Target Particle Control of Smoke Simulation

Jamie Madill* Carleton University

David Mould† Carleton University

ABSTRACT

User control over fluid simulations is a long-standing research problem in computer graphics. Applications in games and films often require recognizable creatures or objects formed from smoke, water, or flame. This paper describes a two-layer approach to the problem, in which a bulk velocity drives a particle system towards a target distribution, while simultaneously a vortex particle simulation adds recognizable fluid motion.

A bulk velocity field is obtained by distributing target particles within a mesh, then matching control particles with target particles; control particles are given a trajectory bringing them to their targets, and a field is obtained by interpolating values from the control particles. A detail velocity field is obtained by traditional vortex particle simulation. We render the final particle system using stochastic shadow mapping. We spend some effort optimizing our processes for speed, obtaining simulations at interactive or near-interactive rates: from 70 to 500 milliseconds per frame depending on the configuration.

Index Terms: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.6.8 [Simulation and Modeling]: Types of Simulation—Animation

1 INTRODUCTION

Control is arguably the most important problem in modeling natural phenomena for special effects. Without a high-level control system, the artist may have to manipulate physical parameters and inject forces in many places to achieve a particular desired result. Ideally, control systems should be intuitive and easy to use; the artist must be able to effectively achieve a desired effect. Since we are using fluid simulation, which is usually already expensive to compute, it is important that the control scheme be as computationally efficient as possible. Our goal is to find a control scheme fast enough for artists to use as they design a scene. We are primarily interested in real-time performance, and have designed a system that sacrifices physical accuracy in favor of speed. Interactive speeds in simulation and rendering gives artists shorter iteration times when shaping new effects, and is essential for games and interactive media.

One useful abstraction is a target that fluid particles travel towards, such as a target density function, or mesh. Many examples of creatures, characters and objects made up of fluid are found in animation and movies, such as the the rivergod of Narnia [30] or Gandalf's galley made of smoke [4]. In these cases, an animator places a target mesh, animates it as though it were a character, and the control system determines how to route the fluid towards this destination. Such a control scheme avoids the need to tune numerous physical parameters.

Our approach is a particle-based target driven control scheme, applied to smoke simulation. We split the velocity field into two components, a base velocity and a detail velocity: a low-frequency bulk velocity field drives the smoke volume towards a target mesh, while a Lagrangian vortex particle turbulence model generates fine-scale fluid motion. For the base velocity, we generate target particles within a control mesh, and match these with control particles placed in the smoke volume, to drive the smoke velocity. Thus we can effectively drive the smoke towards the target shape, while providing high-level control to an animator. The detail velocity, generated with vortex particles, adds the appearance of a turbulent gas. We contribute this two-part approach, using a target particle based approach to fluid control, combined with a Lagrangian particle method; the resulting scheme is computationally efficient and operates at interactive rates.

*e-mail: jamie@nullspace.ca

†e-mail: mould@scs.carleton.ca

2 PRIOR WORK

Foster and Metaxas [7, 8] were among the first in computer graphics to directly solve the incompressible Navier-Stokes equations, using small grids. Stam [25] demonstrated unconditionally stable Eulerian simulation: borrowing ideas from the Lagrangian perspective, Stam's semi-Lagrangian advection avoids creating excess velocity. Subsequent work by Fedkiw et al. [5] introduced the notion of vorticity confinement to counter numerical diffusion introduced by semi-Lagrangian advection. Foster et al. [6, 11], applied the particle level set method to counteract diffusion for free-surface liquids.

Lagrangian approaches are characterized by simulation elements moving through the domain. Smoothed Particle Hydrodynamics (SPH) solves the Navier-Stokes equations by interpolating values in space, given a smoothing kernel [15].

Vortex simulation is a Lagrangian approach using primitives that spin the fluid elements, which together produce turbulent fluid flow [3]. Park et al. [18] use vortex particles for smoke simulation. Their method for no-slip and no-through boundary conditions produces an effect known as vortex shedding, where new vortices form a turbulent wake behind a solid object. Selle et al. [21] demonstrate that vortex particles can counter the effects of numerical diffusion in Eulerian fluid simulation. Others have proposed using extended vortex primitives such as straight lines [9], connected line segments [1, 31], and continuous Fourier series [2]. Pfaff et al. [19] propose a technique for combining vortex sheets with a bulk velocity to produce high-resolution smoke simulation, though without applications in fluid control.

Control is an extremely important aspect of animation; effect production often calls for a the fluid to assume a specific target shape [30]. Many different approaches to control have been explored in recent years [16].

Seminal work by Treuille et al. [29], later improved by McNamara et al. [17] using the adjoint method, uses artist supplied keyframes of a target density function to directly specify the fluid destination. Using nonlinear optimization, they find optimal control forces that move fluid to a target, while minimizing the total amount of control. Their methods were reported as unstable and slow, requiring derivatives of every control parameter.

Subsequent work by Fattal and Lischinski [4], simplified their approach by using a driving force combined with a smoke gathering term. A driving force provides a shape matching velocity, along the gradient of the target density. The gathering term adjusts the source density of the smoke cloud to match that of the density of the target. Their efficient framework has the problem of turbulent motion stopping once the smoke matches the target shape.

Shi and Yu [24, 23] propose a method that is both efficient and allows the smoke to move naturally after reaching the target density. They convert the target density field to a level set, and apply control velocity along the boundary. They also propose applying a similar method to free surface liquids [24], that considers the skeleton of the target mesh to be a center that pulls the liquid inwards, keeping it together. Hong [12] proposes using a potential field for target control. His simple method is very efficient; however, without user interaction, the method may obscure thin features.

Rasmussen et al. [20] propose a different form of control framework than previous target-based methods. Their particle-based detailed control scheme gives the artist the ability to place control particles governing the velocity, level set (liquid injection/erasers), and divergence. Clouds of thousands of particles allow for detailed artist control.

Thurey et al. [28] propose a system for particle fluid control based on the Lattice Boltzmann method. Their control framework uses a single class of control particle instead of many; control particles act as both locally based magnets, that attract nearby fluid particles, and wind forces, that transport fluid particles along the moving path of the control particle. This allows a simpler control scheme than Rasmussen et al.; however, it again lacks target-based control.

Spline curves offer a different choice of control primitive, where fluid moves along a path in space. Gates proposes a spline primitive consisting of a series of directional flow primitives, to generate a divergence free velocity field [9]. Kim et al. [14] augmented the path control spline with vortex particles, Rankine vortices, and the ability to handle self-intersecting paths. Their control system is computationally efficient, adding little overhead, and is able to handle complex paths, with attractive results. Angelidis et al. [2], in their work on vortex filaments, propose a controlled method of adjusting the properties of a vortex ring, to guide the motion of the filaments along a curve.

3 Overview

In our approach to targeted fluid control, we split the velocity field in two: a control velocity field moves the smoke directly towards the target, matching the target shape, while a turbulent velocity field adds the illusion of fluid motion. Our algorithm is composed of the following steps:

1. Initialize marker, control and target particles
2. Compute the bulk control velocity field (Section 4)
   - Generate attraction force on the control particles
   - Compute control velocity field
3. Compute the turbulent vortex velocity field (Section 5)
   - Evaluate vortex velocity field
   - Update vortices (spinning and strength exchange)
   - Spawn new vortex particles
4. Particle update (Section 6)
   - Particle advection
   - Smoke particle redistribution
5. Render marker particles (Section 7.1)

The details of the algorithm are given in sections 4, 5, and 6. We discuss rendering and implementation details in Section 7, and present some results in Section 8. In Section 9 we outline some key observations and future work before concluding in Section 10.

The specific contributions of this work are:

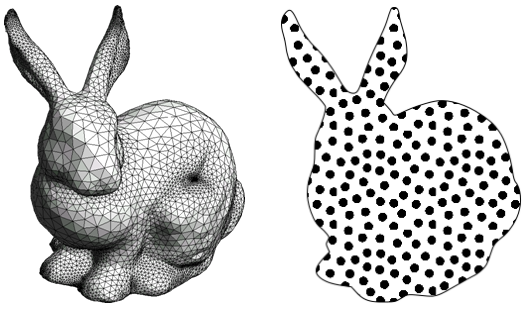

Figure 1: Mesh sampling example

- A unique point-based shape matching algorithm based on point correspondence
- An efficient and parallel direct evaluation method for finite support vortex particles
- A method to combine vortex particles with a shape matching velocity for targeted smoke control

4 Targeted Control Velocity

This section describes how our base velocity system moves the fluid to the target. Before starting the simulation, we first sample the target mesh, generating a set of target particles. Then, we place an equal number of control particles inside the smoke cloud. A correspondence between the target and control particles provides the basis for an attraction force, which then steers the control particle velocity. We then interpolate this control velocity into a bulk velocity field, used to advect particles during simulation.

4.1 Control Particles

We wish to place control particles evenly over the smoke domain. We first compute two scalar fields: a smoke marker particle density field, representing the distribution of the M marker particles over space, and also a control density field, from the existing control particles, if any. By using a separable tent blur with kernel radius r_m (for smoke particles) and r_c (for control particles), we can compute these fields very efficiently. Good values of r_m and r_c are typically quite small, and though they depend on the size of the simulation, they might be as small as 5% of the size of the domain of interest.

We then sample the space of the marker particles uniformly, placing a control particle where the ratio of control potential to marker potential is below a given threshold. On placing a new particle, we update the control density field, and repeat this process C_{init} times. We can choose between two policies on when to initialize control particles: either once whenever new smoke particles are emitted, or once every frame, allowing the control particles to saturate the smoke. In either case, we will have N control particles, where $N \ll M$.

4.2 Target Particles

Our next step is to generate exactly N target particles inside the target shape. For performance, we first convert the target shape to a signed distance field representation, using the approach described by Swoboda [26].

We then perform stratified sampling of a grid for an initial set of target particles, $N' \neq N$. In a second pass, we add or delete random particles until we have exactly N. In the case of undersampling, we add points with a uniform random distribution. Figure 1 illustrates our sampling outcome.

The target shape may be undergoing transformations, or moving through space. To move the points with a changing mesh, we use

linear interpolation over the Delaunay triangulation, or tetrahedralization in 3D (for simplicity, we call the 3D case a triangulation also). We compute the triangulation once, when the mesh is initialized. The triangle containing each target particle, in the mesh's rest configuration, gives a barycentric coordinate identifying the position of that target particle. We store the containing mesh element, and corresponding barycentric coordinate, for each point, and when the mesh moves we can compute a point's transformed position by using its barycentric coordinate with the new mesh vertex positions.

4.3 Attraction Force

Given our cloud of N control and target particles, we wish to find a correspondence between them, and use this to drive the smoke towards the target. In order to find a force that moves the control particles quickly and accurately, we use energy minimization to find a matching with paths of short distance. However, note that, although we prefer short paths, since that will mean faster convergence, we also prefer paths of uniform length, so distant particles will arrive at the target at about the same time as nearby particles. Thus, we choose our energy metric to be *squared* Euclidean distance, penalizing longer paths. The correspondence then gives an attraction force F_a, exerted on the control particles.

We initially assign each control particle to a unique random target. Then, at each time step, we perform a fixed number C_{opt} of optimization steps. We show our optimization algorithm in Algorithm 1. We randomly choose two control particles a and b, and test if exchanging their targets produces an improvement in global energy. The fixed number of iterations per frame gives an incrementally improving, adaptive and coherent solution, and allows us to provide a fixed run-time budget. Although not guaranteed to converge, with control particle counts in the thousands, a few thousand iterations will provide an approximate solution with nearly identical quality to the optimal solution. In other words, after a certain time investment for optimization, the improvements in velocity field quality drop off rapidly.

Figure 2 shows a sample of our the performance of our algorithm in a simple 2D scene with 10,000 particles. After 80,000 iterations, the solution is only improving very slowly. In most of our results we use a few thousand control particles, and hence arrive at a near-optimal solution with far fewer iterations than this example. We are able to perform 50,000 swaps in under 1 ms. This is virtually free, and gives a near-optimal solution within 2-3 frames.

Algorithm 1 Incremental correspondence optimization

```
for 1 to C_opt do
    i ← RANDOM(1,N)
    j ← RANDOM(1,N)
    a ← POSITION(i)
    b ← POSITION(j)
    t_a ← TARGETPOSITION(i)
    t_b ← TARGETPOSITION(j)
    d_old ← |a−t_a|^2 + |b−t_b|^2
    d_new ← |a−t_b|^2 + |b−t_a|^2
    if d_new < d_old then
        SWAP(POSITION(i), POSITION(j))
        SWAP(TARGETPOSITION(i), TARGETPOSITION(j))
    end if
end for
```

Each pair of control and target particle uses an attraction weight w_a, to compute the attraction force F_a. Here, $w_a = (|x - x_t| - A_\phi)A_\rho$, where x is the position of the control particle, x_t is the position of the target particle for this control particle, and A_ϕ and A_ρ are scaling parameters. Then, $F_a = \frac{x-x_t}{|x-x_t|} * \text{SATURATE}(w_a) * S_a$ where S_a is a global attraction strength parameter, and SATURATE clamps

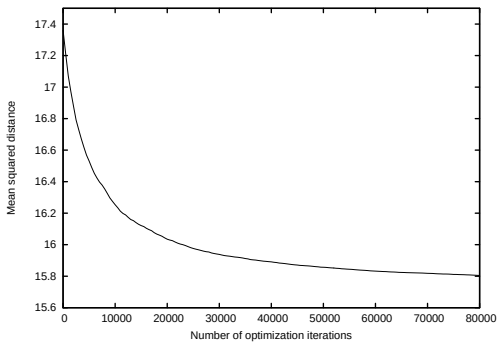

Figure 2: Correspondence optimization convergence with 10,000 particles

w_a to $[0,1]$. This linear function gives an increasing weight with distance, allowing the attraction force to gradually ramp down as particles approach their targets.

During particle update, described below in Section 6, we integrate attraction force to give control velocities. We also apply a simple velocity damping to the control particles: we compute a damped velocity $v_d = v_f * (1 - c)$, where c is a scalar damping strength parameter, usually near zero.

4.4 Velocity Interpolation and Extrapolation

We have a cloud of moving control points, each with a separate velocity; we want to distribute these velocities over space to transport a cloud of marker particles representing the smoke. We determine an interpolation (for velocities between our control points samples), and extrapolation (for velocities outside the region of the samples), using a variant of Shepard's Method [22], i.e., inverse distance weighting. We have a collection of N samples u_i, where each interpolated control velocity $\mathbf{u}_c(\mathbf{x})$ at a point $\mathbf{x}$ is expressed as

$$\mathbf{u}_c(\mathbf{x}) = \sum_{i=1}^{N} \frac{w_i(\mathbf{x})u_i}{\sum_{j=1}^{N} w_j(\mathbf{x})}, \tag{1}$$

where

$$w_i(\mathbf{x}) = \begin{cases} (r_v^2 - d(\mathbf{x},\mathbf{x_i})^2)^3 & \text{if } d(\mathbf{x},\mathbf{x_i}) < r_v \\ 0 & \text{otherwise} \end{cases}. \tag{2}$$

Here d is Euclidean distance, with radius r_v giving the region of influence of a control particle. This distance weighting defines velocity over the radius of a particle, thus it extrapolates velocity outside the convex hull of the data points, to a distance of r_v. In practice this function gives good results, and is very simple to implement.

We chose this method for its speed, simplicity, and ease of extrapolation; however, the interpolation is not exact. The interpolated velocity at control particle position p does not equal the original control velocity. In effect, we have a smoothed velocity field, averaged over nearby particles. Since we are creating a bulk velocity, this is adequate; for applications that require an exact method, the interpolation module can be replaced.

5 Turbulent Vortex Velocity

Our control velocity efficiently matches the source density to the target density; however, it lacks turbulent motion. To add the missing characteristic rolling, turbulent smoke motion, we use the vortex particle method to generate a turbulent velocity field, and combine it with the control velocity for our final velocity field.

5.1 Vortex Particle Method

The equations for vortex velocity are derived directly from the curl of the Navier-Stokes equations:

$$\frac{\partial \omega}{\partial t} + (\mathbf{u} \cdot \bigtriangledown)\omega = (\bigtriangledown \mathbf{u}) \cdot \omega + v \bigtriangledown^2 \omega + \frac{1}{\rho} \bigtriangledown \times \mathbf{f}. \quad (3)$$

We solve the first term in Equation (3) using the vortex particle method. We are most interested in the velocity contribution, $\mathbf{u}_v(x)$ of a vortex particle v at a point x. For our work, we use the vortex energy kernel proposed by Angelidis et al. [2], where the velocity-from-vorticity equation is given as

$$\mathbf{u}_v(x) = \sum_{i=1}^{N_v} [(x - x_i) \times \omega] \, \xi(|x - x_i|^2) \quad (4)$$

for N_v vortices. Here, ξ defines the strength of a vortex, given a squared distance to a point in space. Angelidis et al. define the compact energy kernel, ξ, of a vortex as

$$\xi(x^2) = \begin{cases} \left(4 - \frac{20}{x^2+4}\right)^2 & \text{if } x^2 < 1, \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

This kernel requires no square roots, is smooth throughout the domain, and is zero everywhere where $x^2 \geq 1$. Combined with a vortex radius, σ, which determines the area of effect of each vortex, the vorticity vector ω determines both the direction and magnitude of rotation. Thus the expression for vortex vorticity in our simulation is $\xi(x^2)\sigma$.

5.2 Efficient Vortex Evaluation

Given a collection of vortex particles, each with a radius and vorticity vector, we want to evaluate the velocity at every smoke particle as accurately and quickly as possible. Since we care about performance more than physical correctness, we borrow elements of the Eulerian frame, to accelerate the velocity computation. We can compute the exact vortex velocity at every cell center of a uniform grid directly, and then find an approximate vortex velocity anywhere in our domain of interest, using bi-linear or tri-linear interpolation along the velocity grid. Note that if the grid is too coarse, the structure of the vortices will be lost due to undersampling.

We perform direct velocity from vorticity computation using a "splatting" approach, inspired by 3D graphics hardware texture blending. For every vortex, we find a bounding box of grid cells from its area of influence, given by its radius and magnitude; we then traverse each cell in order, adding the new vortex velocity to each cell center we traverse. This simple approach allows us to exploit SIMD vector instructions to compute several velocity values in parallel. Although the linear interpolation from grid cells introduces a source of error, the error is small enough to preserve the overall effect; the performance advantages are overall more beneficial than perfect accuracy.

5.3 Particle Strength Exchange

The second term of the right hand side of equation (3) is a diffusion term that describes how the spinning vorticity slows down over time. In our Lagrangian vortex particle framework, we implement this diffusion effect using *particle strength exchange* [3, 18] (PSE). Over time, nearby vortices will exchange vorticity, gradually unifying their motion.

We implement PSE by finding neighbouring pairs of vortices. For nearby vortices $i$and j, with vorticity ω_i and ω_j, and radii σ_i and σ_j, we compute new vorticity values as

$$\omega_i' = \omega_i + \upsilon * d_{ij} * (\sigma_j^3 \omega_j - \sigma_i^3 \omega_i), \quad (6)$$

where

$$d_{ij} = \textsc{Saturate}\left(1 - \frac{|p_i - p_j|}{d_{max}}\right). \quad (7)$$

And similarly for ω_j'. Here, SATURATE is a function which clamps the input value to $[0,1]$, and d_{max} is a parameter which governs the maximum distance at which vortex particle can affect another. The parameter υ determines global diffusion strength.

5.4 Vortex Spinning

The first term on the right side of equation (3), is called the stress term. It describes the stretching and tilting of vortices, due to the velocity field. This term only applies in 3D. Expanding this term in three dimensions, where the 3D velocity vector $\mathbf{u} = (u, v, w)$), we get

$$\omega_x \frac{\partial \mathbf{u}}{\partial x} + \omega_y \frac{\partial \mathbf{u}}{\partial y} + \omega_z \frac{\partial \mathbf{u}}{\partial z} = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} \end{bmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = J(\mathbf{u})\omega. \quad (8)$$

$J(\mathbf{u})$ is a 3 by 3 matrix, called a Jacobian matrix. In our implementation, similar to that of Gourlay [10], we compute the Jacobian from the gradient of the velocity field. Since this term can cause energy to be gained or lost from the system, as a result of round-off and truncation due to linear interpolation, we constrain the energy of each vortex by normalizing the new vorticity vectors to the same magnitude as they had before this step. We call this vortex spinning, instead of stretching or tilting, as it only changes the direction of rotation.

5.5 Spawning Vortex Particles

We use dart-throwing to spawn new vortex particles, similar to the process of placing control particles. We store the current turbulent energy of the system in a grid, then randomly throw darts into the volume (or area in 2D); when the marker particle density (described in Section 4.1) is above a given threshold, and turbulent energy is below a threshold, we place a new vortex particle. We randomly assign vorticity and radius. Incrementally, over time, vortex particles will saturate the smoke. As we describe in the next section, vortex particles move with both the control velocity and vortex velocity fields, but control particles only move with their own control velocities.

6 Particle Update

Now that we have described our methods of vortex simulation and initialization, we will describe how we move our particles with the velocity in our system. Since we intend to simulate millions of marker particles at interactive rates, performance is our primary concern. We first combine the interpolated control velocity field, described in Section 4, with the turbulent vortex velocity field described in Section 5.1. We store this combined velocity field in a uniform grid, which can be quickly sampled for an approximate velocity, using bilinear or trilinear interpolation.

6.1 Advection

Vortex particles and smoke marker particles move according to the current combined velocity field. This means the marker and vortex particles are both subject to the influence of the control and vortex velocities. Control particles, in contrast, move according to their individual control velocities, determined from the attraction force.

To find the new position of a particle given a velocity, the problem statement takes the form of an ordinary differential equation. However, in our case, we can make a simplification; because we have a velocity field, we can simply step the position of the particles through this field, instead of recalculating the entire state vector

at every step. In this simplified case, if the position of a particle i is x_i, we have $x_{n+1} = f(\Delta t, x_n)$, where f is a function that computes a new position from the previous position for a given time step value, Δt.

We choose to use a second-order Adams-Bashforth scheme, described by Park and Kim [18], where

$$x_{n+1} = x_n + \Delta t \left(\frac{3}{2}\mathbf{u}(x_n) - \frac{1}{2}\mathbf{u}(x_{n-1}) \right). \quad (9)$$

This higher-order scheme uses the previous frame's velocity to estimate a more accurate particle trajectory. The higher-order accuracy comes at a very low cost in runtime performance; the cost in terms of memory is the expense of storing velocities computed for the every particle in the previous simulation step. Control particles move according to individual forces; we use the Euler method to integrate attraction force before advecting the particles according with the Adams-Bashforth integration mentioned previously.

6.2 Redistribution

When we move the marker particles along with the vortex velocity field, they often end up outside the area of control; chaotic vortex movement may trend away from the region of control. This effect, similar to diffusion, conceals mesh details. Since we are focused on performance, we chose a non-physical technique to address this scattering. Finding marker particles outside of the control area, we randomly place them back in the control area using rejection sampling. The algorithm is listed in Algorithm 2. Note that for sampling control potential, we use the control particle potential field described in Section 4.1. Also the value of ε must be small to ensure the algorithm terminates.

Algorithm 2 Incremental randomized marker particle redistribution

```
ControlRegion ← COMPUTEBOUNDINGBOX(ControlPoints)
for 1 to C_redist do
    i ← RANDOM(1, NumMarkerParticles)
    p ← MarkerParticlePosition(i)
    if CONTROLPOTENTIALAT(p) < ε then
        repeat
            p' ← RANDOMPOINTIN(ControlRegion)
        until CONTROLPOTENTIALAT(p') < ε
        MarkerParticlePosition(i) ← p'
    end if
end for
```

The incremental nature of the algorithm is such that, if C_{redist} is sufficiently large, gradually the particles move back inside the volume. It also has the advantage of being simple to implement, and computationally inexpensive. We have found this method prevents the marker particles from being left outside the region of influence of the control points effectively and unobtrusively.

7 Implementation

In this section we will describe the implementation of our simulation and rendering. Since we chose fast, parallelizable, techniques, we make use of the threading abilities of CPUs as much as possible. We avoid the complex functions usually involved in spatial structures such as octrees and KD-trees, and many steps are trivially parallel. Even with detailed simulations, we usually run at interactive rates.

We make use of SIMD instructions for optimization. Since they operate most effectively by loading groups of data elements simultaneously, instead of storing a single array of packed structures with (x,y,z) data per element, called the 'structure-of-arrays' layout, we prefer the 'array-of-structures' layout. The array-of-structures layout stores data in a structure of two or three large linear single-component arrays; it stores two large float arrays for 2D data, and three large arrays for 3D data. The allows us to skip the overhead of reorganizing our data in memory to fit SIMD operations, and gives good read/write performance.

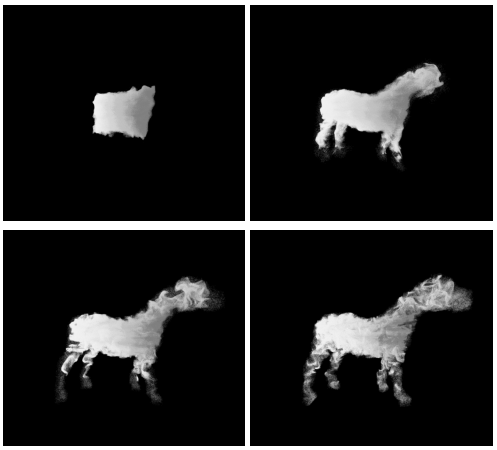

Figure 3: A sample animation matching the shape of a horse

7.1 Rendering Implementation

We use a volumetric rendering proposed by Swoboda [27] named "stochastic shadow mapping." Swoboda's method has the main advantages of being relatively cheap to compute and simple to implement, while giving attractive renders.

Similar to traditional shadow mapping, we use a large 2D shadow map texture (in our results 4096x4096). We draw the smoke particles as points to this texture from the light's point of view, storing distance to the light. We then perform a stochastic shadow test in a second pass: for every particle, we sample the shadow map cells near the test particle with a large kernel (of size as large as 15). We perform a shadow test of the stored shadow depth versus particle depth, and count the number of failed tests. The fraction of occluded samples determines the final opacity value for the particle. The large kernel helps give a smooth, soft result.

Due to the numerous collisions when inserting points into the large shadow map, the smoke will display a fair amount of temporal aliasing. Thus Swoboda proposes two additional steps to produce a temporally smooth render. First, insert particles into the shadow map with a small, random offset. This helps to ensure that no particle or group of particles are obscured completely. Second, use a temporal smoothing step; blending each particle's shadow value value with that from the previous frame offers reasonably alias-free renders with good temporal coherence. We use Intel's TBB library [13] to sort our particles back-to-front relative to the camera for compositing during rendering.

8 Results

In Figure 3, we present a comparative result to Shi and Yu [23]. We show their results in Figure 4. For this test, we used 400,000 marker particles, 10,000 control particles, and roughly 10,000 vortices. They report simulation times of 4 hours and 15 minutes to generate a 1250 frame animation sequence, giving an average simulation time of 12.24 seconds per frame. Our result takes between 120-170 milliseconds per frame. Faster CPUs, multi-core processors, may account for some of this performance difference. Their timing results also do not include render times, which we do concurrently to our simulation. As expected, we do not display the

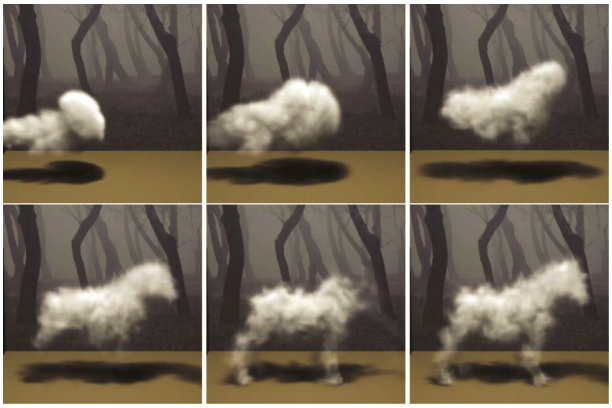

Figure 4: Horse shape matching animation from Shi and Yu

same visual fidelity of the Shi and Yu result. In terms of interactivity, however, our framework is far more suitable for games, or rapid prototyping before rendering a complex scene. Moreover, some difference in quality is due to our simple but efficient rendering algorithm, which could be replaced with a higher quality version if needed.

We present a sequence using an animated target mesh in Figure 5. Skeletal animation does not cause our simulation any significant performance overhead; this scene uses between 120-170 ms per frame with 400,000 marker particles. This scene demonstrates the windy, turbulent motion of our smoke, in contrast to the lazy, rolling smoke of Shi and Yu.

Figure 6 shows the target first taking the form of the letter psi, then shifting to omega. This experiment is similar to that of Fattal et al. [4]; we show theirs in Figure 7. Our result has the advantage of having continuous turbulent motion even after the smoke has reached the destination shape. Their simulation matches the shape border more accurately, and offers somewhat more detail, while our approach offers speed.

We used approximately 2 million marker particles for this test, 10,000 control particles and about 10,000 vortices. We take on average under 500 ms to simulate and render the morph scene per time step. We render the scene simultaneously on the GPU as we simulate the next frame; render time is always dominated by the simulation time. We list detailed timing numbers in Table 1. Our test machine has a 4 core 8 thread i7 CPU, with a GeForce 570 GPU. Fattal et al. report their simulation to take about 10.6 seconds per time step on a Pentium IV, with several time steps per frame, for a total time of 35 minutes of computation time per second of animation.

In contrast to prior work, our simulation steps are usually with linear, parallel operations. One of the most expensive steps is framework is our *NlogN* depth sorting of the particles for rendering; however, this is not part of our simulation time. Our particle evaluation and update are done with simple, parallel, additive SIMD operations.

We do not use expensive pressure solving steps characteristic of previous Eulerian methods. Moreover, previous particle-based methods often use spatial search structures such as uniform grids or KD-trees, while our operations demonstrate coherent, often linear memory access patterns. Our novel direct evaluation scheme for vortex particles with finite radius replaces the dozens, or more, of iterations in an Eulerian linear solver. Our interactive frame rate is not due simply to increases in computing power, but a novel combination of efficient simulation steps. For instance, Fattal et al. require 35 minutes per frame of animation, and if we are targeting a 30 FPS simulation, we require 15 seconds per second of animation

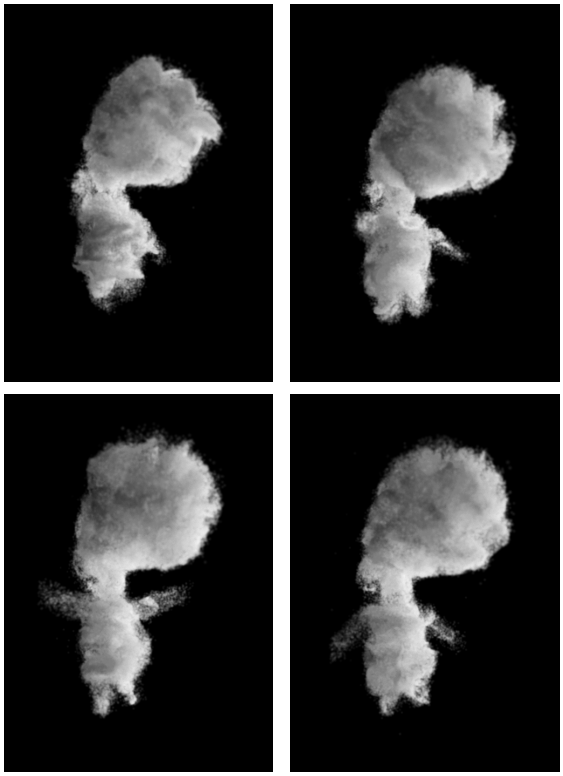

Figure 5: A smoke creature formed from an animated target mesh

Module	Time (ms)	Time (%)
Evaluate Marker Density	102	21%
Evaluate Control Particles	82.6	17%
Advect Marker Particles	117	24%
Redistribute Marker Particles	36.4	7%
Depth Sort	127	26%
All Other Operations	23	5%
Total Time	488	100%

Table 1: Timing results for the letter morph sequence

for our similar test. This is about a 140x speed increase; assuming single-threaded performance increase of 5x and a 4x improvement from threading, this is a relative increase of 7x. In comparison to Shi and Yu, we require about 4.5 seconds per second of animation, while they require roughly 375 seconds. Given a 20x increase in processing power, this is a relative improvement of about 4x.

Table 2 gives a breakdown of the key parameter values we use for the smoke creature sequence. Many parameter values work well for scenes of the same size. For instance, the size of the dynamically generated grids, such as the velocity grids, and particle potential fields, are fairly insensitive. However, too few cells will undersample thin, high-frequency features. Undersampling will cause target particles to miss these thin mesh features. Additionally, the number of target particles must be sufficient to resolve all areas of the target mesh.

It is crucial to choose an appropriate value for S, the global attraction strength parameter. An excessively high S will overpower the turbulent detail velocity; points will rush to their targets quickly. Conversely, with S too low, the smoke will be sluggish and will not be able to match a moving shape. Additionally C_{redist}, the number of redistributions, strongly affects the appearance of the scene. Finding a good attraction strength is a matter of lowering the

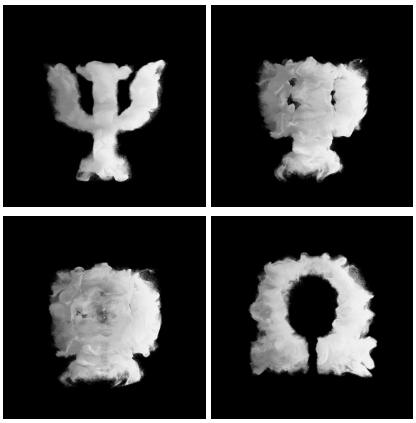

Figure 6: Morph sequence

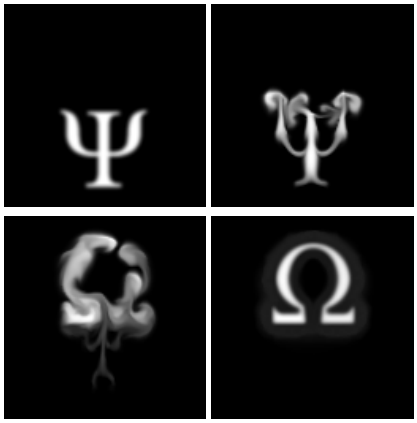

Figure 7: Morph sequence of Fattal et al.

Symbol	Parameter	Value
M	Number of marker points	400000
N	Number of control/target points	10000
r_m	Marker particle radius	0.01
r_c	Control particle radius	0.01
S_a	Global attraction strength	30
d	Attraction velocity damping	0.3
	Mean vortex radius	0.5
	Mean vortex magnitude	4
C_{redist}	Marker particle redistribution iterations	400000

Table 2: Key parameter values for the smoke creature scene

strength parameter if the points move too quickly, and vice-versa. The number of redistributions parameter can also be adjusted by inspection: choose an initial value, and increase if the smoke escapes the mesh too easily. These two sensitive parameters are worth noting for tuning.

Artists must choose radii for control and marker particles for evaluating particle potential fields (described in Section 4.1 and Section 4.4). Setting a narrow size for the control particles allows for sharp detail in the shape matching, and forces marker particles to areas closer to the mesh. Conversely, setting larger radii for control velocity produces a smooth control velocity field. Finding an appropriate size is easily done by adjusting the size gradually, as the simulation runs; first choose a large size and then reduce it until thin features are visible.

Target control in itself provides a tunable high-level parameter, eliminating the need for particle clouds or force fields. The typical workflow for creating a scene would largely be in choosing a target mesh, and using traditional animation tools to manipulate the mesh itself, rather than tuning individual control parameters. We gave some advice about how to tune the most sensitive parameters: particle size, attraction strength, and number of redistributions. Remaining parameters do not strongly affect the simulation and can be left at default settings.

9 Discussion and Future Work

We have presented a paper that provides interactive fluid control for artistic effect. Since our work is novel in that it allows artists real-time control over fluid simulation, we would like to design a test framework to solicit feedback from technical artists. We would like to investigate the benefits of target control and interactive control versus other prior control methods.

It is worth discussing briefly some reasons why our results may lack detail when compared to previous methods. Our focus on performance has led us to use a simplified control framework. We also prefer conservative simulation detail: our velocity fields are on the order of twenty thousand cells, particle counts in the low millions. In film production, artist may often employ particle counts in the hundreds of millions or more. Although our system is scalable, we choose to use particle counts in the millions, as real-time performance becomes infeasible with larger counts, particularly due to rendering times. More particles would, however, increase the quality of results.

One non-physical aspect to our work that could cause noticeable or undesirable behaviour is the non-negligible level of divergence in our control velocity field. Divergence is undesirable because it may cause the smoke to spread out, and later merge back together; this gives an uneven motion. In practice, we do not detect any visual artifacts due to divergence in the control field. Furthermore, the smoke is always guaranteed to arrive at the target exactly, and ensuring the target particles are spread out is enough to guarantee the resting position of the smoke will be spread out. A possible solution to the problem of intermediate divergence is to apply a pressure projection to the control velocity field; although simple to implement, the performance cost of adding a pressure solve would be significant, and lead to less interactive simulations.

We have specifically focused on a different style of animation than previous methods. Our simulations feature windy, highly energetic turbulence as opposed to the gentle motion in previous approaches. This is a deliberate choice, as it works well with our simple marker particle redistribution scheme. In our tests, when we lowered the speed of the smoke, the particles would not stray outside the boundaries of the target shape as quickly. Fewer particles outside the boundary meant our system would redistribute fewer smoke particles throughout the shape, and the simulation would appear clumpy and unbalanced. We do not simulate the buoyant, lazy smoke in the style of Fattal and Lischinski, instead offering faster and more energetic motion with shorter simulation times.

Additionally as we mentioned in Section 8, compared to other methods, our simulations do not match the target boundaries as precisely. Because we do not include boundary conditions, the shape edges are often chaotic and turbulent. In the future, we might investigate the advantages of implementing simple boundary conditions to contain the smoke is near to the target shape. One simple fast implementation of boundary conditions is to use a signed distance test to generate a repulsion force from nearby objects; this would not significantly impact performance, although it would add some divergence to the particle velocity fields, giving a less pleasing effect. Other techniques for implementing boundary condition such as the panel method, as described by Park and Kim [18], bear investigation as fast and high quality boundary effects.

Finally, the divide between base velocity and turbulent velocity sometimes causes an appearance of smoke motion without large-

scale turbulence. The base velocity is entirely turbulence-free, and when the simulation drives motion more from the base velocity than the turbulent velocity, it does not appear to be fluid-like. It may be interesting in future work to look into other directions to approach bulk velocity, such as using a hybrid method, by generating a bulk velocity field using a Eulerian approach similar to Fattal et al. [4], or by replacing the bulk velocity with some method of directed flow from a purely Lagrangian vortex particle frame; that is, to direct vortex particles using a modified vortex kernel or a carefully placed arrangement of vortices.

10 Conclusion

We have presented a method of target control using Lagrangian vortex particles, combined with a particle-based control velocity, tailored for smoke simulation. Our unique shape matching, combined with vortex fluid motion, gives an efficient basis for fluid control. Our specific contributions are this system of fluid target control derived from a separation of bulk and detail velocity that accurately matches a target shape; a novel a driving force from a correspondence between source and target particles, which is simple to implement, fast, and accurate; and an efficient direct velocity-from-vorticity computation for compact vortex kernels.

Our simple and effective methods of simulating smoke gives us a near-interactive simulation for smoke shape matching. We have attempted with this work to give artists and animators a fast control scheme with a high level of abstraction, when designing visual effects with a fluid simulation system. This is one type of application of fluid control; ultimately there are many possible such applications and this work is a small step towards complete artist control over fluid effects and other natural phenomena.

Acknowledgements

The authors wish to thank Mykola Konyk and Michael Gourlay for advice, suggestions and other assistance. This work received financial support from the Ontario Graduate Scholarship Program, NSERC, and Carleton University.

References

[1] A. Angelidis and F. Neyret. Simulation of smoke based on vortex filament primitives. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, number July, pages 87–96, New York, New York, USA, 2005. ACM.

[2] A. Angelidis, F. Neyret, K. Singh, and D. Nowrouzezahrai. A controllable, fast and stable basis for vortex based smoke simulation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 25–32. Eurographics Association, 2006.

[3] G.-H. Cottet and P. D. Koumoutsakos. *Vortex Methods: Theory and Practice*. Cambridge University Press, 2000.

[4] R. Fattal and D. Lischinski. Target-driven smoke animation. *ACM Transactions on Graphics*, 23(3):441–448, Aug. 2004.

[5] R. Fedkiw, J. Stam, and H. W. Jensen. Visual simulation of smoke. In *SIGGRAPH '01 Proceedings*, pages 15–22. ACM, 2001.

[6] N. Foster and R. Fedkiw. Practical animation of liquids. In *SIGGRAPH '01 Proceedings*, pages 23–30, New York, New York, USA, 2001. ACM Press.

[7] N. Foster and D. Metaxas. Realistic Animation of Liquids. *Graphical Models and Image Processing*, 58(5):471–483, Sept. 1996.

[8] N. Foster and D. Metaxas. Controlling fluid animation. In *Proceedings of CGI 1997*, pages 178–188. IEEE, 1997.

[9] W. F. Gates. *Interactive Flow Field Modeling for the Design and Control of Fluid Motion in Computer Animation*. PhD thesis, The University of California at Davis, 1994.

[10] M. J. Gourlay. Fluid Simulation for Video Games. `http://software.intel.com/en-us/articles/fluid-simulation-for-video-games-part-1`, 2012.

[11] S. E. Hieber and P. D. Koumoutsakos. A Lagrangian particle level set method. *Journal of Computational Physics*, 210(1):342–367, Nov. 2005.

[12] J.-m. Hong and C.-h. Kim. Controlling fluid animation with geometric potential. *Computer Animation and Virtual Worlds*, 15(34):147–157, July 2004.

[13] Intel. Intel threading buiding blocks. `http://threadingbuildingblocks.org/`, 2012.

[14] Y. Kim, R. Machiraju, and D. Thompson. Path-based control of smoke simulations. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 33–42. Eurographics Association, 2006.

[15] T. Lenaerts. *Unified Particle Simulations and Interactions in Computer Animation*. PhD thesis, Katholieke Universiteit Leuven, 2009.

[16] S. Limtrakul, W. Hantanong, P. Kanongchaiyos, and T. Nishita. Reviews on Physically Based Controllable Fluid Animation. *Engineering Journal*, 14(2):41–52, Apr. 2010.

[17] A. McNamara, A. Treuille, Z. Popović, and J. Stam. Fluid control using the adjoint method. *ACM Transactions on Graphics*, 23(3):449–456, 2004.

[18] S. I. Park and M. J. Kim. Vortex fluid for gaseous phenomena. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation - SCA '05*, number July, pages 261–270, New York, New York, USA, 2005. ACM Press.

[19] T. Pfaff, N. Thuerey, and M. Gross. Lagrangian vortex sheets for animating fluids. *ACM Transactions on Graphics (TOG)*, 31(4):112:1–112:8, 2012.

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

[21] A. Selle, N. Rasmussen, and R. Fedkiw. A vortex particle method for smoke, water and explosions. *ACM Transactions on Graphics*, 24(3):910–914, July 2005.

[22] D. Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, pages 517–524, New York, New York, USA, Jan. 1968. ACM Press.

[23] L. Shi and Y. Yu. Controllable smoke animation with guiding objects. *ACM Transactions on Graphics*, 24(1):140–164, Jan. 2005.

[24] L. Shi and Y. Yu. Taming liquids for rapidly changing targets. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, number July, pages 229–236, New York, New York, USA, 2005. ACM Press.

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

[26] M. Swoboda. Advanced Procedural Rendering in DirectX 11, 2011.

[27] M. Swoboda. numb res. `http://directtovideo.wordpress.com/2011/05/03/numb-res/`, 2011.

[28] N. Thürey, R. Keiser, M. Pauly, and U. Rüde. Detail-preserving fluid control. In *Proceedings of 2006 ACM SIGGRAPH/Eurographics*, pages 7–12. Eurographics Association, Nov. 2006.

[29] A. Treuille, A. McNamara, Z. Popović, and J. Stam. Keyframe control of smoke simulations. *ACM Transactions on Graphics*, 22(3):716—-723, July 2003.

[30] S. Trojansky. Raging waters: the rivergod of Narnia. In *ACM SIGGRAPH 2008 Talks*, pages 74:1–74:1, 2008.

[31] S. Weiß mann and U. Pinkall. Filament-based smoke with vortex shedding and variational reconnection. *ACM Transactions on Graphics (TOG)*, 29(4):115:1–115:12, 2010.