Eurographics Workshop on Natural Phenomena (2007)
D. Ebert, S. Mérillou (Editors)

Eulerian Motion Blur

Doyub Kim[†] and Hyeong-Seok Ko[‡]

Seoul National University

Abstract

This paper describes a motion blur technique which can be applied to rendering fluid simulations that are carried out in the Eulerian framework. Existing motion blur techniques can be applied to rigid bodies, deformable solids, clothes, and several other kinds of objects, and produce satisfactory results. As there is no specific reason to discriminate fluids from the above objects, one may consider applying an existing motion blur technique to render fluids. However, here we show that existing motion blur techniques are intended for simulations carried out in the Lagrangian framework, and are not suited to Eulerian simulations. Then, we propose a new motion blur technique that is suitable for rendering Eulerian simulations.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1. Introduction

Motion blur is essential for producing high-quality animations. The frame rate of most films and videos is either 24 or 30 Hz, whereas human vision is reported to be sensitive up to 60 Hz [Wan95, CJ02]. Due to the lower frame rate of film and video, when each frame is drawn as a simple instantaneous sampling of the dynamic phenomena, artifacts such as temporal strobing can occur. The graphics community has been aware of this problem, and several motion blur techniques have been proposed to solve this problem.

Fluids are often important elements of a dynamic scene, and for the artifact-free production of such a scene, fluids need to be rendered with motion blur. Since the graphics field already has an abundance of motion blur techniques, one may consider applying existing techniques to the motion blur of fluids. Unfortunately, existing techniques do not produce satisfactory results. This paper describes why the existing solutions do not work for fluids and how to modify existing motion blur techniques to make them applicable to fluids.

Motion blur techniques developed so far are intended for

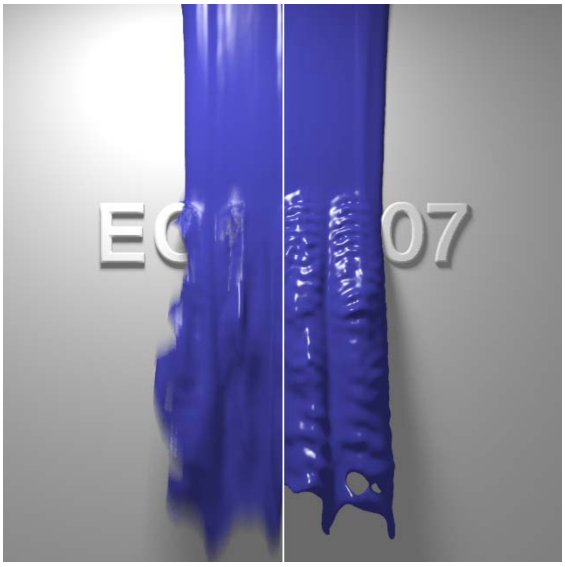


Figure 1: *A motion blurred image (left) produced with the algorithm presented in this paper and an unblurred image (right): A slice of water is falling along the wall, which hits the logo and makes the splash. To factor out the effects caused by the transparent material, we rendered the water as opaque.*

[†] kim@graphics.snu.ac.kr
[‡] ko@graphics.snu.ac.kr

rendering simulations that are performed in the Lagrangian framework.† We will call this type of motion blur techniques *Lagrangian motion blur* (LMB). The majority of objects encountered in 3D graphics scenes (including rigid bodies, articulated figures, deformable solids, and clothes) are simulated in the Lagrangian framework; thus their motion blur can be readily rendered with LMB.

Simulation of fluids, however, is often carried out in the Eulerian framework. Considering the high quality and broad applicability of LMB, and considering there is no specific reason to discriminate fluids from other 3D objects, one may consider employing LMB for rendering fluids. An interesting finding of this paper is that LMB is not suitable for rendering the results generated by an Eulerian simulation. So far no algorithm has been proposed that can properly render motion blur of fluids that are simulated using the Eulerian framework. In this paper, we explain why Lagrangian motion blur should not be used for rendering Eulerian simulations. Insight obtained during this process led us to develop a simple step that can be added to existing motion blur techniques to produce motion blur techniques that are applicable to Eulerian simulations (i.e., *Eulerian motion blur* (EMB)).

2. Previous Work

Motion blur was first introduced to the computer graphics field by Korein and Badler [KB83], and Potmesil and Chakravarty [PC83]. Korein and Badler proposed a method that works on an analytically parameterized motion and creates a continuous motion blur. Potmesil and Chakravarty proposed another method that creates continuous motion blur by taking the image-space convolution of the object with the moving path. We will classify the above sort of motion blur techniques as *analytic methods*.

The next class of motion blur we introduce is the *temporal super-sampling methods*. Korein and Badler [KB83] proposed another method that renders and accumulates whole (not partial) images of the object at several super-sampled instants, resulting in a superimposed look of the object. The distributed ray tracing work of Cook et al. [CPC84] brought improved motion blur results. Their method successfully increased the continuity of the motion blur by retrieving pixel values from randomly sampled instants in time. Recently, Cammarano and Jensen [CJ02] extended this temporal super-sampling method to simulate motion-blurred global illumination and caustics using ray tracing and photon mapping.

The third class of motion blur is known as *image-based methods*. Max and Lerner [ML85] proposed an algorithm to achieve motion blur effect by considering the motion on the image plane. Brostow and Essa [BE01] also proposed an entirely image-based method which can create motion blur from stop motion or raw video image sequences. These methods are suited to cases where the 3D motion is not available or the motion is already rendered. A more complete survey of motion blur techniques can be found in Sung et al. [SPW02]

We assume in this work that the 3D data of the fluid at every frame are available, but the data are not given in a parameterized forms. Therefore the temporal super-sampling method seems to fit to the situation. In this paper, we develop a motion blur technique based on the temporal super-sampling method.

Realistic rendering of fluids has been studied as well as the fluid simulation itself in the graphics community. Fedkiw et al. [FSJ01] visualized smoke simulation using a Monte Carlo ray-tracing algorithm with photon mapping, and Nguyen et al. [NFJ02] presented a technique based on Monte Carlo ray tracing for rendering fire simulations. Techniques for rendering liquids were also developed by Enright et al. [EMF02]. However, motion blur was not considered in those studies.

Müller et al. [MCG03] used blobby style rendering for visualizing water represented with particles, and their method was subsequently improved by Zhu and Bridson [ZB05] to have smoother surfaces. For the visualization of Lagrangian particles, Guan and Mueller [GM04] proposed point-based surface rendering with motion blur. Geundelman et al. [GSLF05] and Lossaso et al. [LIG06] attempted to include the rendering of the escaped level-set particles to create the impression of water sprays.

Motion blur of Eulerian simulation has rarely mentioned/practiced before; To our knowledge, there have been only two reports on motion blur of Eulerian simulations in computer graphics thus far. In rendering water simulation, Enright et al. [EMF02] mentioned that a simple interpolation between two signed distance volumes can be applied in order to find ray and water surface intersection. A few years later, Zhu and Bridson [ZB05] mentioned that the method will destroy surface features that move further than their width in one frame.

3. Computing Motion Blur

The basic principle of motion blur is to add up the radiance contributions over time, which can be expressed as

$$L_p = \int_{t_s} \int_A L(x', \vec{\omega}, t) s(x', \vec{\omega}, t) g(x') dA(x') dt, \qquad (1)$$

† We note the intrinsic differences of the physical quantities used in the Lagrangian and Eulerian frameworks. In the Lagrangian framework, the simulator deals with the quantities *carried* by the moving objects (e.g., the position, velocity, acceleration of the objects). In the Eulerian framework, on the other hand, the domain is discretized into grids and the simulator deals with the quantities *observed* from fixed 3D positions (e.g., the velocity and density of the fluid at a fixed grid point).

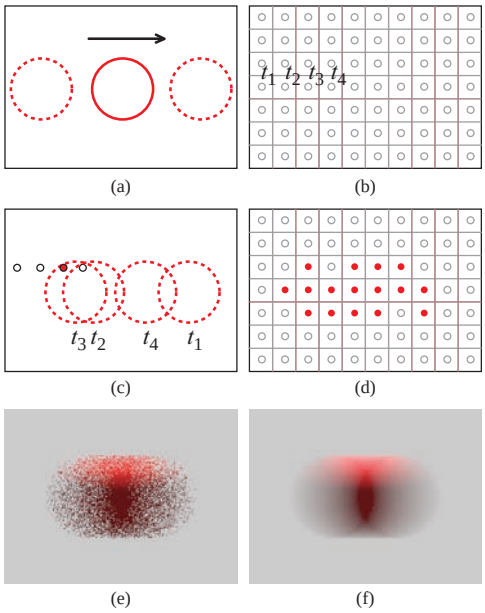


Figure 2: *Motion blur with temporal super-sampling*

where $g()$ is the filter function, $s()$ represents the shutter exposure, and $L()$ is the radiance contribution from the ray [CJ02]. The above principle applies to both Lagrangian and Eulerian motion blurs. In the equation, x is the place where the movement of objects jumps into the motion blur; for the evaluation of x', the locations of the objects at arbitrary (super-sampled) moments need to be estimated, which forms a core part of motion blur.

For the development of a motion blur technique based on temporal super-sampling, we use Monte Carlo integration. It computes the integral in Equation (1) by accumulating the evaluations of the integrand at super-sampled instants.

More specifically, imagine the situation shown in Figure 2 (a) in which a ball is moving horizontally. Suppose that we have to create a blurred image for frame t^n. Let η be the shutter speed. For each pixel, we associate a time sample picked within the interval $[t^n - \eta/2, t^n + \eta/2]$; the samples are taken from both past and future. Figure 2 (b) shows that, for example, the time samples t_1, t_2, t_3, and t_4 (which do not need to be in chronological order) are associated with the four pixels in a row.

For each pixel, we now shoot the ray at the associated time, test for intersection, and estimate the radiance contribution. Shooting a ray at a certain time and testing for intersection implies that the location of the objects at that time should be estimated. Figure 2 (c) shows the object locations at t_1, t_2, t_3, and t_4. In this particular example, only the ray shot at t_3 hits the moving object. Figure 2 (d) shows the final result. Figure 2 (e) shows an image produced with an actual ray tracer. Usually multiple rays are shot for each pixel for better results (Figure 2 (f)), which can be easily done by associating multiple time samples to a pixel.

4. Lagrangian Motion Blur

LMB is used for rendering objects that have explicit surfaces such as rigid bodies, deformable solids, and clothes. The core part of the LMB approach is to compute, from the given 3D data of each frame, ray–object intersection at arbitrary super-sampled instants. In order to do this, the location of the surface at an arbitrary moment has to be estimated. In LMB, the estimation is done by taking the *time-interpolation* of the vertices of the two involved frames; When the positions $\mathbf{x}(t^n)$ and $\mathbf{x}(t^{n+1})$ of the vertices at t^n and t^{n+1} are given, the estimated position $\mathbf{x}_L(\tau)$ at super-sampled time τ is calculated by

$$\mathbf{x}_L(\tau) = \frac{\tau - t^n}{t^{n+1} - t^n}\mathbf{x}(t^{n+1}) + \frac{t^{n+1} - \tau}{t^{n+1} - t^n}\mathbf{x}(t^n). \qquad (2)$$

We now briefly consider the physical meaning of the estimation given by rearranging Equation (2) into the form

$$\mathbf{x}_L(\tau) = \mathbf{x}(t^n) + (\tau - t^n)\frac{\mathbf{x}(t^{n+1}) - \mathbf{x}(t^n)}{t^{n+1} - t^n}. \qquad (3)$$

This equation shows that the estimation is the result of assuming the movement was made with a constant velocity $(\mathbf{x}(t^{n+1}) - \mathbf{x}(t^n))/(t^{n+1} - t^n)$. However, how valid is this assumption? Movement of any object with non-zero mass has the tendency to continue its motion, and thus has an inertial component. When specific information is not available, calculation of the object position based on the inertial movement turns out to give quite a good estimation in many cases, judging from images rendered using LMB. The error of the estimation is proportional to the acceleration.

5. Eulerian Motion Blur

In developing Eulerian motion blur, we assume that the simulation result for each frame is given in the form of 3D grid data. The grid data consists of the level-set (or density) and velocity fields. As in the Lagrangian motion blur, it is necessary to know how a ray traverses the fluid at an arbitrary super-sampled instant. However, rendering Eulerian simulations needs a different type of information: instead of the ray-surface intersection, the required information is the level-set (in the case of water) or density (in the case of smoke) values at the cell corners of all the cells the ray passes.[‡]

‡ When the fluid has a clear boundary, as is the case for water, the surface can be extracted from a Eulerian simulation using the marching cube algorithm [LC87]. In such a case, rendering can be done with ray–surface intersections. However, this approach is not applicable to surfaceless fluids such as smoke which do not have

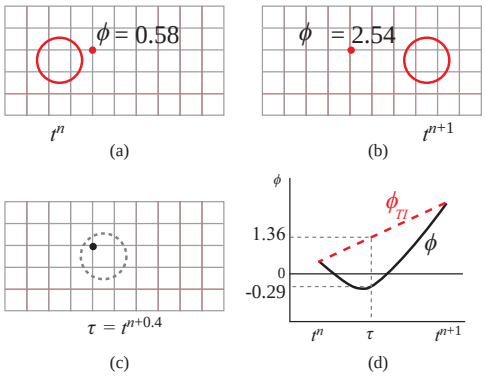


Figure 3: *Characterization of the level-set change in a simple example: (a) the snapshot at t^n, (b) the snapshot at t^{n+1}, (c) the situation at $t^{n+0.4}$, (d) the level-set changes.*

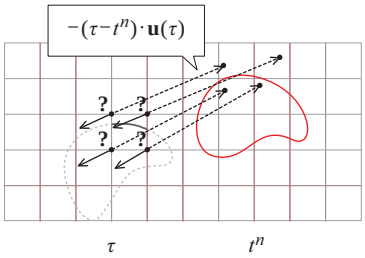


Figure 4: *Estimation of the level-set values for Eulerian motion blur. The grid points marked with ?s are the locations whose level-set values must be estimated. The short solid arrows at those points represent the estimated velocity* $\mathbf{u}(\mathbf{x},\tau)$.

5.1. Why Time-Interpolation Does Not Work

Since the grid data are available only for the frames, we must somehow *estimate* the level-set values at arbitrary time sample τ. For the estimation, Enright et al. [EMF02] presented a method which interpolates the level set data between two frames. Note that this is just as same as LMB-style estimation. An LMB-style solution would be to make the estimation with

$$\phi_{TI}(\tau) = \phi(t^n) + (\tau - t^n)\frac{\phi(t^{n+1}) - \phi(t^n)}{t^{n+1} - t^n}. \qquad (4)$$

Contrary to expectation, the above estimation gives incorrect results. Imagine the simple case shown in Figure 3, in which a spherical ball of water is making a pure translational movement along the horizontal direction at a constant velocity. Figures 3 (a) and 3 (b) show two snapshots taken at t^n and t^{n+1}, respectively. At the marked grid point, the level-set values are $\phi(t^n) = 0.58$ and $\phi(t^{n+1}) = 2.54$. The question is what would be the level-set value $\phi(\tau)$ at $\tau = t^{n+0.4}$ at that position? Since the fluid movement is analytically known in this example, we can find out the exact location of the water ball at τ, as shown in Figure 3(c). At τ, the marked position comes within the body of fluid; therefore $\phi(\tau)$ has a negative value. In fact, we can find out the trajectory of $\phi(t)$ for the duration $[t^n, t^{n+1}]$, which is plotted as a solid curve in Figure 3(d). On the other hand, the time-interpolated result is $\phi_{TI}(\tau) = 1.36$, which is far from what has happened. Variation of $\phi_{TI}(t)$ within the duration follows a straight line and is plotted with a dashed line in Figure 3(d). Here, we note that (1) the time-interpolation gives an incorrect result even in such a simple, non-violent, analytically verifiable case; (2) the error is remarkable; and (3) the error is not related to the grid resolution.

We now investigate why the time-interpolation gives such an incorrect result. When specific information about the movement is not available, exploiting the inertial component of the movement works quite well. The reason the LMB method works so well for Lagrangian simulations can be attributed to the fact that the LMB-estimation of object location exploits the inertia. We can adopt this idea of *exploiting inertia* in the development of Eulerian motion blur. A question that arises here is whether the time-interpolation ϕ_{TI} is exploiting the inertia.

It is critical to understand that it cannot be assumed that the level-set/density change at a grid point will continue to happen at the current rate. The space in which the fluid experiences inertia in the conventional sense is the 3D space. The inertial movement of the fluid in 3D space is *reflected* to the level-set field by updating the level-set according to the equation

$$\frac{\partial \phi}{\partial t} + \mathbf{u}\cdot\nabla\phi = 0. \qquad (5)$$

This equation states that the level-set should be advected in the direction $\mathbf{u}$ at the rate $|\mathbf{u}|$.

5.2. Proposed Method

For the Eulerian motion blur to exploit the inertial movement of fluids, therefore, we propose that the estimation of the level-set values at arbitrary super-sampled instants be based on the level-set advection, rather than the time-interpolation of the level set values. More specifically, we propose to estimate $\phi_E(\mathbf{x},\tau)$ of the level set value at a 3D position $\mathbf{x}$, at a super-sampled time τ with the semi-Lagrangian advection [Sta99, SC91]

$$\phi_E(\mathbf{x},\tau) = \phi(\mathbf{x} - (\tau - t^n)\cdot\mathbf{u}(\mathbf{x},\tau), t^n). \qquad (6)$$

This equation states that $\phi_E(\mathbf{x},\tau)$ takes the level-set value of t^n at the back-tracked position $\mathbf{x} - (\tau - t^n)\cdot\mathbf{u}(\mathbf{x},\tau)$. In the

a distinct boundary. Even for cases where the surface extraction is possible (as in the water), when the topology changes over frames, LMB is difficult because finding the vertex correspondence is a non-trivial process.

above procedure, we have not yet described how to estimate the 3D velocity $\mathbf{u}(\mathbf{x},\tau)$ at τ. However, the velocity can be estimated with another advection, but this time on the velocity:

$$\mathbf{u}(\mathbf{x},\tau) = \mathbf{u}(\mathbf{x} - (\tau - t^n)\cdot\mathbf{u}(\mathbf{x},t^n),t^n). \quad (7)$$

The above procedure works regardless of whether τ is in the past or future side of t^n. If needed, we can increase the accuracy by adopting the 3rd-order CIP method for the advection [SSK05, YXU01].

It is clear when $|(\tau - t^n)\cdot\mathbf{u}(\mathbf{x},\tau)|/\Delta x$ in Equation (6) gets larger, error from the estimation also increases. The artifact from this error can be quite visible when fluid has fast rotational movements. We can simply cure this problem by adopting sub-stepping technique which is commonly used by particle tracing method [EMF02, ZB05], so the each substep moves less than one grid cell.

5.3. Discussion

Normally the time-step taken for the fluid simulation is smaller than the frame duration. So, one may store the data for every simulation time step and perform the Lagrangian motion blur with those dense data, which will result in reduction of the interpolation error. However, the level-set or density data are usually large. So this approach will require a large storage space, and I/O process will critically slow down the simulation. The approach can also be problematic during the rendering stage: (1) For producing motion blurred image of a frame, the renderer will have to load the dense results. (2) Rendering the scene with multiple sets of data can be a lot more complex than the Eulerian motion blur. Those are the practical reasons this paper develops a motion blur technique which works on the sparse data.

6. Experimental Results

The technique presented in this paper was implemented on a Power Mac with G5 2.5 GHz processors. We applied it to several Eulerian simulations as described below. For all fluid simulations, a CFL restriction was 3. That is, the time-step $\Delta t < 3\Delta x/||\mathbf{u}||_{max}$ was used.

6.0.0.1. A Slice of Water Hitting a Logo Figure 1 shows the result of the EMB (with ϕ_E) when it was applied to a scene in which a slice of water falling along the wall hits the logo. This simulation was done by the level-set method with an grid resolution of $240\times320\times80$. The *degree* of motion blur can be controlled by giving different shutter speeds. Figure 5 shows the results; A longer shutter speed generates more blurred results. To factor out the effects caused by the transparent material, we rendered the water in opaque.

6.0.0.2. Chunks of Water Dropping onto Shallow Water Figure 6 shows four (blurred) snapshots taken from the simulation in which chunks of water were dropped onto shallow water. This simulation was done by octree-based level-set method [LGF04] with an effective grid resolution of $128\times192\times128$. Frames were dumped every 1/30 second and the rendering was done with a shutter speed of 1/60 second. Figure 7 shows the (enlarged) results of motion blur for a particular frame. Figures 7 (b) and (c) are the results generated with the level-set time-interpolation ϕ_{TI} and the EMB ϕ_E, respectively. Figure 7 (a) is the result without any motion blur, i.e., the un-blurred version of the scene. Although Figure 7 (a) looks crisper, the video may have temporal-strobing artifacts.

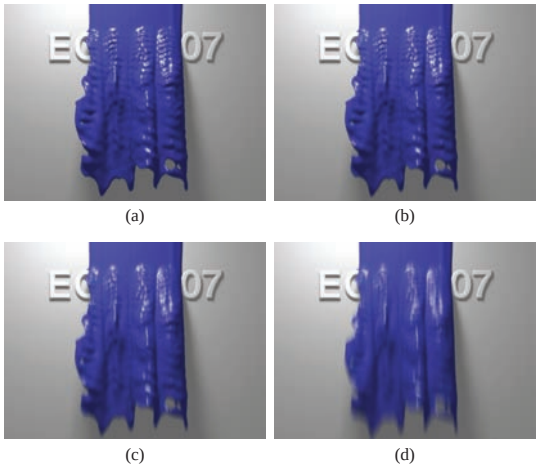


Figure 5: *Results of the Eulerian motion blur with different shutter speeds: (a) 1/500 sec., (b) 1/250 sec., (c) 1/125 sec., (d) 1/60 sec.*

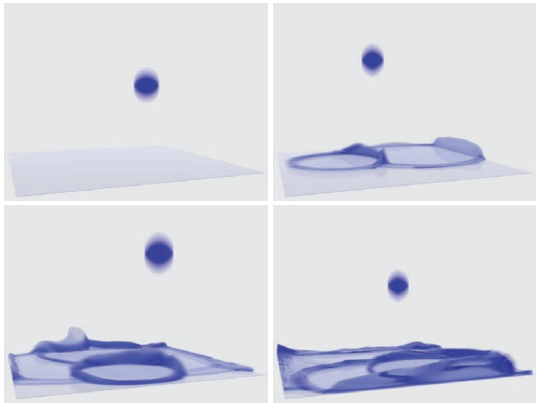


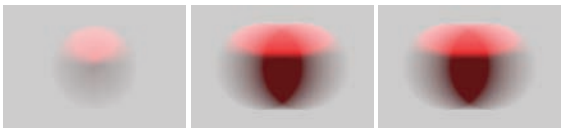
Figure 6: *Sequence of images taken from the simulation in which chunks of water are dropped onto shallow water.*

6.1. Quality Comparison of ϕ_{TI} and ϕ_E

The quality enhancement gained by employing ϕ_E instead of ϕ_{TI} is more clearly visible when they are applied to an

(a) (b) (c)

Figure 7: *Enlarged images of motion blur for a particular frame: (a) no motion blur, (b) motion blur with ϕ_{TI}, (c) motion blur with ϕ_E.*



(a) (b) (c)

Figure 8: *Quality comparison of ϕ_{TI} and ϕ_E in a simple example: (a) with ϕ_{TI}, (b) with ϕ_E, (c) the ground truth.*

analytically characterized fluid movement such as the one shown in Figure 3. Figure 8(a) and (b) show the results of motion blur when ϕ_{TI} and ϕ_E are used, respectively. In this example, evolution of the fluid surface over time is known analytically. Therefore we can use conventional LMB for rendering the ball. The result, shown in Figure 8(c), can be regarded as the ground truth image in this experiment. It can clearly be seen that the result produced with ϕ_{TI} is far from the ground truth image, whereas the result produced using ϕ_E properly generates the blurred accumulation of the horizontal movement.

6.2. Analysis of the Memory and Computation Requirements

We summarize the amount of space and computation needed for the EMB in Table 1, along with the amounts for the cases of no motion blur and motion blur with ϕ_{TI}, for side-by-side comparison. The statistics were taken while the images shown in Figure 7 were rendered. The statistics for motion blur with ϕ_{TI} are included here because it might be helpful for judging the memory and computation requirements of the EMB method in comparison to those of existing motion blur techniques. In this experiment, the image resolution was 640×480, and 64 samples were used per pixel. The same number of samples were used even for the no motion blur case, in order to see the incremental computation taken for the level-set/density estimation steps introduced in this paper. No-motion blur or motion blur with ϕ_{TI} dumps only either the level-set or the density (scalar) field. On the other hand, Eulerian motion blur additionally dumps the 3D velocity field, resulting in four times the dumped storage of the other two cases. In generating a blurred frame for t^n, motion blur with ϕ_{TI} refers the level-set fields of t^{n-1}, t^n, and t^{n+1}; therefore it loads three arrays (three units of space). Regardless of performing motion blur with ϕ_{TI} or ϕ_E, the level-set estimation for a cell requires only simple extra computations ($O(1)$-complexity and almost negligible) in addition to the basic rendering. However, when we measure the time taken to render a whole image (Table 1 (C)), motion blur required 103% more time compared to the case with no motion blur. This was because the total volume that the fluid occupies over a frame duration is significantly larger than the instantaneous volume, as summarized in Table 1 (D). The number of floating point operations for EMB was slightly larger than that for motion blur with ϕ_{TI}.

7. Conclusion

In this paper we classified existing motion blur techniques as the Lagrangian motion blur to emphasize that they are suited to rendering Lagrangian simulations. We showed that using the Lagrangian-style level-set/density interpolation ϕ_{TI} for Eulerian simulations produces incorrect results. We showed that the error is remarkably large and is not reducible by employing finer-resolution grids. The proposed method properly accounts for the inertial component of the fluid movement, by utilizing the advection equation for the estimation of the level-set/density at super-sampled instants. As a result, the proposed method successfully produces conventional kind of motion blur for Eulerian simulations.

Acknowledgment

We would like to thank Oh-young Song for his insightful comments. This work was supported by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (MOST) (National Research Laboratory

	No Motion Blur	Motion Blur with ϕ_{TI}	Motion Blur with ϕ_E
(A) Total dumped space (for the fluid)	$\times 1$	$\times 1$	$\times 4$
(B) Space loaded for generating one frame	$\times 1$	$\times 3$	$\times 4$
(C) Time taken for rendering the frame (in min.)	29	58	59
(D) Number of grid cells the fluid occupied	244711	441177	441177

Table 1: *Comparison of memory and computation requirements.*

M10600000232-06J0000-23210), Ministry of Information and Communication, the Brain Korea 21 Project, and Automation and System Research Institute at Seoul National University.

References

[BE01] BROSTOW G. J., ESSA I.: Image-based motion blur for stop motion animation. *Computer Graphics (Proc. ACM SIGGRAPH 2001) 35* (2001), 561–566.

[CJ02] CAMMARANO M., JENSEN H. W.: Time dependent photon mapping. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering* (2002), pp. 135–144.

[CPC84] COOK R. L., PORTER T., CARPENTER L.: Distributed ray tracing. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques* (1984), pp. 137–145.

[EMF02] ENRIGHT D., MARSCHNER S., FEDKIW R.: Animation and rendering of complex water surfaces. *ACM Transactions on Graphics 21*, 3 (2002), 736–744.

[FSJ01] FEDKIW R., STAM J., JENSEN H. W.: Visual simulation of smoke. *Computer Graphics (Proc. ACM SIGGRAPH 2001) 35* (2001), 15–22.

[GM04] GUAN X., MUELLER K.: Point-based surface rendering with motion blur. In *Proceedings of the 2004 Eurographics Symposium on Point-Based Graphics* (2004).

[GSLF05] GUENDELMAN E., SELLE A., LOSASSO F., FEDKIW R.: Coupling water and smoke to thin deformable and rigid shells. *ACM Transactions on Graphics 24*, 3 (2005), 973–981.

[KB83] KOREIN J., BADLER N.: Temporal anti-aliasing in computer generated animation. In *SIGGRAPH '83: Proceedings of the 10th annual conference on Computer graphics and interactive techniques* (1983), pp. 377–388.

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (1987), vol. 21, pp. 163–169.

[LGF04] LOSASSO F., GIBOU F., FEDKIW R.: Simulating water and smoke with an octree data structure. *ACM Transactions on Graphics 23*, 3 (2004), 457–462.

[LIG06] LOSASSO F., IRVING G., GUENDELMAN E.: Melting and burning solids into liquids and gases. *IEEE Transactions on Visualization and Computer Graphics 12*, 3 (2006), 343–352.

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2003), pp. 154–159.

[ML85] MAX N. L., LERNER D. M.: A two-and-a-half-d motion-blur algorithm. vol. 19, ACM Press, pp. 85–93.

[NFJ02] NGUYEN D. Q., FEDKIW R., JENSEN H. W.: Physically based modeling and animation of fire. *ACM Trans. Graph. 21*, 3 (2002), 721–728.

[PC83] POTMESIL M., CHAKRAVARTY I.: Modeling motion blur in computer-generated images. In *SIGGRAPH '83: Proceedings of the 10th annual conference on Computer graphics and interactive techniques* (1983), pp. 389–399.

[SC91] STANIFORTH A., CÔTÈ J.: Semi-lagrangian integration scheme for atmospheric model - a review. *Mon. Weather Rev. 119*, 12 (1991), 2206–2223.

[SPW02] SUNG K., PEARCE A., WANG C.: Spatial-temporal antialiasing. *IEEE Transactions on Visualization and Computer Graphics 8*, 2 (2002), 144–153.

[SSK05] SONG O.-Y., SHIN H., KO H.-S.: Stable but non-dissipative water. *ACM Transactions on Graphics 24*, 1 (2005), 81–97.

[Sta99] STAM J.: Stable fluids. *Computer Graphics (Proc. ACM SIGGRAPH '99) 33*, Annual Conference Series (1999), 121–128.

[Wan95] WANDELL B. A.: *Foundation of Vision*. Sinauer Associates, 1995.

[YXU01] YABE T., XIAO F., UTSUMI T.: The constrained interpolation profile method for multiphase analysis. *J. Comp. Phys. 169* (2001), 556–593.

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. *ACM Transactions on Graphics 24*, 3 (2005), 965–972.

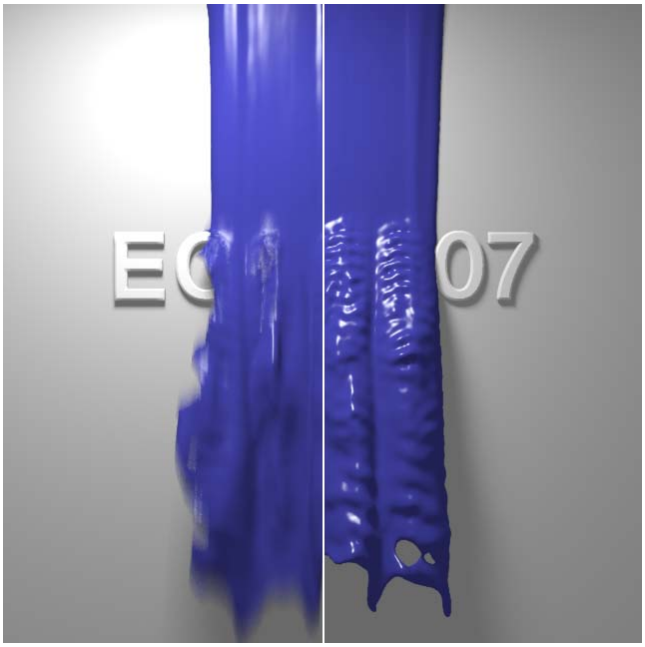


Figure 9: *A motion blurred image (left) produced with the algorithm presented in this paper and an unblurred image (right): A slice of water is falling along the wall, which hits the logo and makes the splash. To factor out the effects caused by the transparent material, we rendered the water as opaque.*

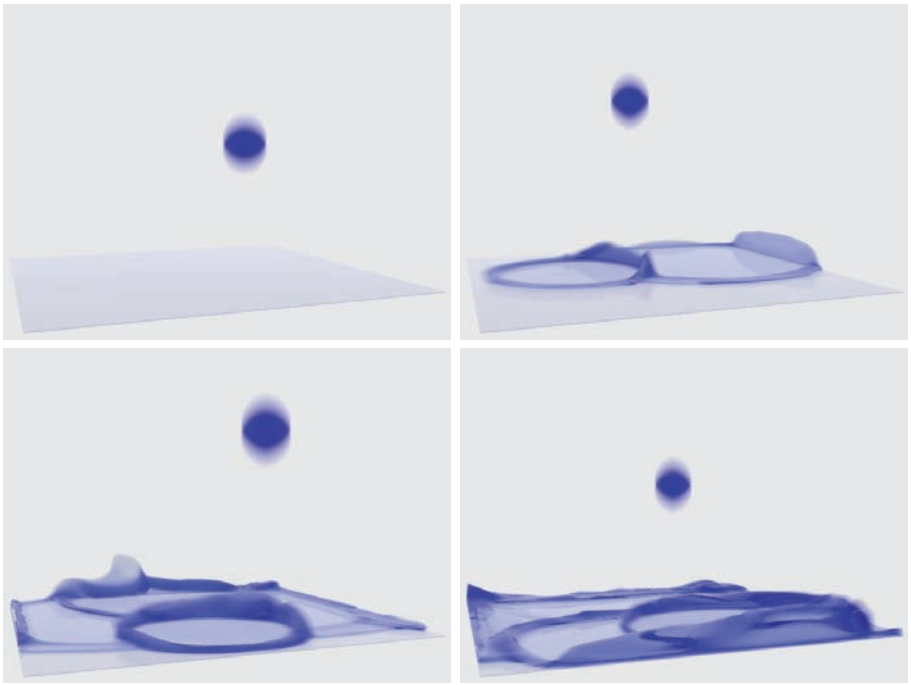


Figure 10: *Sequence of images taken from the simulation in which chunks of water are dropped onto shallow water.*