Editable Mesh Animations Modeling Based on Controlable Particles for Real-Time XR

Xiangyang Zhou, Yanrui Xu, Chao Yao, Xiaokun Wang and Xiaojuan Ban

Abstract—The real-time generation of editable mesh animations in XR applications has been a focal point of research in the XR field. However, easily controlling the generated editable meshes remains a significant challenge. Existing methods often suffer from slow generation speeds and suboptimal results, failing to accurately simulate target objects' complex details and shapes, which does not meet user expectations. Additionally, the final generated meshes typically require manual user adjustments, and it is difficult to generate multiple target models simultaneously. To overcome these limitations, a universal control scheme for particles based on the sampling features of the target is proposed. It introduces a spatially adaptive control algorithm for particle coupling by adjusting the magnitude of control forces based on the spatial features of model sampling, thereby eliminating the need for parameter dependency and enabling the control of multiple types of models within the same scene. We further introduce boundary correction techniques to improve the precision in generating target shapes while reducing particle splashing. Moreover, a distance-adaptive particle fragmentation mechanism prevents unnecessary particle accumulation. Experimental results demonstrate that the method has better performance in controlling complex structures and generating multiple targets at the same time compared to existing methods. It enhances control accuracy for complex structures and targets under the condition of sparse model sampling. It also consistently delivers outstanding results while maintaining high stability and efficiency. Ultimately, we were able to create a set of smooth editable meshes and developed a solution for integrating this algorithm into VR and AR animation applications.

Index Terms—Particle Systems, Control Particles, Virtual Environments, Augmented Reality Application

1 Introduction

Particle control generation technology effectively bridges the gap between geometry-based target models and particles to generate similar editable meshes, increasingly becoming a cornerstone of computer graphics. These techniques have a variety of applications, from creating real-time VR/AR interactive scenes to movie effects. However, in traditional Lagrangian methods, the challenge lies in the excessive reliance on parameter adjustments to control particle generation for individual targets. This challenge leads to difficulties in controlling when multiple diverse targets coexist within a scene. Additionally, existing methods often face challenges when dealing with intricate geometries. Control issues arise, particularly when the target model exhibits complex shapes or sparse sampling, leading to hollowing or particle splashing phenomena. In contrast to 3D reconstruction methods, they are unable to reconstruct intricate hollow models and individual target models.

Traditional Lagrangian methods for particle control, such as the constraint control force proposed by Zhang *et al.* [59] (PBFC) and the skinning mechanism by Lu *et al.* [23] (RSCF), Their skinning mechanisms still exhibit noticeable drawbacks, such as the inability to preserve target details or mitigate control effects accurately. Schoentgen *et al.* [44] (LCAT) has introduced the use of precomputed templates to assist artists in achieving their desired outcomes more effortlessly.

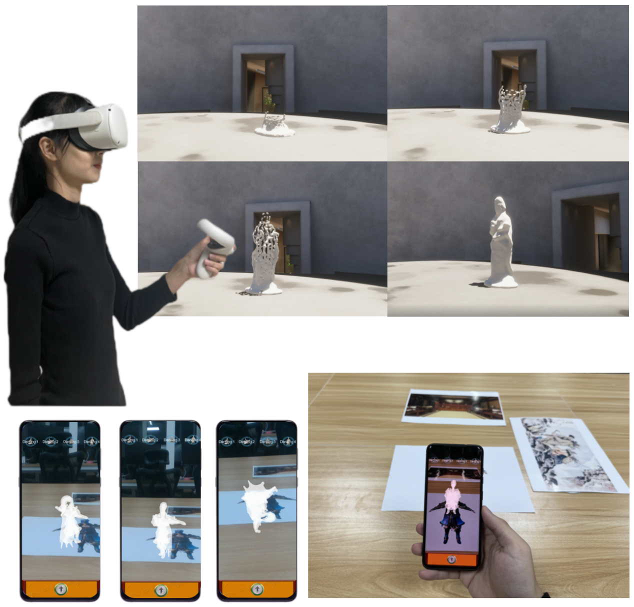


Fig. 1: The control animations crafted as editable meshes, can be applied in real-time to VR and AR applications.

Nevertheless, this technique is primarily suited for short-term dynamic shape control and struggles to maintain object shapes accurately over extended periods. Zhou *et al.* [60] (Weighted) proposed controlling the fluid based on the sampling ratio, but their method only exhibits better control effectiveness in simplified models and may lead to situations of excessive control. As we aim to create a freely editable mesh ultimately, we also compared algorithms for mesh generation using 3D Gaussian Splatting [18] (3DGS) methods. Tang *et al.* [50] could generate a mesh from a single-view image, but due to the limitation of single-view images, the generated mesh lacks a significant amount of information. Additionally, due to constraints inherent in 3DGS, they are unable to generate models with cavities. Antoine *et al.* [14] directly

- *X. Ban, C. Yao and X. Wang are the corresponding authors.*
- *X. Zhou, X. Ban is with Beijing Advanced Innovation Center for Materials Genome Engineering, School of Intelligence Science and Technology, University of Science and Technology Beijing. E-mail: xiaoyongyuan@xs.ustb.edu.cn, banxj@ustb.edu.cn.*
- *C. Yao is with School of Computer and Communication Engineering, University of Science and Technology Beijing and Key Laboratory of Advanced Materials and Devices for Post-Moore Chips, Ministry of Education. E-mail: yaochao@ustb.edu.cn.*
- *X. Wang and Y. Xu are with School of Intelligent Science and Technology, University of Science and Technology Beijing. E-mail: wangxiaokun@ustb.edu.cn, xuyanruiedw@me.com.*

Received 18 September 2024; revised 13 January 2025; accepted 13 January 2025. Date of publication 11 March 2025; date of current version 31 March 2025. This article has supplementary downloadable material available at https://doi.org/10.1109/TVCG.2025.3549573, provided by the authors Digital Object Identifier no. 10.1109/TVCG.2025.3549573

performs mesh reconstruction on the results of 3DGS. By replicating their method, we found that the effectiveness of their mesh generation is not optimal. Furthermore, the meshes they generate cannot be recognized by existing skeletal plugins for automatic skeletal insertion.

To overcome the challenge of controlling the generation of multiple targets, we propose a versatile and user-friendly control framework for creating dynamic animations intuitively and effectively. The paper opted for blue noise sampling [17], which effectively captures the intricate details of the targets. A weighted spatially adaptive control mechanism for particle coupling is proposed. It aligns with the spatial distribution of control particles, enabling precise formation of target shapes. Additionally, boundary correction techniques are introduced to prevent boundary degradation, which significantly reduces the occurrence of particle splashing. A pioneering aspect of this research is the distance-adaptive particle fragmentation mechanism, which dynamically adjusts particle coupling based on particle distances and control parameters. It ensures the avoidance of unnecessary accumulation for the final shape. With these advancements, it achieves relatively precise and rapid dynamic generation of target shapes.

In summary, the main contributions include:

- An adaptive particle bond optimization algorithm leveraging the target model's spatial sampling features for precise control force calculation, which enhances the speed of the generation process,
- An algorithm combining blue noise sampling and boundary correction with an adaptive bond-breaking mechanism, which applies boundary forces to confine particles within the target model and dynamically adjusts based on density and distance, thereby enhancing result precision,
- A solution capable of directly generating editable meshes and automatically embedding skeletal animations. It can be applied in real-time to AR and VR applications.

2 RELATED WORK

Dynamic particle control is essential in the field of animation, and generating editable meshes is also a popular research topic in computer graphics, with many researchers offering various solutions.

2.1 Dynamic Control Of Static Shapes

Control based on key-frame: Treuile *et al.* [53] used optimization techniques to match keyframe shapes in their paper. Meanwhile, Hong and Kim [15] used keyframe potential fields for smoke simulations. While Bergou *et al.* [3] enriched simulations with physical details, Chu *et al.* [7] derived velocity fields from single frames. Though most studies focus on 2D simulations, there's a growing interest in the realism of 3D simulations and their control mechanisms. Pan *et al.* [35, 37] introduced optimization techniques to improve simulation efficiency. Despite their visual success, these methods lack interactive simulation. Later, Pan [36] used direct deformation grids in simulations, showing promising results in areas like smoke simulations but having limited overall applicability.

Control based on target model: Researchers have explored various methods for guiding particle formation using specific control models. Fattal and Lischinski [10] proposed two methods: one for directing smoke to a set density field and another to limit smoke diffusion. Raveendran *et al.* [39] suggested using high-density target control grids to influence simulation pressure, while Zhang *et al.* [59] used models with forces like density and springs to shape particle surfaces. Cornelis *et al.* [8] delved into particle-target interactions, and Lu *et al.* [23] provided a particle control approach from rigging perspectives. Schoentgen *et al.* [44] recommended pre-defined control templates to aid artists in shaping particles.

2.2 Control Techniques For Detail Modules

Enhancement and control of low-resolution details: Nielsen *et al.* [33] used low-res input to enhance high-res smoke simulations, adding more detail while maintaining the overall appearance. Expanding on this, Nielsen *et al.* [31, 32] introduced an Eulerian-based optimization, improving simulation efficiency. Forootaninia *et al.* [11] merged high-frequency particle dynamics with low-frequency guiding fields for detailed smoke simulation. Sato *et al.* [41], building upon this, improved smoke shape control using Forootaninia's method.

Dynamic physical field-based control: McNamara *et al.* [26] utilized gradient-based optimization for particle control under physical constraints. Shi and Yu [46] corrected particle discrepancies using feedback forces, while Sato *et al.* [43] employed time-varying potentials to guide particle movement. Thuerey [51] introduced grid-based space-time deformation for diverse phenomena. Inglis *et al.* [16] worked on aligning guiding and target velocity fields. Meanwhile, Rasmussen *et al.* [38] and Stomakhin and Selle [47] harnessed constraints and boundary conditions for enhanced particle control.

2.3 Data-driven Control Methods

Ren *et al.* [40]focuses on developing a versatile controller for dynamic systems involving fluid and solid interactions. It emphasizes adapting to various dynamic behaviors and multiple tasks without re-training. Morton *et al.* [29] introduce a method based on Koopman theory for predicting fluid flow dynamics. Kim *et al.* [19] propose a CNN-based generative model for efficient fluid simulation. Li *et al.* [21] develop dynamic particle interaction networks. Ummenhofer *et al.* [54] presents a novel approach using continuous convolutional networks for fluid dynamics simulation. Brunton *et al.* [6] provides a comprehensive overview of machine learning in fluid mechanics. Tang *et al.* [49] explore deep reinforcement learning for active flow control in fluid dynamics.

2.4 Integrated Control Technologies Within Systems

Fluid control research began with Foster and Metaxas [12], who adjusted physical properties to guide fluid animations and introduced actuators for interface purposes. Later, Mihalef *et al.* [27] presented an intuitive wave simulation, though with limited wave profiles and scripting capabilities. Advancements in fluid simulation have led to innovative techniques. Bojsen-Hansen and Wojtan [4] devised a method for seamlessly integrating precomputed liquid animations. Manteaux *et al.* [25] introduced a sculpting tool for artists to edit fluid surfaces without altering the entire simulation. Stuyck and Dutre [48] improved this tool in terms of speed and usability. Sato *et al.* [42] developed a technique to merge different fluid data for diverse animation effects, focusing on mesh simulations. Sevinc *et al.* [45] developed a VR system for engineers to design autonomous vehicle test scenarios, which outperforms traditional methods in both accuracy and efficiency. Zhang *et al.* [58] introduced a natural VR control for drone viewpoints, employing adaptive origin updates. Ogawa *et al.* [34] crafted a VR jumping experience, 'PseudoJumpOn', simulating real jump dynamics without physical steps. Lastly, Bozgeyikli *et al.* [5] launched 'Tangiball', a tangible VR football game with enhanced user immersion and no additional motion sickness.

2.5 Mesh reconstruction based on 3DGS radiance fields

The mesh reconstruction method based on 3DGS radiance fields can reconstruct point cloud data from 3DGS into mesh grids. Antoine *et al.* [14] align Gaussian particles with the scene surface using regularization terms, binding Gaussian particles to the mesh surface to generate an editable mesh. Tang *et al.* [50] transform diffusion models and three-dimensional Gaussian matrices into texture meshes, enabling the reconstruction of three-dimensional meshes from single-view images. Joanna *et al.* [55] parameterize each Gaussian component with the vertices of mesh faces, initializing the mesh based on input or estimated meshes and defining the positions of Gaussian particles on the mesh. This allows for shape modifications by adjusting mesh positions, but the method has limited mesh degrees of freedom. Xie *et al.* [57] employ the Material Point Method (MPM) to seamlessly integrate physical simulation and visual rendering, enabling more physically accurate material deformation. However, this method is still constrained by Gaussian reconstruction limitations, offering limited degrees of freedom in simulation and lacking real-time applicability.

Table 1: Meaning of the main formula symbols.

Symbol	Meaning	Symbol	Meaning	Symbol	Meaning
D_p	Particle dataset	λ	Lagrangian multiplier	W	Smoothing kernel function
x_p	The position of particles	a	The acceleration of particles	ω	Control particle weight proportion
v_p	The velocity of particles	Δx	Particle displacement	ρ_c	The density of controlparticles
m_p	The mass of particles	D_c	Control particle dataset	H_c	Support radius of control particles
ρ_p	The density of particles	x_c	The position of control particles	φ	Weighted coefficient
H_p	Support radius of particles	v_c	The velocity of control particles	$\widetilde{\varphi}$	Normalized weight coefficient
F	The force on particles	m_c	The mass of control particles		

3 PRELIMINARY

We employ Position-Based Fluid (PBF) [24] simulation to compute fundamental particle motion, utilizing optimized blue noise sampling [20] of the target model as control particles.

3.1 Fundamental Particle Motion

The PBF approach is an evolution built upon the foundations of the Position-Based Dynamics (PBD) [30] and Smoothed Particle Hydrodynamics (SPH) [28] methodologies. Unlike traditional physical process-solving methods, it directly updates the object based on constraint relationships between spatial positions. The PBF technique enhances PBD by assimilating concepts from SPH, enabling its applicability to particle dynamics. The PBF algorithm can directly satisfy constraint equations by modifying the positions of particles, making it more convenient for particle control.

3.2 Sampling Control Particle

Witkin *et al.* [56] used an implicit surface sampling method to generate boundary particles; however, their approach only allows for changes in sampling results by replacing the implicit function, making it unsuitable for sampling complex models. Bell *et al.* [1] generates basic surface particles using signed distance and allows the particles to be uniformly distributed with a certain offset from the original mesh surface. However, this method produces imprecise results and is only suitable for sampling simple models. We conducted voxelization sampling [9] and blue noise sampling on the model and, through comparative analysis of results and experiments, found that blue noise sampling yields superior outcomes. Consequently, we have adopted the most optimized blue noise sampling technique available [20]. Using blue noise in conjunction with DFSPH [2] spatial sampling on the initial triangular mesh of the model, a more uniform and fuller particle surface can be achieved.

The green module in Figure 2 represents the specific implementation process of the algorithm. When importing data into the experimental environment, the control particles are initially weighted based on spatial sampling results to adaptively adjust the magnitude of applied forces. This facilitates the particles to quickly fill into challenging edge regions, thereby enhancing generation speed. Subsequently, the control particles are designed to attract surrounding particles. Even with precise sampling, sampling gaps inevitably occur in complex large-scale models. To enhance control robustness and efficiency, the algorithm first guides particles to regions with higher densities of control particles to swiftly establish the main outline, followed by a detailed filling process. This ensures the efficiency and effectiveness of the results.

Subsequently, the details are refined. To maintain particles within the model during generation, a corrective force is applied to re-constrain particles that stray outside. It enhances stability. Finally, to ensure that the generated shape remains consistent over time, an adaptive breaking mechanism is introduced to the particle bonds. It prevents excessive expansion of the model over time.

The red module in Figure 2 represents our multi-target scenario design, where multiple target models are not contained within a single model file. Instead, they accept multiple model files as input, generating multiple model datasets. Each model independently controls the fluid.

4 METHOD

This section presents the details of our method. In Figure 2, the blue modules represent the core components of our algorithm, with the first module involving the input of basic data for sampling particles and control particles into the system. The method incorporates three stages: spatially adaptive weighted control of particle bonds (Section 4.2), external and corrective forces acting on the particles (Section 4.3), and spatial distance-adaptive particle bond breaking (Section 4.4).

Our algorithm not only enhances precision and efficiency but also adeptly handles missing sample particles. Furthermore, it generates unique animations based on model features. Through Figure 3, we illustrate the advantages of our algorithm. Despite the presence of gaps in the model's sampling, the final output remains unaffected. Instead, our algorithm leverages sampling features to generate distinctive animations, prioritizing dense regions with controlled particles while filling in the missing sampling areas. Other methods may suffer from control instability due to sampling deficiencies, resulting in particle splattering.

4.1 Basic Control Process

We will introduce our main symbolic meanings in conjunction with Table 1. Particle data $D_p = \left(x_p, v_p, m_p, H_p, \rho_p, W, F, \lambda, a, \Delta x\right)$ and control particle data $D_c = (x_c, v_c, m_c, H_c, \rho_c, W, \omega, \varphi)$ are inputted into the experimental environment. The particle information D_p encompasses several attributes: x_p represents the particle's 3D position in space, v_p its velocity, m_p its mass, H_p its support radius, ρ_p its density, and W its smoothing kernel function. For density estimation, we use the Poly6 kernel, while the Spiky kernel is employed for gradient computations. Additionally, F denotes the external forces acting on the particle, λ the Lagrangian multiplier used for displacement calculations, a the acceleration of the particle, and Δx the displacement for the next time step. The control particle information D_c similarly includes x_c for the 3D position of the control particle in space, v_c is the velocity of control particles, we can make the control particles move through v_c, m_c its mass, H_c its support radius, ρ_c its density, W its smoothing kernel function, and ω is the weight obtained by control particles based on the spatial characteristics of the sampling results, φ is the weighting coefficient obtained through weight value processing.

The green module in Figure 2 represents the specific implementation process of the algorithm. When importing data into the experimental environment, the control particles are initially weighted based on spatial sampling results to adaptively adjust the magnitude of applied forces. This facilitates the particles to quickly fill into challenging edge regions, thereby enhancing generation speed. Subsequently, the control particles are designed to attract surrounding particles. Even with precise sampling, sampling gaps inevitably occur in complex large-scale models. To enhance control robustness and efficiency, the algorithm first guides particles to regions with higher densities of control particles to swiftly establish the main outline, followed by a detailed filling process. This ensures the efficiency and effectiveness of the results.

Subsequently, the details are refined. To maintain particles within the model during generation, a corrective force is applied to re-constrain particles that stray outside. It enhances stability. Finally, to ensure that the generated shape remains consistent over time, an adaptive breaking mechanism is introduced to the particle bonds. It prevents excessive expansion of the model over time.

The red module in Figure 2 represents our multi-target scenario design, where multiple target models are not contained within a single model file. Instead, they accept multiple model files as input, generating multiple model datasets. Each model independently controls the particle.

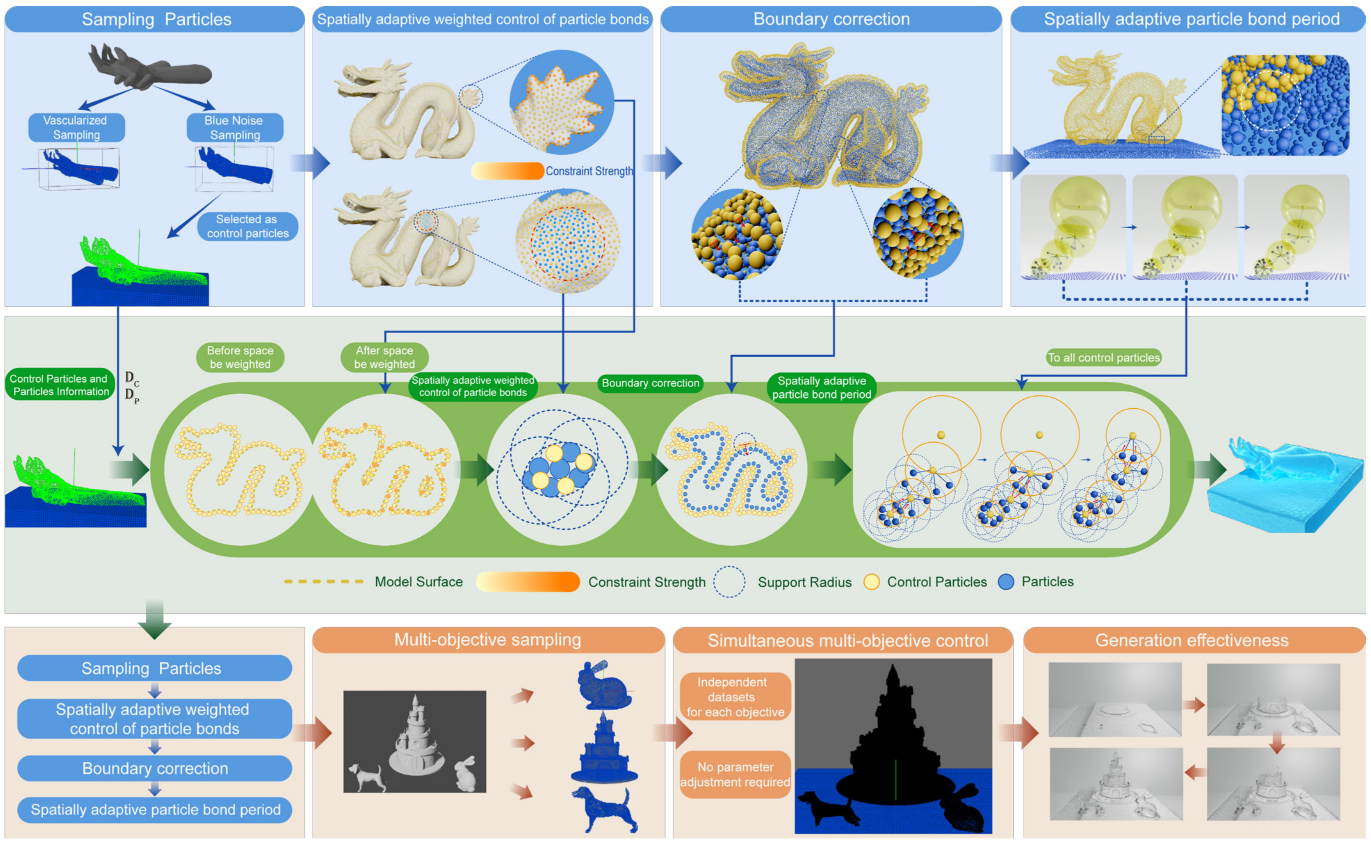

Sampling Particles
Vascularized Sampling
Blue Noise Sampling
Selected as control particles
Spatially adaptive weighted control of particle bonds
Constraint Strength
Boundary correction
Spatially adaptive particle bond period
Control Particles and Particles Information
D_c
D_p
Before space be weighted
After space be weighted
Spatially adaptive weighted control of particle bonds
Boundary correction
Spatially adaptive particle bond period
To all control particles
Model Surface
Constraint Strength
Support Radius
Control Particles
Particles
Sampling Particles
Spatially adaptive weighted control of particle bonds
Boundary correction
Spatially adaptive particle bond period
Multi-objective sampling
Simultaneous multi-objective control
Independent datasets for each objective
No parameter adjustment required
Generation effectiveness

Fig. 2: The blue modules represent the main components of our algorithm, the green modules depict the overall control process, and the red module demonstrates how our method enables the simultaneous generation of multiple targets with different characteristics within a single scene.

4.2 Spatially Adaptive Weighted Control Of Particle Bonds

4.2.1 Adaptive Spatial Weighting

To enhance the representation of surface details on controlled particles and accurately guide particles under weighted field conditions, particularly for complex topological target models with high surface area-to-volume ratios, a weight propagation algorithm is developed based on Smoothed Particle Hydrodynamics (SPH). This allows for the weighted influence of control particles based on their weights, thereby strengthening the control over the external surface of the target model and accelerating the formation of the model shape. The first step involves identifying the information of the weighting center:

$$||\bar{x}|| = \frac{\sum_i (\omega_i \cdot ||x_{c_i}||)}{\sum_i \omega_i}. \tag{1}$$

For control particle i, $||\bar{x}||$ represents the L2 norm of the three-dimensional positional information. The specific weight coefficients ω_i are calculated using the following formula:

$$\omega_i = \frac{W_i\left(x_{c_i} - x_{c_j}, H_c\right)}{\sum_j W_j\left(x_{c_i} - x_{c_j}, H_c\right)}, \tag{2}$$

W is a smoothing kernel function, while the Spiky kernel is employed for gradient computations. In the entire text, i represents a specific property of the current particle, while j represents a specific property of the particles adjacent to i. The weight of a control particle increases as the number of surrounding control particles decreases, and then the control particles are weighted accordingly:

$$\varphi_i = \left| ||x_{c_i}|| - ||\bar{x}|| \right|, \tag{3}$$

φ is the weighting coefficient obtained through weight value processing. For ease of subsequent calculations, the weighted data is normalized:

$$\widetilde{\varphi}_i = \frac{\varphi_i - \varphi_{min}}{\varphi_{max} - \varphi_{min}} + c, \tag{4}$$

c is a constant. The normalized weight coefficient $\widetilde{\varphi}_i$ is always positive before taking the logarithm, where $0.1 > c > 0$.

The magnitude of particle position updates is adjusted based on the weighted values:

$$\Delta x = \alpha \widetilde{\varphi}_i \lambda \nabla x C(x), \tag{5}$$

α is a proportional coefficient that can influence the speed of particle motion. It should not be excessively large, as this may lead to instability in the generation process. $C(x)$ represents a constraint function. The proportional scaling coefficient λ is derived from the equation used in PBF:

$$\lambda = -\frac{C_i\left(x_1, \ldots, x_n\right)}{\sum_k \left|\nabla x_k C_i\left(x_1, \ldots, x_n\right)\right|^2 + \mathscr{E}}. \tag{6}$$

Where $\mathscr{E}$ is a user-specified relaxation parameter that is constant, $\mathscr{E}$ purpose is to prevent the denominator from becoming extremely small, which could cause the value of λ a to become excessively large and lead to instability.

The upper part of the second blue module in Figure 2 represents a spatial weighting schematic. The fewer neighboring control particles around a given control particle, the greater the control force required to ensure rapid and accurate filling of the target shape. In this representation, control particles are denoted by yellow particles, with a more intense yellow indicating a stronger control influence.

4.2.2 Motion Control

We control the flow of the particles by satisfying the density constraints. When the density of control particles is higher, the density of surrounding particles increases to meet the constraints, ensuring that particles

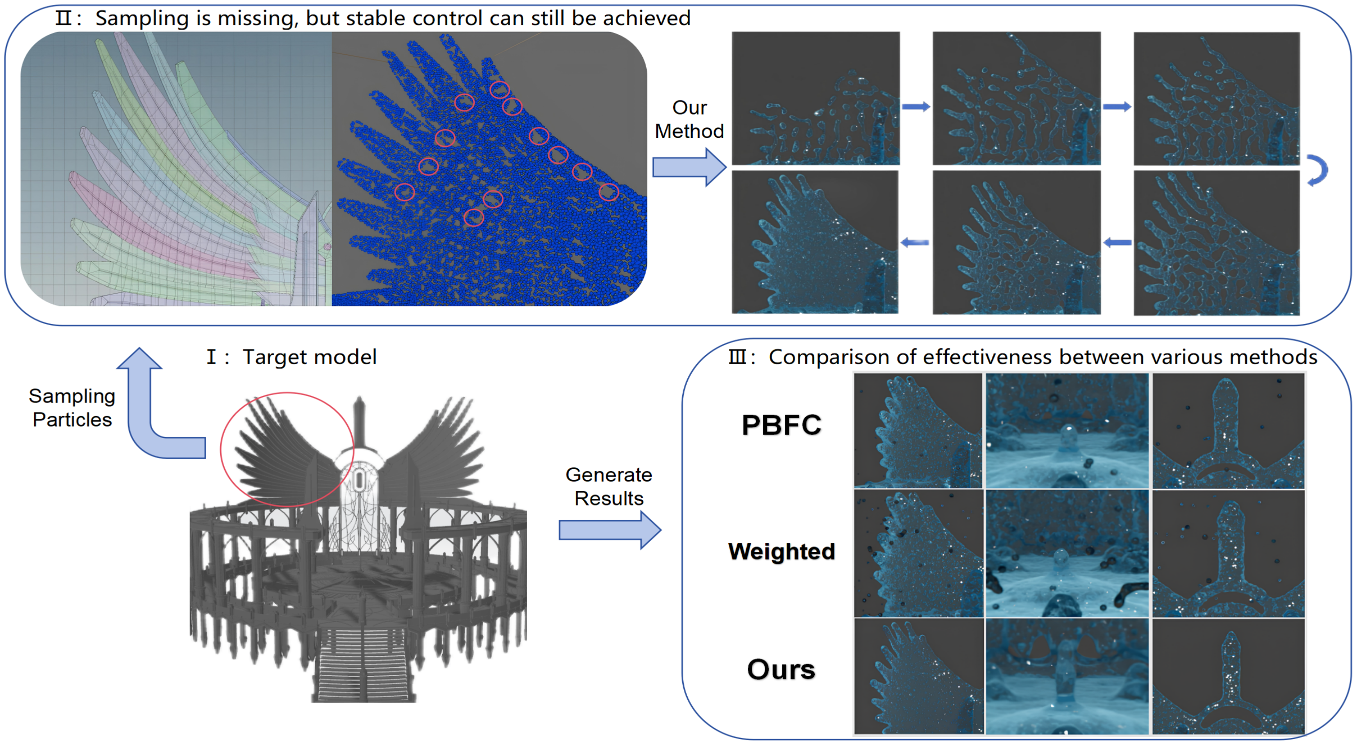

Ⅱ: Sampling is missing, but stable control can still be achieved
Our Method
Ⅰ: Target model
Sampling Particles
Generate Results
Ⅲ: Comparison of effectiveness between various methods
PBFC
Weighted
Ours

Fig. 3: The I module represents the target model, with prominently highlighted red circles indicating the areas of focus for comparison. The II module left demonstrates how modeling the wing section of a building using a quadrilateral mesh may result in sampling deficiencies, as indicated by the highlighted regions within the red circles. The II module right outlines the generation process of the method under sparse sampling, where the target process is precisely shaped based on the spatial distribution characteristics of sampling points. The III module depicts the final comparison between the effects generated by our method and those produced by other methods. Other methods may suffer from control instability due to sparse sampling, leading to particle splattering and shape distortions.

first flow towards the main sampled area and then fill in lower-density regions. This process ensures a faster generation and a better fit for the target model. The constraint algorithm is represented as follows:

$$C_i\left(x_{c_i}\right)=\frac{\rho_{p_i}}{\rho_{c_i}}-1\approx 0. \tag{7}$$

In this context, ρ_{p_i}represents the density of particles x_{p_i} surrounding the control particles, and ρ_{c_i} denotes the density of control particles in the vicinity. The specific equation for calculating density is as follows:

$$\rho_{p_i}=\sum_j m_{p_j}W\left(x_{c_i}-x_{p_j},H_c\right), \tag{8}$$

$$\rho_{c_i}=\sum_j m_{c_j}W\left(x_{c_i}-x_{c_j},H_c\right). \tag{9}$$

Finally, the displacement of the particles should occur along the direction of the negative density gradient:

$$\Delta x_i^{\text{control}}=-\alpha\widetilde{\varphi}_i\frac{\sum_j\lambda m_{p_j}\nabla W\left(x_{c_i}-x_{p_j},H_c\right)}{\sum_j m_{c_j}W\left(x_{c_i}-x_{c_j},H_c\right)}. \tag{10}$$

As depicted in the two-dimensional schematic beneath the second blue module and within the green area in Figure 2, yellow particles represent control particles, red indicates selected control particles, and blue signifies controlled particles. In this approach, particles are attracted and adhere to control particles. However, unlike conventional methods, once the particle count reaches a predetermined density around the control particles, no additional particles are attracted. This prevents particles from exceeding the boundaries formed by the control particles and enables rapid formation of critical contours.

4.3 Boundary Correction

The particles themselves are influenced by external forces F, such as viscosity F_{vis}, surface tension F_{sur}, drag forces F_{drag}, gravity F_{g}, and boundary correction forces F_{corr}. We apply the boundary correction force to rectify the particle's motion.

$$F\left(x_p\right)=F_{\text{vis}}\left(x_p\right)+F_{\text{sur}}\left(x_p\right)+F_{drag}\left(x_p\right)+F_{\text{g}}\left(x_p\right). \tag{11}$$

For the treatment of viscosity F_{vis}, the artificial viscosity approach XSPH is employed to facilitate the conservation of computational resources:

$$F_{\text{vis}}\rightarrow a=\frac{\eta}{h}\sum_j\frac{m_p}{\rho_{p_j}}\left(v_{p_i}-v_{p_j}\right)W\left(x_{p_i}-x_{p_j},H_p\right), \tag{12}$$

η represents the viscosity coefficient I have set ($\eta=0.01$), h denotes the time step, and ρ_{p_j} signifies the density of particles x_{p_j} surrounding a given particle x_{p_i}. The surface tension F_{sur} was calculated using Zorilla's [61] method through Monte Carlo integration:

$$F_{sur}\rightarrow a=-\frac{\sigma\kappa\hat{n}_i}{m_p}, \tag{13}$$

σ is the surface tension coefficient I have set ($\sigma=0.05$), κ represents the local curvature, and $\hat{n}_i$ is the unit normal vector of the particles. The drag force F_{drag} was calculated using Gissler's [13] method:

$$F_{drag}\rightarrow a=\frac{s}{2}\rho_a v_{\text{irel}}^2 C_d A, \tag{14}$$

s is the scaling factor set by us ($s=0.01$). ρ_a represents the density of the air, we choose $\rho_a=1.2931kg/m^3$ in this paper. v_{irel}^2 is the square of the relative velocity difference, that is, the square of the difference in velocity between the particles and the air $\left(v_{\text{irel}} = v_p - v_a, v_a = [0,0,0]\right)$. C_d is the drag coefficient of the particles. A is the cross-sectional area exposed to the particles. Gravity F_{g} is represented by a downward gravitational acceleration $a = [0,-10,0]$. The boundary correction force F_{corr} is an adjustment to the acceleration to eliminate acceleration in the normal direction:

$$F_{corr} \to a = -\hat{n}\left(n \cdot a\right), \tag{15}$$

$$n = -\sum_j m_c \nabla W\left(x_{c_i} - x_{c_j}, H_c\right), \tag{16}$$

n is the negative directional vector controlling the particle density gradient. The dot product is used to obtain the scalar magnitude of the acceleration in this direction, which is then multiplied by the unit vector $\hat{n}$ to obtain a directional vector.

Algorithm 1 Core Algorithm Process

```
1:  for all particles $i$ do
2:      calculate predicted acceleration $a_i^* \Leftarrow a_i + \frac{F}{m_p}$
3:      predict velocity $v_i^* \Leftarrow v_i + \Delta t a_i^*$
4:      predict displacement $x_i^* \Leftarrow x_i + \Delta t v_i$
5:  end for
6:  if particles controlled by control particles then
7:      for all controlled particles $i$ do
8:          calculate acceleration $a_i^* \Leftarrow F_{corr}$, update $v_i^*, x_i^*$
9:          calculate the displacement of controlled $\Delta x_i^{control}$
10:         update particle positions $x_i^* \Leftarrow \mathrm{x}_i^* + \Delta x_i^{\mathrm{control}}$
11:     end for
12: else
13:     for all particles $i$ do
14:         calculate the displacement using the PBF method $\Delta x_i^{\mathrm{PBF}}$
15:         update particle positions $x_i^* \Leftarrow \mathrm{x}_i^* + \Delta x_i^{\mathrm{PBF}}$
16:     end for
17: end if
18: for all particles $i$ do
19:     update velocity $v_i \Leftarrow \frac{(x_i^* - x_i)}{\Delta t}$
20:     update acceleration $a_i \Leftarrow \frac{(v_i - v_i^*)}{\Delta t}$
21:     update position $x_i \Leftarrow x_i^*$
22: end for
```

4.4 Spatial Distance-adaptive Particle Bond Breaking

We introduce an adaptive control that dynamically adjusts the magnitude of control forces to prevent excessive clustering of particles. Specifically, when particles are distant from their control counterparts, the control force is amplified, driving the particles rapidly toward the control points. Conversely, as particles approach their targets, the control force diminishes, ensuring stability within the designated region. To further enhance adaptability, an automatic parameter tuning approach is adopted to adjust parameters based on real-time model states and key performance indicators. This adaptiveness ensures model stability across diverse particle behaviors and environmental conditions while minimizing the need for frequent manual intervention.

Based on the aforementioned observations, a novel adaptive algorithm is proposed, aiming at dynamically adjusting constraint forces through density to achieve more refined, stable, and efficient particle control:

$$\mathrm{f}(\mathrm{x}) = \frac{\rho_{c_i}}{\rho_{p_i}\mathrm{x} + \rho_{c_i}}. \tag{17}$$

When the ρ_{p_i} density is lower, for the same value of x, the corresponding function $\mathrm{f}(\mathrm{x})$ is larger. However, when the density reaches a stable state, the greater the value x, the smaller the function $\mathrm{f}(\mathrm{x})$. Updating the position of particles:

$$\Delta x_i^{adaptive} = \mathrm{f}\left(x_{c_i} - x_{p_i}\right)\frac{x_{c_i} - x_{p_i}}{\left|x_{c_i} - x_{p_i}\right|}, \tag{18}$$

$$H_c{}^* = \mathrm{f}(H_c)H_c. \tag{19}$$

When there are more particles surrounding a control particle, resulting in a higher density, the control force decreases. Furthermore, when $\left|x_{c_i} - x_{p_i}\right| > H_c{}^*$, $\Delta x_{\mathrm{i}}^{adaptive} = 0$. Relying solely on the method of adaptively adjusting constraint strength, minor particle accumulation is observed. To address this, an additional adaptive mechanism is introduced so that particles exceeding the density threshold can be removed from the control.

Algorithm 2 Adaptive Control Algorithm

```
1:  if $\left|x_{c_i} - x_{p_i}\right| > H_c{}^*$ then
2:      for all controlled particles $i$ do
3:          calculate $\Delta x_i^{adaptive} = 0, \Delta x_i^{\mathrm{control}} = 0$
4:          calculate particle positions $x_i \Leftarrow \mathrm{x}_i^*$
5:      end for
6:  else
7:      for all controlled particles $i$ do
8:          calculate $\Delta x_i^{adaptive}$
9:          calculate particle positions $x_i \Leftarrow \mathrm{x}_i^* + \Delta x_i^{adaptive} + \Delta x_i^{\mathrm{control}}$
10:     end for
11: end if
```

As illustrated in the fourth blue module and the corresponding green module in Figure 2, the yellow solid line defines the adaptive control range of the controlled particles, while the blue dashed line represents the search radius of the particles. Particles take the farthest controlled particle as the reference target, searching for controlled particles within this radius. The blue solid line indicates the selected and constrained controlled particles, the red solid line represents the remaining unselected controlled particles, and the absence of a connecting solid line indicates that the particle is not influenced by controlled constraints. It's noteworthy that as the number of particles surrounding a control particle increases, leading to a rise in density, the adaptive control range of the control particle shrinks proportionally to this density variation. Particles exceeding this adaptive control boundary will no longer be influenced by the constraint forces. Importantly, these modifications are designed not to impact the original search radius H_c of control particles. Through this approach, we've optimized control in a more targeted manner, ensuring the precision and stability of the model across diverse scenarios.

5 AUTOMATIC EDITABLE ANIMATION SOLUTION

Creating a mesh that can automatically recognize skeletal free editing is essential for real-time applications in AR/VR. Both particle methods and Gaussian reconstruction techniques often require surface reconstruction of a large number of points. If the reconstruction process is not properly managed, it can produce irregular meshes with excessive polygon counts, resulting in unnecessary mesh data that does not accurately represent the actual model. An excessive number of polygons increases computational load and can hinder recognition by existing skeletal identification plugins, complicating mesh editing and preventing the import of created animation data. This poses significant challenges for users.

Our algorithm addresses these issues. As shown in Figure 4, the first step combines the particle data obtained from the control with our mesh reconstruction method to create a smooth, editable mesh model. This model can then be automatically recognized by existing skeletal plugins, allowing for the insertion of skeletal information. In the second step, users can apply their own animation data in conjunction with the skeletal setup to create a movable model. In the third step, we validated our method's application in both AR and VR, demonstrating promising results. Users can also independently edit rendering styles within these environments.

5.1 Editable Mesh

To achieve the reconstruction of the particle surface, we have improved the marching cubes [22] algorithm. Initially, we partitioned the space

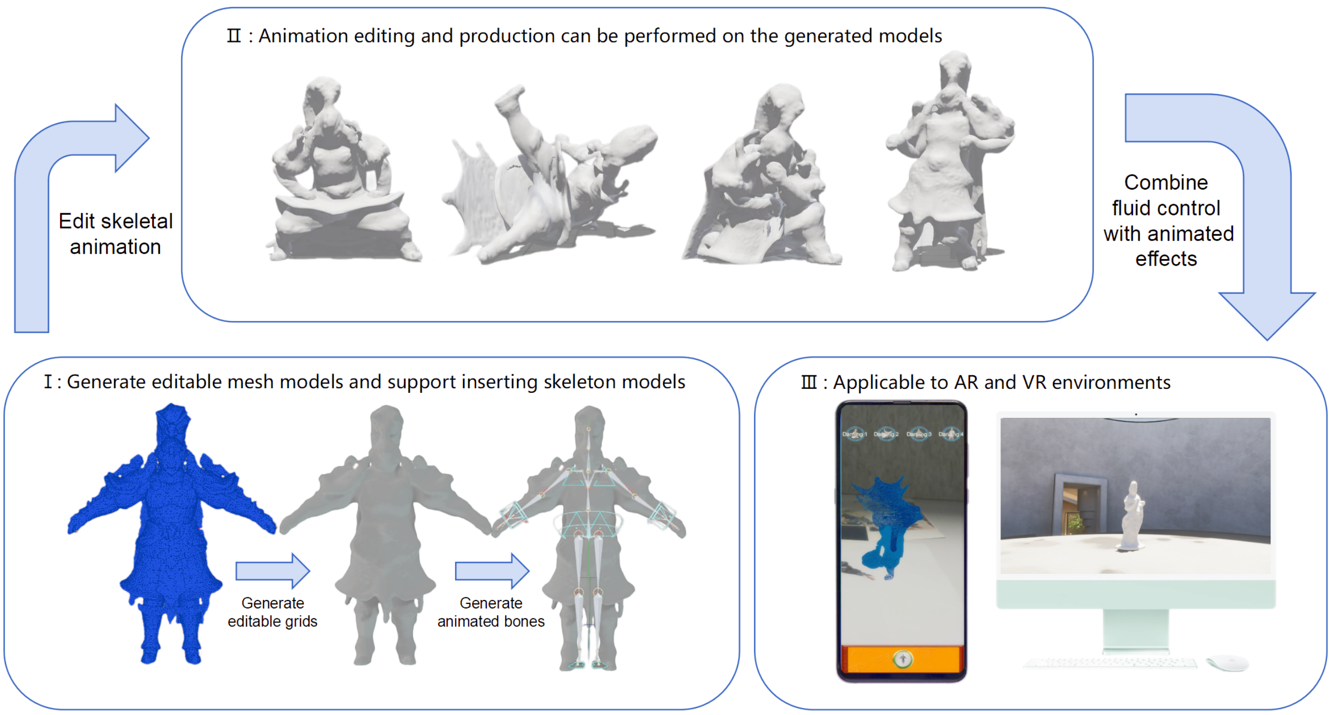

Ⅱ : Animation editing and production can be performed on the generated models
Edit skeletal animation
Combine fluid control with animated effects
Ⅰ : Generate editable mesh models and support inserting skeleton models
Generate editable grids
Generate animated bones
Ⅲ : Applicable to AR and VR environments

Fig. 4: This image demonstrates that the generated editable mesh can be recognized by skeletal plugins available in the market and automatically integrated with skeletons. Developers only need to edit skeletal animations to create their desired animations.

and subdivided it into grids, each grid having eight vertices. It was essential to determine whether a grid represented the interior or exterior of a surface, giving rise to $2^8 = 256$ possibilities. A lookup table was utilized to represent the corresponding triangular facets, with surface vertices calculated using linear interpolation. However, using the basic Marching Cubes algorithm alone could result in uneven surfaces. To address this issue, we implemented smoothing techniques on top of the algorithm.

First, let's define a triangle face as M, with vertices denoted as V and edges as E. To achieve smoother surfaces, we need to displace vertex $V^{'}$:

$$V^{'} = V + \beta \Delta V, \tag{20}$$

$$\Delta V = \frac{\sum_i \frac{W_i(V_i - V_j, H_V)}{\sum_j W_j(V_i - V_j, H_V)} V_i}{S} - V_i. \tag{21}$$

Where β represents the smoothness coefficient within the range of 0 to 1, V_i represents the current vertex, V_j represents all the edges adjacent to the current vertex, W denotes the smoothing kernel function, H_v represents the smoothing length. S represents the sum of the quantities of adjacent particles. By calculating the weighted central position of all adjacent vertices and moving all vertices towards this weighted central position, we can make the constructed mesh surfaces smoother. In order to reduce the size of the model by decreasing the number of triangles while maintaining model accuracy, we adopt a method where, in cases where a single edge has multiple vertices, we retain only the two farthest vertices. This approach ensures that our generated model, when combined with animations, runs efficiently and can be applied seamlessly in virtual environments. The generated model can be automatically recognized by plugins like Mixamo or Auto IK Rigger, which allows for the insertion of skeletal features without issues related to bone mapping. Subsequently, the skeleton can be freely edited in Unity or Blender, enabling the creation of any editable animations.

5.2 Real-time AR and VR Application

To thoroughly validate the performance advantages of our algorithm, we integrated it into AR and VR applications for comprehensive testing. Initially, we focused on reconstructing editable meshes to ensure that the models maintained a smooth and visually appealing appearance. Regularized Marching Tetrahedra [52] is an excellent algorithm to reduce the number of reconstructed meshes, achieving results that are 3/5 of those obtained from the reconstruction of marching cubes. Using our method, we achieved a significant reduction in memory usage; specifically, the memory usage of this model has been reduced to between 1/11 and 1/4 of the original when compared to other control algorithms using the Marching Cubes methods. As the number of particles increases and the model becomes more complex, the proportion of memory reduction increases. This reduction is crucial for optimizing performance in real-time applications. Following the successful reconstruction, we proceeded to combine controlled animations with the freely editable meshes. This integration allowed us to seamlessly import the animated models into the scene. In addition to the overall animations, we also imported individual frame results, which provided greater flexibility and control over the animation process. To ensure that the particle animations were both seamless and of high mass, we meticulously linked all experimental results together. This approach enabled us to create dynamic mesh animations that not only looked realistic but also performed efficiently in both AR and VR environments. The combination of reduced memory usage and enhanced animation capabilities demonstrates the effectiveness of our algorithm, providing users with a robust tool to create immersive and interactive experiences.

6 EXPERIMENTS

This paper conducted extensive experiments on multiple scenes and made comparisons with several state-of-the-art methods. Specifically, we considered the PBFC [59], RSCF [23], LCAT [44], Weighted Method [60], DreamGaussian [50] and Sugar [14]. The experimental setup for this paper was carried out on a Dell desktop equipped with an INTEL i7-9700 processor and 16GB RAM. The control process was implemented using C++. All outputs were rendered offline using the open-source software Blender and subsequently integrated into VR environments.

Table 3 presents a statistical analysis of experimental data. The rightmost column displays basic information for each scene, including

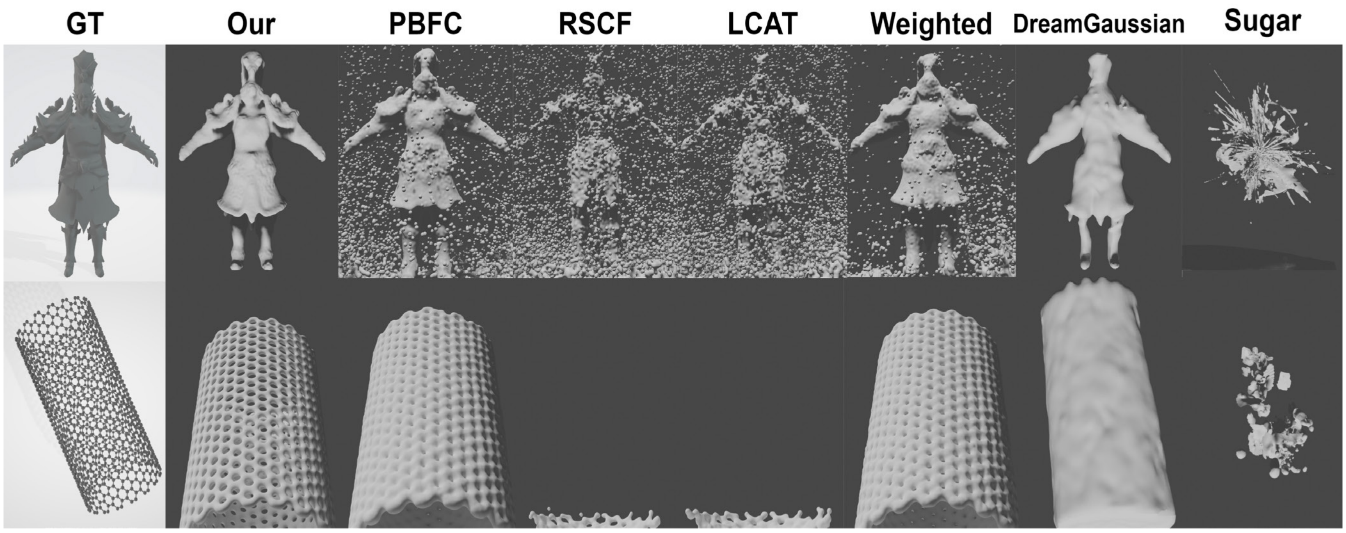
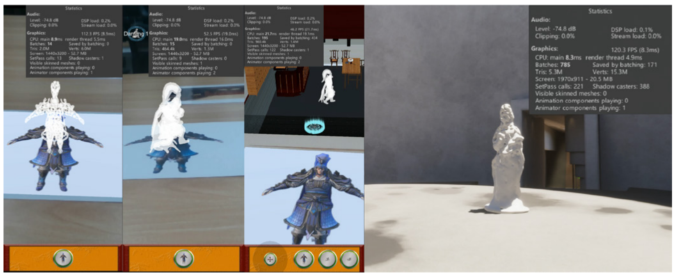


Fig. 5: The figure compares the reconstruction results of seven methods across two different scenarios.



Statistics
Audio:
Level: -74.8 dB DSP load: 0.1%
Clipping: 0.0% Stream load: 0.0%
Graphics: 120.3 FPS (8.3ms)
CPU: main 8.3ms render thread 4.9ms
Batches: 785 Saved by batching: 171
Tris: 5.3M Verts: 15.3M
Screen: 1970x911 - 20.5 MB
SetPass calls: 221 Shadow casters: 388
Visible skinned meshes: 0
Animation components playing: 0
Animator components playing: 1

Fig. 6: Performance and operating parameters in the application.

the scene name, where "Particles" represents the number of particles in the scene, and "Controls" represents the number of control particles. In the multi-target experiment, the number of control particles consists of three parts: the number for "Castle" is 69,405, for "Dog" is 5,666, and for "Bunny" is 8,384. For each scene, the experimental results of five different algorithms are compared. "Controlled Particles" refers to the number of particles influenced by the control particles. "Error Rate" indicates the rate of difference between the final formation and the original three-dimensional positions of the controlled particles. Calculate the average Euclidean distance error between the particles generated in each frame and the positions of the target sampled particles. "Per frame Time" represents the per frame time required to generate the final effect, and "Memory" represents the operational memory occupied during program execution. Figure 7 shows the rendering results of several important experiments in this paper. The building model experiment features the highest number of particles, and the stability and accuracy of our method outperform other comparison algorithms. The warrior model experiment is the most complex; the warrior model generated in this paper, combined with our reconstruction algorithm, enables skeletal mapping and editable applications. The multi-target model experiment validates the stability of our algorithm and demonstrates its capability to adaptively adjust control for any type of model without the need to reset parameter settings.

6.1 Precise Control Results

This method demonstrates more accurate control results in both large and small experimental environments compared to other methods. This can first be observed in Table 3, where the number of controlled particles by this method is closest to the number of control particles. This is due to the algorithm's precision in acquiring the density of the control particles and synchronizing control accordingly.

Table 2: Frame rate table for real-time application performance.

Frame Rate of AR Animation Generation Process	112.3(FPS)
Frame Rate of AR Editing Motion for Generated Results	52.5(FPS)
Frame Rate of VR Animation Generation Process	120.3(FPS)

6.2 Faster Generation Process

This method also shows significant efficiency gains. As observed in Table 3, the generation time is the fastest in most scenarios, except in smaller scenes (hand, dragon, dog) where it is slightly slower than RSCF and LCAT. The RSCF and LCAT methods involve directly and dynamically controlling the motion of control particle templates. These methods exhibit better computational efficiency in smaller scenarios. However, in larger scenes, with an increasing number of control particles, the computational load increases, leading to a decrease in efficiency. The speed of generation is primarily due to the first core algorithm, which utilizes spatial weighting to rapidly fill particles into boundary areas. This approach, combined with controlling particle density, addresses the issue of particle accumulation caused by spatial weighting. This ensures that the main outline of the model is generated quickly and then supplemented, greatly enhancing the efficiency of generation.

6.3 Boundary Issues And Shape Preservation

In the experiment comparison video, featuring the dragon model, it's evident that the method employs boundary correction acceleration. This technique prevents the dragon's body from exceeding the boundary due to control forces, thus avoiding body adhesion. Even if particles escape, they are immediately compressed back into the model, enhancing the robustness of the algorithm and ensuring particle stability. The comparison video of the experiment demonstrates the long-term maintenance of the shape of the model. In this test, the control duration is intentionally extended. Notably, even with prolonged control, this method effectively prevents excessive particle accumulation. This highlights the effectiveness of the approach in maintaining the integrity of the model over long periods.

6.4 Multi-target Control

The results of multi-target control generation demonstrate the superior ability of the approach in controlling particle generation of multiple models with diverse features within the same scene. Coupled with the experimental data presented in Table 3 and Figure 7, the precise control of the algorithm over each particle greatly saves time and memory resources. Unlike other methods that require parameter adjustments to

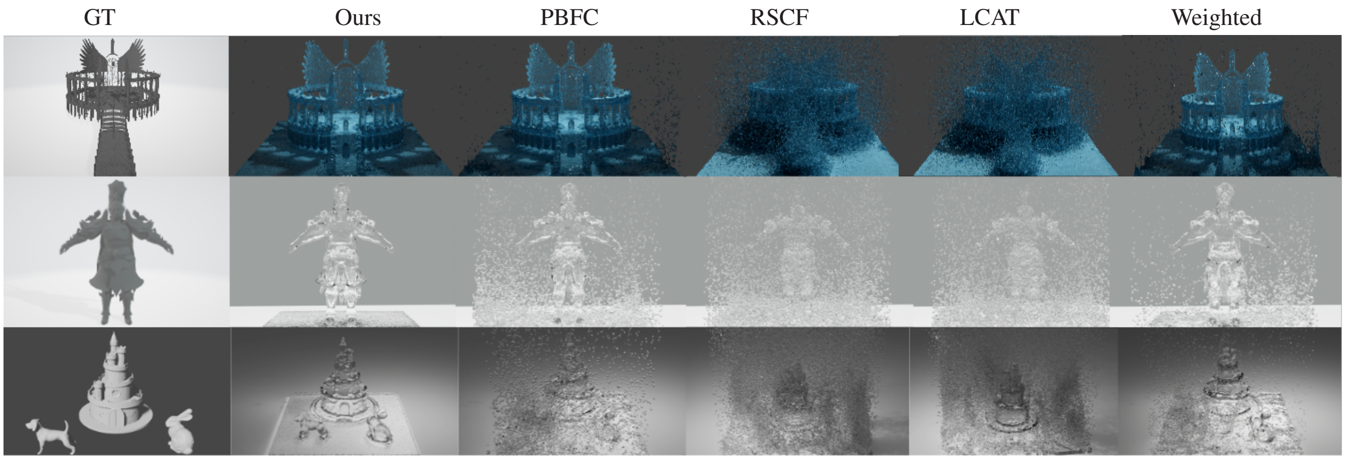


Fig. 7: A unified rendering comparison was made between the building model, warrior model, multi-target model, and the comparison method.

Table 3: The data results of all particle control generation methods.

Scene	Method	Controlled particles	Error rate(%)	Per frame Time(s)	Memory(MB)
Hand	PBFC	7,852	11.21	0.059	353
Particles	RSCF	9,252	8.41	0.061	344
46,610	LCAT	9,053	8.03	0.047	**342**
Controls	Weighted	10,655	6.73	0.066	354
9,582	Our	9,241	**5.38**	**0.037**	344
Dragon	PBFC	11,310	5.01	0.101	352
Particles	RSCF	6,879	26.23	0.083	337
46,610	LCAT	3,995	39.87	0.043	336
Controls	Weighted	12,154	6.14	0.134	351
8,697	Our	8,050	**2.88**	**0.024**	**334**
Dog	PBFC	34,433	11.08	0.112	371
Particles	RSCF	7,522	59.16	failure	344
46,610	LCAT	10,197	42.32	failure	**343**
Controls	Weighted	33,843	11.09	0.156	366
12,038	Our	14,639	**7.60**	**0.043**	355
Vase	PBFC	102,311	22.59	1.212	1,024
Particles	RSCF	18,086	failure	failure	failure
252,810	LCAT	17,599	failure	failure	failure
Controls	Weighted	87,051	17.71	1.231	924
33,001	Our	34,321	**5.94**	**0.433**	**851**
Building	PBFC	308,761	19.23	1.974	5,120
Particles	RSCF	238,917	52.61	2.733	3,891
610,731	LCAT	215,582	53.50	2.199	3,789
Controls	Weighted	276,610	19.33	1.598	3,686
189,612	Our	222,761	**7.08**	**1.296**	**2,970**
Carbontube	PBFC	61,879	9.93	0.540	2,150
Particles	RSCF	1,933	failure	failure	failure
113,288	LCAT	1,954	failure	failure	failure
Controls	Weighted	52,328	9.41	0.479	1,638
21,912	Our	39,678	**4.12**	**0.157**	**826**
Warrior	PBFC	46,869	19.95	0.646	1,998
Particles	RSCF	39,785	16.61	1.644	1,442
113,288	LCAT	42,774	18.50	1.749	1,376
Controls	Weighted	49,675	16.13	0.886	1,776
42,741	Our	35,414	**7.18**	**0.342**	**1,199**
Multi-target	PBFC	73,681	24.39	2.916	9,830
Particles	RSCF	123,969	failure	2.512	5,529
151,686	LCAT	119,643	failure	1.367	5,324
Controls	Weighted	76,123	19.39	1.771	8,806
83,455	Our	34,321	**4.25**	**0.739**	**4,505**

tailor control forces to match model features, this algorithm operates without any need for such adjustments. Instead, it dynamically allocates the force exerted on each particle based on the model's sampling features. This capability enables us to precisely control the generation of multiple target shapes within a single scene.

6.5 Comparison of 3DGS Mesh Reconstruction

Figure 5 compares the latest 3DGS mesh generation methods. From the experimental results, it is observed that due to limited perspectives, DreamGaussian struggles with fully segmenting cavity models in image segmentation, making it challenging to model cavities and difficult to supplement missing information in diffuse models. The results generated by Sugar are comparatively poor, possibly because the training images have excessively uniform colors, resulting in incomplete model generation. However, the 3DGS point cloud models trained on these images pose no issues. Nevertheless, the limitations of 3DGS prevent it from effectively removing background noise, rendering the generated models unsuitable for direct editing.

6.6 Application Performance Testing

In Figure 6 and Table 2, we show how our results are applied in a virtual environment. The animation generation process and subsequent model movement both maintain a satisfactory frame rate (FPS). This performance is important for ensuring a smooth user experience. There is also potential for further development of the application. Currently, we have not optimized it, and with appropriate optimizations in specific scenarios, we could improve operational efficiency. By refining aspects such as rendering and animation handling, we could enhance performance without compromising functionality. Overall, these results indicate the effectiveness of the algorithm while suggesting areas for future improvement.

7 LIMITATIONS

Currently, the algorithm has certain limitations. We are unable to perform more detailed region segmentation for objects. While we can model elements such as armor and capes, we cannot control them individually during animation or simulate them separately. Additionally, due to the relatively low number of particles used in our particle control, some precision is lost during reconstruction. However, we are supplementing our experiments with high-resolution data, and initial results have been promising. This indicates that our algorithm is capable of handling particle control on the order of tens of millions.

8 CONCLUSIONS

This paper utilizes particle control techniques based on the spatial features of target models to achieve higher-precision editable mesh generation without the need for parameter adjustments. It can simultaneously generate multiple targets. This method dynamically adjusts control strength based on the spatial characteristics of the targets, excelling in capturing intricate details of complex models. In multi-target scenarios, it ensures rapid control of high-fidelity shapes while preventing particle clustering. It integrates editing and control animations into virtual environments.

9 ACKNOWLEDGEMENT

This research is supported by: National Science and Technology Major Project of the Ministry of Science and Technology of China (2024ZD0608100), National Natural Science Foundation of China (Nos.62376025, 62332017), Guangdong Basic and Applied Basic Research Foundation (No.2023A1515030177). The authors wish to thank anonymous reviewers and Miss Xiaofan Ling for their valuable suggestions.

References

[1] N. Bell, Y. Yu, and P. J. Mucha. Particle-based simulation of granular materials. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 77–86, 2005. 3

[2] J. Bender and D. Koschier. Divergence-free smoothed particle hydrodynamics. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics symposium on computer animation*, pp. 147–155, 2015. 3

[3] M. Bergou, S. Mathur, M. Wardetzky, and E. Grinspun. Tracks: toward directable thin shells. *ACM Transactions on Graphics (TOG)*, 26(3):50–es, 2007. 2

[4] M. Bojsen-Hansen and C. Wojtan. Generalized non-reflecting boundaries for fluid re-simulation. *ACM Transactions on Graphics (TOG)*, 35(4):1–7, 2016. 2

[5] L. L. Bozgeyikli and E. Bozgeyikli. Tangiball: foot-enabled embodied tangible interaction with a ball in virtual reality. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 812–820. IEEE, 2022. 2

[6] S. L. Brunton, B. R. Noack, and P. Koumoutsakos. Machine learning for fluid mechanics. *Annual review of fluid mechanics*, 52:477–508, 2020. 2

[7] M. Chu, N. Thuerey, H.-P. Seidel, C. Theobalt, and R. Zayer. Learning meaningful controls for fluids. *ACM Transactions on Graphics (TOG)*, 40(4):1–13, 2021. 2

[8] J. Cornelis, M. Ihmsen, A. Peer, and M. Teschner. Liquid boundaries for implicit incompressible sph. *Computers & Graphics*, 52:72–78, 2015. 2

[9] O. Dionne and M. de Lasa. Geodesic voxel binding for production character meshes. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 173–180, 2013. 3

[10] R. Fattal and D. Lischinski. Target-driven smoke animation. In *ACM SIGGRAPH 2004 Papers*, pp. 441–448. 2004. 2

[11] Z. Forootaninia and R. Narain. Frequency-domain smoke guiding. *ACM Transactions on Graphics (TOG)*, 39(6):1–10, 2020. 2

[12] N. Foster and D. Metaxas. Controlling fluid animation. In *Proceedings computer graphics international*, pp. 178–188. IEEE, 1997. 2

[13] C. Gissler, S. Band, A. Peer, M. Ihmsen, and M. Teschner. Generalized drag force for particle-based simulations. *Computers & Graphics*, 69:1–11, 2017. 5

[14] A. Guédon and V. Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5354–5363, 2024. 1, 2, 7

[15] J.-m. Hong and C.-h. Kim. Controlling fluid animation with geometric potential. *Computer Animation and Virtual Worlds*, 15(3-4):147–157, 2004. 2

[16] T. Inglis, M.-L. Eckert, J. Gregson, and N. Thuerey. Primal-dual optimization for fluids. In *Computer Graphics Forum*, vol. 36, pp. 354–368. Wiley Online Library, 2017. 2

[17] M. Jiang, Y. Zhou, R. Wang, R. Southern, and J. J. Zhang. Blue noise sampling using an sph-based method. *ACM Transactions on Graphics (TOG)*, 34(6):1–11, 2015. 2

[18] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. 1

[19] B. Kim, V. C. Azevedo, N. Thuerey, T. Kim, M. Gross, and B. Solenthaler. Deep fluids: A generative network for parameterized fluid simulations. In *Computer graphics forum*, vol. 38, pp. 59–70. Wiley Online Library, 2019. 2

[20] T. Kugelstadt, J. Bender, J. A. Fernández-Fernández, S. R. Jeske, F. Löschner, and A. Longva. Fast corotated elastic sph solids with implicit zero-energy mode control. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 4(3):1–21, 2021. 3

[21] Y. Li, J. Wu, R. Tedrake, J. B. Tenenbaum, and A. Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. *arXiv preprint arXiv:1810.01566*, 2018. 2

[22] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*, pp. 347–353. 1998. 6

[23] J.-M. Lu, X.-S. Chen, X. Yan, C.-F. Li, M. Lin, and S.-M. Hu. A rigging-skinning scheme to control fluid simulation. In *Computer Graphics Forum*, vol. 38, pp. 501–512. Wiley Online Library, 2019. 1, 2, 7

[24] M. Macklin and M. Müller. Position based fluids. *ACM Transactions on Graphics (TOG)*, 32(4):1–12, 2013. 3

[25] P.-L. Manteaux, U. Vimont, C. Wojtan, D. Rohmer, and M.-P. Cani. Space-time sculpting of liquid animation. In *Proceedings of the 9th International Conference on Motion in Games*, pp. 61–71, 2016. 2

[26] A. McNamara, A. Treuille, Z. Popović, and J. Stam. Fluid control using the adjoint method. *ACM Transactions On Graphics (TOG)*, 23(3):449–456, 2004. 2

[27] V. Mihalef, D. Metaxas, and M. Sussman. Animation and control of breaking waves. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 315–324, 2004. 2

[28] J. J. Monaghan. Smoothed particle hydrodynamics. *Annual review of astronomy and astrophysics*, 30(1):543–574, 1992. 3

[29] J. Morton, A. Jameson, M. J. Kochenderfer, and F. Witherden. Deep dynamical modeling and control of unsteady fluid flows. *Advances in Neural Information Processing Systems*, 31, 2018. 2

[30] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 2007. 3

[31] M. B. Nielsen and R. Bridson. Guide shapes for high resolution naturalistic liquid simulation. In *ACM SIGGRAPH 2011 papers*, pp. 1–8. 2011. 2

[32] M. B. Nielsen and B. B. Christensen. Improved variational guiding of smoke animations. In *Computer Graphics Forum*, vol. 29, pp. 705–712. Wiley Online Library, 2010. 2

[33] M. B. Nielsen, B. B. Christensen, N. B. Zafar, D. Roble, and K. Museth. Guiding of smoke animations through variational coupling of simulations at different resolutions. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 217–226, 2009. 2

[34] K. Ogawa, K. Fujita, K. Takashima, and Y. Kitamura. Pseudojumpon: Jumping onto steps in virtual reality. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 635–643. IEEE, 2022. 2

[35] Z. Pan, J. Huang, Y. Tong, C. Zheng, and H. Bao. Interactive localized liquid motion editing. *ACM Transactions on Graphics (TOG)*, 32(6):1–10, 2013. 2

[36] Z. Pan and D. Manocha. Editing smoke animation using a deforming grid. *Computational Visual Media*, 3:369–378, 2017. 2

[37] Z. Pan and D. Manocha. Efficient solver for spacetime control of smoke. *ACM Transactions on Graphics (TOG)*, 36(4):1, 2017. 2

[38] N. Rasmussen, D. Enright, D. Nguyen, S. Marino, N. Sumner, W. Geiger, S. Hoon, and R. Fedkiw. Directable photorealistic liquids. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 193–202, 2004. 2

[39] K. Raveendran, N. Thuerey, C. J. Wojtan, and G. Turk. Controlling liquids using meshes. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2012. 2

[40] B. Ren, X. Ye, Z. Pan, and T. Zhang. Versatile control of fluid-directed solid objects using multi-task reinforcement learning. *ACM Transactions on Graphics*, 42(2):1–14, 2022. 2

[41] S. Sato, Y. Dobashi, and T. Kim. Stream-guided smoke simulations. *ACM Transactions on Graphics (TOG)*, 40(4):1–7, 2021. 2

[42] S. Sato, Y. Dobashi, and T. Nishita. Editing fluid animation using flow interpolation. *ACM Transactions on Graphics (TOG)*, 37(5):1–12, 2018. 2

[43] S. Sato, Y. Dobashi, Y. Yue, K. Iwasaki, and T. Nishita. Incompressibility-preserving deformation for fluid flows using vector potentials. *The Visual Computer*, 31:959–965, 2015. 2

[44] A. Schoentgen, P. Poulin, E. Darles, and P. Meseure. Particle-based liquid control using animation templates. In *Computer Graphics Forum*, vol. 39, pp. 79–88. Wiley Online Library, 2020. 1, 2, 7

[45] F. S. Sevinc Eroglu, A. Chevalier, D. Roettger, D. Zielasko, T. W. Kuhlen, and B. Weyers. Design and evaluation of a free-hand vr-based authoring environment for automated vehicle testing. in 2021 ieee virtual reality and 3d user interfaces (vr). 1–10, 2021. 2

[46] L. Shi and Y. Yu. Controllable smoke animation with guiding objects. *ACM Transactions on Graphics (TOG)*, 24(1):140–164, 2005. 2

[47] A. Stomakhin and A. Selle. Fluxed animated boundary method. *ACM Transactions on Graphics (TOG)*, 36(4):1–8, 2017. 2

[48] T. Stuyck and P. Dutré. Sculpting fluids: A new and intuitive approach to art-directable fluids. In *ACM SIGGRAPH 2016 Posters*, pp. 1–2. 2016. 2

[49] H. Tang, J. Rabault, A. Kuhnle, Y. Wang, and T. Wang. Robust active flow control over a range of reynolds numbers using an artificial neural network trained through deep reinforcement learning. *Physics of Fluids*, 32(5), 2020. 2

[50] J. Tang, J. Ren, H. Zhou, Z. Liu, and G. Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653*, 2023. 1, 2, 7

[51] N. Thuerey. Interpolations of smoke and liquid simulations. *ACM Transactions on Graphics (TOG)*, 36(1):1–16, 2016. 2

[52] G. M. Treece, R. W. Prager, and A. H. Gee. Regularised marching tetrahedra: improved iso-surface extraction. *Computers & Graphics*, 23(4):583–598, 1999. 7

[53] A. Treuille, A. McNamara, Z. Popović, and J. Stam. Keyframe control of smoke simulations. In *ACM SIGGRAPH 2003 Papers*, pp. 716–723. 2003. 2

[54] B. Ummenhofer, L. Prantl, N. Thuerey, and V. Koltun. Lagrangian fluid simulation with continuous convolutions. In *International Conference on Learning Representations*, 2019. 2

[55] J. Waczyńska, P. Borycki, S. Tadeja, J. Tabor, and P. Spurek. Games: Mesh-based adapting and modification of gaussian splatting. *arXiv preprint arXiv:2402.01459*, 2024. 2

[56] A. P. Witkin and P. S. Heckbert. Using particles to sample and control implicit surfaces. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pp. 269–277, 1994. 3

[57] T. Xie, Z. Zong, Y. Qiu, X. Li, Y. Feng, Y. Yang, and C. Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4389–4398, 2024. 2

[58] D. Zhang, C.-M. Pun, Y. Yang, H. Gao, and F. Xu. A rate-based drone control with adaptive origin update in telexistence. In *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*, pp. 807–816. IEEE, 2021. 2

[59] S. Zhang, X. Yang, Z. Wu, and H. Liu. Position-based fluid control. In *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games*, pp. 61–68, 2015. 1, 2, 7

[60] X. Zhou, S. Liu, H. Zeng, X. Wang, and X. Ban. Efficient and high precision target-driven fluid simulation based on spatial geometry features. *Computer Animation and Virtual Worlds*, 35(1):e2202, 2024. 1, 7

[61] F. Zorilla, M. Ritter, J. Sappl, W. Rauch, and M. Harders. Accelerating surface tension calculation in sph via particle classification and monte carlo integration. *Computers*, 9(2):23, 2020. 5

Xiangyang Zhou is a postgraduate student in Computer Science and Technology at the University of Science and Technology Beijing. He received an undergraduate diploma from Inner Mongolia University of Science and Technology in 2022. His research field is computer graphics, virtual reality, and human-computer interaction. Especially particle-based simulation.

Yanrui Xu is a PhD student at the School of Intelligence Science and Technology, University of Science and Technology Beijing, and the Bernoulli Institute, University of Groningen. He earned his Master's degree from the University of Science and Technology Beijing in 2020. His research interests include physics-based fluid simulation.

Chao Yao received an M.E. degree and Ph. D degree from Beijing Jiaotong University (BJTU) in 2010 and 2016. From 2014 to 2015, he was a Visiting Ph. D student with LTS4 Group, Institute of the Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland. He is currently an associated Professor at the University of Science and Technology Beijing (USTB). His research interests include image/video compression, computer vision, and human-computer interaction.

Xiaokun Wang is currently an associated Professor at the University of Science and Technology Beijing (USTB). He received a Ph.D. degree in Computer Science and Technology from the University of Science and Technology Beijing, in 2017. His research interests include computer graphics, virtual reality, and human-computer interaction.

Xiaojuan Ban is a professor in Artificial Intelligence and leader of the Computer Animation Group at University of Science and Technology Beijing, China. She received her Ph.D. degree in control theory and control engineering from the University of Science and Technology Beijing. Her research interests include computer graphics, artificial intelligence, human-computer interaction, big data analysis and 3D visualization.