δ'_i over individual grid cells. Though many possible G'_r could be used, we choose to use the same Gaussian weighting function used for sampling. Effectively, G'_r is the Gaussian weight multiplied by one over the sum of the squares of the Gaussian weights. The reasoning behind this perhaps counterintuitive result is described here.

Assume we have a difference $Q = \delta'_i$ 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

$$Q = \sum_{\vec{x}} G_r(\vec{x}_i, \vec{x}) Q'_x \quad (13)$$

where Q'_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. $Q'_x = Q$. However, if the sample point has a large support region (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 or Qr^3 in 2D/3D), and also affects points far away from the sample center as much as it affects the center itself. This is very undesirable.

We would prefer to instead distribute any additional density closer to the match point center x_i (where it contributes more to the sample), 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 $Q'_x = G_r(\vec{x}_i, \vec{x}) Q''$. We now must find Q'' , the amount to apply via a Gaussian spread to result in the correct overall change, Q . Substituting into equation 13, we have:

$$Q = \sum_{\vec{x}} (G_r(\vec{x}_i, \vec{x}))^2 Q'' \quad (14)$$

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

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

Thus, for each point in space, p , we would increase the density value at p by $\delta'_i G'_r(\vec{x}_i, p)$. Keep in mind that G'_r will typically be defined only within a fixed support radius around each sample point.

This spatial blending function is also used to spread the other errors, $\Delta'_{i,1}$ and $\Delta'_{i,2}$. Directional velocity control $\Delta'_{i,1}$ is applied the same way as with density, with simple vector addition. Applying control on the curl requires an additional computation; for a point p , the change in the vector value due to curl control is:

$$G'_r(\vec{x}_i, p) (\Delta'_{i,1} \times (\vec{x}_i - p)) \quad (16)$$

3.3.4. Iterative spread of control

The discussion above describes matching one control point. In practice, multiple control points with overlapping support are used to achieve more uniform control. When match points' radii overlap, the overlapping area can receive conflicting or compounding control. For example, two overlapping samples could both add density (thus compounding), or one could add and one remove (thus conflicting). To get around this, we use an iterative approach, similar to similar to Jacobi iteration. In each iteration, we recompute the amount of error, and apply control independently at each point.

If there is convergence, the behavior is roughly analogous to a damped (in the conflicting case) or overdamped (in the compounding case) oscillator. The obvious question is under what conditions this iterative application of control will converge, which we discuss next.

3.3.5. Guaranteeing convergence

In this section we show that our iterative spread of control is guaranteed to converge under certain conditions. Assume that we put n match points in the simulation domain. We will change our notation from that used earlier, for easier description, here. Let E_i^k be the error (or the amount of control) measured at match point x_i after the k^{th} iteration. For example, $E_1^0 = \delta'_1$ as defined in Equation 10 for a scalar. We derive the following equation:

$$e^{k+1} = W e^k \quad (17)$$

Where $e^k = [e_1^k \ e_2^k \ \dots \ e_n^k]^T$ is an n -dimensional error vector, $e_i^k = E_i^k \sum_{D_i} G_r(\vec{x}_i, \vec{x})$

and W is a matrix:
$$\begin{bmatrix} 0 & w_{12} & w_{13} & \dots & w_{1n} \\ w_{21} & 0 & w_{23} & \dots & w_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & w_{n3} & \dots & 0 \end{bmatrix}$$
 where

$w_{ij} = \sum_{D_i} (G_r(\vec{x}_i, \vec{x}))^2$ and D_i is the support of the kernel function $G_r(\vec{x}_i, \vec{x})$. Notice that the entries of W are a measure of fractional overlap of sample kernels.

Notice that if the power of matrix W converges to 0, our iterative approach will reduce the error vector to near zero after several iterations. A necessary condition for the power of W to converge is that the spectral radius of W is less than 1. Equation 17 also shows that the convergence of our method does not rely on the preview or the final but only on how the kernels are set up. We therefore use this as a clue to whether the matching will achieve a plausible result. Since computing the spectral radius of W is trivial, we can provide instant feedback to a user who wants to set up a simulation with our method.

4. Implementation and Limitations

The control procedure we have described is very simple, and compared to many other control-oriented methods, is straightforward to implement within a standard simulation. However, there are several practical issues of implementation that should be understood:

- *Placement and size of match points.* Obviously, the placement (x_i) and scale (r) of match points can have an effect on the final results. Some rules of thumb that we found useful from running our examples were:
 - Control seemed more effective when captured by a few match points, rather than the overlapping regions of multiple match points. When multiple match points overlap, the control of the overlap region is affected by a wider region of space, thus limiting the local control.
 - We generally achieved better results when all match points were the same size, rather than many different sizes. A default radius of about 3 times the low resolution grid cell width generally worked well. The standard deviation of the Gaussian kernels is chosen accordingly to make sure that the integral within the kernel is above 0.95.
 - Match points at too large of a scale did not provide much useful control, as there was too much freedom within the region.
 - Match points at very small scale seemed to introduce artificial detail from the control process, rather than from the high resolution simulation. We want detail to emerge from the high-resolution simulation, not from the control process.
 - While user interaction was useful in certain situations, we often found it easier and just as effective to place a regular pattern of match points automatically.
- *Threshold.* In practice, it is useful to set thresholds for error, below which no matching will occur. This reduces unnecessary controls for nearly-matching simulations, and can reduce computation time.
- *Density-based Weighting.* To prevent any sudden increases in or appearance of density during the *density* control process, we found that when matching density, using weights of m_i^r set proportional to the density in the high-resolution grid cell was useful.
- *Curl Control.* We have described a method for controlling for curl in the simulation vector fields (i.e. $D_{i,2}^r$). While it is easy to imagine cases where capturing and matching this behavior would be desirable (e.g. large vortices with tornado-like effects), in our test cases, curl control appears to have only a minimal effect. It may be worthwhile to ignore this control component in implementations.

There are also several limitations of our method:

- *Temporal Sampling.* Our current method applies matching at every frame. Applying our control when previous frames are not controlled could lead to dramatic, implausible results. Improvements in this sense, either by combining with a keyframe matching system or modifying our own method, are a topic for future work (see Section 6).
- *Global Conservation.* Because our control process is introducing or removing velocity and/or density from the system, we do not conserve momentum/density in the entire system. This lack of global conservation has not been noticeable in our tests, and most simulators already suffer from numerical dissipation that makes this point somewhat moot, but this could be an issue for some simulations.
- *Unallowed Values.* Since the control can add or subtract density, it is possible that the control process could cause a scalar value to move out of some allowed range. A situation where this could be problematic is a density value being reduced below 0.0. Again, we have not noticed this problem in any of our tests. It is trivial to introduce caps to enforce ranges of properties, though it is possible that doing so could impact convergence of the control process.
- *Blurring.* By spreading the control values spatially, our scheme may be seen as blurring out the fine-scale details we want to capture. As should be clear from our examples, this does not appear to be the case in practice, but it is easy to imagine how sourced density spread over a region could reduce the contrast in density values in the domain.
- *Inaccuracy.* One possible objection to our approach is that we lose the accuracy that might be gained from the higher resolution simulation (see Figure 4). It is true that some accuracy may be lost, however it is important to realize that this accuracy may be exactly what causes deviation from the desired behavior of the lower resolution simulation. We are making a tradeoff for control, and this will be true for any simulation control system.

5. Results and Discussion

We have implemented our preview-based sampling control system within an Eulerian fluid smoke simulator. Most examples use a basic smoke simulation in which a single density field (the smoke) is advected, starting from a single square/cube source. The smoke itself provides a buoyant force, added as an external force in the Navier-Stokes equations. During the advection step, we used BFECC [KLLR07] and monotonic cubic interpolation [FSJ01]. Our implementation is a single-threaded CPU-based simulation.

Many of our examples are in 2D, since this allows for much clearer understanding of the sampling and behavior of the system, which is often hidden by the complexity of a 3D simulation/rendering. Our method works fine in 3D, and we include some 3D examples to demonstrate this, as well.

Our examples typically use low resolution grids that are 32 in the horizontal direction (32×32 in 3D), and 64 in the vertical. Our high resolutions are a factor of 4 larger in each dimension. For example, our 3D high-resolution simulation has 64 times the number of grid cells of the low-resolution preview, and thus takes about two orders of magnitude longer to compute.

5.1. Results

We present some basic illustrative examples. Please refer to the accompanying video for more, and more complete versions of the simulations.

Regarding timing, the control process varies based on the level of overlap between match points (and thus the number of iterations to convergence). Matching can increase time by as little as 10% (with no overlap of points), but in our examples, it consistently increases time by about 80%.

Figure 1 presents an example of our control method applied in a 3D simulation. The effect of control while maintaining detail is clear.

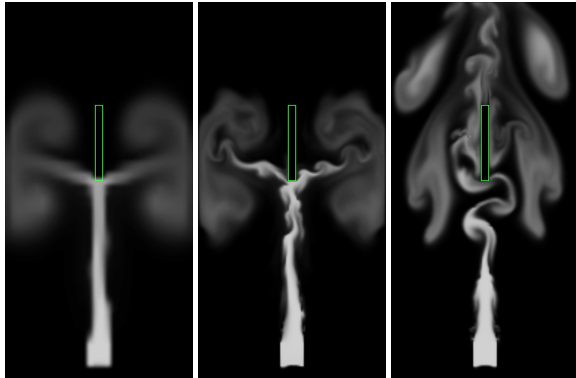


Figure 4: Controlling smoke when hitting a vertical bar. Left is preview, center is controlled high resolution simulation, right is uncontrolled.

The importance of control is further illustrated by the example in Figure 4. In this example, rising smoke hits a vertical bar. The coarse grid limits the ability of the smoke to curl around the bar, while a high resolution simulation curls much closer. The controlled simulation matches the behavior seen in the preview. Note that if one wanted the improved accuracy of the high resolution simulation, you would simply not match that area.

The importance of density control can be seen by the example in Figure 5. While control of the velocity alone captures the overall motion of the smoke, there are significant differences in density that are only captured by including density control (highlighted in the image).

The usefulness of density control is further highlighted by

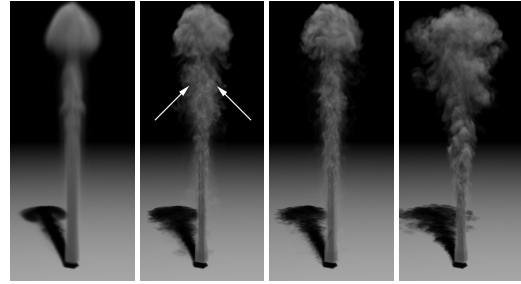


Figure 5: Comparison with and Without Density Control. The preview is at left and an uncontrolled simulation at right. Matching of only velocity (left center) captures motion correctly, but results in major differences in smoke density, as in the indicated areas. Adding density control (right center) fixes this.

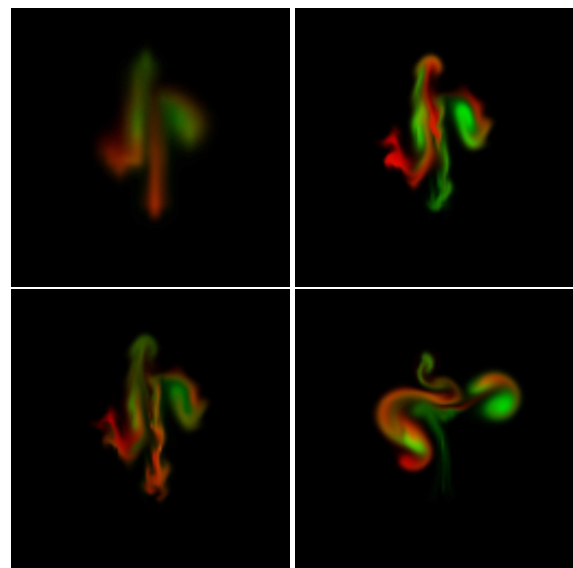


Figure 6: Control of Multiple Densities. Our method is capable of controlling multiple properties at once. The preview is located at upper left, and the uncontrolled high-res simulation at lower right. At upper right is the results of a simulation matching velocity only, while the lower left illustrates matching of the two densities.

the example in Figure 6. For this example (which takes place on a 32×32 grid in the preview), two density fields are advected. One behaves as smoke (rising), while the other moves in the opposite direction (falling). A method that matches only the velocity will be unable to match the results to the preview, since the densities will mix within the high resolution. Since our method can match multiple scalar and vector fields simultaneously, we can achieve control even for situations like this one. This type of example is common in

simulations where multiple fields affect flow (e.g. temperature and particle density).

In addition to these examples, the accompanying video contains examples of other control situations, spatial variation of the match points, multiple kernel sizes, the effect of curl control, user interaction in setting match points, additional 3D examples, and a comparison with another method.

5.2. Discussion and Comparison

There are four groups of prior techniques (discussed in Section 2) that we will briefly contrast our method with. First are keyframe matching techniques. Note that our keyframes arise from fundamentally the same simulation. Since the preview and high-res simulations follow the same physically-based rules, and are matched in the previous frames, the two simulations should never diverge by much. Thus, the control forces/densities required at any one frame are generally minor and not very noticeable (though they can have large effects over several frames). In contrast, prior keyframing approaches in fluid simulation can make no assumptions and thus must go through a much more extensive optimization process to match keyframes.

A second group of techniques are those designed to add additional detail (usually turbulence) to a lower resolution simulation. These methods have the benefit of never requiring the larger computation time and storage of a high-resolution simulation. However, these methods add only certain types of low-level motion (typically turbulent “noise”), and do not necessarily capture the fine-scale motion that an actual high resolution simulation will produce. Further, these methods generally only provide detail to the vector field (implicitly extending to density fields), making it difficult to handle situations with multiple different densities. Note also that nothing prevents the subsequent application of techniques such as wavelet turbulence [KTJG08] to our results.

The third approach, and the one most similar to our own, is that of Nielsen et al. [NCZ*09, NC10]. Like our method, Nielsen et al. use a low resolution preview as guidance for a high-resolution simulation. Their method relies on a modification within the pressure projection step in which the low resolution simulation’s velocity values are upsampled to high resolution, followed by a process that modifies the high resolution velocities appropriately. There are a few important differences between their approach and our own. First, their approach deals only with the velocity field. Although their second paper [NC10] incorporates smoke density, this is used only to determine how to adjust the velocity field itself. As a result, there is no way of matching properties of the low resolution simulation other than the velocity, and multiple density examples (Figure 6) cannot be simulated. Second, because of the way in which their approach requires solutions to a large linear system, it has very large memory requirements, and significant computational expense. Third,

though this is subjective, we believe that our approach is far easier to implement and to integrate into existing systems.

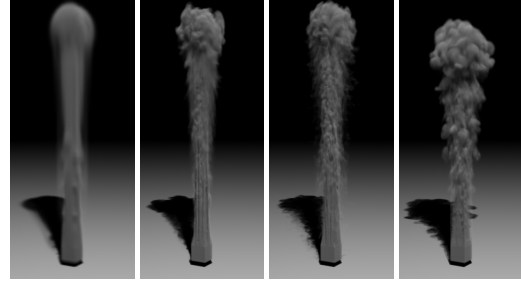


Figure 7: Comparison to Nielsen et al. Preview is at left, Nielsen et al.’s method at center left, our method at center right, and high resolution at right.

We have implemented Nielsen’s method, and provide a comparison for an example in Figure 7 and in the video. Our implementation is of the method found in [NCZ*09], with the additional modifications for weighting by density in [NC10]. We use an alpha value of 0.65 for that implementation. As is seen in the example, our method better captures the density values of the preview simulation, and also appears to maintain fine details of the high resolution simulation better.

6. Conclusion

Our method is simple, is easy to integrate with existing Eulerian simulations, and provides control across multiple simulation properties. As outlined above, our method has specific advantages over other methods that control generation of high resolution fluid simulations, and thus presents a useful alternative. The technique should prove useful especially in production environments.

There are several avenues still open for future work, such as:

- Automatically identifying “interesting” phenomena to aid in match point placement.
- Extensions for mass/momentum conservation, locally and globally, (desirable for *some* types of simulations).
- Application to other domains. We believe the method could be adapted to Lagrangian frameworks, and might be applicable to liquid simulation.
- Solving for control with other methods beyond our iterative Jacobi scheme, such as Gauss-Seidel.
- Adding a temporal component. Currently, we match simulations at every frame. We could instead sample keyframes at only some points, and provide limited control (e.g. by advecting the sample points backward through the velocity field) at earlier times.
- Combining our approach with others, e.g. standard

keyframe matching to hit an initial set of match points, or addition of turbulence.

Acknowledgements

We would like to thank Vivek Sarin and Shu-Wei Hsu for helpful discussions. This work was supported in part by NSF Grant IIS-0917286. This publication is based in part on work supported by Award No. KUS-C1-016-04, made by King Abdullah University of Science and Technology (KAUST).

References

- [ANSN06] ANGELIDIS A., NEYRET F., SINGH K., NOWROUZEZAHRAI D.: A controllable, fast and stable basis for vortex based smoke simulation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2006), pp. 25–32. [2](#)
- [BHW96] BARZEL R., HUGHES J. F., WOOD D.: Plausible motion simulation for computer graphics animation. *Proceedings of the Eurographics Workshop on Computer Animation and Simulation* (1996), 183–197. [2](#)
- [BMWG07] BERGOU M., MATHUR S., WARDETZKY M., GRINSPUN E.: Tracks: toward directable thin shells. *ACM Trans. Graph.* 26 (2007), 50:1–50:10. [2](#)
- [CF00] CHENNEY S., FORSYTH D. A.: Sampling plausible solutions to multi-body constraint problems. *Computer Graphics (SIGGRAPH 2000)* (2000), 219–228. [2](#)
- [FF01] FOSTER N., FEDKIW R.: Practical animation of liquids. *Proceedings of SIGGRAPH '01* (2001), 23–30. [2](#)
- [FL04] FATTAL R., LISCHINSKI D.: Target-driven smoke animation. *ACM Trans. Graph.* 23, 3 (2004), 441–448. [3](#)
- [FM96] FOSTER N., METAXAS D.: Realistic animation of liquids. *Graphical Models and Image Processing*, 58(5) (1996), 471–483. [2, 3](#)
- [FM97] FOSTER N., METAXAS D.: Controlling fluid animation. *Computer Graphics International* (1997), 178–188. [2](#)
- [FSJ01] FEDKIW R., STAM J., JENSEN H. W.: Visual simulation of smoke. *Proceedings of SIGGRAPH '01* (2001), 15–22. [7](#)
- [HK04] HONG J.-M., KIM C.-H.: Controlling fluid animation with geometric potential: Research articles. *Computer Animation and Virtual Worlds 15*, 3-4 (2004), 147–157. [2](#)
- [KLLR07] KIM B., LIU Y., LLAMAS I., ROSSIGNAC J.: Advections with significantly reduced dissipation and diffusion. *IEEE Transactions on Visualization and Computer Graphics 13* (2007), 135–144. [7](#)
- [KMT06] KIM Y., MACHIRAJU R., THOMPSON D.: Path-based control of smoke simulations. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2006), pp. 33–42. [2](#)
- [KTJG08] KIM T., THÜREY N., JAMES D., GROSS M.: Wavelet turbulence for fluid simulation. *ACM Trans. Graph.* 27 (2008), 50:1–50:6. [3, 9](#)
- [Lin94] LINDBERG T.: *Scale-Space Theory in Computer Vision*. Kluwer Academic Publishers, 1994. [4](#)
- [MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003), pp. 154–159. [2](#)
- [Me107] MELEK Z.: *Interactive Simulation of Fire, Burn, and Decomposition*. PhD thesis, Texas A&M University, 2007. [3](#)
- [MK07] MELEK Z., KEYSER J.: *Preview Directed Control of Complex Fluid Simulations*. Tech. Rep. tamu-cs-tr-2007-8-1, Texas A&M University Dept. of Computer Science, 2007. [3](#)
- [MTPS04] MCNAMARA A., TREUILLE A., POPOVIĆ Z., STAM J.: Fluid control using the adjoint method. *ACM Trans. Graph.* 23, 3 (2004), 449–456. [3](#)
- [NC10] NIELSEN M. B., CHRISTIANSEN B. B.: Improved variational guiding of smoke animations. *Computer Graphics Forum* 29 (2010), 705–712. [3, 9](#)
- [NCZ*09] NIELSEN M. B., CHRISTENSEN B. B., ZAFAR N. B., ROBLE D., MUSETH K.: Guiding of smoke animations through variational coupling of simulations at different resolutions. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2009), pp. 217–226. [3, 9](#)
- [NSCL08] NARAIN R., SEWALL J., CARLSON M., LIN M. C.: Fast animation of turbulence using energy transport and procedural synthesis. *ACM Trans. Graph.* 27 (2008), 166:1–166:8. [3](#)
- [PCS04] PIGHIN F., COHEN J. M., SHAH M.: Modeling and editing flows using advected radial basis functions. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2004), pp. 223–232. [2](#)
- [Pop01] POPOVIĆ J.: *Interactive design of rigid body simulations for computer animation*. PhD Thesis (2001). [2](#)
- [PSE*00] POPOVIĆ J., SEITZ S. M., ERDMANN M., POPOVIĆ Z., WITKIN A.: Interactive manipulation of rigid body simulations. *Computer Graphics (SIGGRAPH 2000)* (2000), 209–218. [2](#)
- [PTC*10] PFAFF T., THÜREY N., COHEN J., TARIQ S., GROSS M.: Scalable fluid simulation using anisotropic turbulence particles. *ACM Trans. Graph.* 29 (2010), 174:1–174:8. [3](#)
- [REN*04] RASMUSSEN N., ENRIGHT D., NGUYEN D., MARINO S., SUMNER N., GEIGER W., HOON S., FEDKIW R.: Directable photorealistic liquids. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2004), pp. 193–202. [2](#)
- [SB08] SCHECHTER H., BRIDSON R.: Evolving sub-grid turbulence for smoke animation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2008), pp. 1–7. [3](#)
- [SDE05] SCHPOK J., DWYER W., EBERT D. S.: Modeling and animating gases with simulation features. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2005), pp. 97–105. [2](#)
- [Sta99] STAM J.: Stable fluids. *Proceedings of ACM SIGGRAPH '99* (1999), 121–128. [2](#)
- [SY05a] SHI L., YU Y.: Controllable smoke animation with guiding objects. *ACM Trans. Graph.* 24, 1 (2005), 140–164. [2](#)
- [SY05b] SHI L., YU Y.: Taming liquids for rapidly changing targets. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2005), pp. 229–236. [2](#)
- [TKPR06] THÜREY N., KEISER R., PAULY M., RÜDE U.: Detail-preserving fluid control. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2006), pp. 7–12. [2](#)
- [TMPS03] TREUILLE A., MCNAMARA A., POPOVIĆ Z., STAM J.: Keyframe control of smoke simulation. *ACM Trans. Graph.* 22, 3 (2003), 716–723. [3](#)