View-Dependent Adaptive Animation of Liquids

Janghee Kim, Insung Ihm, and Deukhyun Cha

Various adaptive mesh refinement techniques are often employed in numerical simulations for increasing spatial and temporal resolution beyond the limits imposed by available CPU time and memory space. Recently, an octree-based adaptive mesh structure was successfully used in fluid animation to place more grid cells efficiently in visually interesting regions of fluids. In an attempt to optimize the use of computational resources further in fluid animation, this paper extends this adaptive technique by modifying the mesh refinement scheme so that the camera's viewing properties are dynamically exploited during the simulation. Based on a simple adaptive mesh structure, we show that the new meshing strategy can save a substantial amount of computation time and memory space by using a view-dependent adaptive approach. The experimental results reveal that the proposed technique provides a good compromise between the computational effort and the simulation's fidelity, and may be used quite effectively in 3D animation production.

Keywords: Liquid animation, Navier-Stokes equations, adaptive simulation, view-dependent simulation.

I. Introduction

Even with the recent advances made in the production of realistic fluid animations, physically based fluid simulation on large computational domains still remains a challenge in computer graphics. The demand for high-resolution grids has become ever greater because they are essential for simulating large-scale natural phenomena or producing fluid effects with rich visual detail. Unfortunately, the number of cells of a 3D rectilinear grid increases cubically with respect to the resolution, easily resulting in an impractical requirement of computation time and memory space.

The use of adaptive simulation grids is a natural solution for alleviating this severe computational problem, and has frequently been dealt with in computational fluid dynamics. Recently, an adaptive or level-of-detail approach was applied successfully in the computer graphics community in an attempt to produce small-scale visual detail of fluids efficiently, such as rolling smoke and sheeting water [1]. In that work, an octree-based mesh structure was employed to place relatively more grid cells in visually interesting regions under some mesh refinement criteria. By adapting the Navier-Stokes equations solver to the adaptively refined octree data structure, it was possible to simulate fluids on grids of high effective resolution at reasonable computational costs.

The key idea of adaptive fluid simulation is to focus more heavily on the important regions of flows. The effectiveness of such an adaptive simulation process greatly relies on the proper selection of mesh refinement criteria that decide regions of interest. In contrast to smoke and gas simulation, tracking a liquid's interface accurately is an essential part of computations in liquid simulations. In particular, the tracking process is important for computer animation because the quality of the final rendered images is directly affected by the liquid's surface, taken implicitly through a level set computation or explicitly

Manuscript received Feb. 21, 2006: revised Oct. 13, 2006.

This Research was supported by the Information Technology Research Center (ITRC) support program from the Ministry of Information and Communication (MIC) of Korea.

Janghee Kim (phone: + 82 42 860 5923, email: rolldice@etri.re.kr) is with Digital Content Research Division, ETRI, Daejeon, Korea.

Insung Ihm (email: ihm@sogang.ac.kr) and Deukhyun Cha (email: seaboy7@grmanet.sogang.ac.kr) are with Department of Computer Science, Sogang University, Seoul, Korea.

through an iso-surface extraction. Therefore, distributing more mesh elements in the region of a liquid's interface is generally regarded as a refinement criterion of the highest priority.

When fluids are simulated in the production of animations, the view dependency may be explored as another criterion in the hope of raising the computational efficiency further. Consider, for example, a scene where a camera keeps tracking a speedboat that moves around in a large physical area of the sea. For a computer animation intended to produce images seen from the camera, it might not be a clever strategy to allocate the equivalent amount of computational resources to the entire simulation domain regardless of the visibility. In such a situation, a more effective simulation could be achieved by placing more refined mesh elements inside the camera's viewing volume.

This paper briefly discusses a view-dependent adaptive fluid simulation that is quite feasible for 3D animation production. In particular, we show that a remarkable amount of memory space and computation time can be saved during the fluid simulation by dynamically considering the camera's viewing parameters. For this, we extend the idea of the adaptive technique presented by Losasso and others [1], and demonstrate how a view-dependent simulation on a simple adaptive grid can effectively cull uninteresting grid cells from the fluid computation. Our technique might not be well suited to simulations where numerical preciseness has the highest priority, as is common in computational fluid dynamics, because exploring the view dependency in addition to the adaptive mesh refinement introduces additional numerical errors. However, we show that the proposed technique provides a good compromise between the computational effort and the fidelity of the fluid simulation, which is useful when some numerical precision may be traded off for computational efficiency, as is often the case in computer graphics.

II. Related Work

Although there is a rich body of literature regarding the numerical simulations of the Navier-Stokes equations in the computational fluid dynamics community, the work by Foster and Metaxas [2] is probably the first in the field of computer graphics that attempted to solve the full three-dimensional equations to generate complex water effects. Foster and Fedkiw [3] presented an improved liquid simulation method, where a semi-Lagrangian integration scheme, introduced to the graphics community by Stam [4], was adopted as a stable solution for liquid motion. Enright and others [5] applied the particle-level set method [6] to liquid simulation to improve the hybrid front tracking method, proposed by Foster and Fedkiw [3]. This simulation technique turned out to be successful at producing more realistic animation of complex water scenes. In parallel with these efforts, many simulation techniques also have been developed in computer graphics to realistically animate water [7]-[9] smoke and gas [4], [10]-[12], and fire, flames, and explosions [13]-[16].

These simulation techniques lead to very impressive animations of a variety of fluid effects. However, they still suffer from the intensive, sometimes impractical, requirement of memory space and computation time when fluids are simulated on high-resolution grids. To optimize the computational resources, Rasmussen and others [17] simulated highly detailed large-scale smoke-like phenomena by combining a 2D high-resolution flow field with a moderate-sized 3D Kolmogorov velocity field. As mentioned in the previous section, Losasso and others [1] employed a dynamic octree-based mesh structure to adaptively simulate water and smoke. A different adaptive approach for buoyancy-driven flows was proposed by Shah and others [18], where the simulation grid of fixed-resolution follows the motion of the flow by modifying its spatial location and shape. A moving grid windowing method was also proposed by Rasmussen and others [19] to track only the visually interesting portion of a liquid animation. Together with this method, a loosely coupled multiple grid sourcing technique was used to allow for the efficient calculation of large-scale gravity-driven liquid behavior.

Although its usefulness was demonstrated only recently in the computer graphics community, the adaptive mesh refinement has been an important research topic in computational fluid dynamics, where the aim is to increase the effective spatial and temporal resolution of numerical simulations beyond the limits imposed by computational resources. Starting with the work by Berger and Oliger [20], various adaptive data structures and computational schemes have been proposed for the simulation of both steady and unsteady flows on structured and unstructured meshes (refer to, for example, [21]-[24] for extensive references). While the idea of view-dependent adaptive computing is widely applied in computer graphics, particularly in the view-dependent polygonal mesh simplification initiated independently in [25]-[28], no attempt, to the best of our knowledge, to explore the advantage of the view-dependent method has been made for 3D fluid simulations except in the work by Hinsinger and others [29], where ocean surface meshes are generated adaptively based on the camera projection. In fact, it may not be an appropriate research direction in computational fluid dynamics because the concept of view dependency is rather unclear and could only harm the numerical fidelity of physically based simulations. However, in computer animation where the camera-dependent computation is important and some sacrifice of numerical accuracy for computational

efficiency is often allowed, it is quite desirable to examine the possible advantage of view-dependent computation in fluid animation.

III. Simple and Efficient Implementation of a Dynamically Varying Adaptive Grid

Designing an adequate mesh refinement scheme is an essential part of adaptive simulations in which dynamically evolving flow interfaces must be tracked efficiently. Several variations of quadtree/octree data structures have been exploited to represent complex evolving interfacial flows [1], [21], [23], [24]. A great advantage of such hierarchical structures is that they are well suited to flexible local refinement of the computational domain. However, when implemented carelessly, this unrestricted but irregular meshing capability causes substantial overheads in maintaining and accessing the dynamic tree structures. The fluid simulation is often limited by the available memory space when the computation is performed on high-resolution grids. In this case, optimizing the memory cost for storing the adaptive grid becomes of great concern.

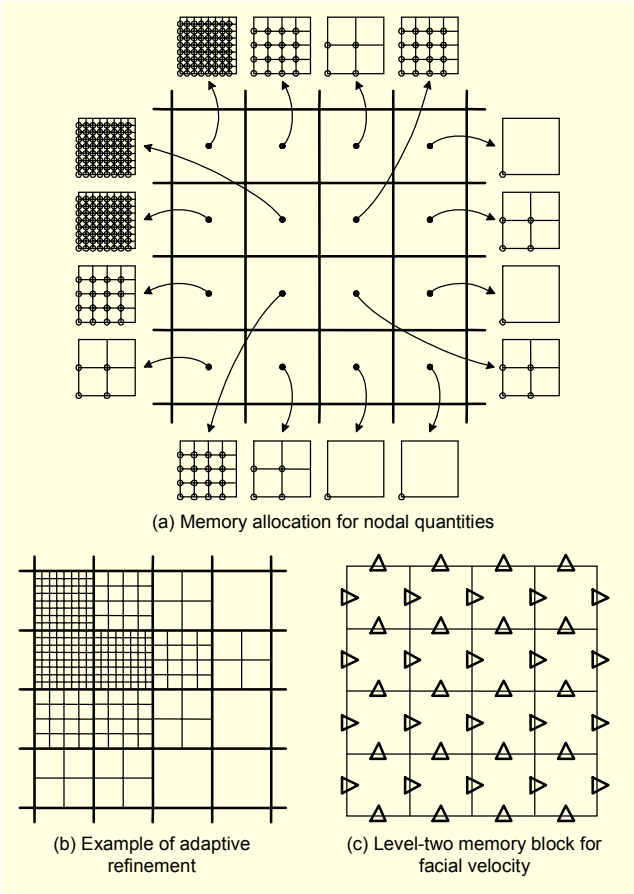

Fig. 1. Adaptive mesh representation scheme.

In this work, we propose to use a simple data structure that offers an efficient implementation of view-dependent adaptive mesh refinement. Our approach is similar to those employed in the previous works, for instance, [20], [30], [31]. Figure 1 illustrates the overall scheme of the proposed hierarchical grid structure. An entire computational domain in 3D space is subdivided with a coarse base grid, denoted as level 0. Finer grids are then laid over the base cells as required so that each base cell is uniformly refined using a properly determined level. For a gradual refinement, we demand that the levels of the six neighboring base cells may differ by one at most. (See Fig. 1(b) which illustrates an example of grid refinement in a region made up of 16 base cells). While not fundamentally necessary, this restriction simplifies the calculations required at the cell boundaries. It may be possible to save memory by allowing a bigger difference in levels, but only at the expense of increased discretization errors at the boundaries between different levels of refinement.

Figure 1(a) depicts a realization of our adaptive grid scheme in which a pair (level indicator, pointer) is stored per base cell. The pointer identifies the memory block that actually holds the data defined in the corresponding cell region. In the current implementation, we maintain the fluid's properties, including velocity, at the nodes of the grid. To store the nodal values, memory blocks of size $2^i \times 2^i \times 2^i$ are allocated to level-i cells. Note how the memory blocks are aligned toward the cell's anchor points to avoid data redundancy.

When the Poisson equation in the projection stage is solved to enforce the divergence-free constraint, memory blocks are allocated for storing the pressure and facial velocity temporarily. The same-sized memory blocks can be allocated for the pressure values that are defined in the centers of grid elements.

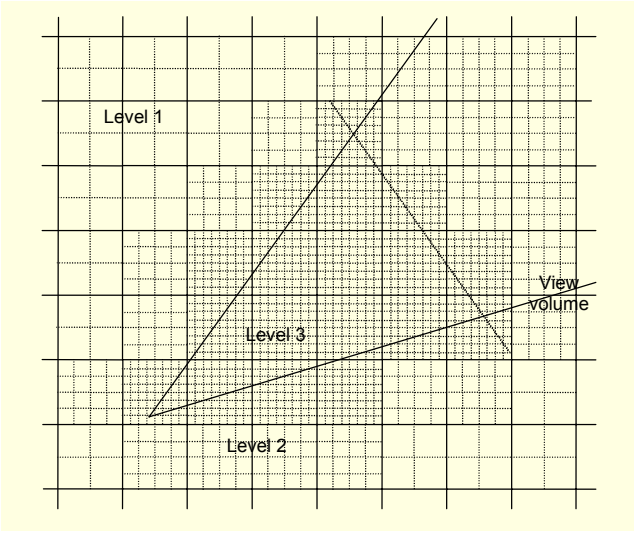

Fig. 2. View-dependent mesh refinement around the interface band.

On the other hand, memory blocks of size $2^{i+1} \times 2^{i+1} \times 2^{i+1}$ must be assigned to level-*i* cells to store the temporary velocities that are defined at the centers of faces (refer to Fig. 1(c)). This allocation strategy may entail some inefficiency, because the facial velocities are duplicated between neighboring base cells with the same level. However, we find the overheads are not serious in view-dependent adaptive mesh refinement.

In this representation, no recursive data structure is adopted to represent the adaptive mesh structure. Instead, each base cell is associated with a level indicator and a pointer to the array that holds the appropriate data values. Our scheme may have the disadvantage that the mesh refinement is somewhat restricted in that each base cell must be refined to the same level. However, this possible restriction leads to a very simple implementation of dynamic adaptive grids without having to handle the complex tree structure. The proposed data structure provides an efficient $O(1)$ access to a data element on the grid, thus reducing computation overheads. More importantly, the dynamic modification of the mesh becomes very easy through a simple memory management scheme where a set of heaps of memory blocks is maintained level-by-level. When the refinement level of a base cell changes as the simulation proceeds, its memory block is simply replaced by one at the new level as coarsening or refinement is performed.

Note that, for the computation of level set advection, we use a uniform grid of full level defined only within narrow bands as its implementation with adaptive grid becomes quite inefficient. Since the narrow band region is rather small, adopting the uniform grid does not deteriorate the performance.

IV. View-Dependent Mesh Refinement

The main goal of using an adaptive grid scheme is to concentrate computational resources in the region of interest, while still allowing an acceptable treatment of the remaining regions. When an evolving interface is tracked during the fluid simulation, its boundary is usually of the most concern. Therefore, it is natural to apply adaptive refinement to a band around the interface. It has been shown in previous work that this adaptive strategy can yield a remarkable reduction of computational effort, while still providing high numerical accuracy.

If fluids are simulated for the purpose of computer animation, it is worth considering additional simulation parameters that may help improve the computational efficiency further. Above all, the camera, or its viewing volume, allows us to focus on a more specific region of the computational domain. Whereas the simulation is performed on the entire domain, the visually interesting regions may be actually restricted to the viewing volume. In an attempt to optimize the use of computational resources, we propose placing more grid cells in the region inside the viewing volume. This implies that we further focus on the physics taking place there. Note, however, that the evolving interface outside the viewing volume affects the entire behavior of the fluids, and thus must still be tracked with good accuracy, although at a relatively lower resolution.

Now, we summarize a set of generic criteria that may be incorporated into our view-dependent adaptive base-cell refinement system:

- Visible base cells have higher refinement levels than invisible base cells.
- Base cells intersecting the band around moving interfaces are refined with higher precision. In particular, those inside the viewing volume are refined with the highest possible precision.
- Base cells intersecting the moving interface band, even if they are out of view, are refined with some appropriate level, because pressure computation with an overly coarse refinement may cause the entire system to deteriorate.
- The levels of adjacent base cells may differ by one at most.
- When the camera's near and far clipping planes are far away, as is often the case when an open sea surface is simulated, the distance from the camera may be used to determine the visible base cell's level to give closer cells higher refinement levels.

Notice that choosing a specific set of criteria is somewhat problem-dependent, so the refinement procedure must be adapted carefully to reflect the characteristics of a given simulation.

V. Results

To verify the effectiveness of view-dependent adaptive simulation, we have implemented a liquid simulator based on the particle-level set method [6]. For a performance comparison, the liquid simulation core was extended to support the adaptive grid structure explained in section III. In doing this, we used the approximate projection operator derived by Losasso and others [1], which leads to an easy-to-solve symmetric positive definite linear system in the projection stage.

The particle-level set method basically consists of two major stages: an update of the velocity field using the Navier-Stokes equations and the evolution of the level set and particles for interface tracking. The most memory- and time-consuming part during the velocity update is the projection computation, in which the memory space for the pressure field must be allocated, and a large linear system must be built and solved for

the pressure to keep the velocity field divergence-free. Using the (view-dependent) adaptive grid offers two major advantages in this velocity update stage. First, the memory required to store the entire velocity and other related attribute fields is greatly reduced. Second, the order of the linear system to be inverted for the pressure computation is reduced when the adaptive mesh is used, which remarkably lowers the timing and spatial complexity of the velocity update computation.

Notice that the efficiency of our implementation is affected by several other parameters inherent in the specific interface tracking method used. For instance, when the particle-level set method is used, the number of particles that are seeded to correct the level set interface also influences the timing and spatial performance. To understand more precisely the exact effect of the view-dependent adaptive computation in fluid simulation, we focus on the velocity update stage only. In particular, the resource requirements in the projection stage are closely analyzed.

1. Test Scene I

Figure 4 shows three pairs of liquid animations that were created from the same scene by applying three different simulation schemes: **uniform**, **adaptive**, and **v-adaptive**. Using the uniform method, the simulation was performed on a fixed, uniform grid of resolution 160×200×320. Adaptively refined grids of the same effective resolution were applied in the other two methods. In this experiment, we partitioned the computational domain into base cells (level 0) of size 8×8×8. This implies that four different levels of local refinement are possible in the simulation.

In testing the adaptive scheme, the highest level (level 3) was assigned to the base cells intersecting with a band around the liquid's interface. Then the adaptive coarsening process was applied to the base cells so that their levels decreased gradually down to level 0 away from the interface.

A more sophisticated refinement strategy, outlined in section IV, was employed in the v-adaptive method to exploit the view-dependency information. Figure 2 illustrates an example of mesh refinement along the liquid's interface. During the mesh refinement, the viewing volume is subdivided into two regions according to the distance from the camera. Unlike the adaptive method in which all the base cells that intersect with the interface band have the highest refinement level, only the cells that are contained, at least partially, in the front sub-region of the viewing volume are refined up to level 3. These base cells represent the most critical region, to which the computational resources are allocated most heavily. Then, the level of the base cells decreases gradually across the interface band. In this process, we guaranteed a minimum refinement level (1 in this experiment) to these cells to ensure there was minimum precision near the liquid's interface. Once the mesh is refined around the interface band, the remaining cells are marched away from the interface, coarsening the mesh down gradually to level zero.

A. Performance Statistics

To investigate how the respective meshing methods perform in the velocity update stage, we considered the following quantities:

- The entire computation time taken to update the velocity field using the Navier-Stokes equations (time)
- The order of the linear system solved in the projection step (order), where an *order*×*order* symmetric matrix is built from the Poisson equation
- The number of nonzero elements in the above linear system (nonzero)
- The amount of memory space required to assemble and solve the linear system (memory). This measure includes all the memory space required to store the nonzero elements, their indexes, and the temporary working space required by the linear system solver. The linear system was inverted in double precision.

Table 1 summarizes the experimental results, measured on a PC equipped with a 2.8 GHz Intel Pentium 4 CPU and 2 GB RAM. The respective figures represent the performance, averaged over the entire 560 time steps of the simulations. Time indicates the entire computing time per time step, taken to update the velocity field by integrating the Navier-Stokes equations, while the remaining measures reveal details about the projection stage where the Poisson equation must be solved to make the velocity field divergence-free. The order and nonzero headings respectively denote the order and the number of nonzero elements of the generated linear system. On the other hand, memory includes all the memory space necessary for storing the nonzero elements, their indexes, and the temporary working space required by the linear system solver. The statistics in the table indicate a quite favorable result for both adaptive meshing strategies compared to the uniform grid computation. When the adaptive mesh structure is applied independently of the camera (adaptive scheme), we observe a factor of 10.5 saving in computation time compared to the uniform grid computation (uniform scheme). The factor increases up to 24.4 if the view dependency is additionally exploited (v-adaptive scheme).

The savings result mainly from the most time-consuming projection stage, in which a large linear system for pressure must be built from the Poisson equation and solved iteratively. When the brute-force fixed grid is used, a linear system of

order 2,038,250×2,038,250 is generated on average for an inversion (order). On the other hand, its order reduces significantly when the liquid is simulated on the adaptive grid. In particular, we find that only a 126,069×126,069 system is built to solve for the pressure when the view-dependent adaptive scheme is employed. The figures in the nonzero and memory columns reveal similar performance enhancements.

Figure 3 illustrates how the performance measures of time and order vary during the simulations. One remarkable thing in this particular example is that the measures for the uniform and adaptive methods are rather uniform, while those for the v-adaptive method decrease as the simulation proceeds. The reason for this phenomenon is evident. In the experiment, we used a camera with a field of view that is 80 degrees wide horizontally and 60 degrees wide vertically. When the simulation starts, the viewing volume covers almost the whole scene (see the third row of Fig. 4), in which case almost no performance gain is achieved by exploiting the view dependency. However, as the camera moves forwards, the region behind it grows, which allows a relatively more efficient computation for the v-adaptive method.

Table 1. Comparison of performance measures for the 160×200×320 grid. The three mesh refinement strategies were tested on a test scene, illustrated in Fig. 4.

	Time (s)	Order	Nonzero	Memory (MB)
Uniform	126.7	2,038,250	8,071,956	300.65
Adaptive	12.1	297,849	1,181,681	43.98
V-adaptive	5.2	126,069	498,212	18.57

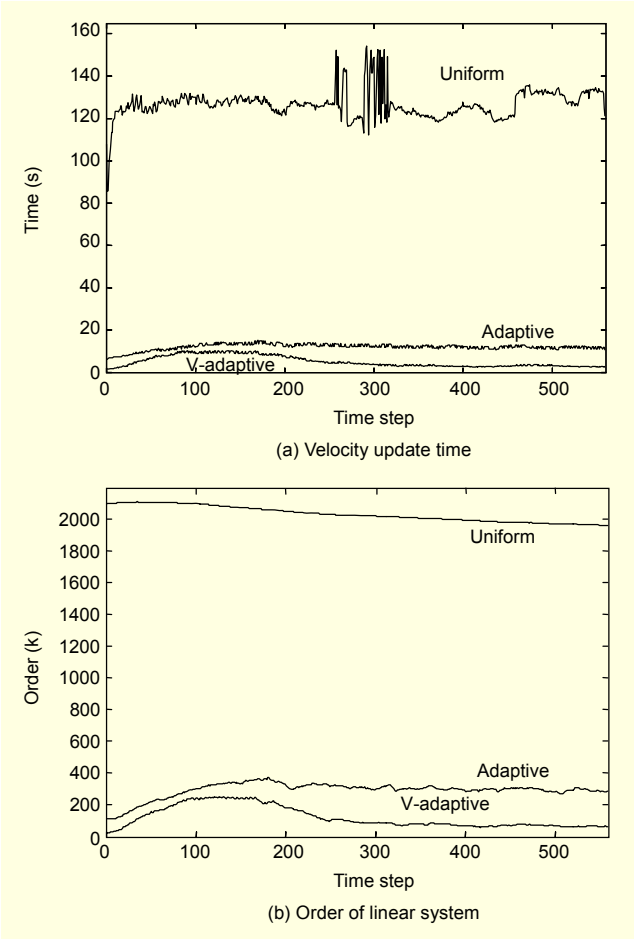

Fig. 3. Variations of two performance measures.

B. Animation Qualities

While the experimental results indicate a remarkable benefit of the adaptive simulation process, particularly combined with the principle of view-dependent computing, it definitely comes with numerical errors, accumulated while the simulation proceeds. However, as the generated animations demonstrate, we observe that the visual errors induced by the lost numerical precision are acceptable enough for the view-dependent adaptive simulation technique to be used effectively in 3D animation production. The animation sequences in the first three rows of Fig. 4 correspond to the three methods: uniform, adaptive, and v-adaptive, where the entire computational domain is depicted. It is evident that the numerical errors introduced during the simulation change the details of the liquid's interface (compare the corresponding frames in the fourth column closely). When simulated on the view-dependent adaptive grid, this visual error is more prominent in the region outside the viewing volume. This phenomenon derives naturally from our strategy that tends to allocate less computational resources to the invisible region.

Despite this problem, the view-dependent simulation turns out to be quite effective, as demonstrated in the actual views taken from the camera position (refer to the next three corresponding rows of Fig. 4). While the details of the surface, generated with the view-dependent adaptive grid, differ from those made by the other two methods, the general appearance of the liquid is well preserved in the simulation, which is often good enough for fluid animation.

2. Test Scene II

To further analyze the effect of the view-dependent simulation, we applied the two adaptive schemes to another test scene, discretized with a 440×280×440 grid (see Fig. 7). In this example, the uniform-grid simulation was excluded from consideration because it demands excessive computation time and memory space, and the linear systems it generates during the pressure computation are too large for our system. We used the same mesh refinement rule for this test scene as for the first

Fig. 4. Animation sequences comparing the three simulation schemes (frame nos.: 40, 90, 160, 210, and 400). The first three rows show snapshots of the entire computational domain in the order of uniform, adaptive and v-adaptive, respectively, while the remaining rows illustrate the corresponding views taken from the moving camera. The effective resolution achieved in this experiment is 160×200×320.

test scene. Figure 5 illustrates two examples of the computational grid that are adaptively refined considering the view information.

A. Performance Statistics

The first two rows of Table 2 show the average performance compiled over the entire 1000 time steps of the v-adaptive and adaptive simulation techniques with and without the view consideration. In Table 2, "V-adaptive" indicates the test case when the applied field of view is 80 degrees wide horizontally and 60 degrees wide vertically. As indicated in the table, the view-dependent adaptive technique saved about 45% computing time compared to the adaptive-only technique. We can also observe a noticeable reduction in the memory requirement as the linear system becomes smaller in size. Refer also to Fig. 6 for the per-frame performance.

The camera's field of view evidently influences the savings, as shown in the next two rows of Table 2. Here, "V-adaptive (4/5)" and "V-adaptive (2/3)" represent the simulations in which the field of view was narrowed down by factors of 4/5 and 2/3, respectively. The results agree well with intuition as

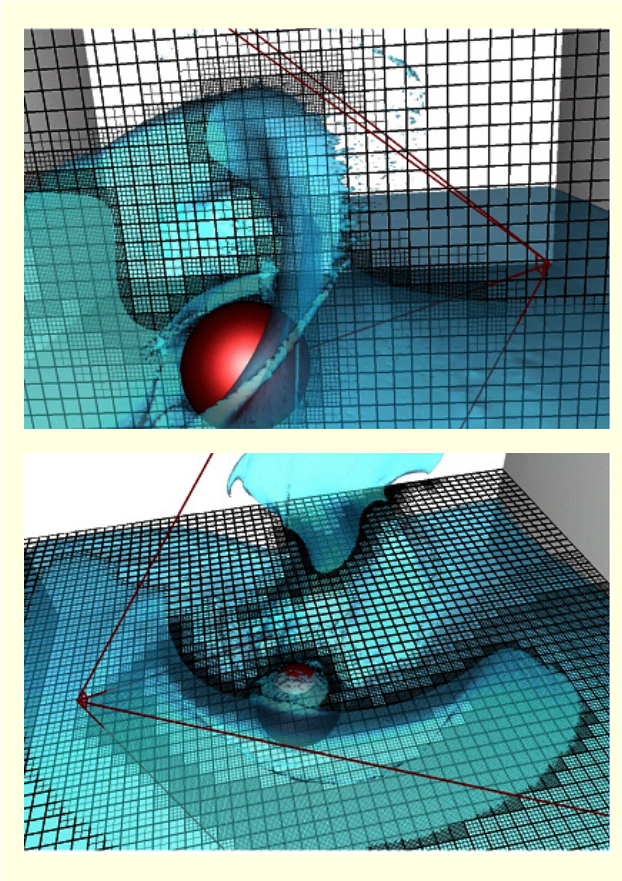

Fig. 5. Two examples of the view-dependent mesh refinement. These pictures show two snapshots of the view-dependent simulation where the orthogonal cutting planes reveal how the computational resources are allocated considering the view information.

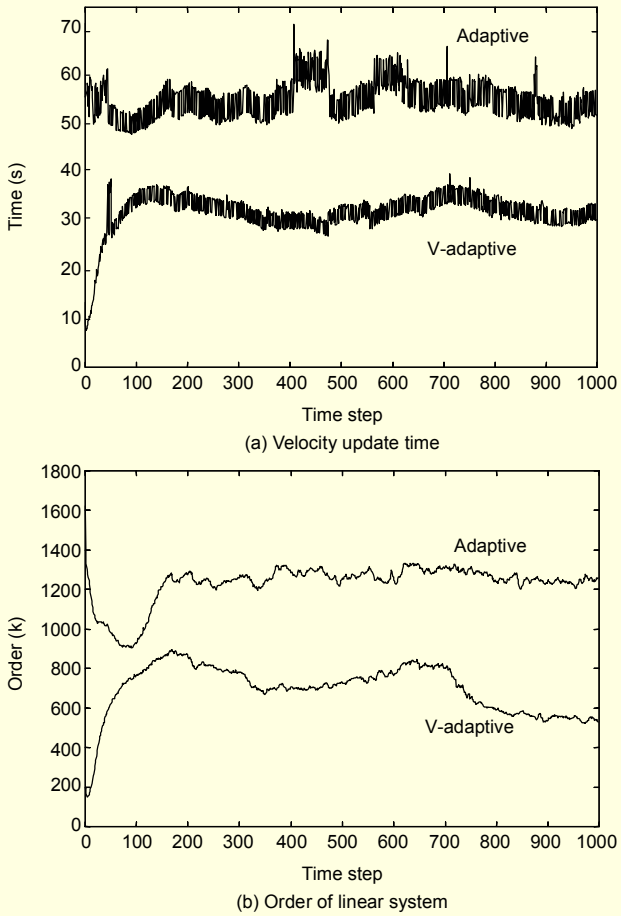

Fig. 6. Variations of the two performance measures. After about 200 time steps, the ratio of the viewing volume to the entire scene is almost the same, which results in a steady performance enhancement for the view-dependent adaptive simulation.

Table 2. Comparison of performance measures for the 440×280×440 grid.

	Time (s)	Order	Nonzero	Memory (MB)
Adaptive	55.9	1,230,087	4,889,637	181.87
V-adaptive	31.2	696,576	2,768,842	102.99
V-adaptive (4/5)	27.1	612,350	2,438,379	90.64
V-adaptive (2/3)	23.5	524,396	2,088,444	77.63

the task of the velocity update becomes lighter with a narrower view angle. Note, however, that narrowing the field of view too much with respect to the entire computational domain may cause a severe deficiency in numerical precision because most of the computational resources are allocated to a very small part of the domain.

Several variations of the refinement strategy could possibly overcome such a situation. One is to modify the coarsening process so that each level forms a band of base cells with thickness greater than one, as in Fig. 2. Another strategy is to increase the minimum level assigned to the base cells. For instance, in the two test scenes, we guaranteed that the invisible base cells, intersecting with the band around the liquid's interface had at least level 1 in the hope of preserving the minimum numerical precision near the liquid's surface. By lifting this minimum level to 2, for instance, more resources can be allocated to the regions outside the viewing volume, securing a minimum accuracy in the entire domain.

B. Animation Qualities

As in the first test scene, the results are quite encouraging. Figure 7 compares the animation sequences created using the two simulation techniques, where the first half shows the entire scenes during the simulation, the adaptive scheme followed by the v-adaptive method. The second half demonstrates the actual result images, seen from the camera. While it is true that

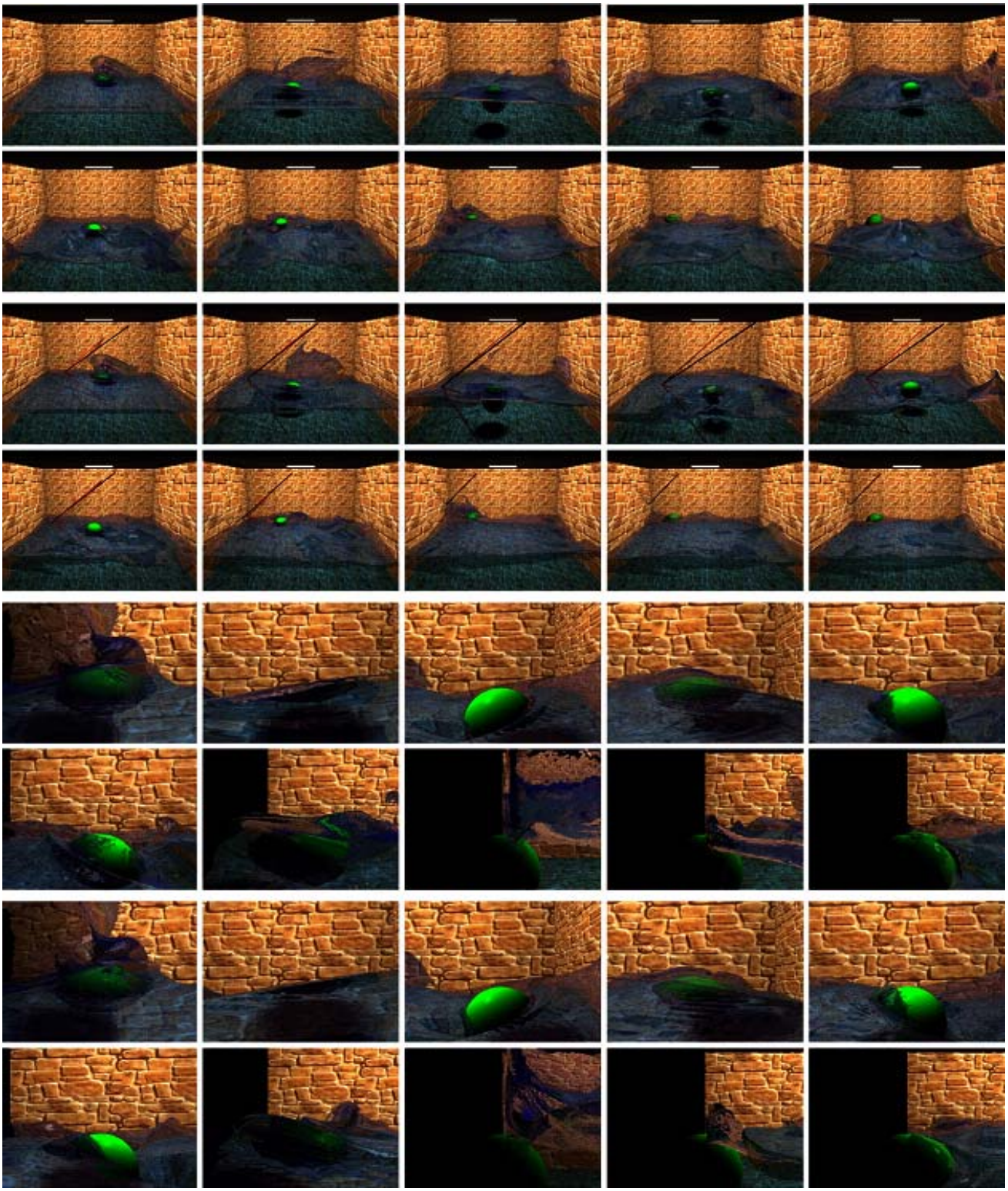

Fig. 7. Animation sequences comparing the two adaptive simulation schemes (frame nos.: 100, 200, 300,⋯, 1000). Again, the entire computational domain and the views taken from the actual camera are demonstrated for Adaptive and V-adaptive, respectively. The effective resolution applied in this experiment is 440×280×440. Although the numerical errors introduced by the view-dependent computing are accumulated additional to those caused by the adaptive mesh refinement, this experiment shows that a clever allocation of CPU time and memory space may lead to an effective view-dependent adaptive simulation technique quite suitable for 3D animation production.

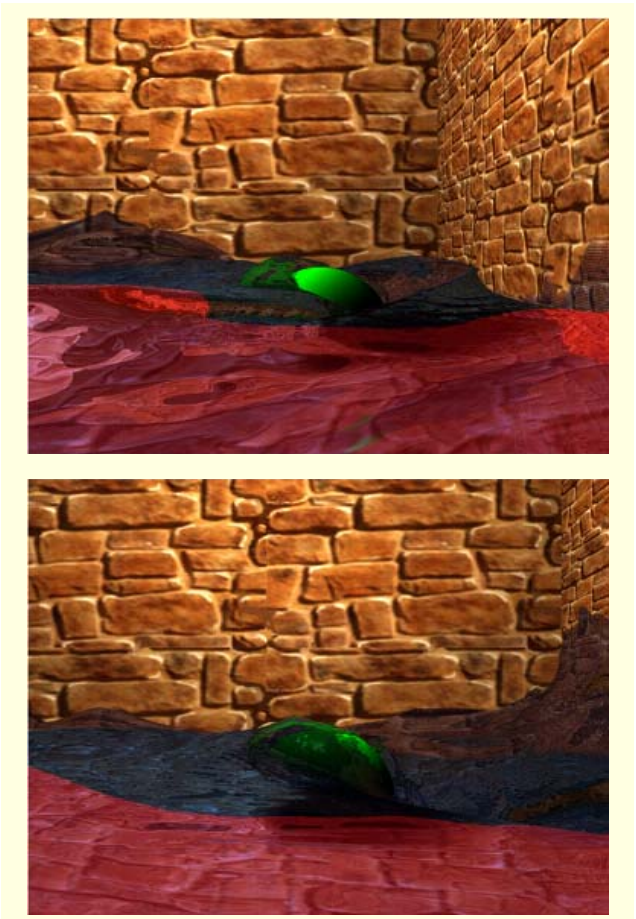

Fig. 8. Two sample snapshots illustrating the effect of the view-dependent refinement.

some visual differences in the animations are observed between the two methods, we conclude that clever usage of the economized computational resources still produces a very faithful simulation result. We believe that the slight visual errors found would be quite acceptable in the 3D animation industry.

Figure 8 demonstrates the effect of employing a higher simulation resolution in the fluid animation. The images were rendered by slightly shifting the viewing camera backwards to see how varying the level of refinement affects the details of the liquid animation. In this image, the area behind the camera was colored red to distinguish it from the region visible from the camera. We partitioned the viewing volume into two sub-regions according to the distance from the camera, and assigned them different refinement levels. The uncolored area represents the region to which the highest level (3 in this test scene) is assigned, while the far area, colored green, was simulated with the refinement level 2 (the green region appears in the video). Furthermore, the levels decrease gradually down to level 1 along the surface in the red area. Clearly, we can see that the details on the surface are well depicted inside the viewing volume, especially in the front part, while the surface becomes smoother in the back region as a result of using lower resolutions.

In conclusion, employing the adaptive mesh in fluid simulations causes some numerical inaccuracy. The situation becomes worse when accompanied by the view-dependent mesh refinement. However, the view-dependent adaptive simulation technique is quite feasible for 3D animation production, where some amount of numerical precision may be traded off for computational efficiency. This is evident considering the benefit achieved through the optimized CPU time and memory space.

VI. Conclusions

Physically based fluid animation is still a challenging problem when a high-resolution grid must be used. In this paper, we briefly discussed a view-dependent technique appropriate for adaptively simulating liquids. While the view-dependent computation may appear strange from the viewpoint of computational fluid dynamics, it is found to be quite feasible for simulating fluids for 3D animation production. Several tests with different scenes indicate that the described technique provides a good compromise between the computational effort and the simulation's fidelity, as limited computational resources are concentrated where they are most necessary.

While we proposed a simple and efficient data structure for representing dynamically varying adaptive meshes, the idea of view-dependent simulation presented here will be effective in any other adaptive mesh structure as long as it provides an efficient mechanism for handling the frequent dynamic refinement and coarsening processes.

References

[1] F. Losasso, F. Gibou, and R. Fedkiw, "Simulating Water and Smoke with an Octree Data Structure," *ACM Trans. Graphics (ACM SIGGRAPH 2004)*, vol. 23, no. 3, 2004, pp. 457-462.

[2] N. Foster and D. Metaxas, "Realistic Animation of Liquids," *Graphical Models and Image Processing*, vol. 58, no. 5, 1996, pp. 471-483.

[3] N. Foster and R. Fedkiw, "Practical Animation of Liquids," *Proc. ACM SIGGRAPH 2001*, 2001, pp. 23-30.

[4] J. Stam, "Stable Fluids," *Proc. ACM SIGGRAPH 1999*, 1999, pp. 121-128.

[5] D. Enright, S. Marschner, and R. Fedkiw, "Animation and Rendering of Complex Water Surfaces," *ACM Trans. Graphics (ACM SIGGRAPH 2002)*, vol. 21, no. 3, 2002, pp. 736-744.

[6] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell, "A Hybrid Particle Level Set Method for Improved Interface Capturing," *J. Computational Physics*, vol. 183, no. 1, 2002, pp. 83-116.

[7] J. Hong and C. Kim, "Animation of Bubbles in Liquid," *Computer Graphics Forum (Eurographics 2003)*, vol. 22, no. 3, 2003, pp. 253-262.

[8] O. Song, H. Shin, and H. Ko, "Stable but Nondissipative Water," *ACM Tran. Graphics*, vol. 24, no. 1, 2005, pp. 81-97.

[9] J. Kim, D. Cha, B. Chang, B. Koo, and I. Ihm, "Practical Animation of Turbulent Splashing Water," *Proc. Eurographics/ACM SIGGRAPH Symp. Computer Animation 2006*, 2006, pp. 335-344.

[10] N. Foster and D. Metaxas, "Modeling the Motion of a Hot, Turbulent Gas," *Proc. ACMSIGGRAPH 1997*, 1997, pp. 181-188.

[11] R. Fedkiw, J. Stam, and H. Jensen, "Visual Simulation of Smoke," *Proc. ACM SIGGRAPH 2001*, 2001, pp. 23-30.

[12] I. Ihm, B. Kang, and D. Cha, "Animation of Reactive Gaseous Fluids Through Chemical Kinetics," *Proc. Eurographics/ACM SIGGRAPH Symp. Computer Animation 2004*, 2004, pp. 203-212.

[13] G. Yngve, J. O'Brien, and J. Hodgins, "Animating Explosions," *Proc. ACM SIGGRAPH 2000*, 2000, pp. 29-36.

[14] A. Lamorlette and N. Foster, "Structural Modeling of Flames for a Production Environment," *ACM Trans. Graphics (ACM SIGGRAPH 2002)*, vol. 21, no. 3, 2002, pp. 729-0735.

[15] D. Nguyen, R. Fedkiw, and H. Jensen, "Physically Based Modeling and Animation of Fire," *ACM Trans. Graphics (ACM SIGGRAPH 2002)*, vol. 21, no. 3, 2002, pp. 721-728.

[16] B. Feldman, J. O'Brien, and O. Arikan, "Animating Suspended Particle Explosions," *ACM Trans. Graphics (ACM SIGGRAPH 2003)*, vol. 22, no. 3, 2003, pp. 708-715.

[17] N. Rasmussen, D. Nguyen, W. Geiger, and R. Fedkiw, "Smoke Simulation for Large Scale Phenomena," *ACM Trans. Graphics (ACM SIGGRAPH 2003)*, vol. 22, no. 3, 2003, pp. 703-707.

[18] M. Shah, J. M. Cohen, S. Patel, P. Lee, and F. Pighin, "Extended Galilean Invariance for Adaptive Fluid Simulation," *Proc. Eurographics/ACM SIGGRAPH Symp. Computer Animation 2004*, 2004, pp. 213-221.

[19] N. Rasmussen, D. Enright, D. Nguyen, S. Marino, N. Summer, W. Geiger, S. Hoon, and R. Fedkiw, "Directable Photorealistic Liquids," *Proc. Eurographics/ACM SIGGRAPH Symp. Computer Animation 2004*, 2004, pp. 193-202.

[20] M. Berger and J. Oliger, "Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations," *J. Computational Physics*, vol. 53, no. 3, 1984, pp. 484-512.

[21] A.M. Khokhlov, "Fully Threaded Tree Algorithms for Adaptive Refinement Fluid Dynamics Simulations," *J. Computational Physics*, vol. 143, no. 2, 1998, pp. 519-543.

[22] B. Bennett and M. Smooke, "Local Rectangular Refinement with Application to Nonreacting and Reacting Fluid Flow Problems," *J. Computational Physics*, vol. 151, no. 2, 1999, pp. 684-727.

[23] S. Popinet, "Gerris: a Tree-Based Adaptive Solver for the Incompressible Euler Equations in Complex Geometries," *J. Computational Physics*, vol. 190, no. 2, 2003, pp. 572-600.

[24] V. Sochnikov and S. Efrima, "Level Set Calculations of the Evolution of Boundaries on a Dynamically Adaptive Grid," *Int'l Journal for Numerical Methods in Engineering*, vol. 56, 2003, pp. 1913-1929.

[25] J. Xia and A. Varshney, "Dynamic View-Dependent Simplification for Polygonal Models," *Proc. IEEE Visualization 1996*, 1996, pp. 327-334.

[26] H. Hoppe, "View-Dependent Refinement of Progressive Meshes," *Proc. ACM SIGGRAPH 1997*, 1997, pp. 189-198.

[27] B. Koo, Y. Choi, C. Chu, J. Kim, and B. Choi, "Shrink-Wrapped Boundary Face Algorithm for Mesh Reconstruction from Unorganized Points," *ETRI Journal*, vol. 27, no. 2, 2005, pp. 235-238.

[28] D. Luebke and C. Erikson, "View-Dependent Simplification of Arbitrary Polygonal Environments," *Proc. ACM SIGGRAPH 1997*, 1997, pp. 199-208.

[29] D. Hinsinger, F. Neyret, and M. Cani, "Interactive Animation of Ocean Waves," *Proc. Eurographics/ACM SIGGRAPH Symp. Computer Animation 2002*, 2002, pp. 161-166.

[30] M. Minion, "A Projection Method for Locally Refined Grids," *J. Computational Physics*, vol. 127, no. 1, 1996, pp. 158-178.

[31] L. Howell and J. Bell, "An Adaptive Mesh Projection Method for Viscous Incompressible Flow," *SIAM Journal in Scientific Computing*, vol. 18, no. 4, 1997, pp. 996-1013.

Janghee Kim received the BS and MS degrees in computer science from Sogang University, Seoul, Korea, in 2003 and 2005, respectively. Since then, he has been with the Digital Content Research Division (DCRD) of ETRI. His research interests are focused on all kind of dynamics, especially fluid dynamics, and rendering techniques including GPU-based real-time computing.

Insung Ihm received the BS degree in computer science and statistics from Seoul National University, Seoul, Korea in 1985; the MS degree in computer science from Rutgers University, New Jersey, USA in 1987; and the PhD degree in computer science from Purdue University, Indiana, USA in 1991. He is currently a Professor in Computer Science at Sogang University, Seoul, Korea. His research interests include computer graphics, scientific visualization, and high-performance computing.

Deukhyun Cha received the BS and MS degrees in computer science from Sogang University, Seoul, Korea, in 2002 and 2004, respectively. Since then, he has been a PhD student in the Department of Computer Science at Sogang University. His research interests are focused on fluid dynamics, photo-realistic fluid rendering, and GPU-based real-time computing.