# Time Reversal and Simulation Merging for Target-Driven Fluid Animation

Sivakumaran Gowthaman
Carleton University
Ottawa, ON, Canada
gowtham.maran1@gmail.com

Eric Paquette
École de technologie supérieure
Montreal, QC, Canada
eric.paquette@etsmtl.ca

David Mould
Carleton University
Ottawa, ON, Canada
mould@scs.carleton.ca

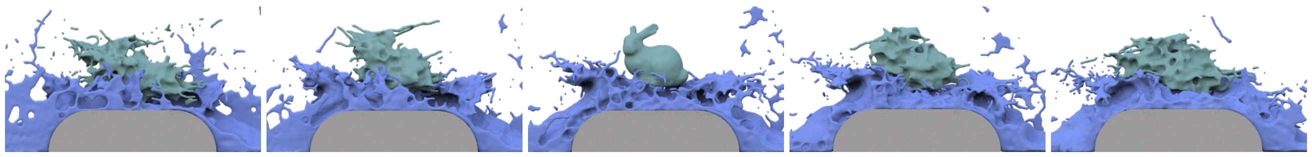

**Figure 1: Frames from art-directed fluid simulation. A target simulation (green) combines with the source simulation (blue); the target simulation forms the target shape, while the source simulation provides surrounding fluid motion.**

## ABSTRACT

We present an approach to control the animation of liquids. The user influences the simulation by providing a target surface which will be matched by a portion of the liquid at a specific frame of the animation; our approach is also effective for multiple target surfaces forming an animated sequence. A source simulation provides the context liquid animation with which we integrate the controlled target elements. From each target frame, we compute a target simulation in two parts, one forward and one backward, which are then joined together. The particles for the two simulations are initially placed on the target shape, with velocities sampled from the source simulation. The backward particles use velocities in the opposite direction as the forward simulation, so that the two halves join seamlessly. When there are multiple target frames, each target frame simulation is computed independently, and the particles from these multiple target simulations are later combined. In turn, the target simulation is joined to the source simulation. Appropriate steps are taken to select which particles to keep when joining the forward, backward, and source simulations. This results in an approach where only a small fraction of the computation time is devoted to the target simulation, allowing faster computation times as well as good turnaround times when designing the full animation. Source and target simulations are computed using an off-the-shelf Lagrangian simulator, making it easy to integrate our approach with many existing animation pipelines. We present test scenarios demonstrating the effectiveness of the approach in achieving a well-formed target shape, while still depicting a convincing liquid look and feel.

## CCS CONCEPTS

• **Computing methodologies** → **Physical simulation**; **Motion processing**.

## KEYWORDS

animation, fluid simulation, visual effects, motion control, target-driven fluid animation

**ACM Reference Format:**
Sivakumaran Gowthaman, Eric Paquette, and David Mould. 2022. Time Reversal and Simulation Merging for Target-Driven Fluid Animation. In *ACM SIGGRAPH Conference on Motion, Interaction and Games (MIG '22), November 3–5, 2022, Guanajuato, Mexico.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3561975.3562952

## 1 INTRODUCTION

Animated visual effects applications range from movies, TV shows, and advertisements to real-time interactive applications in virtual/augmented reality and computer games. In such applications, it is tedious for artists to animate phenomena like smoke, clouds, fire, and water. Computational fluid simulation is completely governed by physical laws. Once the initial state is set for the simulator, it simulates forward in time. The advantage of computational fluid simulation is that it generates reproducible results. At the same time, this constrains the effect artists as they lack creative control over the simulation process, beyond specifying the initial state.

In contrast to computational fluid dynamics, computer graphics prioritizes the subjective ambiguous quality of believability over the objective unambiguous physical accuracy. Art direction of fluid motion is one of the major requirements of animators where the simulated fluid takes the form of desired target shapes at the desired frames within the animation; for example, in the often fantastical settings of games, magical effects or mythical creatures may manifest as recognizable shapes made of water. However, in a typical fluid solver, animators can only change the settings of the initial state and would have to repeat the simulation in a trial and error manner until they end up with the expected art-directed fluid motion effect. This is a time-consuming and expensive process that does not guarantee the expected results.

In order to provide artistic control for art-directed fluid motion, previous strategies have used control forces through user-defined guides in terms of control particles and meshes. Even though this resulted in a considerable amount of control over fluid motion, it tampers with the natural motion of the fluid, which is the primary property that identifies it as fluid in the first place. If, in an effort to restore the natural fluid motion look, the control forces are relaxed, there will be a low probability of attaining the desired art-directed fluid motion.

In this paper, we propose an approach that facilitates the process for artists to have a desired fluid shape. We separate the liquid into two separate simulations: *source* and *target*. The source simulation is a conventional simulation controlled by initial conditions. It provides the contextual liquid behavior into which the target simulation will be integrated. The target simulation matches the target shape; it consists of a forward component, simulating the fluid forward in time, and a backward component, which also simulates the fluid forward in time but which will be played backwards. Our source simulation does not need to be constrained to a specific initial simulation state in order to have the desired effect later in the simulation. At any desired point in time, the intended target shape is achieved without introducing artificial control forces that tamper with the appearance of natural fluid motion.

Our approach runs target fluid simulations both forward and backward in time from the desired fluid shape at the desired animation frame. We sample velocity vectors from a source simulation to set the initial velocity of particles in our forward and backward simulations. When the backward and forward simulations are concatenated, the fluid target shape is achieved at the desired frame. Since we are intermixing multiple simulations, we need a strategy for merging them. We propose to select a subset of target particles to be used in the surface reconstruction and rendering, and to ignore the others. Judicious selection of visible particles limits discrepancies visible in the merged simulation.

The plausibility of the final result rests on the assumption that the backward simulation (simulated in forward direction but played in reverse order) looks believable. In general, a video played in reverse order can be detected as such. In the case of these fluid simulations, the viewer's job is much harder, for three reasons: one, much of the motion is ballistic, which is reversible; two, the portion of the motion played backwards covers a small fraction of the scene (most of which is the source simulation); three, both simulations, and especially the source, contain fairly complex motions, and the human viewer has a limited ability to predict accurately the motion. The turbulent motion motivates the manifestation of the target particles; their motion is governed by a simulation, and hence inherits the plausibility of the simulated physics.

Our main contributions can be summarized as follows:

- A division of the simulation into source and target, and division of the target simulation into a forward portion and backward portion.
- An approach to distribute target particles to reproduce well-formed target shapes;
- An approach to set initial conditions of the target simulations, allowing a good integration with the source simulation;
- An approach to select an appropriate set of particles for surface reconstruction and rendering.

## 2 PREVIOUS WORK

Foster and Metaxas [Foster and Metaxas 1997] presented the idea of fluid control using embedded controllers. The idea of control particles was introduced by Foster and Fedkiw [Foster and Fedkiw 2001] as a mechanism to incorporate user-defined body forces. Rasmussen et al. [Rasmussen et al. 2004] further modified this by using the particle level-set method to include distinct control particles for fluid parameters such as divergence, viscosity, and velocity.

Thurey et al. [Thürey et al. 2009] proposed a multiscale decomposition of the velocity field and apply control forces only to the coarse-scale components of the flow. This preserves the small-scale detail in a natural manner, by avoiding the artificial viscosity that force-based control systems frequently generate. Nielsen and Bridson [Nielsen and Bridson 2011] presented a method that uses a low-resolution simulation as a guide shape for a higher resolution. This keeps the low-resolution version's general shape and motion while adding high-resolution detail. Madill and Mould [Madill and Mould 2013] proposed a two-layer approach in which a bulk velocity drives a particle system towards a target distribution by using matching control particles while a vortex particle simulation adds recognizable fluid motion. Raveendran et al. [Raveendran et al. 2014] presented a semi-automatic method that blends two different simulations. Even though control-particle methods provide a means to guide the fluid, manipulations can only be achieved through precise changes in the starting state. It is still difficult and time-consuming to get a precise shape in the middle of the simulation.

Some methods generate guiding forces to achieve the target fluid motion. The concept of smoke density targets [McNamara et al. 2004; Pan and Manocha 2016; Treuille et al. 2003] relies on user-specified keyframes to define target shapes at specific points in the simulation. Fattal and Lischinski [Fattal and Lischinski 2004] combined a driving force term to direct the smoke towards the target, with a smoke gathering term that prevents the smoke from diffusing too much. These guiding force methods incorporate new and non-physically based equations into the simulation process. An attraction constraint, which propels the fluid toward the target shape, is an example of these artificial forces.

Raveendran et al. [Raveendran et al. 2012] generate a volume-preserving morph that allows the animator to produce a plausible fluid-like motion from a sparse set of control meshes. They strive to maintain the fluid momentum by reintroducing unpredictability into the simulation and allowing for relaxed control. Pan et al. [Pan et al. 2013] allow local fluid motion editing through the use of curves and meshes, and global influence through guide forms to control the fluid's overall silhouette. The Fluxed Animated Boundary (FAB) technique [Stomakhin and Selle 2017] enforces the boundary constraints of a user-defined control shape and material flow field. One limitation of the FAB method is that creating physical velocities requires attention and care.

The retiming of a simulation could be another possible method for art direction. A deep neural network method by Giraud et

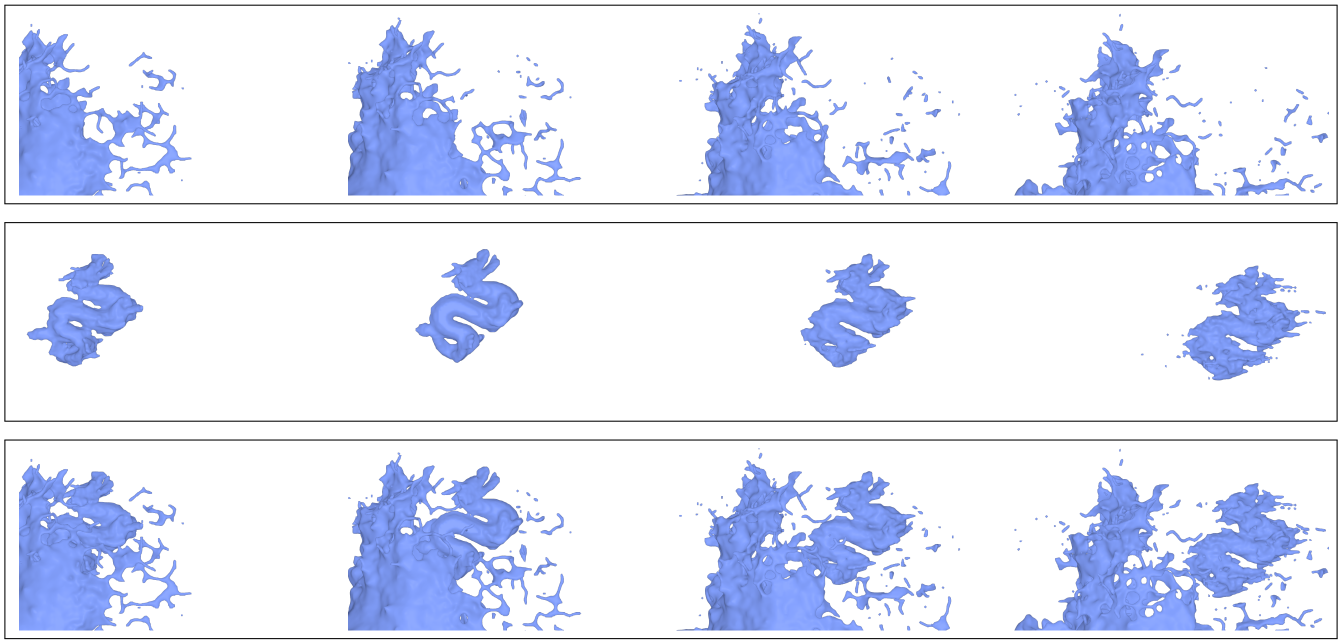
**Figure 2: The basic process. First, a source simulation is created (top). Second, a target simulation is created, moving both forward and backward in time from a target shape (middle). Third, the particle sets from all simulations are merged and rendered together (bottom).**

al. [Giraud-Carrier 2020] is trained from the frames in an original fluid simulation sequence and reproduces a set of time samples that achieve the desired new sequence speed. If there is a target shape in the original simulation, the desired frame in which the target shape has to be formed can be altered using this method. Nevertheless, this approach at the moment is not fully developed and is not able to generate a target shape that is not found in the original simulation.

## 2.1 Time-reversed Simulation

Time-reversed simulation is a highly promising class of methods for art directing fluid animation. In such methods, fluid is simulated backward from a target shape. When the result of the reversed simulation is played forward in time, the fluid appears to flow naturally into the intended shape.

Twigg and James [Twigg and James 2008] introduced the time-reversal idea for a rigid body simulator. Reverse simulation can appear odd because it presents circumstances in which entropy decreases when played forward. Their method for rigid bodies is to simply jitter the backward simulator's starting configuration. For fluid flow, this was insufficient. Oborn et al. [Oborn et al. 2018] began with a target shape and then ran a simulation in the time-reversed direction, resulting in smoke that appears to organically fit the target shape when played forward in time. This enables artists to construct simulations with quick turnaround times that fit an exact art-directed shape at any point in the simulation. To accomplish a backward simulation of smoke, this method makes use of a novel set of equations to reverse entropy. While time-reversed simulations appear to be promising, previous efforts have introduced non-physically based components, and this results in a less natural fluid motion. Our approach uses time reversal but pushes the non-physical aspects to the simulation merging, and all motion that is shown arises from the simulation. Especially when the motion is ballistic, as in splashes and waterfalls, the reversibility of the trajectory means that even the backward motion is highly realistic.

## 3 METHOD

Our process involves creating two separate, noninteracting fluid simulations: see Figures 1 and 2. The *source simulation* is the main simulation, providing the context and environment in which the target shape appears. The *target simulation* provides the evolution of the target shape; it is divided into forward and backward simulations, which will then be concatenated to produce the full animation. See Figure 3 for a visualization of this process. The steps in the method can be summarized as follows:

(1) Create and compute the source simulation.
(2) Obtain initial conditions for the target simulation. Appropriately fill the target shape with particles. For each target simulation particle, derive its velocity from the source simulation.
(3) Carry out the two halves of the target simulation: one forward, one backward. Join the two halves together to construct the entire target simulation sequence.
(4) Conduct a selection of which particles to keep and remove when merging the source and target simulation particle sets.
(5) Reconstruct the surface and render.

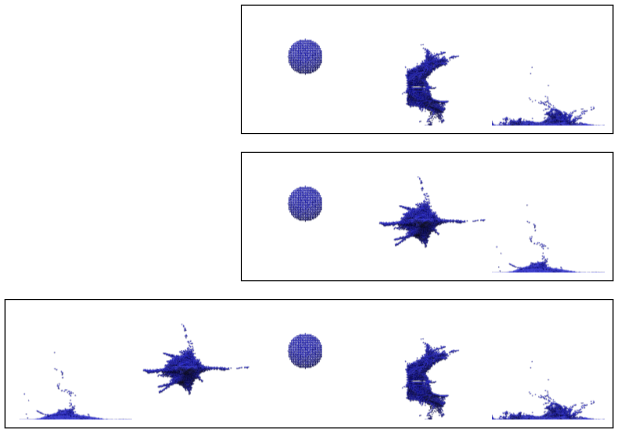
**Figure 3: Example of a target simulation. Top: Forward target simulation. Middle: Backward target simulation. Bottom: Complete target simulation (merged backward and forward).**

In the following subsections, we provide more detail on the scheme outlined above. We will begin by discussing the case of a static target shape appearing at a single frame, followed by a description of how to handle persistent and animated target shapes (Section 3.2). A static target shape is handled by augmenting the source simulation with a single target simulation. When target shapes appear in multiple frames, we construct multiple separate target simulations by merging multiple pairs of forward and backward simulations.

## 3.1 Simulation Details

In preparation for the art-directable portion of the fluid motion, the animator first creates the source simulation, representing the surroundings where the effect will appear. The animator will choose the time and placement of the target shape, typically after inspecting the source simulation to guide the decision. The target simulation is carried out in two halves: one, conventionally, *forward* in time, and the other, the *backward* simulation, with opposite initial velocities. This backward simulation will be animated in reverse order; note how reversing the initial velocities before simulation, then reversing it again when playing it back, causes the velocities to match exactly at the moment when the forward and backward components connect.

For both the source or target simulations, our approach is agnostic to the specific simulator, as long as it is a Lagrangian (particle-based) fluid simulator and that it preserves the same set of particles through time. This is an advantage of our approach as it can use an off-the-shelf simulator instead of requiring modification of a simulator to add special forces or fields.

Below, we discuss the two key considerations for using our technique: first, how to determine the initial conditions for the target simulation; second, how to integrate the source and target simulations. Because we use simulation without guiding forces or other elements, the choice of initial conditions is critical in getting the desired effect. Further, because we perform the source and target simulations separately, they are not naturally integrated and some effort must be expended in postprocessing to enhance the illusion of a unified effect.

In order to describe the postprocessing carefully, we require some formal notation. We have some number of simulations $S^{(s)}$, where $S^{(0)}$ is the source simulation and one or more target simulations are given unique indices $s > 0$. The source simulation begins at global time zero and has a long temporal extent, while other simulations have shorter temporal extents set relative to the global time as determined by the animator. Each simulation contains a number of particle trajectories; a particle trajectory $p_{s,i,t}$ is written with subscript $s$ to indicate which simulation generated it, subscript $i$ for its index within the simulation, and subscript $t$, for time, to access a particular moment within the simulation. Trajectories from the source simulation are only defined for positive $t$; trajectories for target simulations have both a forward and backward component and are defined for a range of times $-S^{(s)}_{\text{maxtime}} < t < S^{(s)}_{\text{maxtime}}$ relative to the start time of the simulation to which they belong.

*3.1.1 Initial Conditions.* Here, we restrict our discussion to the intial conditions of the target simulation. Any existing workflow can be used to create the source simulation and we impose no restrictions there.

For the target simulation, we intend to design initial conditions that (i) produce the target shape at the desired time, and (ii) mimic the surrounding source motion. The former requirement informs our choice of initial particle positions, while the latter requirement affects our assignment of initial velocities. In designing a policy for initial velocities, we note the tradeoff between ease of use for the animator and the resulting complexity of motion. Our objective is to create plausible and dramatic motion without imposing excessive burden on a human animator or director. The user must choose a few numerical parameters (or use default values), plus make decisions about the location and timing of the target shape and the spatial region from which the target shape's velocities will be sampled. Naturally, the user must select the target shape itself, which in our implementation is represented by a polygonal mesh. (In principle, any representation which allows point sampling could be used.) A list of parameters and default settings is shown in Table 1.

In order to reproduce the target shape with fluid particles, it might appear that we should simply populate with fluid particles the volume contained within the target shape. Indeed, this is a valid strategy. However, as only the fluid surface is visible, particles within the volume typically contribute less to the appearance of the shape: we can save simulation effort by leaving the volume hollow. More importantly, filling the volume leads to sluggish behaviour as the particles restrict one another's movement; using a hollow shell of particles provides a lighter, more energetic feeling to the animation. We have used the shell system for all examples shown in this paper.

To create a shell of particles, we compute a signed distance field from an input mesh, then use a Poisson disc sampling to populate the interior of the mesh (sign negative) within a distance $\kappa_s$ of the surface. The particle spacing is dictated by the simulation settings; for simulations in this paper, we used a spacing of $h = 0.01$.

For the velocities, several strategies are possible, including defining and evaluating an analytic function, or allowing the animator to create a velocity field. While our method is compatible with any

**Table 1: List of numerical parameters. Values given are representative for the animations we show in the paper.**

| **Parameter** | **Symbol** | **Typical setting** |
|---|---|---|
| Shell thickness | $\kappa_s$ | 0.01–0.02 |
| Initial particle spacing | $\kappa_t$ | 0.01 |
| Target simulation duration | – | 0.5 s |
| Target simulation duration (middle frames) | – | 0.15 s |
| Velocity assignment fraction | $f$ | 0.25 |
| Velocity amplification | $\gamma$ | 1.0–1.3 |

of these, creating an adequately complex distribution can be time-consuming. Fortunately, we have a complex distribution at hand, namely, the source simulation; we employ a strategy of sampling velocities from the source simulation, as follows. The animator defines a region of the source simulation from which velocities will be obtained. This is often a region relevant to the target simulation, such as its immediate surroundings, or the site of a splash. Source particles within the region are eligible to be sampled. Additional restrictions may be imposed; e.g., the animator could decide that the particles must have upward vertical velocity.

Having established a set of eligible source particles, we then sample from them and assign their velocities to randomly selected target particles. Some fraction $f$ of target particles ($f = 0.25$ for most examples in this paper) are assigned velocities in this manner; the remainder are assigned velocities by interpolating among those already assigned, using Shepard's method [Shepard 1968]. Occasionally, the animator may want to make the motion more vigorous; we provide a single-parameter mechanism for doing so, multiplying all velocities by a scalar factor $\gamma$. By default we use $\gamma = 1$, but we occasionally used $\gamma$ values as high as 1.3.

Once initial conditions for the target simulation are available, we can compute the simulation: one half forward in time (Figure 3-top), using the initial velocity, and one half backward in time (Figure 3-middle), running the simulator forward but reversing the initial velocity of each particle. The outcome of the backward simulation is reversed and the forward and backward simulations are concatenated to form the full target simulation (Figure 3-bottom), which is then merged with the source simulation as discussed next.

#### 3.1.2 *Merging Source and Target Simulations.*

At this stage, we have a separate source simulation and target simulation. They have been simulated separately and do not interact. The target simulation is intended to last only a short time (a maximum duration specified by the animator). Within this period, however, we will render only a subset of the available particles, hiding some portions of the particle trajectories to improve the plausibility of the animation.

The intent is to hide target particles starting from the moment when they come into close proximity with source particles. We refer to the detection of proximity among particles from different simulations as a *collision event*, since it parallels conventional collision detection. The collision event provides a visual explanation for the sudden appearance of the target particle, if a backward portion of the trajectory is hidden, or for its disappearance, if a forward portion is hidden. Note that the plausibility of the explanation depends on the activity level within the source simulation, with more violent motion being more unpredictable and hence providing a stronger illusion of continuity. Conversely, suppose the target particles were to rise suddenly from a still body of water; the viewer could detect the inconsistency with trivial ease. For this reason, we advise that the source simulation contain high levels of activity, containing splashes, sprays, and other violent motion that can disguise the non-physical addition and removal of target particles.

When a collision event occurs, one trajectory is higher priority and is unchanged, while one is lower priority and a portion will be hidden. Recall that collision events always involve trajectories from two different simulations. In the case of a single source and a single target simulation, the trajectory from the source simulation is always higher priority. The same system is used for creating animated fluid objects, in which case multiple target simulations are created and need to be merged. We discuss this case next.

## 3.2 Persistent and Animated Target Shapes

As described, the target simulation creates fluid motion that forms into the target shape for a single moment. However, animators may wish to have a longer-lasting target shape, or even to animate the target shape with keyframe animation or other techniques before it dissolves.

We propose breaking an input animation into separate target shapes, each offset in time by some small amount, typically one frame. We will then execute a separate target simulation for each of these target shapes. The target simulations do not interact physically with one another, even though they overlap in time. Instead, we postprocess the simulations to hide some particles, providing an illusion of interaction when the particles appear or disappear. When we have many target simulations close together in time, we recommend making them very short. The exceptions are the backward portion of the first target simulation, and the forward portion of the last target simulation, which have no time-adjacent simulation and are given a full duration (by default, 0.5 seconds).

We compare particle trajectories between simulations, and when particles from different simulations get too close, one trajectory will be terminated and its particle will be hidden for all subsequent frames. More formally, suppose we have a collision event and decide to remove part of particle trajectory $p_{s,i,m}$. For $m > 0$, we hide the rest of the trajectory $p_{s,i,t}$ where $t >= m$, and if $m < 0$, we hide the portion of the trajectory where $t =< m$. The intent is to keep more particles close to the target frame so that we have a well-formed shape, while having the liquid particles disappear reasonably quickly away from the target frame.

In any collision event, one of the trajectories will have higher priority and one will have lower. Consider a collision between

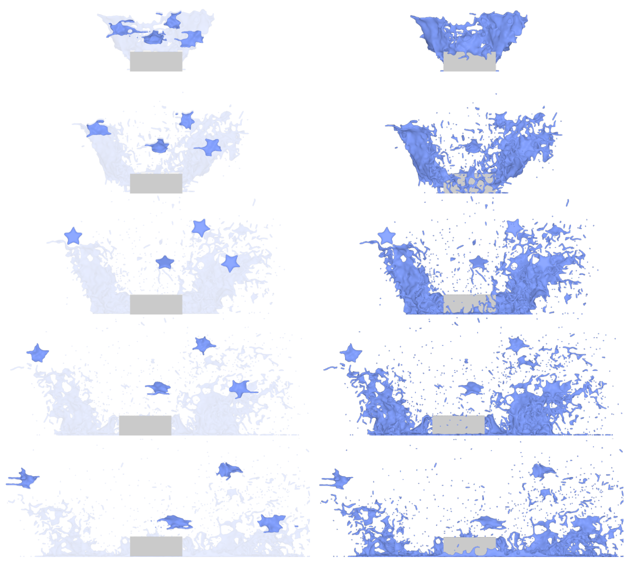
**Figure 4: Left: Star target shape simulation. Right: Merged source and target simulations.**

trajectories $p_{s_1,i,u}$ and $p_{s_2,j,v}$, $s_1 \neq s_2$. Priority is determined using the following policy:

- Trajectories from the source simulation always have highest priority. If neither $s_1$ nor $s_2$ refer to the source simulation, apply the following rules.
- If $\mathrm{sign}(u) \neq \mathrm{sign}(v)$, higher priority goes to the trajectory with positive sign.
- With $\mathrm{sign}(u) = \mathrm{sign}(v)$, higher priority goes to $p_{s_1,i,u}$ if $|u| < |v|$, otherwise $p_{s_2,j,v}$ has higher priority.

For each frame where the animator set a target shape, all particles from earlier target simulations that are inside the target shape are hidden. In addition, we hide particles that are far from the target shape (more than distance $\kappa_t$). Preventing such particles from being rendered improves the visibility of the target shape at later frames, and the presence of the target shape motivates the disappearance of these trajectories from the scene.

## 4 RESULTS

We tested our approach on various scenarios involving different target shapes. We first present our results involving static shapes, and then our results with dynamic target shapes.

### 4.1 Static

We designed different scenarios to test our approach with static target shapes. Our source simulations range from a sphere of liquid splashing into a container, to various forms of dam breaks. We use a variety of target shapes: a dragon with fine details, stars with sharp features, and the bunny which has a relatively smooth surface.

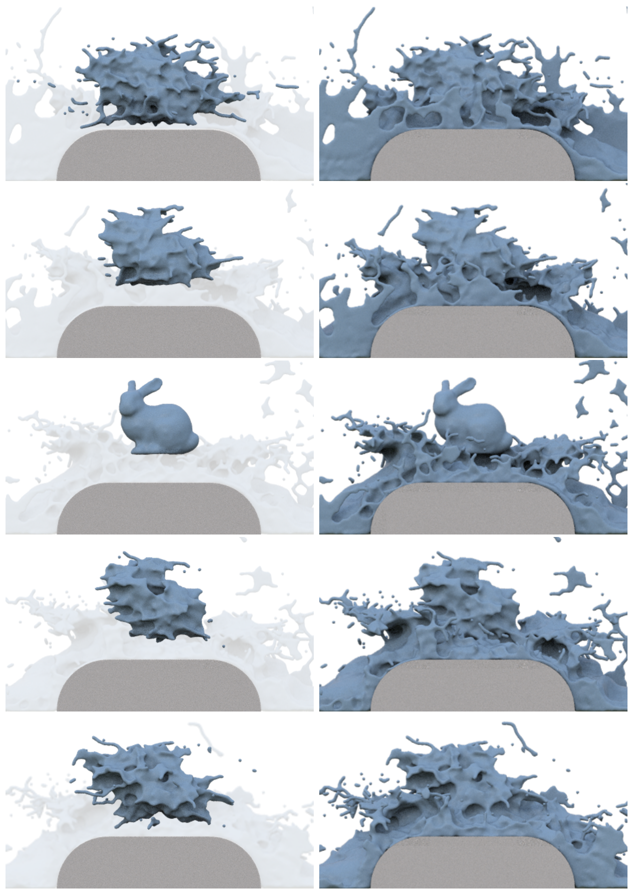
**Figure 5: Left: Bunny target shape simulation. Right: Merged source and target simulations.**

#### 4.1.1 *Splash.*

In this simulation, we drop a globe of water into a partly filled bucket, causing a splash. We observed the resulting behaviour to identify distinct clusters of droplets, and placed a target shape (a star) at each of three clusters, plus a fourth star by itself above the middle of the bucket. Velocities for each target shape were sampled from a nearby region of the source; the vertical component of velocity was set to zero, and the horizontal component multiplied by a factor $\gamma \approx 1.3$ (a slightly different factor for each star). Example frames are shown in Figure 4, and a video is available in the supplemental material. The resulting combined rendering shows the splash seeming to create the stars, with the formation and subsequent dissipation of the shapes looking quite believable.

#### 4.1.2 *Double dam break.*

Here, we release two columns of water from either end of the simulation domain. Barriers are placed on the domain floor, arranged to create an upward disturbance in the water flow. We placed a bunny object in the middle of the domain, timing its placement with the progress of the source simulation. Example frames are shown in Figure 5, and a video is available in the supplemental material. Target velocities are taken from a large horizontal region, encompassing particles travelling in both the positive and negative $x$ directions, causing the bunny to appear to be formed by two separate splashes cooperating and then passing through one another.

*4.1.3 Dam break.* This simple scenario demonstrates the difference between using a single target shape and using multiple shapes. A column of water is released from one side of the simulation domain and strikes a barrier, producing a spray of water. After observing the outcome of the simulation, we placed a target shape (dragon) slightly ahead of the main mass of splashing water. Sample frames from the animation are shown in Figure 6(top). The velocities are obtained by sampling a bounding box containing the splash; the resulting distribution causes the dragon to disintegrate, while the positioning of the target relative to the source splash provides a justification for the sudden appearance of the target shape.

## 4.2 Animation

Animated target shapes were tested with two source scenarios: a dam break and a waterfall. We used two strategies for the animated target shapes: one test simply translates a dragon object without changing its shape, while other experiments involve target objects that change shape during the animation.

*4.2.1 Flying dragon.* Here, we revisit the single dam break scenario above: a column of water is released, the flow strikes a barrier and sprays upward, and a dragon model is placed ahead of the splash. The difference now is that we place several target shapes along a manually-constructed approximation of a ballistic arc. Sample frames from the animation are shown in 6. Compared to its single-target predecessor, the dragon is much more stable, disintegrating only once the final shape is finished. This simple scenario serves as an introduction to the multi-target process.

*4.2.2 Running horse.* In this scenario, we extract several poses from a horse model animation and place them in sequence in front of a waterfall. The horses are placed underneath and a little in front of the falling water so that the body is covered and the head is clear. This setup was inspired by the scene at the Ford of Bruinen in the movie *The Fellowship of the Ring*, in which horses manifest in the rushing waters of the river. Frames from the animation appear in Figure 7.

The initial velocities are taken from a box above and behind the horse, near the top of the waterfall. The horse is stable during the target frames, although obscured by the source simulation; once the last target frame is reached, the horse seems to revert to regular water.

*4.2.3 Face in waterfall.* We created a second waterfall and added visual interest to the scene by using a rotating turbine to stir the water accumulating on the ground. Within the waterfall, we placed several frames from a face animation; the face's expressions change in time with the spurts of water. Initial velocities were taken from the region surrounding the face. As with the other animated sequences, once the last frame is reached the face falls apart. Here we can see the importance of meshing source and target particles into a single surface. A frame from the animation is shown in Figure 8, with the full video presented in the supplementary material.

## 5 DISCUSSION

Our method is quite effective for single target shapes. The two halves of the target simulation transition smoothly, and the integration of the target simulation and source simulation is plausible, aided by the unpredictability of the complex motion in the source. Earlier work on target shape simulation concentrated on smoke, rather than the liquid that we are simulating here.

We further demonstrated how the basic method can be extended to sequences involving animated models. The same basic concepts apply, with short-lived target simulations used in middle frames, and longer-lived simulations for the first and last keyframe. Past methods that used time reversal to provide target shape control over fluid simulation did not attempt to animate the target shapes at all.

A significant advantage of our approach is that it is compatible with any existing particle-based fluid solver or particle effect engine. There is no need to modify the internals of the solver; rather, multiple simulations are run separately, with coordinated assignment of initial conditions, and the outputs of the simulations are integrated in postprocessing. The drawback of this strategy is that physical interaction between particles in different simulations is not possible, which we view as our primary limitation. We elaborate on limitations next.

## 5.1 Limitations

Our decision to separate the target and source simulations helps with controllability and simplifies the pipeline, but limits the physical interaction between the art-directed effect and the source simulation. We advise disguising the lack of interaction with high complexity of simulation. However, in scenes with still water or simple motions, the lack of interaction will be apparent. Lack of interaction between source and target simulations is our primary limitation, and improving the coupling between the simulations is the most obvious direction for followup work.

Continuity between the forward and backward portions of the target simulation is excellent when the target shape is static and appears only for a single frame. Continuity between the surrounding simulations and the animated mesh is much weaker; the initial velocities for the simulation may not match the velocities from the animation. Seamless integration of the target simulation and the animated mesh is another good direction for future work.

The transformation of the apparently disordered fluid into the target shape may look excessively perfect. One suggested workaround is to augment the target shape with a collection of distractors. The source simulation is also a source of distraction. Mechanisms to further embellish the movement and visually improve integration of the multiple simulations can be explored in the future.

As described, we sample the target shape with particles. The fidelity with which the target shape can be represented is limited by the particle spacing: surface detailing or fine-scale structures will not be reproduced if below the particle distance.

We described a system that works with particle-based simulations. Purely grid-based simulations are not supported, primarily because our approach for merging simulations is based on particle trajectories. Modifying the merging process could open the door to allowing grid-based solvers to use our method.

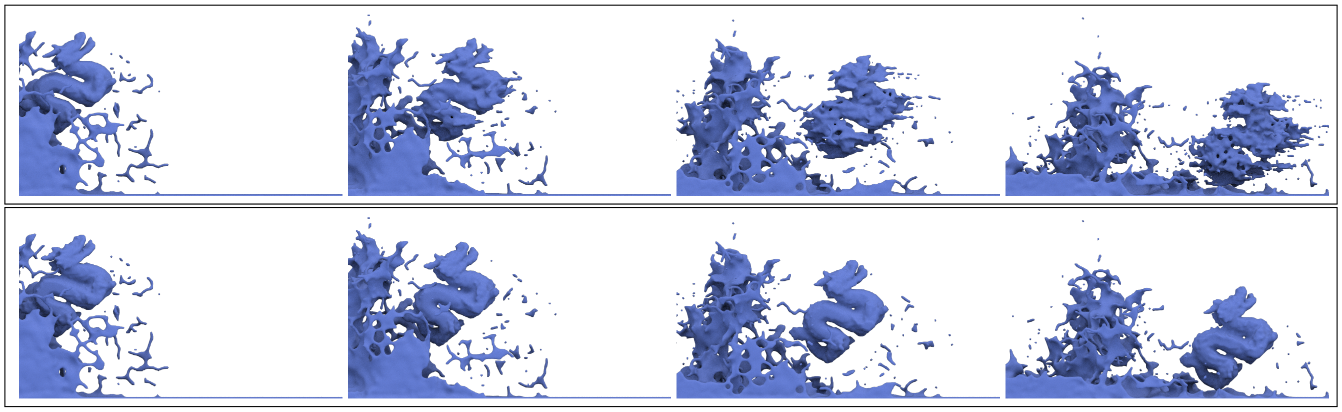

**Figure 6: Dragon emerging from splash. Top: single target shape (first frame shown). Bottom: multiple target shapes. Using multiple target shapes ensures that the dragon shape is stable over a period of time.**

## 5.2 Performance

**Table 2: Single targets. Timing figures are per frame; particle count is for the target simulation.**

| Scenario | Shape | Particles | Source | Target |
|---|---|---|---|---|
| splash | star | 2000 | 10s | 6s |
| dam break | dragon | 6600 | 14s | 6s |
| double dam break | bunny | 7500 | 16s | 7s |

In the previous section, we showed several art-directed fluid simulations. Here, we report some statistics summarizing the computational expense of the simulations. The computational cost of calculating initial conditions for the target simulation was negligible for single target (less than one second); for multiple frames it is more expensive, since cross-simulation comparisons are needed, but still below one second per target simulation. Of greater interest are the simulation times, shown in Table 2 and Table 3. We report simulation times averaged over active frames so as to normalize for the length of each animation. Postprocessing requires an additional second or two for single target shapes, and as much as 3s per target simulation to consolidate all simulations in the scenarios with multiple target shapes. Timing is with respect to a single-threaded implementation on a PC equipped with a i7-600 Intel processor and 16GB RAM.

The particle counts for the target simulations are in the low thousands, a small fraction of the overall particle count. The source simulations typically had on the order of 500k particles. Because we have separated the source and target simulations, and the greater part of the computational expense is in the source simulation, iteration on the art-directed portion of the animation is expedited: the animator can experiment with different settings, target shapes and placements, and so on, without needing to rerun the entire simulation. For our implementation, the reported simulation times for multiple target frames are increased by the need to write particle data to disk, in order to enable later cross-simulation trajectory comparisons and merging.

**Table 3: Multiple targets. Timing figures are per frame; particle count is typical number of target particles per frame.**

| Scenario | # Targets | Particles | Source | Target |
|---|---|---|---|---|
| flying dragon | 16 | 8000 | 23s | 9s |
| running horse | 48 | 23,000 | 10s | 15s |
| face in waterfall | 48 | 14,500 | 18s | 32s |

## 6 CONCLUSION

In this work, we have presented a new art-directable control approach for liquid simulations by connecting forward and time-reversed backward target simulations and merging them into a source simulation. Vigorous motion in the source simulation provides visual motivation for the target fluid animation, while use of simulation for the target particle motion ensures a convincing sense of fluid motion.

Our proposed art-directable control approach achieves the desired fluid target shape at the desired frame without introducing artificial control forces that tamper with the natural fluid motion. Our time-reversal art-directable control approach is dedicated to liquid animation, as opposed to the majority of work in this domain that is intended for smoke animation.

One direction for future work can involve investigating how to make the system operate in real time. The main bottleneck is the fluid solver; a simpler real-time particle effect engine could be employed instead, with loss of visual quality but vastly increased speed. In such a setting, the fluids would be directly rendered as particles rather than first being converted to a mesh.

Future work will also involve exploring more elaborate strategies for merging source and target simulations, as well as better integrating animated meshes with the surrounding simulations. A multi-pass strategy could be employed to iteratively modify the source simulation by progressively adding target elements, and a similar idea could be used for multiple target shapes in sequence. Such a strategy should work well with our approach as the simulations do not interfere with each other. Our main insight of combining backward and forward simulations, where the backward

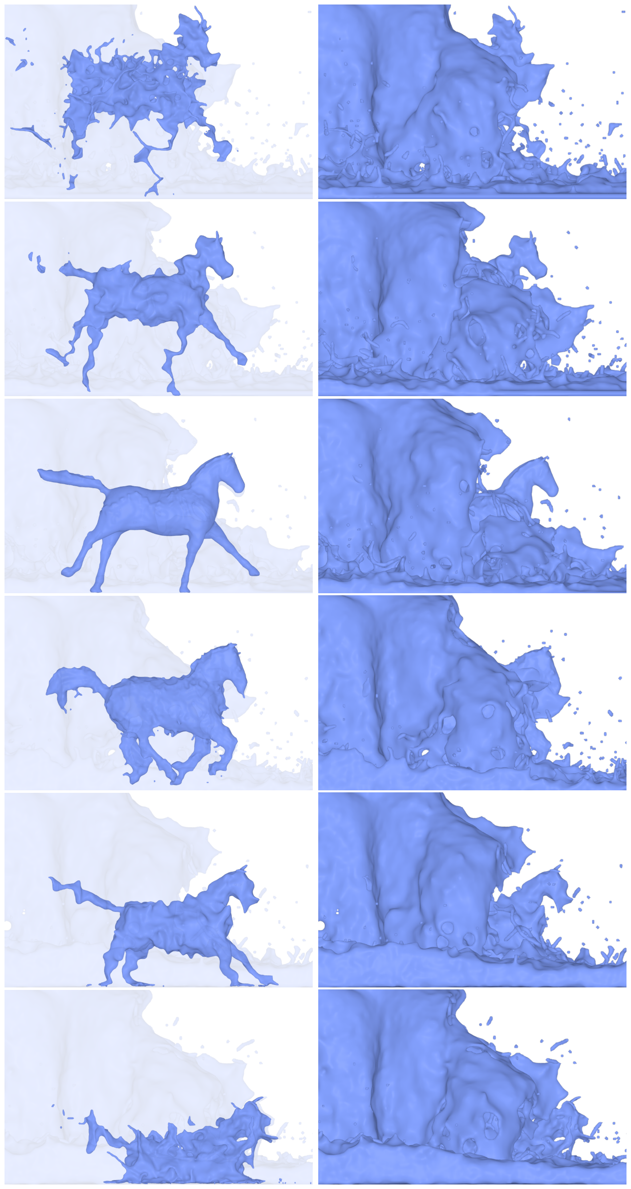

**Figure 7: Running horse target simulation (left) and combined simulation (right). First two frames: prior to animation keyframes, horse shape has not yet formed. Middle two frames: during animation, horse is fully formed. Final two frames: after final keyframe, horse is dissolving.**

simulation involves a forward simulation played in reverse order, ensures excellent quality in the resulting fluid motion.

## REFERENCES

Raanan Fattal and Dani Lischinski. 2004. Target-Driven Smoke Animation. *ACM Trans. Graph.* 23, 3 (aug 2004), 441–448.

Nick Foster and Ronald Fedkiw. 2001. Practical animation of liquids. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 23–30.

Nick Foster and Dimitris Metaxas. 1997. Controlling fluid animation. In *Proceedings Computer Graphics International*. IEEE, 178–188.

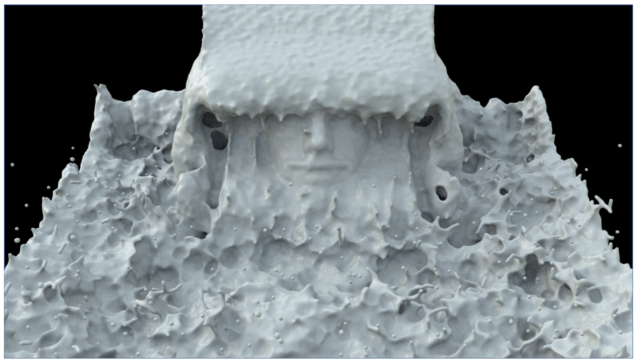

**Figure 8: A frame from the face simulation.**

Samuel Giraud-Carrier. 2020. *Retiming Smoke Simulation Using Machine Learning*. Ph. D. Dissertation. Brigham Young University.

Jamie Madill and David Mould. 2013. Target particle control of smoke simulation. In *Proc. of Graphics Interface*. CHCCS, 125–132.

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

Michael B. Nielsen and Robert Bridson. 2011. Guide Shapes for High Resolution Naturalistic Liquid Simulation. *ACM Trans. Graph.* 30, 4, Article 83 (jul 2011), 8 pages.

Jeremy Oborn, Sean Flynn, Parris K Egbert, and Seth Holladay. 2018. Time-Reversed Art Directable Smoke Simulation.. In *Eurographics (Short Papers)*. 1–4.

Zherong Pan, Jin Huang, Yiying Tong, Changxi Zheng, and Hujun Bao. 2013. Interactive localized liquid motion editing. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 1–10.

Zherong Pan and Dinesh Manocha. 2016. Efficient optimal control of smoke using spacetime multigrid. *arXiv preprint arXiv:1608.01102* (2016).

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

Karthik Raveendran, Nils Thuerey, Christopher J Wojtan, and Greg Turk. 2012. Controlling liquids using meshes. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

Karthik Raveendran, Chris Wojtan, Nils Thuerey, and Greg Turk. 2014. Blending liquids. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–10.

Donald Shepard. 1968. A Two-Dimensional Interpolation Function for Irregularly-Spaced Data. In *Proceedings of the 1968 23rd ACM National Conference*. 517–524.

Alexey Stomakhin and Andrew Selle. 2017. Fluxed animated boundary method. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–8.

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

Adrien Treuille, Antoine McNamara, Zoran Popović, and Jos Stam. 2003. Keyframe Control of Smoke Simulations. *ACM Trans. Graph.* 22, 3 (jul 2003), 716–723.

Christopher D. Twigg and Doug L. James. 2008. Backward Steps in Rigid Body Simulation. *ACM Trans. Graph.* 27, 3 (aug 2008), 1–10.