Review of Editing Fluid Animation Using Flow Interpolation

Authors Syuhei Sato, Yoshinori Dobashi, and Tomoyuki Nishita conducted a 2018 study concerning the improvement of fluid animations, in terms of their realistic feeling and computational expensiveness. They documented their findings in an article titled "Editing Fluid Animation Using Flow Interpolation," published by the Association for Computing Machinery in their journal titled *ACM Transactions on Graphics* in 2018. Cited in this article was another article published in *ACM Transactions on Graphics* in 2004 titled "Target-Driven Smoke Animation." The 2004 article, by authors Raanan Fattal and Dani Lischinski, is used as a secondary reference.

Syuhei Sato, Yoshinori Dobashi, and Tomoyuki Nishita (also referred to as the "researchers") conducted their study (the "study") with the intent to develop a computational model that produced more realistic fluid animations which required fewer computing resources. The term "fluid animation" refers to computer generated renderings of liquids and gasses, and the researchers focused their efforts on improving the rendering of smoke. Re-use of current fluid animations is common because they require large amounts of resources to generate. Fluid animations are used in industries such as film and video game creation. The researchers attempted to reduce the amount of re-use necessary by developing more efficient algorithms to generate additional renderings given input animations.

Fluid animations are becoming increasingly necessary to today's multimedia and content creation. An increasing demand for fluid animations calls for the generation of more renderings at a rate that current methods can't scale too. In modern computer graphics, it's important to have the ability to render animations efficiently and reliably.

"Target-Driven Smoke Animation," one of the article's primary references, reports the findings of Fattal's and Lischinski's study of computer-generated smoke renderings. This article describes some of the challenges encountered when attempting to produce digital animations of smoke. At its core, smoke animation at the time were described as "highly computationally intensive, because it solves a very difficult multi-dimensional non-linear optimization problem" (Fattal & Lischinski, 2004). Existing methods involved in the animation generating process involved attempting to estimate input parameters that would be present in an environment where smoke would exist. These parameters included wind speed, temperature, and other environmental factors. Optimization of an animation involved brute-force algorithmic approaches, which would simply not scale when more complex animations were desired.

Fattal and Lischinski instead focused their study on developing a target-driven algorithm that would estimate what the next state of the smoke animation would be based on its current state. They perform a few modifications to a general flow animation equation known as the Euler equation, which calculates fluid velocity at time τ. This algorithm was modified by adding a computed force named "the driving force", and offset by subtracting a value for desired viscosity, specified by the animator.

The most important modification to this equation is the incorporation of "the driving force." The goal of this formula is to "exert forces on the fluid in such a manner that the resulting flow will carry the smoke to the prespecified target density" (Fattal & Lischinski, 2004). It does this with two input parameters – the current density of the smoke, and the target density. The driving force's purpose is to incorporate the current and desired densities of smoke when attempting to estimate the velocity of the dissipating smoke. The modified Euler equation is controlled by four constant global parameters: a

smoothing constant, a driving force strengthening or weakening constant, a viscosity constant, and a speed dissipation constant.

The work of Fattal and Lishinski influenced Sato, Dobashi and Nishita in that their algorithm requires an input of a set of velocities of smoke over time. The researches stated that they used both the Navier-Stokes equation mentioned in the Fattal and Lishinski's article or the Euler equation which theirs was built off of.

The algorithm developed in the study conducted by Sato, Dobashi and Nishita aims to efficiently generate a new fluid animation using the previous fluid animation(s). The goal of this process is to allow a user to seamlessly cut and paste parts of an existing animation together in order to form a unique animation. There are two important processes undertaken in the method. One is called frame matching, and the other is called flow interpolation. At an abstract level, frame matching is a process that finds the point most similar in a user specified area in both source animation the target animation, and flow interpolation does the work of copying and making the source object fit into the target.

The frame matching process ensures that the region from the source animation (source region) is placed into the best possible time in the target animation (the target region). This algorithm compares the densities of the two regions over time and calculates "a temporal offset such that the correlation [between the densities] is maximized" (Sato, Dobashi, & Nishita, 2018). The purpose of doing this calculation is because the point in time when the densities are the closest to each other is also the point in time where the source and target region are the most similar to each other. The temporal offset is the difference in time between $\tau 0$ and the time that the most correlated densities exists.

After the frame matching process is complete, the algorithm begins the flow interpolation process. The first step is inserting the source region into the target region. This functions like a simple copy and paste, without worrying about smoothing yet. The step that this process relies on, and subsequently the reason it works as well as it does, is that of smoothing (blending) the region surrounding the target region. The researches state that they define an arbitrarily sized region surrounding the border of the target region, called the interpolation region. This is where the majority of the computation is done. The researchers developed their algorithm with the intent of minimizing the divergence of the velocities in the interpolation region. The researches defined a function (which they call the "energy function") that takes into account incompressibility and momentum, as failing to do so resulted in too smooth flows in the interpolation region. The article states "To obtain a smooth velocity field, we want to minimize the spatial change in the velocities. This can be achieved by minimizing the L2 norm of the gradient of the velocity" (Sato, Dobashi, & Nishita, 2018).

To touch on the results of the study, first it's important to discuss the increased efficiency of using the researchers' implementation. Similar to the smoke animations that Fattal and Lischinski sought to improve, the best way to accomplish the task that flow interpolation accomplishes would be to re-run a full simulation, tweaking input parameters. The researchers' algorithm works much faster re-running a full simulation, as can be observed in Exhibit A, which was included from the researchers' article. Two additional diagrams (Exhibits B and C) have been included to demonstrate how the process works, and what final results look like, respectively. The researchers were able to effectively implement an algorithm that can combine fluid animations efficiently. In the future, these findings can be applied in animation software, giving animators the opportunity to generate realistic, new fluid animations.

Exhibit A – Comparison of Fluid Interpolation and Full Simulation

Figure	Grid Size		Size of Interpolation Region		T_{match}	T_{interp}	T_{sim}
	Input	Final	Width	Total			
Figure 6	$192 \times 192 \times 256$	$192 \times 192 \times 256$	12	128×10^4	968	11	210
Figure 7(d)	$192 \times 192 \times 256$	$192 \times 192 \times 320$	4	$(192 \times 192 \times 4) \times 16$	–	7.5	280
Figure 7(g)	$192 \times 192 \times 256$	$240 \times 192 \times 256$	4	$(4 \times 192 \times 256) \times 12$	–	8.5	310
Figure 8(b)	$256 \times 256 \times 64$	$256 \times 256 \times 64$	12	21×10^4	90	1.4	65
Figure 8(d)	$256 \times 256 \times 64$	$256 \times 256 \times 64$	12	38×10^4	221	2.5	65
Figure 8(f)	$256 \times 256 \times 64$	$256 \times 256 \times 64$	12	41×10^4	257	3.0	65
Figure 9	$256 \times 256 \times 64$	$256 \times 256 \times 64$	12	44×10^4	–	3.1	65
Figure 10	$384 \times 384 \times 512$	$384 \times 384 \times 640$	8	$(384 \times 384 \times 8) \times 16$	–	100	3500
Figure 1	$768 \times 256 \times 192$	$768 \times 764 \times 192$	8	$(768 \times 8 \times 192) \times 2$	–	15	6300

T_{match} and T_{interp} are computation times corresponding to the frame matching and flow interpolation, respectively. T_{sim} is measured by running a fluid simulation with the grid size the same as the final results. T_{match} is measured in seconds, and T_{interp} and T_{sim} are measured in seconds per frame.

(Sato, Dobashi, & Nishita, 2018)

Exhibit B – Diagram of the Flow Interpolation Process

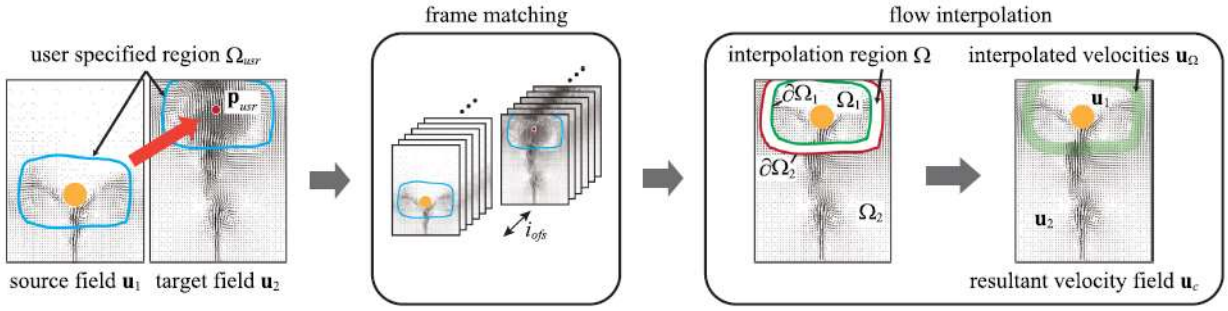
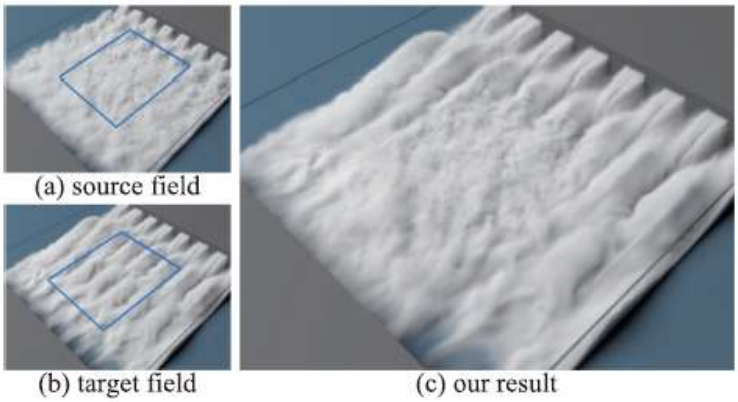


Fig. 2. Overview of our method. $\mathbf{p}_{usr}$ is a user-specified position where Ω_{usr} is placed in a target field, i_{ofs} indicates a temporal offset, and $\partial\Omega_1$ and $\partial\Omega_2$ are boundaries of Ω. For visual clarity, 3D velocity fields are shown as 2D velocity fields. The orange circles indicate obstacles.

*Note that in the frame matching step, the stacked frames represent frames over time.

(Sato, Dobashi, & Nishita, 2018)

Exhibit C – Sample Input and Output



(Sato, Dobashi, & Nishita, 2018)

All exhibits are included for further explanation and to provide visual detail.

References

Fattal, R., & Lischinski, D. (2004). Target-Driven Smoke Animation. *ACM Trans. Graph. 23, 3*, 439-446.

Sato, S., Dobashi, Y., & Nishita, T. (2018). Editing Fluid Animation Using Flow Interpolation. *ACM Trans. Graph. 37, 5*, 173-185.