The Journal of the Society for Art and Science, Vol. 19, No. 1, pp. 1-8 (2020)

AABB Pruning: Pruning of Neighborhood Search for Uniform Grid Using Axis-Aligned Bounding Box

Daiki Takeshita (Member)

National Institute of Technology, Akita College

take@akita-nct.ac.jp

Abstract

In a particle method, it is necessary to search for neighboring particles within the influence radius of a particle. This search is called a neighborhood search, and, when using a uniform influence radius, a data structure based on a uniform grid is often used. However, because there are some grids where registered particles are all outside the influence radius depending on the particle position in the grid, the neighborhood search with data structures based on the uniform grid is not efficient. In this study, the efficiency of the neighborhood search with the uniform grid was improved. In this method, an axis-aligned bounding box (AABB), which includes particles registered in each grid, is acquired, and it is determined whether the AABB is within the influence radius of the particle. When the AABB exists outside the influence radius, all particles registered in this grid are outside the influence radius, and thus, it is possible to perform the pruning of the neighborhood search. This method, implemented on a central processing unit, matched the search result of the uniform grid, and the sum of the calculation times for data structure construction and the neighborhood search was reduced by 20.7%.

Keywords: Particle method, Neighborhood search, Uniform grid, AABB, Pruning

1. Introduction

Fluid computer graphics (CG) is mainly used in the entertainment field, and there is interest in an automatic generation of this animation. This animation is generated using a calculation result of a discretization of the Navier–Stokes equation. For this calculation, a grid method that divides space into grids, a particle method that uses nonconnected particle groups, and a hybrid method that uses both are proposed. In particle methods, many methods based on smoothed particle hydrodynamics (SPH) [1] have been proposed [2-11].

In SPH, physical quantities are determined by superposition of kernels. Therefore, it is necessary to search for neighboring particles within the influence radius of the kernel. This search is called a neighborhood search [2], and, when all particles are searched, the calculation amount is $O(n^2)$ at particle n; thus, the calculation amount is generally reduced by the data structure. When the uniform influence radius is used, data structures based on a uniform grid [14, 16] are often used [2, 16]. However, data structures based on a uniform grid have some grids in which all registered particles are outside the influence radius, depending on the particle position in the grid, and the neighborhood search is not efficient (see Figure 1).

In this study, position-based fluids (PBF) [8] was used as the particle method based on SPH with the uniform influence radius. The purpose of this study was to improve the efficiency of neighborhood search with the uniform grid. In this method, an axis-aligned bounding box (AABB) including particles registered in each grid is acquired, and it is determined whether the AABB is within the influence radius of the particle. When the AABB exists outside the influence radius, all particles registered in this grid are outside the influence radius, so it is possible to perform the pruning of the neighborhood search. The proposed method is implemented in parallel on the central processing unit (CPU), and the sum of calculation time for the data structure construction and the neighborhood search is reduced by 20.7% on average compared with the uniform grid. This method matches with the result of the neighborhood search by the uniform grid, and there is

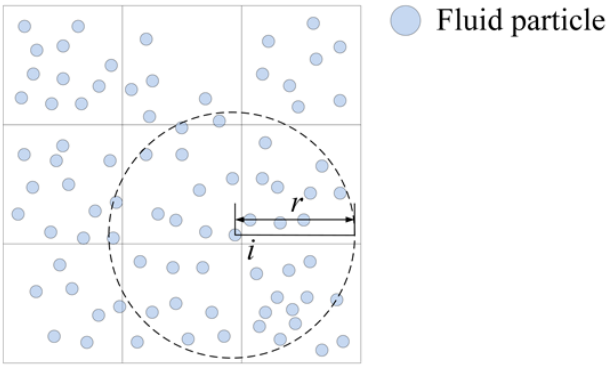


Figure 1. Neighborhood search with the uniform grid (2D): in the upper left, lower left, and upper right grids, there are no fluid particles within the influence radius r of the particle i.

no difference in the generated image.

2. Related works

CG studies related to SPH [1] were detailed in a report [2] by Ihmsen et al.. The initial study on CG using SPH was an interactive CG of water by Müller et al. [3]. This method does not consider the incompressibility of the fluid and does not maintain the volume of water. Currently, methods attempting to realize incompressibility capable of creating more-realistic animation have become mainstream [4-11], and one of them is PBF [8]. This method is a fluid calculation model using a framework of position-based dynamics [12]. In comparison with predictive–corrective incompressible SPH (PCISPH) [5], this method permits a larger time step and shows similar behavior of the fluid to PCISPH when a position correction repetition processing of several tens of times is calculated. In addition, it is possible to use it as a real-time application by setting a particle number from 80k to 128k and a position correction from 2 to 4, which is useful as a method with a reduced calculation amount. PBF has been enhanced through unified particle physics [10] for real-time applications and includes bidirectional fluid rigid coupling and gas animation in its fluid model. In this study, the CG of water is focused on, and PBF [8] is adopted.

Data structures for the neighborhood search in the particle method were described in detail in a previous report [2] and a work [16] by Ihmsen et al.. These data structures were based on a uniform grid [14, 16], k-d tree [20, 21, 26], Verlet lists [22-24], and an octree [25]. In the neighborhood search for the particle method, the construction and search with the uniform grid is performed with a time complexity of O (n). In hierarchical data structures, such as kd tree and octree, this complexity is O ($n \log n$). Therefore, the uniform grid is suitable for the neighborhood search with a uniform influence radius. In fact, for the uniform influence radius, a previous study on SPH [34] shows the experimental result that the computation time of the uniform grid is approximately 3-5 times faster than that of the tree structure. In particular, a parallel implementation involves high memory transfer, which limits the performance of hierarchical data structures [16]. Verlet lists construct lists of particles that are larger than the influence radius and use them for the neighborhood search at multiple time steps. These particle lists are reconstructed in consideration of particle movement distances. An implementation of this method is usually based on the uniform grid. Verlet lists work quickly with a small number of particles. However, this data structure is slower than data structures based on the uniform grid [14, 16], because it requires a large amount of memory when the number of particles is large [2]. For these reasons, data structures based on the uniform grid were focused on in this study.

In the basic uniform grid, the space is divided with grids, and register particles exist in each grid. The grid width is set to the influence radius. In a searching process, it is determined whether particles registered in a 3 × 3 × 3 neighborhood grid are within the influence radius. In the data structure based on the uniform grid, there are index sort [13-15], an extension method of index sort [28], Z-index sort [16], an implementation of Z-index sort on a graphics processing unit (GPU) [27], spatial hashing [16-19], and compact hashing [16]. In these data structures based on a uniform grid, there are some grids in which all registered particles are outside the influence radius, depending on the position in the grid of particles, and the search is not efficient. There are many data structures based on the uniform grid, however, in this study, the basic uniform grid was focused on because it is the simplest data structure.

Changing the size of a grid cell in a uniform grid changes the number of particles to be searched. The optimal size of this value for implementation on a CPU was investigated elsewhere [16]. When the grid width d_g is less than the influence radius r, the number of particles registered per grid decreases, and the number of searched particles also decreases. However, because the number of grids to be searched increases, as the number of queried grids increases, memory transfer increases. In that study [16], it was shown that, when the grid width was set to $d_g = r / 2$, the processing time increased compared with the case of $d_g = r$, and the optimal grid width was the influence radius. Therefore, the influence radius r was used for the grid width in the present study.

3. Pruning of the neighborhood search

In this study, the efficiency of the neighborhood search using a uniform grid was improved. In this method, AABB, which is a kind of bounding volume, is used for the pruning of the neighborhood search. The following describes the implementation of PBF in this study, a reason for selecting AABB from many bounding volumes, and details of the proposed method.

3.1 Implementation of PBF

This study is based on PBF [8] and considers the surface tension, the vorticity confinement, and the artificial viscosity. The data structure that is the basis of the improvement is a uniform grid. According to previous work [8], construction of a data structure and the neighborhood search in PBF are performed only once in one step. Therefore, in the neighborhood search, the indices of the particles within the influence radius of each particle are stored and then used for calculation of particle behavior. At a wall boundary, the penalty method prevents particles from advancing inside the wall, and the wall particles are not placed. In this case, the number of particles used decreases; therefore, the amount of calculation is reduced. However, water particles tend to stick to wall boundaries. In the rendering, the anisotropic kernel was used, water surfaces were constructed by the marching cube, and polygons were displayed with OpenGL. The implementation was done on a CPU, and the program was parallelized using OpenMP.

3.2 Selection of the bounding volume

In this study, an AABB, which is a kind of boundary volume, was used for the pruning of the neighborhood search. The bounding volume has been detailed elsewhere [29]. Boundary volumes often use a sphere [30, 31], an AABB, an oriented bounding box (OBB) [32], discrete-orientation polytopes in eight directions (8-DOP), and convex hulls [33]. Although other boundary volumes exist, because the overlap decision cost is expensive, there are few opportunities to use them.

When using bounding volumes for the pruning of the neighborhood search, the construction cost is important. It is possible to construct a bounding volume as preprocessing, when the shape of an object does not change in the collision detection of the object. By contrast, in the neighborhood search, it is necessary to reconstruct the bounding volume corresponding to the movement of a particle in each step. When it takes a long time to construct it, it will not lead to a reduction in the entire computational time. In the proposed method, it is also important that the cost to determine whether the bounding volume is within the influence radius of the particle is small. Therefore, the AABB is selected in the proposed method. Since the AABB composed of each grid is guaranteed to be smaller than the grid, pruning with the AABB is efficient. Furthermore, the construction of an AABB is a process in which the maximum and minimum values of the xyz coordinates of a box containing particles are determined, and this can be done only through comparison and assigning of values. Therefore, the calculation can be performed very robustly without including calculation errors resulting from arithmetic operations.

3.3 Pruning of the neighborhood search for the uniform grid using an AABB

In this method, an AABB including particles existing in each grid is constructed. Then, a distance d_a between the AABB of each grid and a particle i is calculated, and the distance d_a is compared with the influence radius r to determine whether the AABB is within the influence radius r of the particle i (see Figure 2(a)). When the AABB is outside the influence radius r, all particles registered in this grid are outside the influence radius r, so it is possible to perform the pruning of the neighborhood search. When it is within the influence radius r, a distance d is calculated between the particle i and each particle registered in the grid, and it is judged whether it is within the influence radius r of the particle i (see Figure 2(b)). In the implementation, an expensive square root calculation can be avoided by judging the pruning with a square of the distance d_a and the influence radius r. An efficient method of calculating the square of the distance d_a is available in the literature [29]. It is obtained by comparing maximum and minimum values of xyz coordinates that consist of the AABB with xyz coordinates of the particle i and adding distances beyond a range of the AABB. The specific procedure of the proposed method using the distance d_a and the influence radius r is shown below.

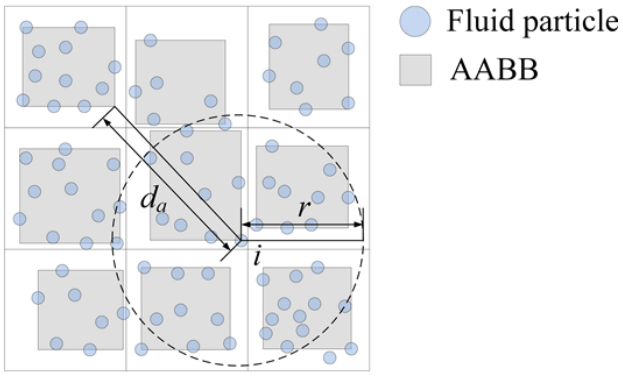
Construction of a data structure

1) Construct a data structure by the uniform grid.
2) In each grid, construct the AABB including particles.

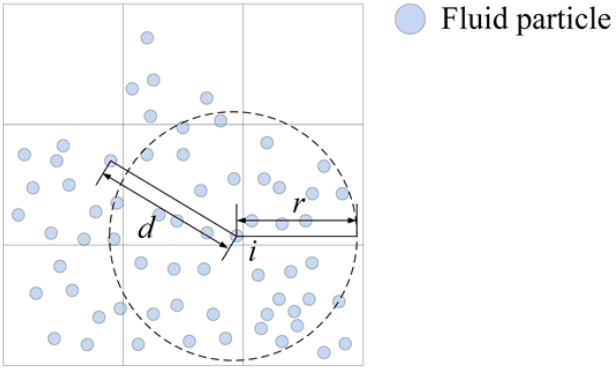
Neighborhood search

1) In 3 × 3 × 3 neighborhood grids centered on the grid where particle i is registered, calculate the distance d_a between the AABB in each grid and particle i.
2) When $d_a \leq r$, it is determined whether particles registered in this grid are within the influence radius r of particle i.

It is obvious that the AABB in the grid where the particle i is registered is within the influence radius r of the particle i. However, to exclude the grid in which the particle i is registered from the judgment of the pruning, many comparison processes with respect to the grid index are involved, and it does not lead to a reduction of the calculation time. Therefore, it is treated the same as the AABBs of other grids. The construction of the AABB is performed on each grid, and the determination of whether the AABB is within the influence radius r of



(a) Determining whether the AABB is within the influence radius r of the particle i: In this example, upper left and upper right grids are outside the influence radius r, and it is possible to perform the pruning of the neighborhood search.



(b) Figure not displaying pruned fluid particles: Determine whether displayed fluid particles are within the influence radius r of the particle i. In a lower left grid, all registered particles are outside the influence radius r, but depending on a position of the particle i and AABB, they may not be pruned.

Figure 2. Pruning of the neighborhood search using the AABB (2D)

the particle i is performed for each particle; therefore, parallel processing can be performed. However, when constructing the uniform grid by parallel processing, the process of registering particles in each grid is performed serially, because it will be in the race condition [2].

4. Results

In this section, the results of experiments comparing the uniform grid and the proposed method are presented. Animation example 1 is a basic experiment setup called a "dam break." Animation example 2 is a setup in which six water chunks are dropped on a water surface. Animation example 3 is a setup in which two plates are placed and water flows from the top of the space. Animation example 4 is a setup in which waves are generated by sliding the wall on the left side of the space. Animation example 5 is a setup in which two cylinders are placed and water flows from the upper left side of the space. In these examples, 1,000,000, 1,277,760, 1,084,800, 1,000,000, and 1,044,000 water particles are used, and the number of calculation steps, including sub-steps, is 1000, 1200, 2000, 1000, and 2000, respectively. In these experiments, the calculation of particle position correction in PBF was performed three times. The resulting images are shown in Figures 3 through 7. To compare the generated images, Figure 3 shows the results of the uniform grid and the proposed method; there is no difference in the generated images. This was also confirmed in animation examples 2 through 5.

Table 1 shows data on the calculation times. These data are average values at all steps. In the table item "(1) Uniform grid," the basic uniform grid was used for the neighborhood search. "(2) AABB pruning" is the proposed method. "Construction" is the construction time of each data structure, and "Search" is the time of the neighborhood search by each data structure. "Construction and search" is the sum of time for "Construction" and "Search." "Particle behavior calculation" is the time taken by the process that calculates the behavior of particles. This process includes the PBF, surface tension, vorticity confinement, and artificial viscosity. "Total" is the total calculation time of "Construction," "Search," and "Particle behavior calculation." These calculation times do not include rendering. A program was implemented on a CPU and parallelized by OpenMP. The CPU was 4.2 GHz with four cores, and memory was 32 GB.

Data structures take longer to build as they become more complex. However, an appropriate data structure reduces the time for the neighborhood search. Therefore, "Construction and search" was the focus. From Table 1, in animation example 1, it was 1.703 s in "(1) Uniform grid" and 1.367 s in "(2) AABB pruning." The proposed method reduced the calculation time by 19.7% compared with the uniform grid. In animation examples 2 through 5, the calculation time was similarly reduced by 18.7%, 23.5%, 18.8%, and 23.0%, respectively. The average value of these was 20.7%. Figure 5 shows a graph of the reduction rate in the sum of the calculation times for the construction and the search in each step. The vertical axis of the graph is the reduction rate in the calculation time of the proposed method for the uniform grid, and the horizontal axis is the number of steps. Figures 5 (a) to 5 (e) show the data in animation examples 1 through 5. The best results are 30.0%, 30.9%, 31.2%, 33.4%, and 29.9%, respectively; and the worst results are 8.2%, 7.2%, 4.1%, 11.7%, and 6.1%, respectively. In the early steps of the animation, the reduction rate was relatively high, and, as the animation progressed, the reduction rate tended to increase gradually after falling once. In the proposed method, when particles are isolated and the number of nearby particles decreases, the number of particles contained in the AABB decreases, and the effect of the pruning decreases. By contrast, when particles are uniformly distributed, the effect of the pruning is relatively high.

Table 2 shows "Average of search target particles" and "Average of particles in the influence radius." These are the sums of all particles and are average values at all steps. "Average of search target particles" is the average of particles for which it was judged whether the particle was within the influence radius in the neighborhood search. "Average of particles in the influence radius" is the average of particles that were in the influence radius of the particle i as a result of the neighborhood search. "Average of search target particles" in animation example 1 was 427299320.7 for "(1) Uniform grid" and 282275071.5 for "(2) AABB pruning." Therefore, the proposed method pruned 33.9% in animation example 1. Similarly, it pruned 33.3%, 32.4%, 34.7%, and 34.1% in animation examples 2 through 5, respectively. The average of these was 33.7%. In "Average of particles in the influence radius," the results of "(1) Uniform grid" and "(2) AABB pruning" match.

The proposed method is based on the uniform grid and constitutes the AABB in each grid. The AABB requires maximum and minimum values of the xyz coordinates as data, and it is implemented with six real variables. These data are added to the memory amount of the uniform grid according to the number of grids. The memory amount at program execution is shown below. This memory amount was measured with an initial particle placement. In animation example 1, the number of initially placed particles is 1,000,000, and the number of grids was $92 \times 64 \times 50 = 294{,}400$. The memory amount was 6659.0 MB for "(1) Uniform grid" and 6,672.3 MB for "(2) AABB pruning." The memory amount increase of the proposed method compared with the uniform grid was 13.3 MB. This was 0.20% of the memory amount used by "(2) AABB pruning." Similarly, in animation examples 2 through 5, they are 0.18%, 3.22%, 0.83%, and 4.26%, respectively. The memory amounts used by animation examples 3 and 5, with a small number of initially placed particles, are slightly higher than the other animation examples. The memory amount added to the uniform grid by the proposed method was small.

5. Conclusion

In this study, PBF was used as the particle method based on SPH using a uniform influence radius, and the efficiency of the neighborhood search by the uniform grid was improved. Data

structures based on a uniform grid have some grids in which all registered particles are outside the influence radius, depending on the particle position in the grid, and the search is not efficient. In this study, the AABB, which is a kind of boundary volume, was used for the pruning of the neighborhood search. The proposed method was implemented in parallel on a CPU, and the sum of the calculation times for the data structure construction and the neighborhood search was reduced by 20.7% on average compared with the uniform grid. This method matches the result of the neighborhood search by the uniform grid, and there is no difference in the generated image.

References

[1] R. A. Gingold and J. J. Monaghan, Kernel Estimates as a Basis for General Particle Methods in Hydrodynamics, Journal of Computational Physics, 46, pp.429-453, 1988.

[2] M. Ihmsen, J. Orthmann, B. Solenthaler, A. Kolb and M. Teschner, SPH Fluid in Computer Graphics, Eurographics 2014 - State of the Art Reports, 2014.

[3] M. Müller, D. Charypar and M. Gross, Particle-Based Fluid Simulation for Interactive Applications, Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp.154-159, 2003.

[4] M. Becker and M. Teschner, Weakly compressible SPH for free surface flows, Proceedings of the 2007 ACM SIGGRAPH/ Eurographics symposium on Computer animation, pp.209-217, 2007.

[5] B. Solenthaler and R. Pajarola, Predictive-Corrective Incompressible SPH, Proceedings of ACM SIGGRAPH 2009, pp.40:1-40:6, 2009.

[6] K. Bodin, C. Lacoursiere and M. Servin, Constraint Fluids, IEEE Transactions on Visualization and Computer Graphics, 18, 3, pp.516-526, 2012.

[7] X. He, N. Liu, S. Li, H. Wang and G. Wang, Local Poisson SPH For Viscous Incompressible Fluids, Journal of Computer Graphics Forum, 31, 6, pp.1948-1958, 2012.

[8] M. Macklin and M. Müller, Position Based Fluids, ACM Transaction on Graphics, 32, 4, pp.104:1-104:12, 2013.

[9] M. Ihmsen, J. Cornelis, B. Solenthaler, C. Horvath and M. Teschner, Implicit incompressible SPH, IEEE Transactions on Visualization and Computer Graphics, 20, 3, pp.426-435, 2014.

[10] M. Macklin, M. Müller, N. Chentanez and T.-Y. Kim, Unified Particle Physics for Real-time Applications, ACM Transaction on Graphics, 33, pp.153:1-153:12, 2014.

[11] J. Bender and D. Koschier, Divergence-free smoothed particle hydrodynamics, Proceedings of the 14th ACM SIGGRAPH/ Eurographics Symposium on Computer Animation, pp.147-155, 2015.

[12] M. Müller, B. Heidelberger, M. Hennix and J. Ratcliff, Position Based Dynamics, Journal of Visual Communication and Image Representation, 18, 2, pp.109-118, 2007.

[13] T. J. Purcell, C. Donner, M. Camimarano, H. W. Jensen and P. Hanrahan, Photon mapping on programmable graphics hardware, Proceedings of the ACMSIGGRAPH/ EUROGRAPHICS conference on Graphics hardware, pp. 41–50, 2003.

[14] S Green, Cuda particles, NVIDIA whitepaper 2 (3.2), 1, 2008.

[15] J. Kalojanov and P. An Slusallek, A parallel algorithm for construction of uniform grids, Proceedings of the 1st ACM conference on High Performance Graphics, pp.23-28, 2009.

[16] M. Ihmsen, N. Akinci, M. Becker and M. Teschner, A Parallel SPH Implementation on Multi-core CPUs, Journal of Computer Graphics Forum, 30, 1, pp.99-112, 2011.

[17] M. Teschner, B. Heidelberger, M. Müller, D. Pomeranets and M. Gross, Optimized Spatial Hashing for Collision Detection of Deformable Objects, Proceedings of Vision, Modeling, Visualization, pp.47–54, 2003.

[18] E. Guendelman, R. Bridson and R.Fedkiw, Nonconvex rigid bodies with stacking, ACM Transactions on Graphics, 22, 3, 871–878, 2003.

[19] N. Bell, Y. Yu and P. J. Mucha, Particle-based simulation of granular materials, Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 77–86, 2005.

[20] B. Adams, M. Pauly, R. Keiser and L.Guibas, Adaptively sampled particle fluids, ACM Transactions on Graphics - Proceedings of ACM SIGGRAPH 2007, pp.48:1-48:7, 2007.

[21] F. Sin, A. W. Bargteil and J. K. Hodgins, A pointbased method for animating incompressible flow, Proceedings of the 2009 ACM SIGGRAPH /Eurographics Symposium on Computer Animation, pp. 247–255. 3, 5, 2009.

[22] L. Verlet, Computer experiments on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules. Phys. Rev. 159, 1, 98–103, 1967.

[23] S. Hieber, Particle-Methods for Flow-Structure Interactions. PhD thesis, Swiss Federal Institute of Technology, 2007.

[24] B. Pelfrey and D. House, Adaptive neighbor pairing for smoothed particle hydrodynamics. Advances in Visual Computing 6454, 192–201. 3, 2010.

[25] B. Vermuri, Y. Cao and L. Chen, Fast collision detection algorithms with applications to particle flow. Computer Graphics Forum 17, 2 (1998), pp.121-134, 1998.

[26] H. Samet, The design and analysis of spatial data structures, Addison–Wesley, 1990.

[27] P. Goswami, P. Schlegel, B. Solenthaler and R. Pajarola, Interactive SPH Simulation and Rendering on the GPU, Eurographics/ ACM SIGGRAPH Symposium on Computer Animation, pp. 1–10, 2010.

[28] S.-K. Im and K.-H. Chan, Fast Particle Neighbor Searching for Unlimited Scene with Fluid Refraction Improvement, International Journal of Modeling and Optimization, Vol. 6, No. 2, pp.71-76, 2016.

[29] C. Ericson, Real-Time Collision Detection, CRC Press, 2004.

[30] E.Welzl, Smallest enclosing disks (balls and ellipsoids), H.Maurer Ed., New Results and New Trends in Computer Science,

Lecture Notes in Computer Science 555, SpringerVerlag, pp. 359–370, 1991.

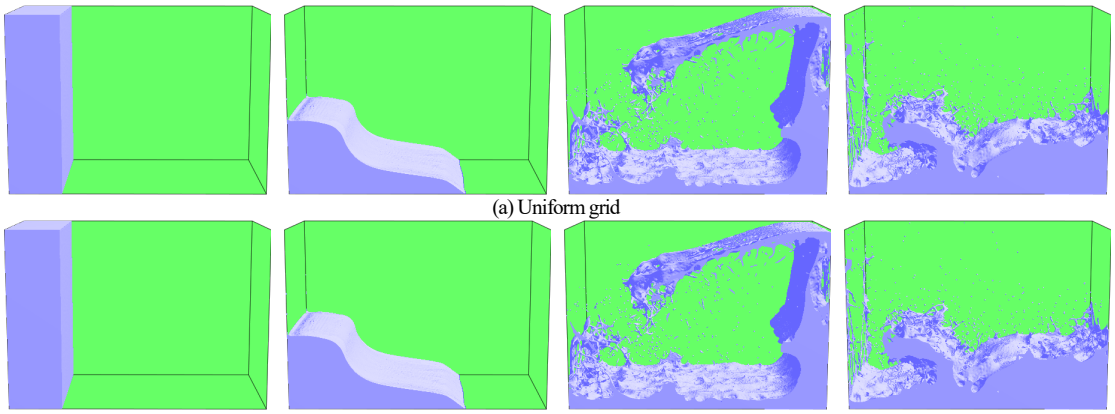
[31] B. Gärtner, Fast and Robust Smallest Enclosing Balls, Proceedings 7th Annual European Symposium on Algorithms (ESA), Lecture Notes in Computer Science 1643, Springer-Verlag, pp.325-338, 1999.

[32] G. Barequet and S. Har-Peled, Efficiently Approximating the Minimum-Volume Bounding Box of a Point Set in Three Dimensions, Journal of Algorithms, Volume 38, Issue 1, pp.91-109, 2001.

[33] C. B. Barber, D. P. Dobkin and H. Huhdanpaa, The Quickhull Algorithm for Convex Hulls, ACM Transactions on Mathematical Software, Vol. 22, No. 4, 1996.

[34] L. Hernquist and N. Katz, TreeSPH: A unification of SPH with the hierarchical tree method, Astrophysical Journal Supplement Series, Vol. 70, pp.419-446, 1989.

Daiki Takeshita received the Ph.D. degree in electrical engineering and computer science from Iwate University in 2005. Since 2018, he has been an Associate professor with the National Institute of Technology, Akita College. His research interests include computer graphics, image processing, and human interfaces. He is a member of the Society of Art and Science, ACM, and IEEE Computer Society.



(a) Uniform grid

(b) AABB pruning

Figure 3. Animation example 1: images of 0.0, 1.4, 4.0, and 6.0 s. A setup called a dam break. There is no difference in the images generated by the two methods.

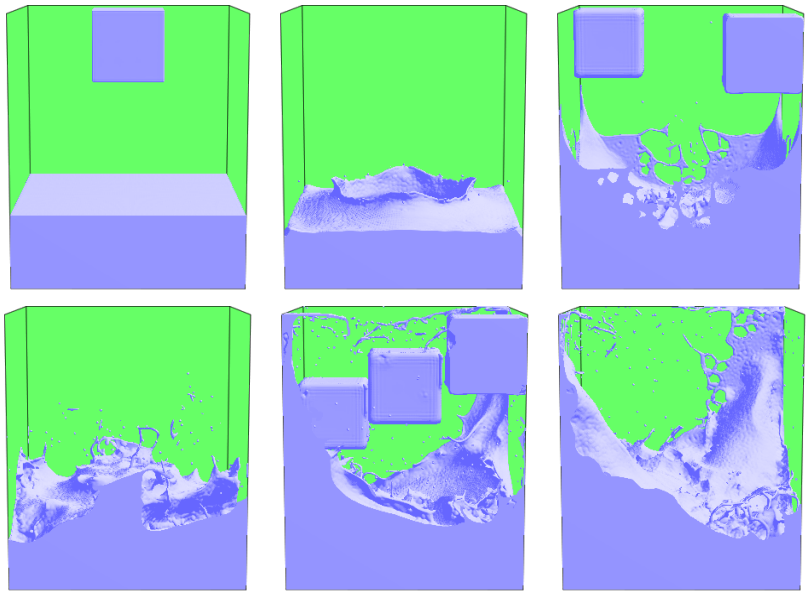


Figure 4. Animation example 2: images of 0.0, 1.6, 3.2, 4.8, 6.2, and 9.0 s. A total of six water chunks dropped on the water surface.

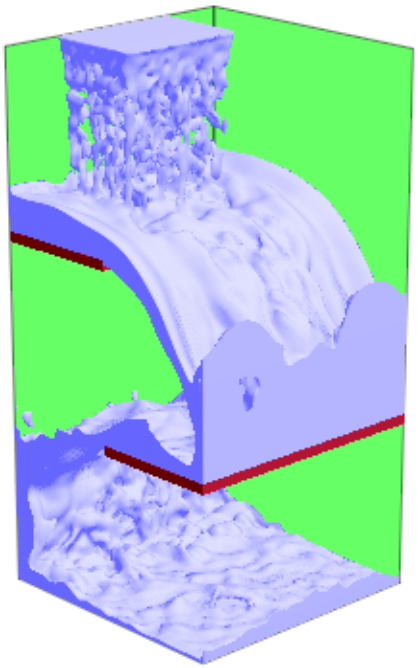


Figure 5. Animation example 3: an image of 8 s. Two plates are placed and water flows from the top of the space.

Figure 6. Animation example 4: an image of 5.56 s. A wave is generated by sliding the wall on the left side of the space.

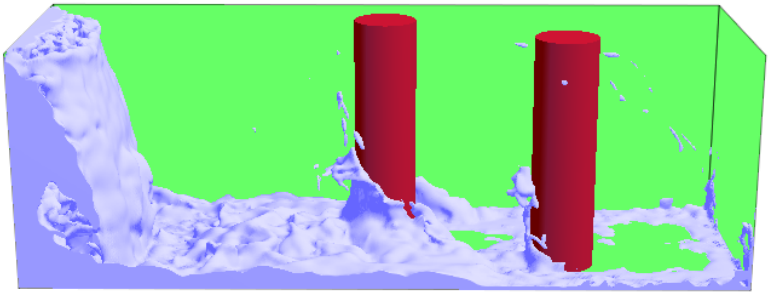
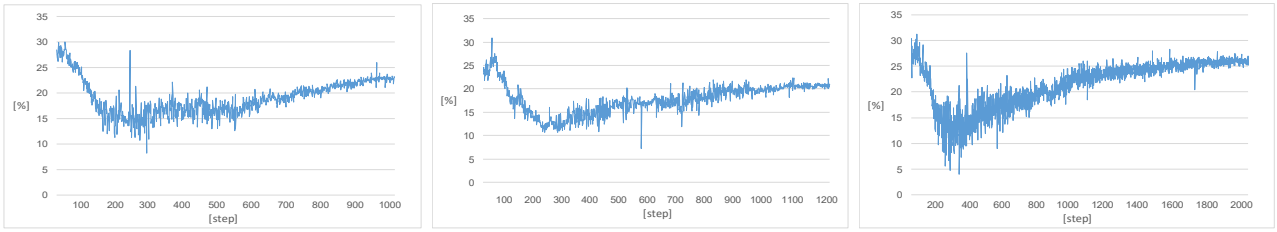
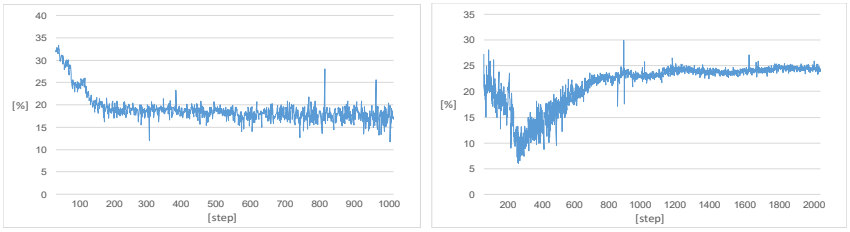


Figure 7. Animation example 5: an image of 3.76 s. Two cylinders are placed and water flows from the upper left side of the space.



(a) Animation example 1 (b) Animation example 2 (c) Animation example 3



(d) Animation example 4 (e) Animation example 5

Figure 5. Reduction rate in the sum of computation time for the construction and the search: a vertical axis of a graph is a reduction rate in a calculation time of the proposed method compared with a uniform grid, and a horizontal axis is a number of steps.

Table 1. Calculation time

Animation	Method	Construction [s]	Search [s]	Construction and search [s]	Particle behavior calculation [s]	Total [s]
Example 1	(1) Uniform grid	0.091	1.613	1.703	10.711	12.415
	(2) AABB pruning	0.099	1.269	1.367	10.699	12.067
Example 2	(1) Uniform grid	0.147	1.613	1.760	10.931	12.691
	(2) AABB pruning	0.156	1.276	1.432	10.955	12.387
Example 3	(1) Uniform grid	0.107	2.411	2.518	14.105	16.622
	(2) AABB pruning	0.105	1.822	1.927	14.101	16.028
Example 4	(1) Uniform grid	0.120	0.792	0.912	6.987	7.899
	(2) AABB pruning	0.097	0.644	0.741	6.973	7.713
Example 5	(1) Uniform grid	0.116	2.501	2.617	13.159	15.776
	(2) AABB pruning	0.147	1.869	2.016	13.131	15.147

Table 2. Average of search target particles and average of particles in the influence radius

Animation	Method	Average of search target particles	Average of particles in the influence radius
Example 1	(1) Uniform grid	427299320.7	70278482.2
	(2) AABB pruning	282275071.5	70278482.2
Example 2	(1) Uniform grid	508952655.3	81610148.5
	(2) AABB pruning	339531792.0	81610148.5
Example 3	(1) Uniform grid	347312760.0	62073916.7
	(2) AABB pruning	234841325.1	62073916.7
Example 4	(1) Uniform grid	407398749.8	66779049.6
	(2) AABB pruning	266156459.9	66779049.6
Example 5	(1) Uniform grid	367846109.7	62117093.3
	(2) AABB pruning	242378598.4	62117093.3