Computers & Graphics 110 (2023) 118–124

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Special Section on AniNex

Anisotropic screen space rendering for particle-based fluid simulation

Yanrui Xu [b], Yuanmu Xu [b], Yuege Xiong [b], Dou Yin [b], Xiaojuan Ban [a,b,d,*], Xiaokun Wang [b,c,e,*], Jian Chang [e], Jian Jun Zhang [e]

[a] Beijing Advanced Innovation Center for Materials Genome Engineering, University of Science and Technology Beijing, Beijing, 100083, China
[b] School of Intelligence Science and Technology, University of Science and Technology Beijing, Beijing, 100083, China
[c] Beijing Key Laboratory of Knowledge Engineering for Materials Science, University of Science and Technology Beijing, Beijing, 100083, China
[d] Key Laboratory of Perception and Control of Intelligent Bionic Unmanned Systems, Ministry of Education, University of Science and Technology Beijing, Beijing, 100083, China
[e] National Centre for Computer Animation, Bournemouth University, Poole, BH12 5BB, United Kingdom

ARTICLE INFO

Article history:
Received 1 December 2022
Accepted 18 December 2022
Available online 23 December 2022

Keywords:
Real-time rendering
Screen space rendering
Fluid simulation
Smoothed particle hydrodynamics

ABSTRACT

This paper proposes a real-time fluid rendering method based on the screen space rendering scheme for particle-based fluid simulation. Our method applies anisotropic transformations to the point sprites to stretch the point sprites along appropriate axes, obtaining smooth fluid surfaces based on the weighted principal components analysis of the particle distribution. Then we combine the processed anisotropic point sprite information with popular screen space filters like curvature flow and narrow-range filters to process the depth information. Experiments show that the proposed method can efficiently resolve the issues of jagged edges and unevenness on the surface that existed in previous methods while preserving sharp high-frequency details.

1. Introduction

The physics-based fluid simulation has great application value in computer-aided Engineering (CAE) and Computer Graphics (CG) fields. Among all the simulation methods, the *particle-based* approaches like *Smoothed Particle Hydrodynamics* (SPH) have received much attention for their algorithmic efficiency and application flexibility [1]. However, for the visualization of particle-based simulation results, tracing surfaces for particle-represented fluid has been the computation bottleneck of the whole process. Hence, how to efficiently visualize the simulated fluid while maintaining proper fidelity has become a hot research topic.

Instead of directly tracing the particle surface and constructing the polygon mesh, the *screen space rendering* approach avoids these expensive processes by only generating the surface with respect to the camera using the depth and thickness information. This way, particles are handled as *point sprites*, 2D textures facing the camera (usually in a round shape). The edges between textures are smoothed out to produce fluid surface effects for the screen [2]. This approach avoids the expensive mesh construction procedure, making real-time fluid simulation viable for gaming and state-of-the-art virtual reality applications [3].

However, several inherited artefacts of screen space rendering hinder its further exploitation and integration into a broader range of applications. For example, despite subsequent improvements in the original method, surfaces extracted from particles are uneven and often too blurred, making it impossible to distinguish a clear boundary between the foreground and the background. Inspired by Yu and Turk [4], we propose an anisotropic transformation scheme for point texture, which is applied before filtering the depth image to generate smooth surfaces in discontinuous regions and, at the same time, preserve vivid details.

2. Related work

Physics-based fluid simulation approaches can be mainly divided into two categories: Eulerian methods [5], which utilize a background grid to discretize space into cells, and meshless Lagrangian methods [6] using particles to represent fluid volumes. For video games, virtual reality, and other real-time graphics applications, particle-based simulation such as SPH [7–9] is often the preferred approach due to its high efficiency and flexibility [10].

To visualize the Lagrangian simulation results, traditionally, fluid surfaces are first traced and constructed as polygon meshes using the marching cubes algorithm [11–13]. Meshes are then rendered as fluid. The major drawback of this two-step scheme is that the mesh generation process consumes considerable computational resources and can be pretty time-consuming for detailed

* Corresponding author at: School of Intelligence Science and Technology, University of Science and Technology Beijing, Beijing, 100083, China.
E-mail addresses: banxj@ustb.edu.cn (X. Ban), wangxiaokun@ustb.edu.cn (X. Wang).

https://doi.org/10.1016/j.cag.2022.12.007

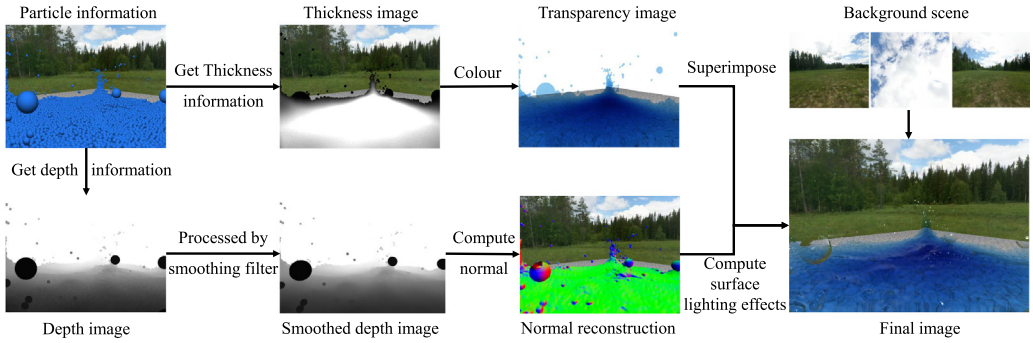


Fig. 1. Schematic flow diagram of screen space rendering pipeline for fluid rendering: First, transform the fluid particles to the screen space and obtain the depth information and thickness information. Second, post-process the depth image, reconstruct the normal on the depth image and render the fluid surface using normal and thickness information.

results. Some researchers have improved the marching cubes algorithm to achieve real-time mapping with a tiny number of particles, but its quality is still unsatisfactory [14].

The screen space rendering technique is an alternative approach for visualizing fluid surfaces. It directly draws the fluid surfaces without the need to produce a surface mesh, sacrificing fidelity for efficiency. Müller et al. [2] first applied this method for fluid rendering. The choice of smoothing filter is the critical factor in determining the rendering result. As shown in Fig. 1, the jagged depth information derived from fluid particles needs to be smoothed to get the fluid surface.

A simple Gaussian filter for surface smoothing [2] tends to produce unwanted blur effects and is often not ideal. The bilateral Gaussian filter [3,15] preserves sharp fluid boundaries but leads to excessive flattening of discontinuous regions. In addition, the bilateral Gaussian filter is not separable, and an approximate separation can result in visual artefacts. Truong and Yuksel [16] introduced a narrow-range filtering technique that smooths the depth map by directly using a narrow range of depth values. This method provides improved surface quality in surface smoothness and preserving boundaries near discontinuities. Recently, Oliveira and Paiva [17] solved the particle deficiency problem [16] by performing a particle classification mechanism. Liu et al. [18] proposed a low-cost differentiable screen-space rendering algorithm for augmented reality (AR).

However, we find that the unevenness of fluid surfaces caused by the distribution of particles still needs to be appropriately handled in screen space rendering. Inspired by work from Yu and Turk [4], we provide anisotropic transformation for particle textures prior to the derivation of depth image to get a smoother fluid surface.

3. Real-time screen space fluid rendering

Fluid rendering in the screen space is performed by drawing the 3D fluid directly into the 2D screen without generating surface meshes and is closely related to image processing algorithms [3]. The schematic diagram describing the procedure of the surface rendering algorithm is shown in Fig. 1 and is detailed in the following.

3.1. Depth information

Deriving the Depth Information First, the distance z_i between each particle and camera pixel i is calculated from the camera viewpoint [19]. To obtain the fluid surface from the camera point of view, the point sprite [20] method is used to render the particles as spheres, which allows us to extract the depth values from the video memory buffer directly.

Smoothing Depth Information Directly using spherical point sprites as depth information usually produces uneven fluid surfaces. To reduce the rugged effects, specific filters are applied to the depth image, flattening the abrupt bumps of particles by averaging z_i for local pixels. One of the most common smoothing filters is a Gaussian filter [21], with the formula:

$$G\left(\mathbf{p}_i, \mathbf{p}_j, \sigma_i\right) = \frac{1}{2\pi\sigma_i^2} e^{-\frac{\|\mathbf{p}_i - \mathbf{p}_j\|}{2\sigma_i^2}}, \tag{1}$$

where j is the neighbour pixel of i, σ is the standard deviation parameter determined by z_i [16], and $\mathbf{p}$ describes the location for each pixel in the screen space. The Gaussian convolution can obtain smooth particle depth information. However, it can cause over-smoothing, leading to some loss of edges. Therefore, variants such as the bilateral Gaussian method are often used to obtain better results.

3.2. Thickness information

Fluid distribution in the real world is not uniform, so it is necessary to calculate the amount of fluid between the camera and the nearest opaque object, called the thickness image. When shading, the thickness image is used to calculate the colour and transparency of the fluid [22]. The formula is given as follows:

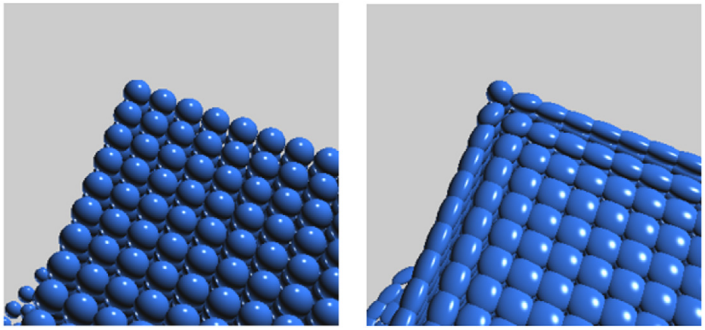
$$T(\mathbf{p}_i) = \sum_{j=0}^{n} G\left(\left\|\mathbf{p}_i - \hat{\mathbf{x}}_j\right\|\right), \tag{2}$$

where n represents all fluid particles, $\hat{\mathbf{x}}_j$ corresponds to the projection position of particle $\mathbf{x}_j$ in the screen space. This calculation is correct when the particles do not overlap. Because of the incompressibility of the SPH method (particles seldom overlap), the thickness information can be calculated by Eq. (2).

3.3. Normal reconstruction and surface shading

After obtaining the vertex coordinates in camera space, the normal needs to be reconstructed from the coordinates to handle light reflection and refraction on the surface. The partial derivatives are calculated according to the finite difference method.

Afterwards, the refraction of the fluid [23] and Blinn–Phong illumination [24] are calculated. The relationship between the transmittance of the fluid and the thickness of the fluid is obtained according to the Beer–Lambert algorithm [25] $I(d) = I_0 d^{-kd}$, where d is the thickness of the fluid, I_0 is the value of light intensity and k is the attenuation coefficient vector of the fluid. Then the final fluid colouring result is obtained.



(a) isotropic (b) anisotropic

Fig. 2. Comparison of isotropic and anisotropic fluid particles.

4. Anisotropic transformation of point sprites for fluid particles

In fluid simulation, a finer-grained spatial discretization can be achieved by increasing the simulation resolution by representing fluid using smaller particles, generating smoother edges when rendered. However, increasing the resolution for 3D fluid simulation requires complex computations, which seriously affects the performance in real-time. To combine both efficiency and quality, our approach is based on the weighted principal components analysis (WPCA) of SPH fluid particles. We obtain anisotropic kernel functions to obtain anisotropic fluid particles based on their distribution and use the eigenvalue magnitudes on the eigenvectors as the basis for fluid particle deformation to achieve efficient smoothing and detail preservation. The effects of this anisotropic transformation on the screen space are presented in Fig. 2.

4.1. Tracing surface using smoothing kernel

In particle-based fluid simulation, fluid particles are represented as spheres, which are then used to perform screen space projection based on point sprites or surface reconstruction using the marching cubes algorithm. Since the discrete fluid particles are isotropic, for a fluid particle i whose centre is located at the spatial coordinate $\mathbf{x}_i$, the approximated value ϕ_i of the colour field at its location can be averaged and weighted by all its neighbouring particles as:

$$\phi_i = \sum_j \frac{m_j}{\rho_j} W\left(\|\mathbf{x}_i - \mathbf{x}_j\|, h\right), \tag{3}$$

where m_j and ρ_j are the volume and density of the neighbouring fluid particle j, W is called the smoothing kernel or kernel function, which is a normalized Gaussian-like function [1], h is the support radius of the kernel function, beyond which W takes the value of 0. In this paper, we use an anisotropic smoothing kernel $W(\mathbf{r}, \mathbf{B})$ to replace the isotropic function $W(r, h)$ by introducing the anisotropy matrix $\mathbf{B}$ into the kernel function:

$$W(\mathbf{r}, \mathbf{B}) = \beta \det(\mathbf{B}) P(|\mathbf{rB}|), \tag{4}$$

where β is the scaling factor, P is a symmetric decaying spline with finite support.

The anisotropy matrix $\mathbf{B}$ makes the smoothing kernel anisotropic by stretching and rotating the relative position $\mathbf{r}$ between particles. Fig. 2(a) shows the fluid particles in the normal state, and Fig. 2(b) shows the fluid particles after anisotropic processing. It can be easily observed that particles become more compact and smooth at the edges of the fluid when using the anisotropic representation.

4.2. Deriving anisotropy matrix

The anisotropic kernel function mentioned above can be considered a generalization of the isotropic kernel function. Let $\mathbf{I}$ be a d-dimensional unit matrix. When $B = \frac{1}{h}\mathbf{I}$, $W(\mathbf{r}, \mathbf{B})$ returns to $W(r, h)$. To make the values of the kernel function larger along the tangential direction of the fluid surface than the normal direction, or to stretch the kernel along the edges of the fluid, WPCA [26] is performed to find the eigenvalues and eigenvectors according to neighbouring fluid particles. Since the Lagrangian simulation algorithm is naturally non-uniform in particle distribution, the use of the WPCA method effectively avoids the sensitivity to outliers in the traditional principal component analysis (PCA) method and enhances the efficiency of surface smoothing.

First, a weight value ω_{ij} is determined for each pair of fluid particles. A weighted covariance matrix is then constructed using the weight values, and singular value decomposition (SVD) is performed on the covariance matrix. The obtained matrix principal axes, as well as the eigenvalues, are finally used to construct the anisotropy matrix $\mathbf{B}$. The weighted average position of the fluid is given as $\bar{\mathbf{x}}_i = \sum_j \omega_{ij}\mathbf{x}_j$, where ω_{ij} is the weight of the fluid neighbour particle j to particle i. In this paper we compute ω_{ij} as:

$$\omega_{ij} = \frac{W\left(\left|\mathbf{x}_i - \mathbf{x}_j\right|, h\right)}{\sum_j W\left(\left|\mathbf{x}_i - \mathbf{x}_j\right|, h\right)}. \tag{5}$$

This results in a larger normalized weighting when the particles are closer. The displacement of the weighted average position from the original position can indicate the distribution trend of the fluid surface. Based on this trend, a covariance matrix $\mathbf{C}_i$ can be constructed for the fluid particle i as:

$$\mathbf{C}_i = \sum_j \omega_{ij}\left(\mathbf{x}_j - \bar{\mathbf{x}}_i\right)\left(\mathbf{x}_j - \bar{\mathbf{x}}_i\right)^T. \tag{6}$$

A singular value decomposition for the covariance matrix yields $\mathbf{C}_i = \mathbf{W}\mathbf{\Sigma}\mathbf{W}^T$, where the diagonal matrix $\mathbf{\Sigma} = diag(\sigma_1, \sigma_2, \ldots, \sigma_d)$ contains all the eigenvalues from the largest to the smallest. To obtain a stable anisotropic deformation effect, the minimum of eigenvalues is restricted to a threshold with respect to the largest eigenvalue. The value of σ_k^r is chosen as $\sigma_k^r = max(\sigma_1/\kappa, \sigma_k)$ to adjust the smaller eigenvalues so that the ratio of the largest eigenvalue to the smallest one is limited. In our experiments, we choose $\kappa = 4$. Furthermore, no anisotropic kernel function is used when the number of neighbouring particles is less than or equal to 20, and a scale control factor s makes $|s\mathbf{C}| \approx 1$ to keep the particle volume from changing significantly. We select $s = 1400$ in our experiments. The regularized covariance matrix can be expressed as:

$$\mathbf{C}^r = W\mathbf{\Sigma}^r W^T. \tag{7}$$

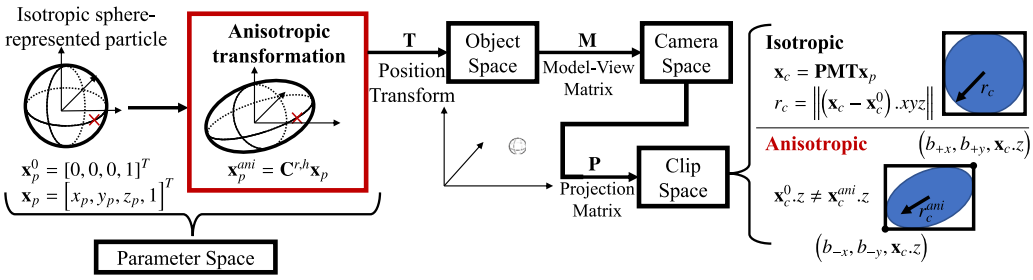
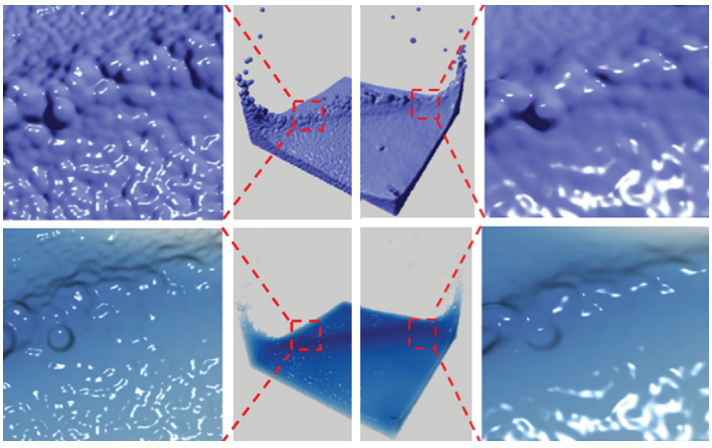


Fig. 3. The pipeline of rasterizing the fluid particle onto the screen space. For representing fluid particles using an isotropic sphere. The stretching effects from the projection matrix P are negligible. So particles always have a round appearance at the clip space, and the radius r_c can be measured from the bias between the position of sphere centre $\mathbf{x}_c^0$ and any point on the sphere surface $\mathbf{x}_c$ at the clip coordinate. To perform a more desirable fluid effect, we apply the anisotropic transformation for each particle at the parameter space (red box). Furthermore, the resulting shape on the clip space is elliptical. The boundary of the ellipse cannot be derived using the traditional method because the point on the resulting surface may not lie on the projected plane (not having the same depth $\mathbf{x}_c^0.z \neq \mathbf{x}_c^{ani}.z$).



(a) isotropic (b) anisotropic

Fig. 4. Experimental comparison of isotropy and anisotropy in the dam break scenario.

4.3. Transforming sphere particles onto the screen space

As shown in Fig. 3, the key implementation of the anisotropic transformation converting the spherical particle to an ellipsoid in this paper is by applying a 4×4 homogeneous anisotropic transformation matrix $\mathbf{C}^{r,h}$ onto the sphere at the parameter space, which takes the form:

$$\mathbf{C}^{r,h} = \begin{bmatrix} \mathbf{C}^r & & 0 \\ & & \vdots \\ 0 & \cdots & 1 \end{bmatrix}. \tag{8}$$

To determine the size of the projected ellipse for an anisotropic point sprite, we apply the method proposed by Sigg et al. [27], where the positions (b_{-x}, b_{-y}) and (b_{+x}, b_{+y}) at the clip coordinate can be computed by inversely transforming the position back to the parameter space. For $\mathbf{x}_c$ inside the point sprite, $\mathbf{n}_c^T \mathbf{x}_c \leq 0$ with $\mathbf{n}_c = (1, 0, 0, b_x)^T$ must be satisfied. To trace b_x back to the parameter space, we have:

$$\mathbf{n}_p = \left(\mathbf{P} \cdot \mathbf{M} \cdot \mathbf{T} \cdot \mathbf{C}^r\right)^T \mathbf{n}_c, \tag{9}$$

where

$$\mathbf{T} = \begin{bmatrix} r_c & 0 & 0 & x_o \\ 0 & r_c & 0 & y_o \\ 0 & 0 & r_c & z_o \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{10}$$

is a scaling matrix to satisfy $\left\|\mathbf{x}_p.xyz\right\| = r_c$ in Fig. 3. Here x_o is the x component of particle position at the object space. And the notation $.xyz$ in Fig. 3 denotes the vector consisting of the first three elements in a four-dimensional position vector $\mathbf{x}$.

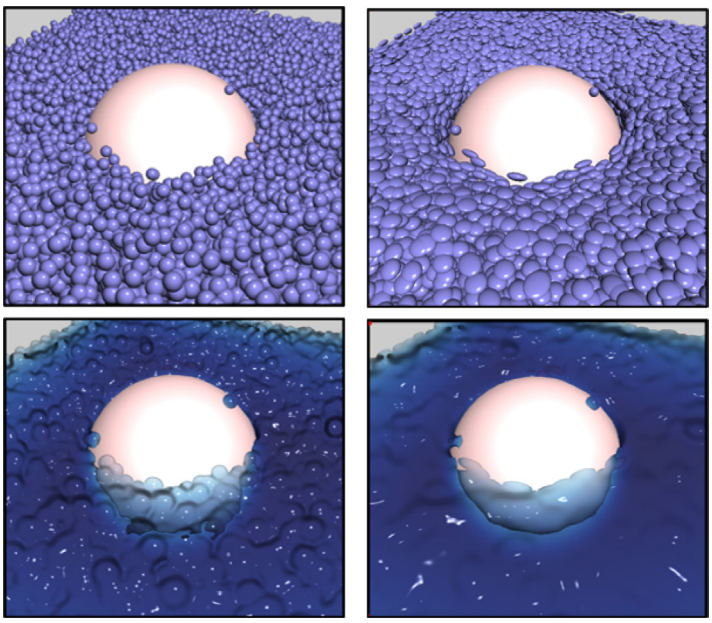
5. Results and analysis

The experiments below are performed using a hardware platform of AMD Ryzen 7 5800H @3.20 GHz, 32 GB memory, and NVIDIA RTX 3060. The 3D graphics API OpenGL is used for particle rendering, and C++ is used as the hardware graphics interactive language to process logical operations. In addition, GLSL colouring language is used to calculate the fluid optical effect in GPU. The real-time performance of fluid is maintained during all experiments.

5.1. Anisotropic processing results

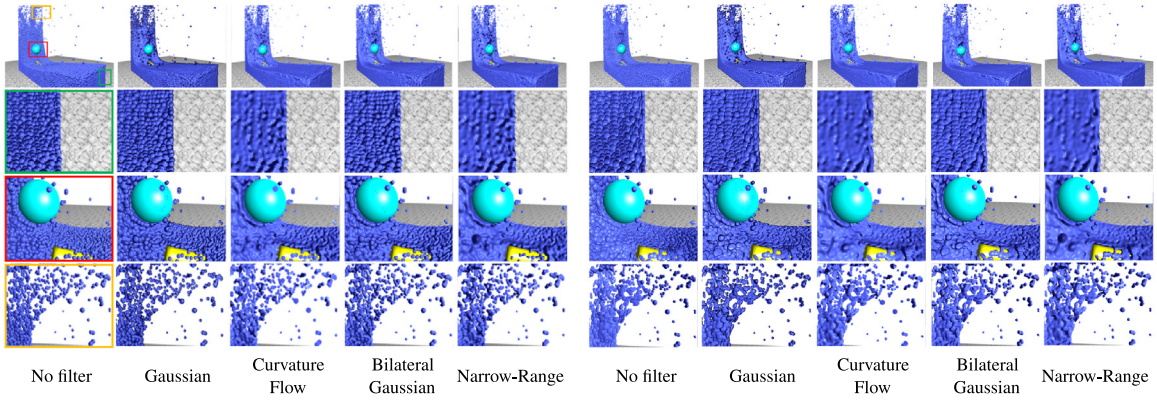
In this subsection, the anisotropic algorithm is adopted to process fluid particles to reduce the roughness of the fluid depth map, which can improve the final rendering effect of the fluid surface. As shown in Fig. 4,(a) and (b) respectively represent the surface rendering results when two water blocks collide under isotropic and anisotropic conditions. It can be observed that the overall surface rendering results processed by the anisotropic algorithm are smoother, and the illumination shading is more realistic with fine highlights.

In addition, experimental verification is conducted for a fluid–solid coupling scenario, as shown in Fig. 5. In the fluid–solid coupling scenario, real-time fluid rendering based on the anisotropic algorithm shows better surface results with smoother surfaces, especially at the interface between the rigid body and liquid.



(a) isotropic (b) anisotropic

Fig. 5. Experimental comparison of isotropy and anisotropy in the fluid–solid coupling scenario.



(a) isotropic point sprites representation (b) anisotropic point sprites representation

Fig. 6. Experimental comparison of isotropy and anisotropy in the dam break scenario. Columns 1–5 is the (a) part of this figure, demonstrating surface constructed without anisotropic transformation; Columns 6–10 is the (b) part of this figure, showing how anisotropic transformation works with and without different smoothing filters.

5.2. Combination with popular smoothing filters

In this subsection, the anisotropic algorithm is combined with various popular smoothing filters to verify the effectiveness and practicability of the proposed scheme under actual application scenarios. The Gaussian filter, the bilateral Gaussian filter and the narrow-range filter use the same number of iterations ($iter = 2$). The curvature flow filter uses more iterations ($iter = 80$) to obtain flat surface results at the cost of performance loss.

Fig. 6 demonstrates the scenario where a dam-break fluid collides with multiple differently shaped solid objects. Complex boundary geometries can be observed in this case. We conducted this experiments using various kinds of smoothing filters considering both isotropic and anisotropic conditions. Figs. 6(a) and 6(b) show the surfaces constructed using isotropic and anisotropic point sprites respectively. It can be seen that the proposed anisotropic transformation scheme can enhance surface performance with almost every state-of-the-art smoothing filter. The second to the third row of Fig. 6 exhibit how anisotropic transformation reduces the unevenness of particle distribution on the free surface and coupling boundaries. And the fourth row shows our method can also help to gather the sparsely distributed fluid particles representing splashes into more well-organized structures.

Fig. 7 is another experiment that further shows the anisotropic surface generation effects of the fluid under different filters with a more complex fluid-rigid coupling boundary. The Gaussian filter is a separable filter with high computational efficiency. However, it still produces a large amount of noise on the surface, which is far from the desired ideal result. Meanwhile, the boundary morphed into particles due to a lack of neighbouring particles in the splashing area (second row of Fig. 7). The Bilateral Gaussian filter solved the boundary problem in the particle-deficiency areas and reduced some noise on the surface. However, it produces an over-flattening phenomenon in the discontinuous surface details, and the edge information between particles is lost. After 80 iterations,

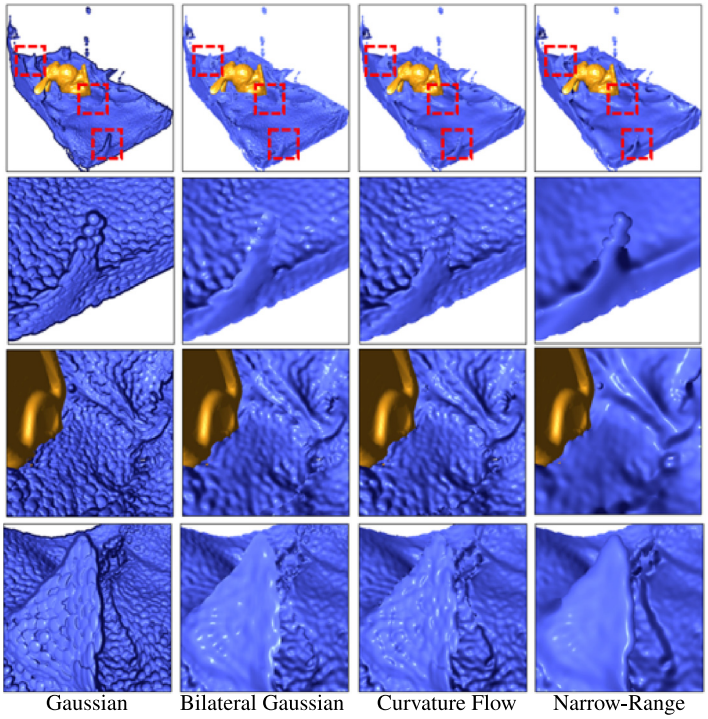


Fig. 7. From left to right are Gaussian filter, bilateral Gaussian filter, curvature flow filter and narrow-range filter.

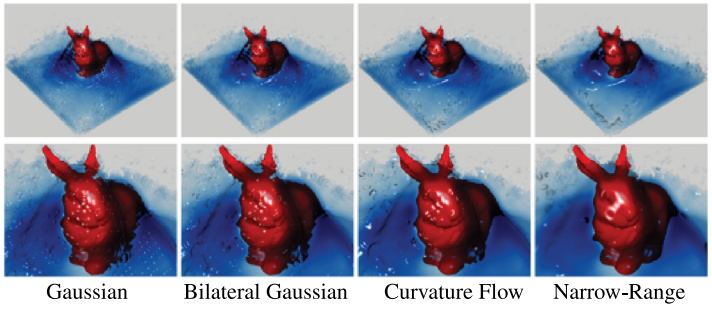


Fig. 8. From left to right are Gaussian filter, bilateral Gaussian filter, curvature flow filter and narrow-range filter.

the curvature flow filter produces a good surface effect in some areas. However, this method is unstable and time-consuming. The narrow-range filter method effectively solves these problems with better surface effects and time efficiency.

Fig. 8 shows the influence of different filtering algorithms with transparent fluids. It can be seen that compared with other algorithms, the narrow-range filter algorithm can produce a smoother fluid effect. Moreover, the overall highlight can form an obvious bright area, which is more consistent with the water highlight effect in the real world.

5.3. Performance analysis of the anisotropic transformation

To evaluate how the refined anisotropic transformation affects the efficiency of the screen space rendering. We carried out the experiment of Fig. 4 using multiple configurations with various numbers of fluid particles, alternative smoothing filters, and the on/off status of the anisotropic transformation.

The relationship between particle number and corresponding frame rate under different methods is shown in Fig. 9. We can see that the anisotropic transformation barely affects the efficiency of the whole screen space rendering pipeline. And although the curvature flow filter can achieve better surface effects with a higher number of iterations, its rendering time also increases. In contrast, the narrow-range filter method not only gives a better surface effect but also keeps the rendering time almost linearly increased with respect to the number of particles.

6. Conclusion

We propose an anisotropic real-time surface rendering scheme based on the screen space approach. The method uses WPCA to determine an anisotropic transform for particle point sprites, solving the problem of jagged edges and uneven surfaces of the fluid in traditional screen space rendering. We also compare the rendering effect and efficiency of various popular smoothing filters by experiments. It is concluded that the narrow-range filter is the preferred method for screen space rendering because it can obtain a better surface effect and maintain an acceptable frame

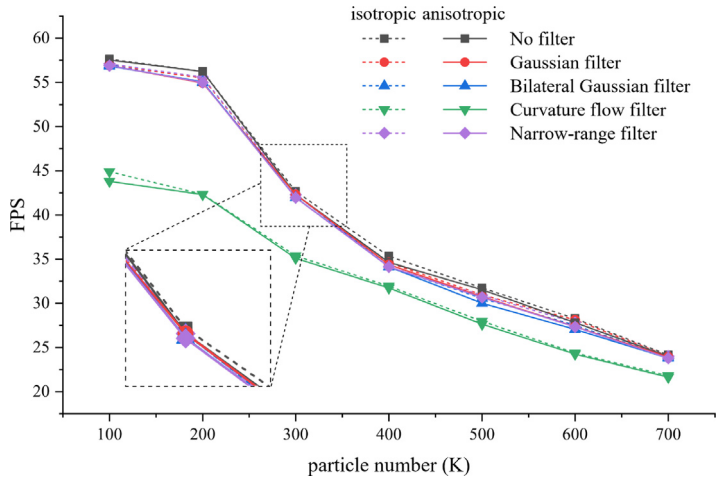


Fig. 9. Frame rate comparison of different algorithms.

rate. In the future, we will study the screen-space rendering of complex phenomena, such as the real-time visualization of multiphase fluid mixing, through the screen-space mapping of material type information.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is confidential.

Acknowledgements

This research was funded by: Horizon 2020-Marie Skłodowska-Curie Action-Individual Fellowships (No. 895941), National Natural Science Foundation of China (No. 61873299), Key Research and Development Project of Hainan Province (No. ZDYF2020031), Fundamental Research Funds for the Central Universities, China (QNXM20220043).

References

[1] Ihmsen M, Orthmann J, Solenthaler B, Kolb A, Teschner M. SPH fluids in computer graphics. In: Lefebvre S, Spagnuolo M, editors. Eurographics 2014 - State of the Art Reports. The Eurographics Association; 2014.
[2] Müller M, Keiser R, Nealen A, Pauly M, Gross M, Alexa M. Point based animation of elastic, plastic and melting objects. In: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. SCA '04, Goslar, DEU: Eurographics Association; 2004, p. 141–51.
[3] Green S. Screen space fluid rendering for games. In: Proceedings for the game developers conference. Moscone Center San Francisco, CA; 2010.
[4] Yu J, Turk G. Reconstructing surfaces of particle-based fluids using anisotropic kernels. ACM Trans Graph 2013;32(1).
[5] Shakib F, Hughes TJ, Johan Z. A new finite element formulation for computational fluid dynamics: X. the compressible Euler and Navier-Stokes equations. Comput Methods Appl Mech Engrg 1991;89(1):141–219, Second World Congress on Computational Mechanics.
[6] Becker M, Teschner M. Weakly compressible SPH for free surface flows. In: Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. SCA '07, Goslar, DEU: Eurographics Association; 2007, p. 209–17.
[7] Ihmsen M, Cornelis J, Solenthaler B, Horvath C, Teschner M. Implicit incompressible SPH. IEEE Trans Vis Comput Graphics 2014;20(3):426–35.
[8] Bender J, Koschier D. Divergence-free smoothed particle hydrodynamics. In: Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation. SCA '15, New York, NY, USA: Association for Computing Machinery; 2015, p. 147–55.
[9] Wang X, Liu S, Ban X, Xu Y, Zhou J, Kosinka J. Robust turbulence simulation for particle-based fluids using the rankine vortex model. Vis Comput 2020;36(10):2285–98.
[10] Carensac S, Pronost N, Bouakaz S. Optimizations for predictive–corrective particle-based fluid simulation on GPU. Vis Comput 2022;1–13.
[11] Lorensen WE, Cline HE. Marching cubes: A high resolution 3D surface construction algorithm. ACM SIGGRAPH Computer Graphics 1987;21(4):163–9.
[12] Wang X, Ban X, Liu X, Zhang Y, Wang L. Effective reconstructing surfaces algorithm of anisotropic kernels orienting SPH fluids. Journal of Computer-Aided Design & Computer Graphics 2016;28(9):1497–505.
[13] Yang W, Gao C. A completely parallel surface reconstruction method for particle-based fluids. Vis Comput 2020;36(10):2313–25.
[14] Wang X, Ban X, Liu X, Zhang Y, Wang L. Efficient extracting surfaces approach employing anisotropic kernels for SPH fluids. Journal of Visualization 2016;19(2):301–17.
[15] Neto LdSR, Apolinário Jr. AL. Real-time screen space cartoon water rendering with the iterative separated bilateral filter. SBC J Interact Syst 2017;8(1):20–32.
[16] Truong N, Yuksel C. A narrow-range filter for screen-space fluid rendering. Proc ACM Comput Graph Interact Tech 2018;1(1).
[17] Oliveira F, Paiva A. Narrow-band screen-space fluid rendering. Comput Graph Forum 2022;41(6):82–93.
[18] Liu C, Wang L, Li Z, Quan S, Xu Y. Real-time lighting estimation for augmented reality via differentiable screen-space rendering. IEEE Transactions on Visualization and Computer Graphics 1–1.
[19] Cornells N, Van Gool L. Real-time connectivity constrained depth map computation using programmable graphics hardware. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), Vol. 1. IEEE; 2005, p. 1099–104.
[20] Shreiner D, Sellers G, Kessenich J, Licea-Kane B. OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.3. Pearson Education; 2013.
[21] Levin A, Fergus R, Durand F, Freeman WT. Image and depth from a conventional camera with a coded aperture. ACM Trans Graph 2007;26(3):70–es.
[22] Young IT, van Vliet LJ. Recursive implementation of the Gaussian filter. Signal Process 1995;44(2):139–51.
[23] Conde M, Vega C, Patrykiejew A. The thickness of a liquid layer on the free surface of ice as obtained from computer simulation. J Chem Phys 2008;129(1):014702.
[24] Blinn JF. Models of light reflection for computer synthesized pictures. 11, (2):New York, NY, USA: Association for Computing Machinery; 1977, p. 192–8,
[25] Swinehart DF. The Beer-Lambert law. J Chem Educ 1962;39(7):333.
[26] Koren Y, Carmel L. Visualization of labeled data using linear transformations. In: IEEE symposium on information visualization 2003 (IEEE Cat. No. 03TH8714). IEEE; 2003, p. 121–8.
[27] Sigg C, Weyrich T, Botsch M, Gross M. GPU-based ray-casting of quadratic surfaces. In: Proceedings of the 3rd eurographics / IEEE VGTC conference on point-based graphics. SPBG '06, Goslar, DEU: Eurographics Association; 2006, p. 59–65.