2017-10-01

Time Reversed Smoke Simulation

Jeremy Michael Oborn
Brigham Young University

Time Reversed Smoke Simulation

Jeremy Michael Oborn

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Seth Holladay, Chair
Parris Egbert
David Wingate

Department of Computer Science

Brigham Young University

ABSTRACT

Time Reversed Smoke Simulation

Jeremy Michael Oborn
Department of Computer Science, BYU
Master of Science

Physics-based fluid simulation often produces unpredictable behavior that is difficult for artists to control. We present a new method for art directing smoke animation using time reversed simulation. Given a final fluid configuration, our method steps backward in time generating a sequence that, when played forward, is visually similar to traditional forward simulations. This will give artists better control by allowing them to start from any timestep of the simulation. We address a number of challenges associated with time reversal including generating a believable final configuration and reversing entropy.

Keywords: Fluid Simulation, Time Reversal, Texture Synthesis

ACKNOWLEDGMENTS

Thanks go to Seth Holladay, Parris Egbert, Sean Flynn and Bryan Morse for their input and ideas. I'd also like to thank my family for all their support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Visual effects and animation play a vital role in the entertainment of today. Movies, TV shows, games, and advertisements are all made possible by armies of animators. In the pursuit of imitating reality, these artists need to be able to depict a wide range of different types of fluid phenomena such as water, smoke, fire, and clouds. As such, fluid simulation has become a well-established tool in visual effects and animation pipelines. Fluid simulators have been used in practically every modern animated feature and visual effects blockbuster. They give artists a reliable way to create complex, photorealistic fluid motion without having to hand-craft every frame of the animation, which is a process that would be beyond the scope of any realistic production budget (see Figure 1.2).

A typical fluid simulation tool requires the fluid to be set to an initial state at the start of the simulation, and then steps forward in time, calculating results at fraction-of-a-second intervals. Once the simulation starts, artists have no control over what it does. To change the fluid's behavior or address feedback, they must adjust the initial state and then re-simulate to see what affect their adjustments had on the shape of the fluid at later timesteps. A single iteration of this process can take hours, meaning that a full cycle of addressing artistic feedback can stretch over days or weeks. While this process has allowed artists to achieve tremendous amounts of detail on enormous scales, it also restricts how much influence they have on the shape and behavior of the final result.

Art direction of fluid simulation often demands that the simulated fluid fill a very specific shape at some timestep in the middle of the simulation in order to create an appealing

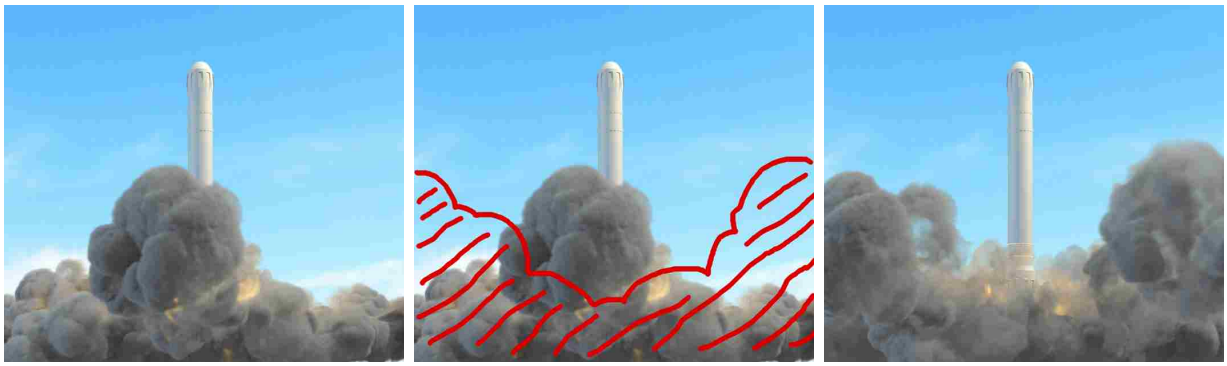

Figure 1.1: An example of the iteration process for art-directing fluid simulations. A director will often have very specific feedback about the shape of the fluid at a certain timestep. The artist must address this by modifying the initial fluid configuration and re-simulating until it matches as closely as possible.

composition (see Figure 1.1). However, this is a very expensive, time-consuming process involving many iterations of careful adjustments to the initial state. Previous methods have attempted to address this problem by manually placing control forces calculated from a set of user-defined guides in the form of meshes or control particles. While they do give artists more control over the simulation, they tend to cancel out the natural motion that defines the material as a fluid. Attempts to counteract this by relaxing the guide forces only re-introduce unpredictable behavior to the simulation.

In contrast to other computational fluid dynamics research, our primary goal in computer graphics is not precise physical accuracy. Instead, we pursue the much more ambiguous quality of believability. This can be very subjective. However, when dealing with common natural phenomenon such as smoke, it is easier to verify than one might suppose. Is the phenomena quickly and easily recognizable? Would it be unintentionally distracting in the background of a film or tv show? Does it achieve the artist's goals? These are the kinds of questions we consider to determine believability.

We propose a system that allows artists to specify a fluid state at any time of the simulation, as opposed to being restricted to the very beginning. Our system can then solve both forward and backward from that point. The most important visual moment of any

fluid simulation is usually towards the middle or end of the sequence and not at the very beginning. This will allow artists to achieve the exact desired shape at the most meaningful point in the simulation, without compromising the natural motion of the fluid.

Our research describes a fluid simulation process that steps backward in time. There are a number of problems we must overcome to achieve this. First, we must generate a configuration that appears to have been the result of a forward simulation. We describe a texture synthesis method that can generate such a state from a simulated example state. Second, reverse simulation also produces results where entropy is incorrectly decreasing. We prevent this from happening with a self-attraction force and a modified divergence constraint. We also describe a method for credibly reversing density dissipation.

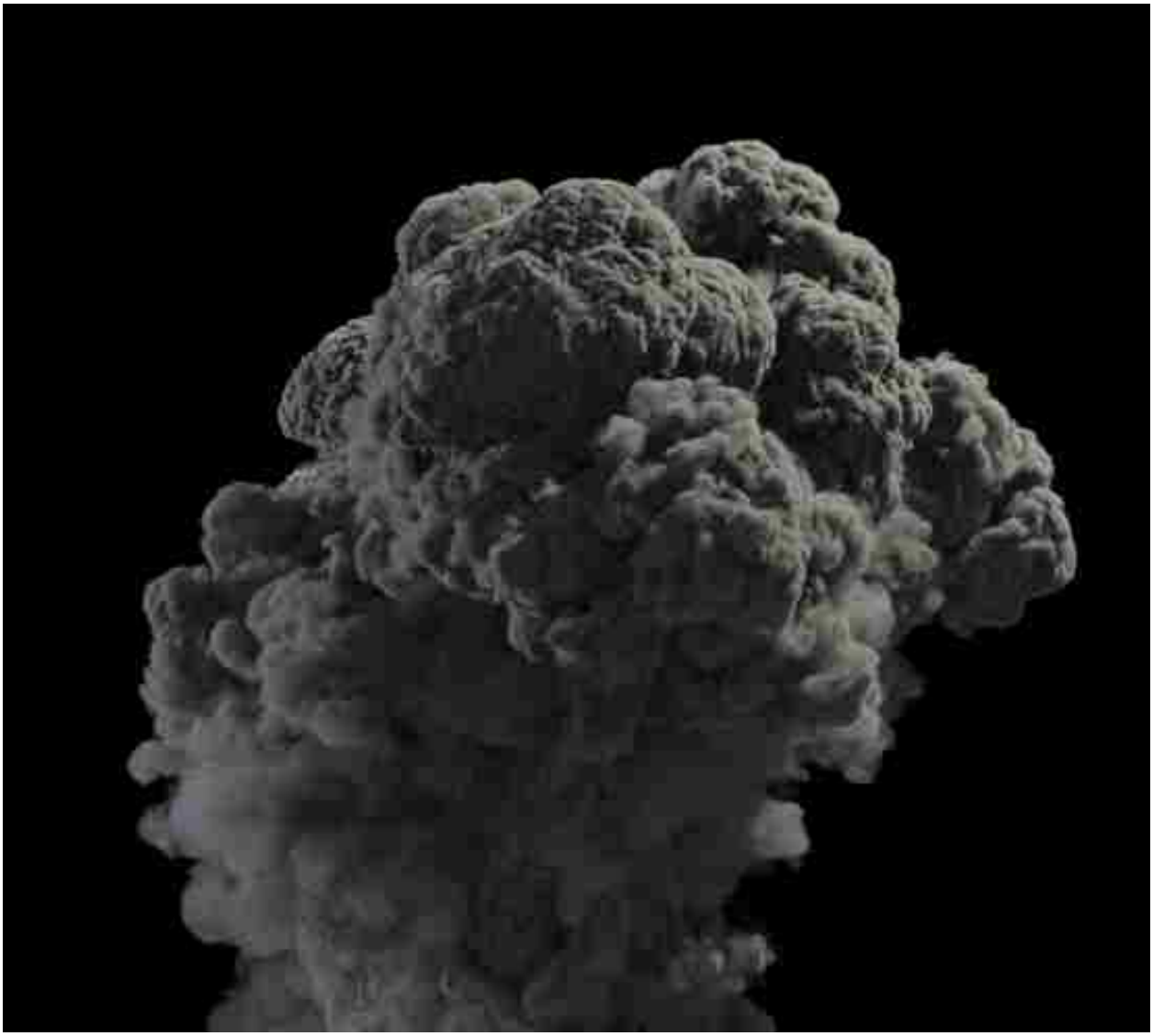

Figure 1.2: A high resolution smoke simulation

Chapter 2

Related Work

Robust 3-dimensional fluid simulation for computer graphics was first made possible by the work of Stam [23] using an unconditionally stable semi-Lagrangian integration scheme. Fedkiw et al. [7] improved upon that method by adapting vorticity confinement from engineering models in order to reduce numerical dissipation. Foster and Fedkiw [9] introduced a particle level set method for modeling liquid surfaces. We will use their technique for density correction made necessary by our self-attraction force. Enright et al. [5] extended this approach with a thickened surface tracking technique that was critical for the feature film Shrek. Zhu and Bridson [32] introduced a particle/grid hybrid approach that performs Lagrangian advection on the particles and non-divergent projection on the grid. Since the introduction of these methods, their practical use for animation and visual effects has been tremendous. However, their inherent unpredictability has motivated years of research in efficient control techniques.

The idea of fluid control was first introduced by Foster and Metaxas [10] with the concept of embedded controllers. These controllers are functions of time and space, e.g. an expanding sphere for explosions. The amount of control is limited by the user's ability to write a function describing the motion they desire.

Much of the research on this topic has focused on generating guide forces from target shapes. Treuille et al. [26] introduced the concept of smoke density targets. Their method requires an additional optimization step to solve for the control forces that would result in those targets. McNamara et al. [13] augmented this method by adapting the adjoint method

from optimal control theory to calculate derivatives. Recent work by Pan and Manocha [17] further improved this method with a number of optimizations using a spacetime multigrid solver. These approaches are computationally expensive, requiring a small number of control forces and coarse simulation grids. Fattal and Lischinski [6] introduced a much faster method by using a driving force term to hit target shapes, and a smoke gathering term to prevent diffusion. The driving force is defined as the gradient of a specially blurred version of the target smoke density. These methods exhibit the behavior of an unseen force morphing the fluid into a target, which can look unnatural. With time reversed simulation, the fluid will appear to more naturally flow into the desired shape in a way that seems unintentional.

Raveendran et al. [20] use meshes as inputs for their guide forces. They create volume preserving morphs between shapes by minimizing a linear system. They try to maintain fluid momentum by allowing for relaxed control, reintroducing unpredictability into the simulation. Stomakhin and Selle [24] introduced the Fluxed Animated Boundary method which enforces boundary conditions of a user-defined control shape and material flow field. Pan et al. [18] completely discard the idea of global guide shapes, and instead allow local editing of fluid motion by drawing curves or placing meshes. However, global guide shapes are necessary in order to achieve certain types of artistic objectives involving the overall silhouette of the fluid. Our goal is to make these global guide shapes work.

Other control techniques are built around the idea of using simpler simulation methods to generate control particles, which are then used to influence the fluid simulation. Foster and Fedkiw [9] first proposed the idea of control particles as a way to introduce user-defined body forces. This was later expanded by Rasmussen et al. [19] to allow separate control particles for velocity, viscosity, divergence and the level set. Thurey et al. [25] introduced a method to preserve detail in the presence of control particles by decomposing low and high frequency velocity and only applying control forces to the low-frequency part. Nielsen and Bridson [16] proposed a process that uses a low resolution simulation (which is faster to compute and easier to control) as a guide shape for a higher resolution simulation. This maintains

the general shape and motion of the low resolution version, while adding high resolution detail. Raveendran et al. [21] introduce a system that blends the results of two entirely separate simulations. While still an improvement upon default simulation techniques, these methods are all inherently limited by the amount of control an artist can achieve with a low resolution fluid simulation or particle system—manipulation is only possible through careful adjustments of the initial state. Achieving an exact shape in the middle of the simulation is still difficult and time-consuming.

Duponcheel et al. [3] explored the time reversibility of the Euler equations as an accuracy benchmark for fluid solvers and showed that an energy-preserving integration scheme and accurate time-stepping are important for time reversibility. One such integrator was introduced by Mullen et al. [15] which is perfectly energy preserving and can be made time reversible. Liu et al. [11] introduced a model-reduced variational Eulerian integrator that is also time reversible. However, we did not use these methods as we were able to achieve plausible results with the modified MacCormack method described in Selle et al. [22] with second order accuracy.

Twigg and James [27] introduced a time reversed rigid body simulator. Their method will influence our approach, especially when it comes to choosing the most visually believable solution among potentially infinite correct possibilities. Their work explores this in the case of friction—when an object is at rest, how did it get there? Did it slide, skid, or bounce? They overcome this by introducing elements of user control and using Markov models of data from forward simulations. We will adapt elements of this approach in the case of dissipation for fluids. They also bring up the challenge of entropy loss—reverse simulation can look unnatural because it creates situations where, when played forward, entropy is decreasing. Their solution for rigid bodies is to simply jitter the starting configuration of the backwards simulator. We found that this was insufficient for fluid flow.

In order to generate a believable fluid state for our reverse simulator to start from, we make use of texture synthesis by sampling Markov Random Fields as described in Efros and

Leung [4] and Wei and Levoy [28]. Our shape map approach bears resemblance to texton maps from Zhang et al. [30] and feature maps from Wu and Yu [29].

We also make use of level set methods, or signed distance fields as they are sometimes called, to track the surface of the fluid. We calculate level sets using the fast sweeping method described by Zhao [31]. Level sets were first used in fluid simulation for computer graphics by Foster and Fedkiw [9]. Each grid cell in a level set structure contains the distance to the closest point on the fluid surface. This data structure give us a spatial context that proves invaluable for generating fluid configurations that can step backward in time.

There has been a lot of promising research in fluid simulation control. Recent research focuses mainly on directing fluids with more efficient simulation techniques. These lower resolution simulations allow artists to iterate more quickly, however they can still only be modified through adjustments to the initial state. Global guide shape techniques show promise, but have high computational costs and produce unrealistic fluid behavior. Our solution, time reversed simulation, solves these problems by allowing the user to artistically control an arbitrary point in the simulation. This results in animations that are easily art directable, produce realistic results, and are computationally efficient.

Chapter 3

Time Reversed Simulation

3.1 Thesis Statement

Time reversed fluid simulation, facilitated by entropy reversal and texture synthesis, will allow artists to achieve an exact, art-directed shape at any desired timestep.

3.2 Overview

Our time reversible simulator is an extension of the popular semi-Lagrangian smoke simulator introduced by Stam [23]. The traditional forward simulator requires an initial state at time t_0 and advances each timestep, t_i, according to:

$$t_{i+1} = t_i + \Delta t \tag{3.1}$$

Where i is any given timestep between 0 and n. Our reverse simulator requires an initial state at time t_n and advances a given timestep, t_i, in the reverse direction by:

$$t_{i-1} = t_i - \Delta t \tag{3.2}$$

The result of this time reversed process is intended to be presented advancing forward in time. The fluid state at time t_n can also be passed into a forward simulator and the results of the separate simulations can be concatenated in sequence as a single, continuous flow. The

result will be presented as follows:

$$t_0 \ldots t_n \ldots t_f$$

Where t_0 is the initial timestep, t_n is the art directed timestep, and t_f is the final timestep of the sequence. The sequence from t_0 to t_n is generated by our reverse simulator. We will show that stepping backwards in time is not trivial, and requires additional steps in the algorithm.

Algorithm 1 Time Reversed Fluid Simulation Algorithm

1: Generate end state (section 3.3)
2: **for** $i = n$ to 0 **do**
3: External forces
4: Self-attraction force (section 3.4.2)
5: Backwards dissipation (section 3.5)
6: Modified divergence projection (section 3.4.3)
7: Advection
8: Density correction (section 3.4.3)
9: **end for**

3.3 "Initial" End State

Our simulator will accept a polygonal mesh as input for the fluid state at time t_n, which will allow artists to take advantage of the many robust sculpting and modeling tools available. We need to generate an "initial" end state from this mesh for our backwards simulator to start from. This state is chosen by the artist and can be defined at any timestep, t_n, of the simulation, although from here on we will refer to it as the "end" state. The end state must be generated in such a way that it resembles a configuration that could have feasibly appeared in the middle of a forward simulation. A uniform volume can look unnatural in the middle of a simulation–chaotic fluid motion doesn't just happen to flow into a perfect uniformly distributed density. In some cases this may be artistically desired behavior in order to exaggerate the abnormality of the motion. However, we also want to provide the flexibility for situations where this is not the case. Using a noise function to break up the texture isn't a

Figure 3.1: Examples of texture synthesis. Given a finite sample from an example, the goal is to synthesize other samples from the same texture.

sufficient solution. The visual quality of fluid is closely related to the simulation parameters, which are hard to correlate with a noise function. The spatial variation of density in a fluid during the simulation will not match an arbitrary noise function applied to the end state without incessantly tweaking parameters–one of the main things we are trying to eliminate with our solution–and even then it won't be an exact match. Twigg and James [27] discuss a number of limitations for time reversed simulation that are caused by an end state that is not the result of forward simulation. We attempt to mitigate these limitations by generating an end state with as many properties of an actual simulated fluid state as possible.

3.3.1 Texture synthesis

In order to generate this end state with fluid qualities, we make use of texture synthesis methods. The goal of texture synthesis is to generate novel samples of a texture based on a given finite example (see Figure 3.1). Adapting this approach for fluid simulation will require an additional example fluid input to our algorithm. This example fluid state should be the result of a forward simulator run for n timesteps with the same simulation parameters that will be used on the backward simulator. We introduce a texture synthesis method to generate a unique end state for our backward simulator with the shape of the given target mesh and similar visual qualities to the given example state.

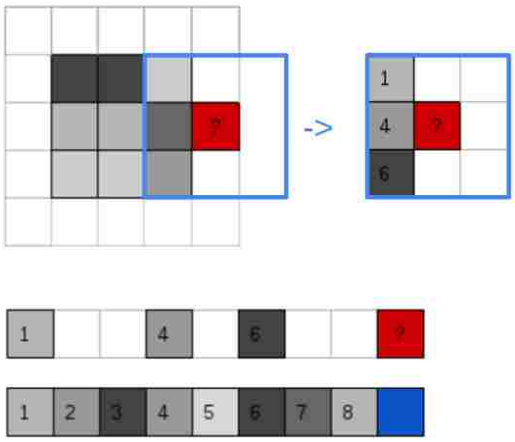

Figure 3.2: Texture synthesis uses a variation of the k-nearest neighbors algorithm. To synthesize cell i, we search through all cells j in the example state where $c_i = c_j \pm 1$ and set cell i equal to a random sample from the set of cells j with similar neighborhoods.

We use example-based texture synthesis methods as described in Efros and Leung [4] and Wei and Levoy [28]. These methods work by synthesizing one pixel at a time from an initial seed. The algorithm searches for pixels in the given example texture that have similar neighborhoods and samples from these (see Figure 3.2). This process is effective for synthesizing homogeneous textures; however, they suffer from poor spatial awareness of larger scale texture variation. This is a problem in our case because a simulated fluid state usually does not have homogeneous texture, but instead has wide variations at large scales. We need these texture variations to match up with the target shape in the same way that they did on the example shape.

To combat this lack of spatial awareness, Zhang et al. [30] employ user-defined texton maps that designate the prominent texture elements in the example texture. This allows them to synthesize textures with varying scale and direction while allowing users to control exactly how those features vary. Wu and Yu [29] extended this idea in the form of automatically generated feature maps in order to capture high level structural information. In these approaches, only samples from the example texture that have the same value in

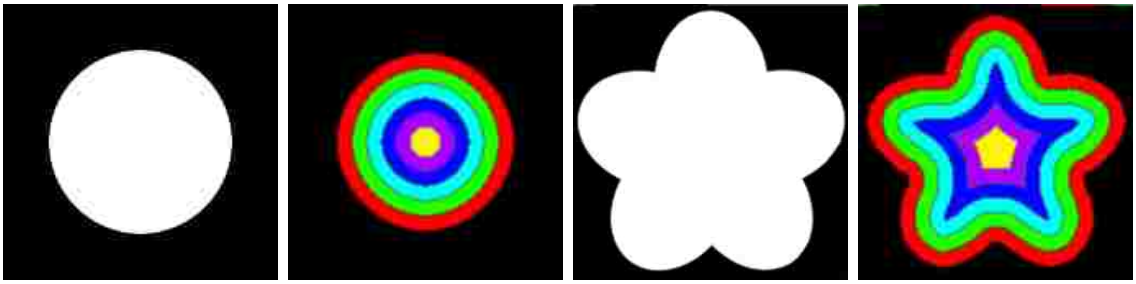

Figure 3.3: The shape map provides a way to correlate location between the target and example shapes. As we iterate through each cell in the example, we calculate it's shape map value and only use samples in the example shape with similar values.

the map as the pixel being synthesized are used. We employ this idea in the form of a 3-dimensional shape map to more accurately correlate shape and texture features.

A shape map is generated from the level set of its given shape. We generate the level set using the fast sweeping method described by Zhao [31], and use the result to populate the shape map. The value at each cell of the shape map is given as follows:

$$c_i = \lfloor \phi_i / \delta \rfloor \tag{3.3}$$

Where ϕ_i is the level set value at cell i and δ is a constant describing how many cells wide each band of the shape map will be. We found this generally worked best with values of δ between 3 and 5. During synthesis, we only look for matches in the example with a value of $c_i \pm 1$. This means we are essentially searching through 3 bands of the shape map at a time. We found that limiting the search to 1 band sometimes gave bad results due to not having enough samples.

The synthesis algorithm proceeds by calculating the shape map values and neighborhood feature vector of each cell in the example state. This is stored in a 3-dimensional jagged array, where the first dimension index is the shape map value. Each index holds the neighborhood feature vectors of all the samples in the example with that shape map value. After this pre-computation step, we iterate through the target shape, synthesizing one cell at

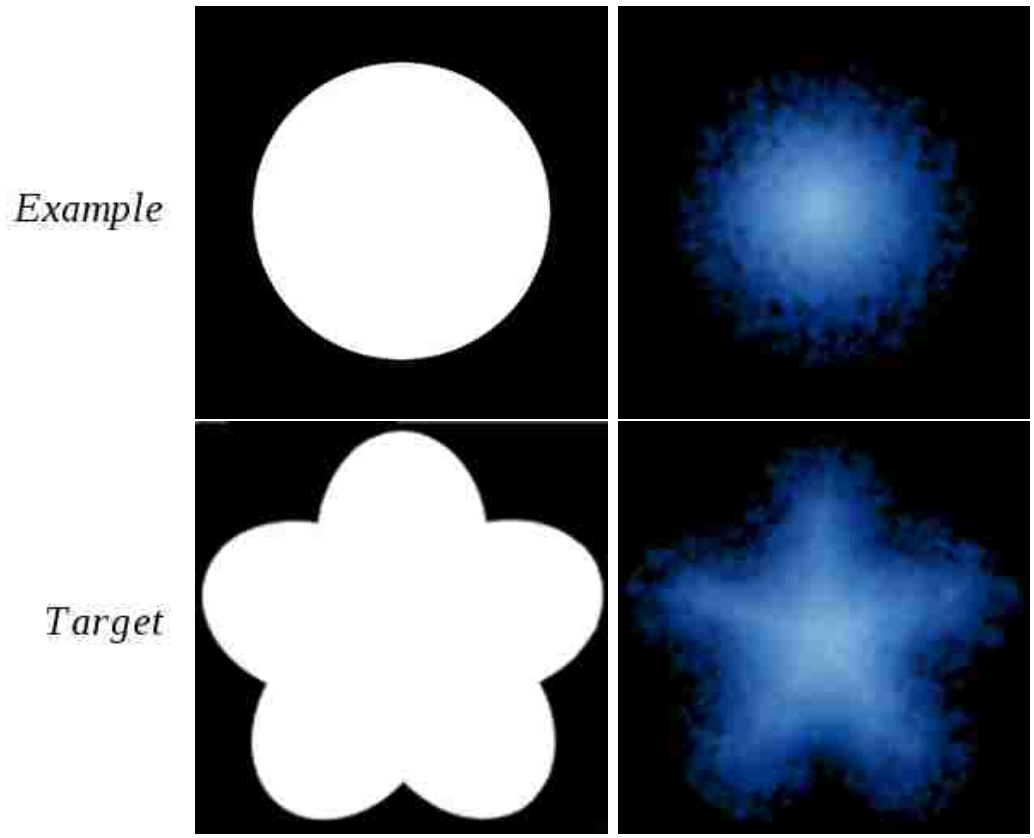

Figure 3.4: A 2-dimensional texture synthesis example with our shape map approach. With the example shape (top-left), example texture (top-right), and target shape (bottom-left) as input, the algorithm fits the example texture to the target shape (bottom-right)

a time. For each synthesis step, we calculate an incomplete neighborhood feature vector and a shape map value c_i. We then search through the example state array to find close matches we can sample from. However, we narrow our search down to only 3 rows of the example state array—those with the following indices: $[c_i - 1, c_i, c_i + 1]$. This ensures that we get a kind of correspondence between the two shapes based on level set values (see Figure 3.3).

This allows the synthesis algorithm to use information about larger scale texture variation and ensures that texture features from the example state will show up at the same depth in the target shape. As you can see from the 2-dimensional example in Figure 3.4, the example texture in the original shape has been fit to the new target shape. We use this method to synthesize the density, velocity, and temperature fields of our fluid end state.

We also experimented with a simple pre-roll technique to generate the end state due to the fact that our texture synthesis approach is very computationally expensive. We input the target shape as the initial state to the forward simulator and run it for a small number of timesteps (between 5 and 10). The goal is to let it run long enough to develop texture features, but not so long that it significantly distorts the target shape. The result of this short simulation will then be used as the end state for the reverse simulation. This is a much faster process, and although it will result in a slightly distorted target shape, in many cases it is not an issue. If art-direction requires matching the exact target shape, the texture synthesis method should be utilized.

3.4 Entropy Reversal

3.4.1 Reversibility Paradox

The incompressible Euler equations are time-reversible:

$$\nabla \cdot \mathbf{u} = 0 \tag{3.4}$$

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\rho}\nabla p \tag{3.5}$$

Duponcheel et al. [3] showed that this property is preserved in fluid solvers that have an energy-preserving integration scheme and accurate time-stepping. Such a solver can be made to step backwards in time by simply reversing the velocity field and running the simulator as usual. Reversing a velocity field that is the result of several timesteps of a forward simulation and running the accurate solver for the same number of timesteps will produce visually plausible behavior, and, if it is accurate enough, can even recover the initial state. However, this requires some assumptions about the initial conditions of the simulator—namely, an end state that is the result of a forward simulation. In our case we will have no such end state—we define one based on the user's input target shape as described in section 3.3. Without such a state, the reverse simulator does not follow the second law of thermodynamics, which states that entropy must always increase. This becomes obvious when the reverse simulation is played forward—disorder grows into order and the motion feels unnatural. The reversibility paradox posed by Loschmidt in 1876 [12] has prompted many theories addressing the incompatibility between reversible dynamics and irreversible processes. For our purposes, we found that the most useful explanation of the nature of this conflict is due to "an explicitly asymmetric assumption about the way in which real systems were formed in the first place" [2]. Because we are essentially reversing this assumption by starting from a user-defined end state, we will need to introduce a constraint outside of the dynamics in order to achieve visually plausible reversed behavior.

3.4.2 Self-Attraction Force

We need to force our reverse simulation to decrease in entropy as it progresses. Inspired by simulations of the gravitational collapse of massive interstellar clouds such as Monaghan and Lattanzio [14], we use a self-attraction force similar to Newtonian gravity.

$$F_e = \frac{Em_1m_2}{r^2} \tag{3.6}$$

We replace the gravitational constant G with a (generally much larger) user-defined constant E that describes how marked the entropy reduction will be. Accurately calculating this force for each cell requires $O(h^2)$ calculations where h is the number of fluid cells.

$$\mathbf{f}_e^i = \sum_{j=1}^{h} \frac{Em_im_j(\mathbf{r}_j - \mathbf{r}_i)}{||\mathbf{r}_j - \mathbf{r}_i||^3} \tag{3.7}$$

We can reduce it to $O(h * log(h))$ by approximating it using a Barnes-Hut tree [1]. However, even with the approximation, this step is still a computational bottleneck. We experimented with setting a max traversal depth for the tree and found that in most cases a depth of 1 is all that is needed for the purpose of reversing entropy. In this case, the self-attraction force is reduced to a center of mass calculation $\mathbf{R}$:

$$\mathbf{f}_e^i = \frac{Em_im_{total}(\mathbf{R} - \mathbf{r}_i)}{||\mathbf{R} - \mathbf{r}_i||^3} \tag{3.8}$$

We found that that this force was very effective at small magnitudes. Adding limited amounts of energy back into the system can actually improve the believability of the fluid flow. This is because, in the forward direction, energy dissipates due to viscosity. In the backward direction, introducing small amounts of energy can have the effect of approximating viscosity.

This force alone will cause the fluid to contract as the reverse simulation progresses. However, there is a limit to how much it can decrease entropy due to the incompressibility constraint. This constraint is an important component of the Navier-Stokes equations which

requires the fluid velocity field to be divergence free. It enforces mass conservation and is vital to achieving believable fluid flow. However, we relax this constraint in order to decrease entropy.

3.4.3 Modified Divergence Projection

Allowing the gas to compress will facilitate a more dramatic decrease in entropy. It will also have the positive side effect of reversing gaseous free expansion, which is the process by which gas expands to fill its container. During free expansion, the density of the gas decreases as it takes up more space. In reverse, we want the density to increase as it takes up less space. We accomplish this with the method introduced by Feldman et al. [8] of enforcing non zero divergence to modify fluid behavior. In their case they enforce a positive divergence in order to cause rapid expansion for explosions. In our case, we enforce a negative divergence which will cause the fluid to contract. For traditional incompressible fluids we force the velocity to be divergence free.

$$\nabla \cdot u = 0 \tag{3.9}$$

This is done by solving a Poisson equation for a scalar pressure field.

$$\nabla^2 p = \nabla \cdot u_i \tag{3.10}$$

$$u_{i+1} = u_i - \nabla p \tag{3.11}$$

In order to allow the fluid to compress to reduce entropy, we force the divergence of the fluid velocity to be equal to the divergence of the self-attraction force F_e (equation 3.8).

$$\nabla \cdot u = \nabla \cdot F_e \tag{3.12}$$

We solve by making the following change to equation 3.10.

$$\nabla^2 p = \nabla \cdot u - \nabla \cdot F_e \tag{3.13}$$

Modifying the divergence in this way allows the fluid to compress at most to the extent defined by the magnitude of the self-attraction force. This has the potential to hurt believability, as a very large self-attraction force will cause the smoke to diminish rapidly and compress into a singularity. However, we found that the force was effective at small magnitudes. Figure 3.5 shows such a case.

At this point we have caused the gas to take up less space. However, density hasn't changed, which violates conservation of mass. In order to avoid this, we need to correct for the density that is lost during semi-lagrangian advection. We do this with the hybrid particle level set approach introduced by Foster and Fedkiw [9]. Their method tracks particles through the voxel grid and after advection uses the particles to udpate the level set. In our case we apply this to the density field. Before the advection step, we scatter 8 particles in each cell with nonzero density and assign them a value calculated from the density field. These particles are advected through the velocity field alongside the rest of the fields on the grid. After advection, we correct the density field anywhere mass was lost by comparing with the particle density values.

3.5 Dissipation

In fluid simulation we often want to simulate dissipation, especially when we are modeling steam or mist. Dissipation creates the effect of the smoke gradually disappearing. We describe a process for creating this effect in a time reversed simulation. Dissipation is modeled in the forward process by the following equation:

$$\rho_{t+1} = max(0, \rho_t - k_d \Delta t) \tag{3.14}$$

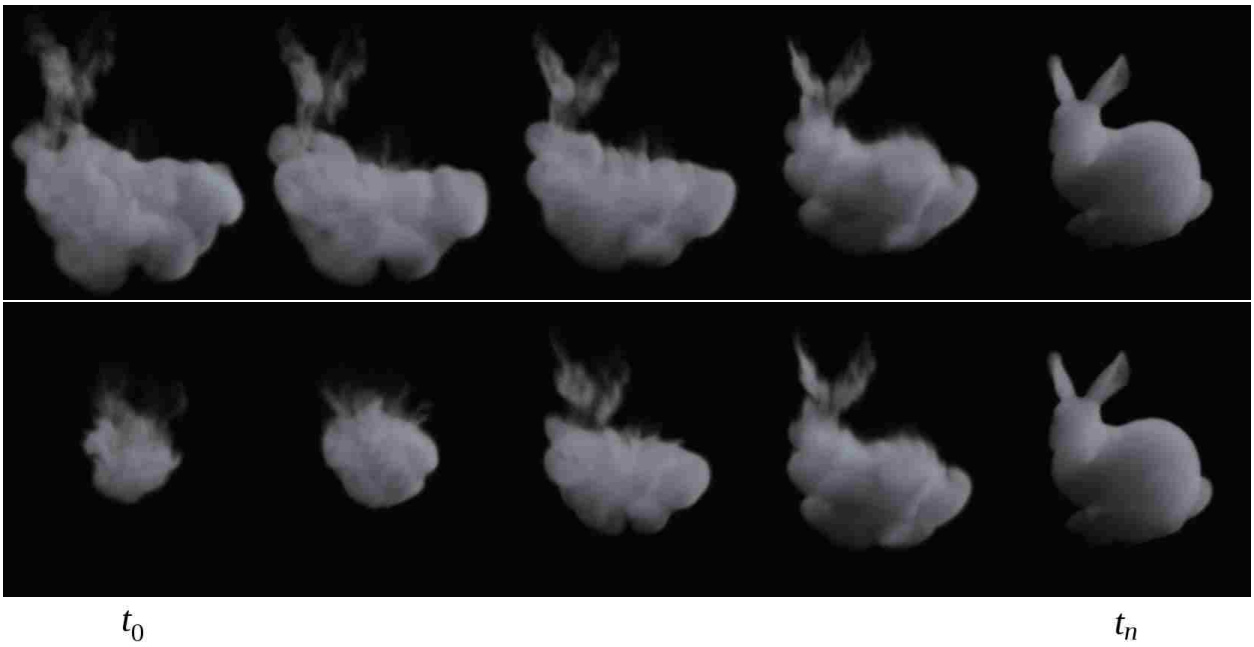

Figure 3.5: An example of a time reversed simulation with (bottom) and without (top) entropy reduction. Both were simulated backwards in time, but here we present them forward in time (as an audience will view them).

Where k_d is the dissipation constant. For cells with nonzero density, the reversal is trivial:

$$\rho_{t-1} = \rho_t + k_d \Delta t \tag{3.15}$$

However for cells with zero density there are an infinite number of possible results. Given an empty cell at time t_i, it could have any density value in the range $[0, k_d \Delta t]$ at time t_{i-1}. We need to be able to choose a reasonable value based on context. One option is to sample data from forward simulations in the manner that was outlined for the reverse rigid body simulator described by Twigg and James [27] in order to recover energy that has dissipated in the form of heat. We can do this for density dissipation by leveraging the example fluid state used to synthesize the end state in Section 3.3. If the example state were generated by a forward simulator that didn't discard dissipated density, we could sample that data to make reasonable guesses about where to add density back in. We do this by redefining

dissipation in the forward process as follows.

$$\rho_{t+1} = \rho_t - k_d \Delta t \tag{3.16}$$

Where ρ is initialized to $-k_d \Delta t$. This will introduce negative values into our density field. We discard values less than zero for rendering purposes, but we will use them in the synthesis stage to generate the end state density field. We do this by simply passing our density field with negative values as the example state input to the synthesis method described in Section 3.3. With a density field defined in this manner, reverse dissipation can be calculated by equation 3.15 from above.

In the case where the example state doesn't have discarded density data, we found that a fairly simple solution also achieves good results. We set the initial density values outside the target shape based on the following equation

$$\rho_0 = -\beta \phi_i + \alpha \psi(i) \tag{3.17}$$

where $\psi()$ is a noise function, ϕ_i is the level set value for cell i and β and α are user-adjustable parameters. α defines how much noise to add while β determines how quickly the shape of the fluid shrinks due to dissipation. With $\alpha = 0$ There will be no random variation and density will dissipate uniformly in all directions. High values of β cause the fluid to shrink rapidly while low values cause it to shrink slowly.

Chapter 4

Results and Discussion

We present a number of believable results of time reversed simulation in action. In all of our examples the defining timestep was determined to lie somewhere in the middle of the simulation. The end state for this timestep was generated from a user-defined target mesh. The starting timestep was determined to be of less importance—therefore, the first timesteps of the simulation are not constrained to any specific shape or target.

Figure 3.5 shows the results of a time reversed simulation (presented forward in time) of smoke forming the shape of a bunny with and without the self-attraction force. As can be seen, entropy is clearly decreasing in the wrong direction of time in the top sequence. Our entropy constraint improves this significantly. The bottom sequence was simulated with the max traversal depth of the Barnes-Hut tree set to 0, essentially only adding a center of mass calculation (see timing data in Table 4.1).

Figure 4.1 shows another simulation with the target bunny shape. This case illustrates how our texture synthesis approach can improve believability for a reverse simulation. Shown are two time reversed simulations which both utilize the self-attraction force, along with

	Reverse	Forward
bunny (Fig 3.5)	0.889	0.717
bunny (Fig 4.1)	0.0175	0.0168
cougars (Fig 4.2)	6.247	5.716
hand (Fig 4.3)	15.865	2.239

Table 4.1: Comparison of average simulation times for each of our results in seconds per frame. The corresponding forward simulations were set up with the final state of the reverse simulation as the initial state.

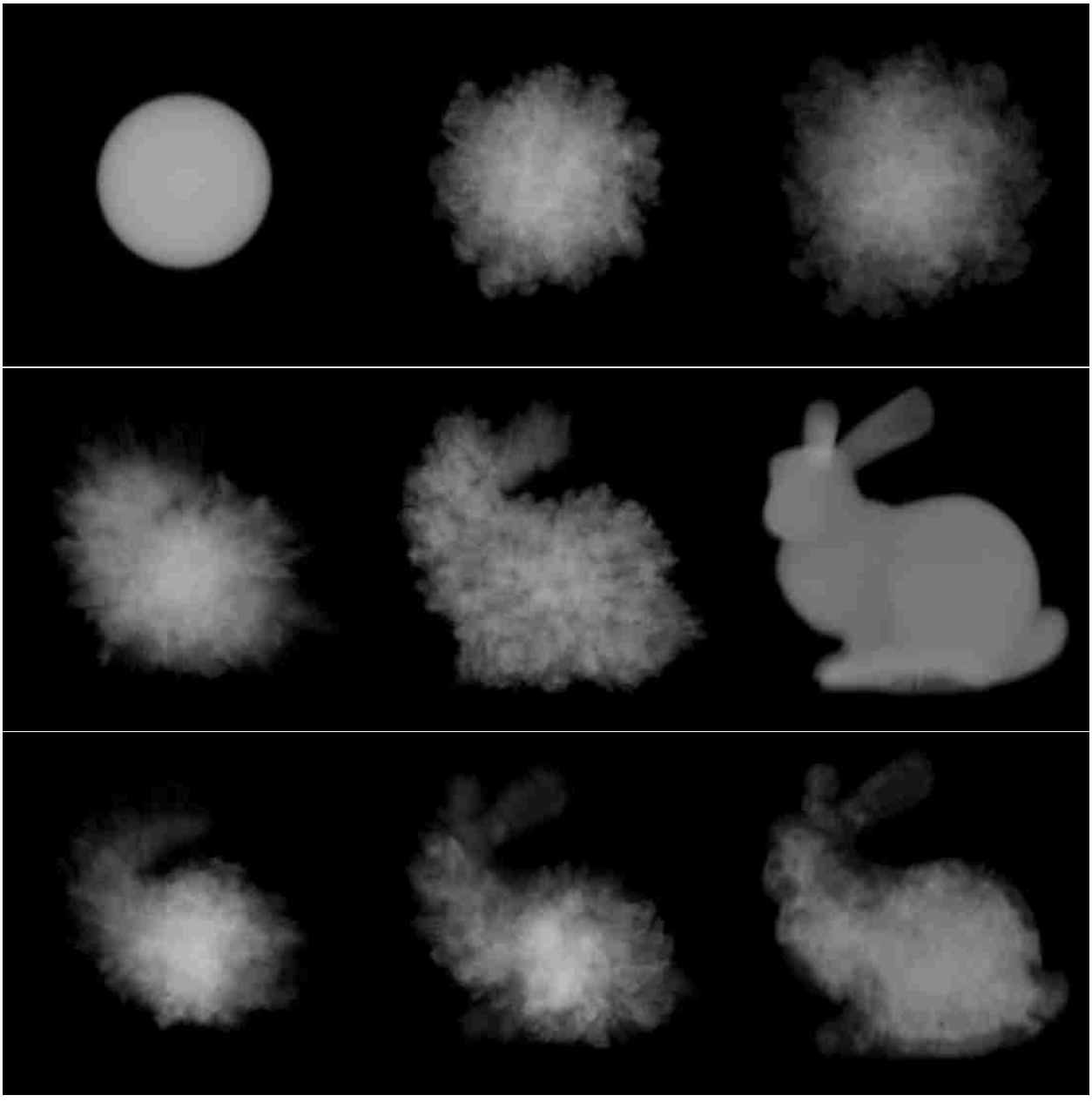

Figure 4.1: An example of our texture synthesis method applied to a fluid simulation. The top sequence shows the example forward simulation. The middle sequence is a reverse simulation without texture synthesis. The bottom sequence is a reverse simulation with an end state that was synthesized from the last timestep of the example.

the example state source simulation (top). The middle sequence does not use our texture synthesis method, while the bottom sequence does. With an end state that is a solid shape the motion can feel unnatural as the fluid flows into a tightly organized form. With our method the end state has a much more natural, organic quality and the texture features match those of the fluid simulation parameters. The drawback is that this method is very slow, especially in 3 dimensions. It involves a heavy, one-time calculation of the end state, but requires no additional computation after the simulation starts.

Figure 4.2 shows an application of time reversed dissipation. A cloud of smoke gradually dissipates to form words. This example was generated using our simplified reverse dissipation method using the target mesh level set values, which is an inconsequential addition to runtime. This example demonstrates our time reversed dissipation technique in isolation—the self-attraction force and texture synthesis methods were not used.

Figure 4.3 shows a large scale example using our self-attraction force. In this case, the force was calculated using the full Barnes-Hut tree, which increased computation time noticeably (see Table 4.1). Using the full tree improved the result for this broad, complex shape. With the force acting on a more local area—instead of the global center of mass—it helps achieve a larger scale effect. This result also leveraged the simplified pre-roll technique instead of the full texture synthesis method to generate the end state.

4.1 Discussion

One major benefit of our method is that it doesn't replace existing control methods—it only increases their utility. Time reversed simulation can be combined with existing techniques to give artists more flexibility. With our method an artist can choose to start the simulation in a target shape from any arbitrary timestep. With the addition of a control technique such as the one described by Fattal and Lischinski [6] the artist can also choose additional, secondary target shapes before or after that point. Time reversed simulation allows the artist to be

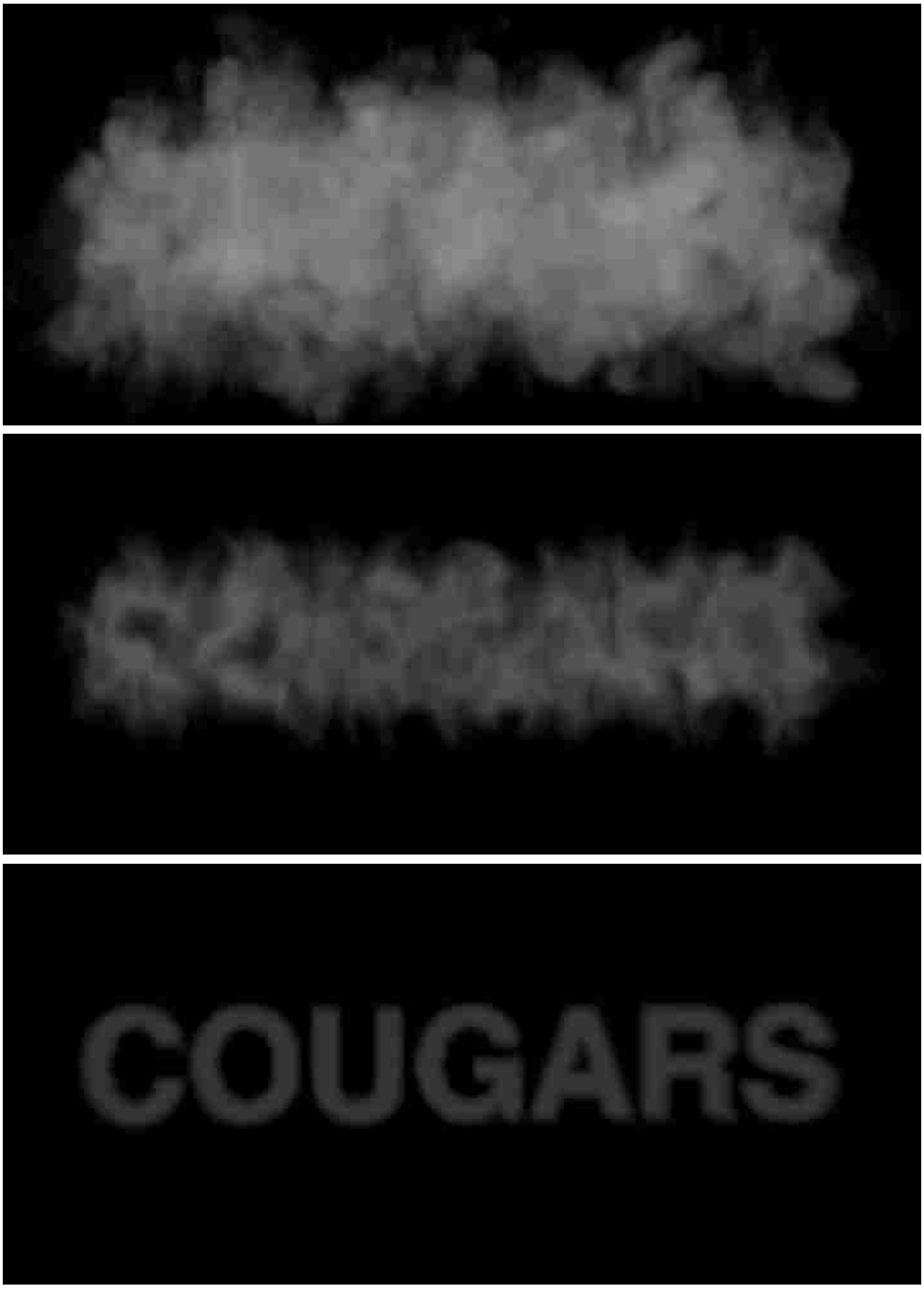

Figure 4.2: An example of time reversed dissipation. The smoke gradually dissipates to form words.

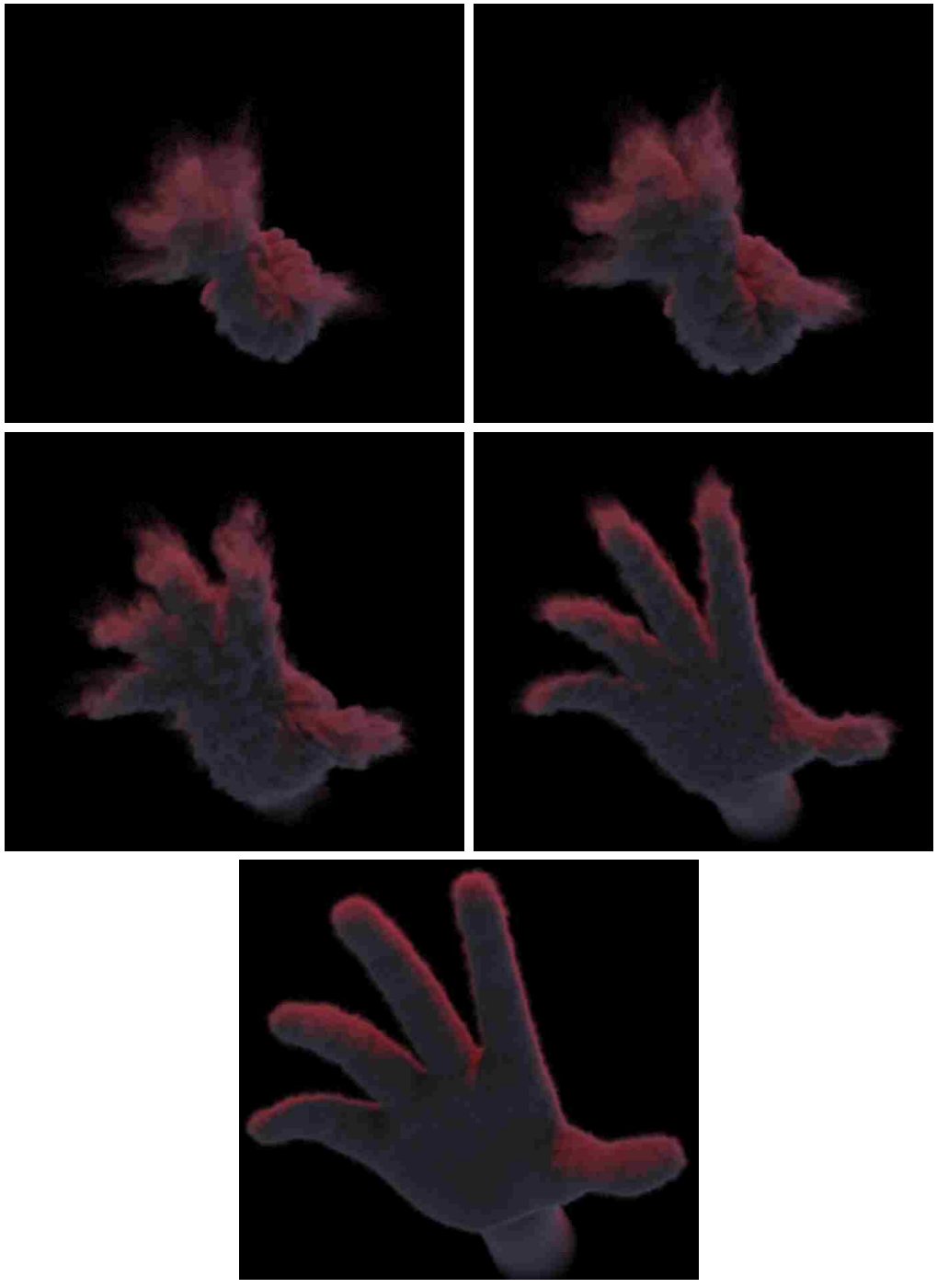

Figure 4.3: A large scale example of a time reversed simulation using our self-attraction force with the full Barnes-Hut Tree.

sure that the most important target shape will be hit exactly, no matter where in time that target lies.

We have shown that our reverse simulation method is fast and efficient. We have also shown that it produces results that exhibit several important characteristics of forward simulations. It follows the 2nd law of thermodynamics, with entropy increasing in the forward direction of time, and the end state fluid configuration can be generated with similar visual qualities to the forward simulated states. This ensures that our results are visually believable. Another major advantage of our method is that it always hits the target shape exactly. With many other control methods, there is still an element of unpredictability. This unpredictability is eliminated with our time reversed simulation. In summary, our method achieves an exact target, runs quickly, and is visually believable. These qualities make our method a valuable addition to the fluid control tools available to artists.

4.2 Conclusion

We have described a time reversed simulation method that produces visually plausible behavior. Our method generates a believable end state for the reverse simulator to start from, and it ensures that entropy will decrease as the reverse simulation evolves. Our method produces results comparable to previous target-matching optimization methods with simulation times very close to those of a contemporary forward simulator. It also has the enormous benefit of being able to guarantee an exact match to the target shape. Time reversed simulation can change the way artists approach fluid simulation by allowing them to define the shape of the fluid at the most important timestep, whether it be at the beginning, middle, or end of the simulation.

4.3 Future Work

Time reversed fluid flow can look unnatural, even after forcing entropy to decrease. This is a difficult problem without a clear solution. This is likely related to the fact that the end

state velocity field isn't the result of a forward simulation. One possible way to reduce the uncanny motion is to improve the synthesis methods in section 3.3. This could perhaps be achieved by leveraging a large dataset of fluid states instead of just one example. Also, the synthesis stage is the most computationally expensive part of our process. It took several hours to synthesize the end state shown in Figure 4.1. Improving efficiency in this area would be an important step for this approach to become practical for the average user.

We have essentially ignored viscosity in our approach, as has been common in fluid simulation for computer graphics. This is due to numerical dissipation adding it for free. However, with time reversal, this dissipation is happening in the wrong direction of time. Our entropy constraint counteracts this to some extent by adding energy back into the system, but it is inaccurate and insufficient for very high viscosity behavior. One possibility to overcome this would be to use an energy-preserving integrator such as those described in Mullen et al. [15] or Liu et al. [11]. Our process for reversing dissipation could then be adapted for viscosity.

Our method would need to be adapted to work for simulating non-gaseous phenomena such as water. We have mentioned that the initial state of a fluid simulation is often of very little consequence. However this is not always the case. For example, when simulating water, it is often important that the initial state be a calm, flat surface. This represents a challenge for time reversed simulation, as generating an end state that would result in such an initial state would likely require an entire separate simulation. Also, our divergence modification method would look unnatural for water as it does not exhibit the same free expansion property of gases.

References

[1] Josh Barnes and Piet Hut. A hierarchical o (n log n) force-calculation algorithm. *Nature*, 324(6096):446–449, 1986.

[2] Paul Charles William Davies. *The physics of time asymmetry*. Univ of California Press, 1977.

[3] Matthieu Duponcheel, Paolo Orlandi, and Grégoire Winckelmans. Time-reversibility of the euler equations as a benchmark for energy conserving schemes. *Journal of Computational Physics*, 227(19):8736–8752, 2008.

[4] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1033–1038. IEEE, 1999.

[5] Douglas Enright, Stephen Marschner, and Ronald Fedkiw. Animation and rendering of complex water surfaces. *ACM Transactions on Graphics (TOG)*, 21(3):736–744, 2002.

[6] Raanan Fattal and Dani Lischinski. Target-driven smoke animation. *ACM Transactions on Graphics (TOG)*, 23(3):441–448, 2004.

[7] Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. Visual simulation of smoke. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, pages 15–22. ACM, 2001.

[8] Bryan E Feldman, James F O'brien, and Okan Arikan. Animating suspended particle explosions. *ACM Transactions on Graphics (TOG)*, 22(3):708–715, 2003.

[9] Nick Foster and Ronald Fedkiw. Practical animation of liquids. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, pages 23–30. ACM, 2001.

[10] Nick Foster and Dimitris Metaxas. Controlling fluid animation. In *Proceedings of the Computer Graphics International Conference*, pages 178–188. IEEE, 1997.

[11] Beibei Liu, Gemma Mason, Julian Hodgson, Yiying Tong, and Mathieu Desbrun. Model-reduced variational fluid simulation. *ACM Transactions on Graphics (TOG)*, 34(6):244, 2015.

[12] Joseph Loschmidt. *Über den Zustand des Wärmegleichgewichtes eines Systems von Körpern mit Rücksicht auf die Schwerkraft: I [-IV].* aus der KK Hof-und Staatsdruckerei, 1876.

[13] Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. Fluid control using the adjoint method. *ACM Transactions On Graphics (TOG)*, 23(3):449–456, 2004.

[14] Joseph J Monaghan and John C Lattanzio. A refined particle method for astrophysical problems. *Astronomy and astrophysics*, 149:135–143, 1985.

[15] Patrick Mullen, Keenan Crane, Dmitry Pavlov, Yiying Tong, and Mathieu Desbrun. Energy-preserving integrators for fluid animation. *ACM Transactions on Graphics (TOG)*, 28(3):38, 2009.

[16] Michael B Nielsen and Robert Bridson. Guide shapes for high resolution naturalistic liquid simulation. *ACM Transactions on Graphics (TOG)*, 30(4):83, 2011.

[17] Zherong Pan and Dinesh Manocha. Efficient optimal control of smoke using spacetime multigrid. *arXiv preprint arXiv:1608.01102*, 2016.

[18] Zherong Pan, Jin Huang, Yiying Tong, Changxi Zheng, and Hujun Bao. Interactive localized liquid motion editing. *ACM Transactions on Graphics (TOG)*, 32(6):184, 2013.

[19] Nick Rasmussen, Douglas Enright, Duc Nguyen, Sebastian Marino, Nigel Sumner, Willi Geiger, Samir Hoon, and Ronald Fedkiw. Directable photorealistic liquids. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 193–202. Eurographics Association, 2004.

[20] Karthik Raveendran, Nils Thuerey, Chris Wojtan, and Greg Turk. Controlling liquids using meshes. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 255–264. Eurographics Association, 2012.

[21] Karthik Raveendran, Chris Wojtan, Nils Thuerey, and Greg Turk. Blending liquids. *ACM Transactions on Graphics (TOG)*, 33(4):137, 2014.

[22] Andrew Selle, Ronald Fedkiw, Byungmoon Kim, Yingjie Liu, and Jarek Rossignac. An unconditionally stable maccormack method. *Journal of Scientific Computing*, 35(2): 350–371, 2008.

[23] Jos 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.

[24] Alexey Stomakhin and Andrew Selle. Fluxed animated boundary method. *ACM Transactions on Graphics (TOG)*, 36(4):68, 2017.

[25] Nils Thürey, Richard Keiser, Mark Pauly, and Ulrich Rüde. Detail-preserving fluid control. *Graphical Models*, 71(6):221–228, 2009.

[26] Adrien Treuille, Antoine McNamara, Zoran Popović, and Jos Stam. Keyframe control of smoke simulations. *ACM Transactions on Graphics (TOG)*, 22(3):716–723, 2003.

[27] Christopher D Twigg and Doug L James. Backward steps in rigid body simulation. *ACM Transactions on Graphics (TOG)*, 27(3):25, 2008.

[28] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pages 479–488. ACM Press/Addison-Wesley Publishing Co., 2000.

[29] Qing Wu and Yizhou Yu. Feature matching and deformation for texture synthesis. *ACM Transactions on Graphics (TOG)*, 23(3):364–367, 2004.

[30] Jingdan Zhang, Kun Zhou, Luiz Velho, Baining Guo, and Heung-Yeung Shum. Synthesis of progressively-variant textures on arbitrary surfaces. *ACM Transactions on Graphics (TOG)*, 22(3):295–302, 2003.

[31] Hongkai Zhao. A fast sweeping method for eikonal equations. *Mathematics of Computation*, 74(250):603–627, 2005.

[32] Yongning Zhu and Robert Bridson. Animating sand as a fluid. *ACM Transactions on Graphics (TOG)*, 24(3):965–972, 2005.