PhyVR: Physics-based Multi-material and Free-hand Interaction in VR

Hanchen Deng[1], Jin Li[*1], Yang Gao[†1,2], Xiaohui Liang[1,3], Hongyu Wu[1], and Aimin Hao[1,2,4]

[1]State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing, China
[2]Research Unit of Virtual Body and Virtual Surgery Technologies (2019RU004), Chinese Academy of Medical Sciences, Beijing, China
[3]Zhongguancun Laboratory, Beijing, China
[4]Pengcheng Laboratory, Shenzhen, China.

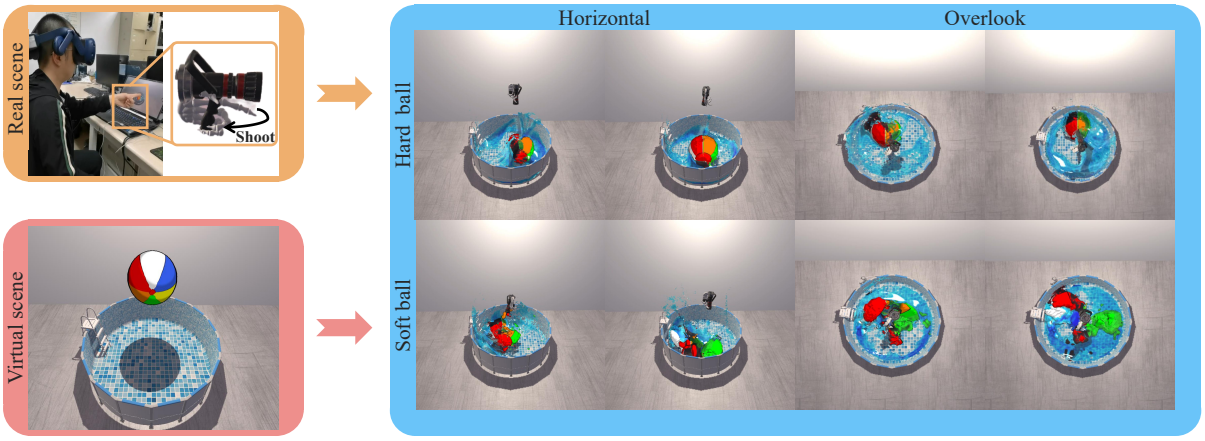

Figure 1: Examples of interactive demonstrations. The top row shows the process of controlling a water firehose to shoot a hard elastic sphere, while the bottom row shows the process of controlling the water firehose to shoot a soft elastic sphere.

Abstract

The realistic interaction with physical phenomena is a crucial aspect of human-computer interaction (HCI) in virtual reality (VR). However, the real-time performance of physical simulation, interactive computation, and rendering is the bottleneck of physics-based VR HCI. To address these challenges, we propose a novel physics-oriented framework for multi-material objects and free-hand interaction, termed PhyVR. This framework enables users to interact with diverse virtual phenomena dynamically. At the algorithm level, we develop a unified particle system to describe both the virtual multi-materials and the user's avatar for the efficiency issue, optimize collision detection, and accelerate the HCI algorithms with a variable fine-coarse particle sampling scheme. At the rendering level, we introduce a hybrid particle-grid anisotropic algorithm for surface reconstruction, enabling real-time and visually convincing fluid rendering. Comprehensive experiments and user studies demonstrate that our framework effectively captures various physical interaction phenomena, providing an enhanced user experience and paving the way for expanding VR-related HCI applications.

Index Terms: Human-centered computing—Human computer interaction (HCI)—Interaction devices—; Computing methodologies—Computer graphics—Animation—Physical simulation

1 Introduction

Achieving realistic and natural physical phenomenon simulations in virtual environments is of paramount importance for enhancing the immersive VR HCI experience. However, achieving vivid and efficient interaction with physical phenomena in VR still presents several challenges. First, solving complex partial differential equations representing physical motions, such as fluids and flexible bodies, demands significant time. Second, achieving realistic real-time rendering of physical phenomena is crucial yet challenging. This process involves managing varying levels of transparency and deformation while maintaining optimal performance. Furthermore, real-time responses between actual user actions and virtual phenomena require algorithms that are both highly robust and efficient. Achieving synchronization between virtual and real-world physics calls for precise coordination and synchronization mechanisms. Addressing these challenges is essential for developing VR experiences that are both engaging and authentic.

To addres the challenges mentioned above, we propose a novel real-time interactive framework, *PhyVR*, for seamless interaction between the detected gestures and multi-material objects in virtual scenes, which provides users with an immersive and realistic VR experience. This framework facilitates real-time user interaction with various virtual materials, including fluid, elastic, plastic, and solid objects, through the tracking of actual hand movements. By using PhyVR, we aim to maintain a high sense of presence and interaction performance, even in complex scenes, to achieve a more diverse and immersive HCI experience in VR applications, thereby significantly broadening the scope of applications for the fusion of VR and physical simulations. In essence, there are three innovative contributions:

- We develop a fast computation scheme that involves both physical materials and the user's avatar behaviors within a unified particle system, thus, all the virtual-real fusion interactions can be tackled through physical simulation using a numerical

*Co-first authors
†Corresponding author:gaoyangvr@buaa.edu.cn

solver.

- We employ an adaptive particle sampling strategy for the avatar hand model to optimize collision detection. Precise interaction regions utilize fine particles to enhance interaction accuracy, while other regions use coarse particles to address efficiency concerns.

- We propose a novel hybrid particle-grid anisotropic implicit fluid surface reconstruction scheme that achieves a balance between efficiency and mitigating fluid volume compression associated with particle-based methods, thereby advancing real-time and visually realistic fluid rendering.

2 Related Work

2.1 Physics Simulation

Physics simulation plays a pivotal role in enabling realistic HCI in VR. In the field of physical phenomenon modeling, three primary simulation perspectives are utilized: Lagrangian, Eulerian, and hybrid Lagrangian-Eulerian. The Lagrangian perspective focuses on particle modeling, with the Smoothed Particle Hydrodynamics (SPH) algorithm [5, 9, 18] being a widely adopted technique for fluid simulation. The Eulerian perspective divides space into a grid structure, primarily concentrating on the grids [4]. As a hybrid Lagrangian-Eulerian technique, the Material Point Method (MPM) algorithm was initially proposed by Sulsky et al. [26]. Expanding upon the MPM paradigm, Hu et al. [14] proposed the Moving Least Squares Material Point Method (MLS-MPM). This advancement enabled the simulation of hitherto unsupported phenomena within conventional MPM, including material cutting, dynamic open boundaries, and intricate two-way coupling with rigid bodies. Beyond introducing these new capabilities, the MLS-MPM approach significantly enhances simulation efficiency by harnessing the power of GPU parallel computing and employing an innovative stress divergence discretization technique.

Leveraging these advancements, we extend the MLS-MPM algorithm to seamlessly integrate with the Unity3D game engine, thereby facilitating real-time multi-material modeling within the context of VR environments. This integration not only broadens the scope of possibilities for interactive experiences but also emphasizes the accelerated computational speeds intrinsic to the MPM approach.

2.2 Real-time Simulation of Physical Phenomena

Real-time simulation of physical phenomena in VR is vital and necessary for enhancing the realism and immersion of user experience. In the field of fluid simulation, Eroglu et al. [8] developed a real-time tool for creating 3D fluid artwork in VR. This tool enables artists to draw fluid-like sketches in 3D space and manipulate them using six degrees of freedom input devices. Turning to the domain of soft body simulation, Pan et al. [22] presented a VR medical simulator for cholecystectomy, allowing users to perform real-time surgical operations with high stability and fidelity. Expanding the horizon to encompass multi-material simulation, Xiao et al. [28] proposed an immersive VR simulator for neonatal endotracheal intubation, enabling flexible multi-modal medical simulation scenarios. Notwithstanding its advancements, this simulator encounters challenges in fluid rendering quality due to time limitations.

The cornerstone of fluid rendering lies in surface reconstruction. In Eulerian perspective, fluid surface reconstruction is typically based on mesh, such as level-set methods [21], particle level-set methods [6] [7], semi-Lagrangian contouring [25], volume-of-fluid methods [12] and explicit surface tracking [2]. Regrettably, these algorithms often exhibit inefficiencies and waste as they reconstruct the entirety of the surface, even including the obscured backside of the fluid. From a Lagrangian simulation perspective, screen space surface reconstruction is a popular method. Müller et al. [19] directly built the surface mesh from the depth map smoothed by fluid particles, eliminating the need for level-set construction. While this approach was groundbreaking, it still adhered to the explicit mesh reconstruction idea. Laan et al. [27] completely abandoned explicit mesh reconstruction and instead directly constructed the normal to the fluid surface based on the depth map. This algorithm achieved excellent results in terms of efficiency and quality. Yu et al. [30] implemented anisotropic surface reconstruction in the SPH algorithm using anisotropic kernels. They calculated the covariance matrix of the mean particle density using principal component analysis to extract particle deformations and rotations. The scheme substantially improves the quality of fluid surface reconstruction.

2.3 Free-hand Interaction in VR

In recent years, substantial research has focused on free-hand interaction in VR. This approach enables users to interact with virtual objects naturally and intuitively, enhancing the user experience and making VR applications more accessible and engaging. With advancements in HCI devices within virtual reality, the field of 3D HCI has gained significant momentum. The most common and natural method is free-hand gesture tracking by kinematic skeleton tracking sensors [11]. Leveraging gesture recognition devices capabilities, e.g. *Leap motion*, researchers have explored gesture and object interactions as key areas of investigation. These interaction methods can be broadly categorized into two distinct approaches: pre-training methods and real-time computation methods.

In pre-training methods, a common approach is to capture and record key gestures, such as grasping, pressing, touching, clicking, and others, in advance. These pre-recorded gestures are then extracted from the pre-trained data when similar situations are detected during real-time interaction [3] [29]. Additionally, certain machine learning techniques have been employed to process and recognize these pre-trained gestures [17] [1]. These studies primarily emphasize gesture recognition, yet their effectiveness in interactions remains limited. In contrast to physics-based methods, they exhibit shortcomings in both interactive effects and the representation of complex physical phenomena. While these interaction methods can perform well in specific scenarios, they do have some limitations in terms of convenience and adaptability.

Physical-based methods are known to have certain drawbacks in terms of efficiency. The approach proposed by Höll et al. [13] utilizes contact points sampled on the hand model's mesh to determine contact occurrence and implements friction using a friction model based on Coulomb's law. A more advanced scheme involves treating the hand as a soft object and considering the finger muscles [10, 16], incorporating contact area and friction into the calculations. Alvaro et al. [23] proposed an algorithm for simulating nonlinear elasticity under frictional contact. Nasim and Kim et al. [20] introduced an intriguing approach where two virtual hand models are generated simultaneously for each hand. When a grasp occurs, one of the virtual hands is frozen, and the force is calculated based on the distance between the two virtual hand models. However, this scheme is limited to handling grip interactions and may not be suitable for other types of interactions.

3 HCI with Physical Phenomena

An overview of our proposed architecture is illustrated in Fig. 2. It comprises several significant components, including an object representation structure (unified particle expression), a physics-based HCI, and a real-time rendering procedure. The primary objective of this architecture is to enable natural and intuitive interaction with virtual objects by seamlessly integrating physics-based behaviors with the user's actual hand movements. The subsequent sections will delve into the specifics of the interaction algorithm, providing a comprehensive understanding of our approach and capabilities.

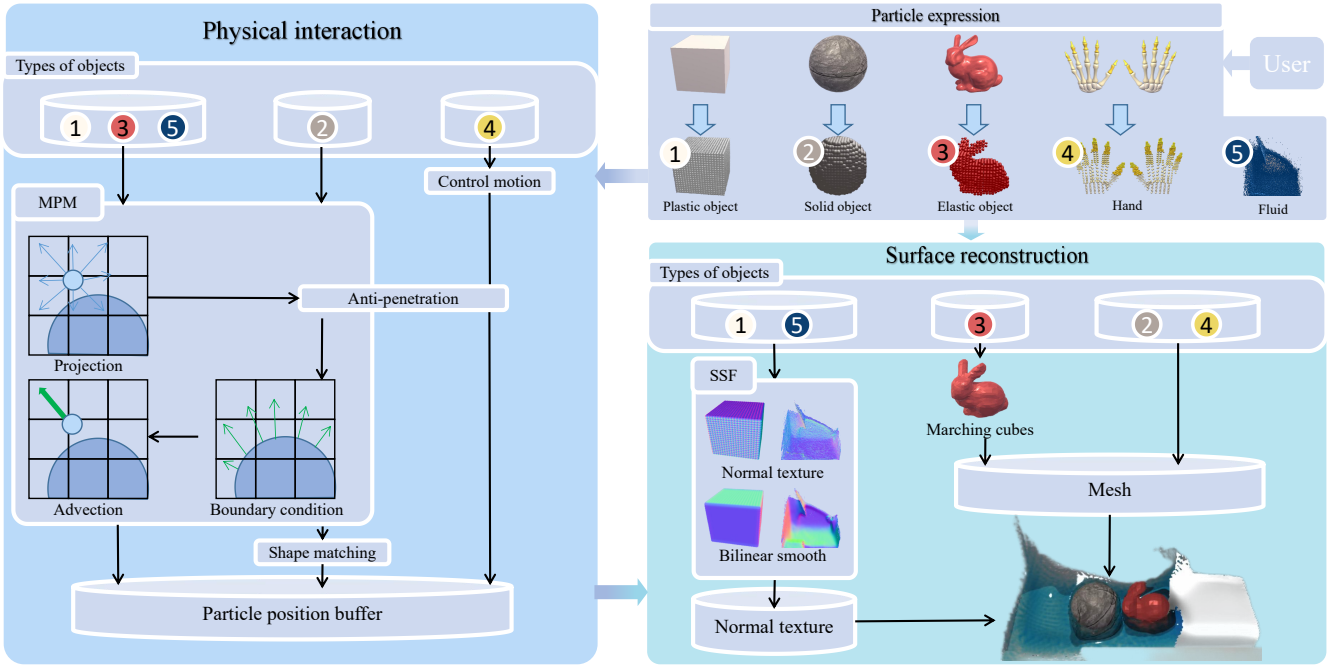

Figure 2: System architecture showing interaction flow. The voxelized objects and hand models are integrated into a unified particle system based on MPM algorithms for physics simulation and human-computer interaction, and different real-time surface reconstruction and rendering schemes are used for different objects.

Table 1: A table with variable definitions.

Symbol	Definition
m	The mass of cell or particle
$\mathbf{V}^0$	The initial volume of the hand particle
w	The B-spline weight of particle at cell
J	The volume expansion scalar
ρ	The density of hand particles in cell
$\mathbf{v}$	The velocity of cell or particle
$\mathbf{x}$	The position of cell or particle
$\mathbf{C}$	The affine matrix of particle
$\mathbf{E}$	The elasticity term of particle
$\mathbf{F}$	The deformation gradient of particle
$\mathbf{P}$	The first Piola-Kirchhoff (PK1) stress tensor of particle
$\mathbf{G}$	The anisotropic matrix of the particle
$\mathbf{A}^n$	The superscript represents the time step
$\mathbf{A}_p$	The subscript (p) represents the particle
$\mathbf{A}_i$	The subscript (i) represents the cell

3.1 Unified Particle System

To address the efficiency issue, we propose a unified particle system that serves as a versatile representation for virtual multi-materials and the user's avatar. This system is designed to optimize collision detection and accelerate HCI algorithms. By utilizing the unified particle system, we can efficiently handle interactions between different materials and the user's avatar, resulting in improved performance and responsiveness. This approach allows for seamless and efficient simulations, enhancing the overall user experience in virtual environments. Fig. 1 showcases the coupling of fluid and an elastic ball, with the user controlling a water firehose to shoot the fluid. The top row displays a hard elastic ball, where Young's modulus (material stiffness) E is 300 and Poisson's ration (ability to preserve volume under deformation) ν is 0.3. The bottom row presents a soft elastic ball, where E is 100 and ν is 0.1. The water's velocity is so fast that upon impact with the elastic ball, the ball fragments into multiple pieces, akin to being sliced by a knife. Among most existing techniques, the detected hand models are mesh-based. To address the MPM-based multi-materials and the hands' avatar uniformly, we sample the mesh hand models into unified particle models. Compared with the mesh hand model, this solution naturally offers the advantage of being more suitable for detecting contact points and expanding the contact surface. Furthermore, it can incorporate all the behaviors into a unified physical solver to address efficiency issues.

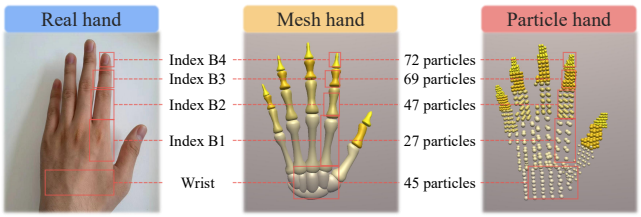

Figure 3: Variable fine-course voxelization strategies for different phalanges.

In terms of the effect on physical interaction, higher particle densities resulting from voxelizing the skeleton lead to better force application to virtual objects and a reduced likelihood of hand model penetration. In terms of system efficiency, increasing the particle density leads to a decrease in overall efficiency. With regard to the positioning of the phalanxes, our objective is to enable more sensitive and detailed interaction at the fingertips. To accomplish this, we incrementally increase the particle density after voxelization, starting from the bones further away from the fingertips and progressing towards the bones closer to the fingertips for each finger. This approach ensures a greater concentration of particles in the regions that require enhanced precision during interaction.

To this end, as shown in Fig. 3, The skeletal hand model consists of a total of 20 bones: four bones for each of the index finger, middle finger, ring finger, and little finger, three bones for the thumb, and a single bone for the wrist. To facilitate further processing, we voxelize these individual bones separately. We introduce a variable fine-coarse particle sampling scheme that effectively balances the level of detail and computational resources required for accurate simulations

and interactions. This approach enhances the overall efficiency and responsiveness of the system, enabling seamless and immersive experiences in virtual environments. The choice of voxelization densities for different phalanges directly impacts the performance of physical interactions and the system's operational efficiency.

3.2 MLS-MPM Implement

Particle projection. We implement the coupling and simulation of multi-material objects based on the MLS-MPM approach [14]. First, we project the particles' mass and velocity onto adjacent cells, and the projection equations are presented as follows:

$$m_i^n = \sum_p w_{i,p} m_p, \tag{1}$$

$$(m\mathbf{v})_i^n = \sum_p w_{i,p} \left\{ m_p \mathbf{v}_p + \left[m_p \mathbf{C}_p^n - \mathbf{E}_p^n \right] \left(\mathbf{x}_i - \mathbf{x}_p^n \right) \right\}, \tag{2}$$

where the variables used in the algorithm are shown in Table 1. The elastic term $\mathbf{E}$ is calculated as follows:

$$\mathbf{E}_p^n = -\frac{4\Delta t}{\Delta x^2} \sum_p \mathbf{V}^0 \mathbf{P}_p^n \left(\mathbf{F}_p^n \right)^T, \tag{3}$$

$$\mathbf{P}_p^n = 2\mu \left(\mathbf{F}_p^n - \mathbf{UW} \right) + \lambda \left(J_p^n - 1 \right) J_p^n \left(\mathbf{F}_p^n \right)^{-T}, \tag{4}$$

$$\mathbf{F}_p^{n+1} = \left(\mathbf{I} + \Delta t \mathbf{C}_p^n \right) \mathbf{F}_p^n, \tag{5}$$

where μ and λ are the Lamé constants, μ denotes the shear variables of the material and λ denotes the compressibility, $\mathbf{W}$ and $\mathbf{U}$ are obtained through the singular value decomposition $\mathbf{F} = \mathbf{U}\sum\mathbf{W}$.

Particle advection. During the particle advection, each particle updates its velocity $\mathbf{v}$ and affine transformation matrix $\mathbf{C}$ based on neighboring cell velocities, subsequently altering its position. The advection equations are as follows:

$$\mathbf{v}_p^{n+1} = \sum_p w_{i,p} \mathbf{v}_i^{n+1}, \tag{6}$$

$$\mathbf{C}_p^{n+1} = \frac{4}{\Delta x^2} \sum_p w_{i,p} \mathbf{v}_i^{n+1} \left(\mathbf{x}_i - \mathbf{x}_p^n \right)^{\mathbf{T}}, \tag{7}$$

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \Delta t \mathbf{v}_i^{n+1}. \tag{8}$$

3.3 VR Interaction

Real hand control. We bind the particles of each bone to the mesh of the bone and follow the bone to move together, including the translation and rotation of the bone. We treat hand particles as elastic material. By discretizing the hand particles with elastic finite elements, we project the hand particles onto the grid, which affect the virtual particles and realize the physical interaction.

Penetration prevention from virtual objects. During the physics simulation, it is beneficial to update the hand particles' velocities rather than their positions directly. There are reasons: First, updating velocities allows for better stability; Second, velocity-based updates lead to smoother transitions between frames; Finally, velocity-based updates make it easier to handle collisions with other objects or boundaries.

This approach helps prevent particle penetration and maintain the stability of the system. However, since the hand particles' motion is entirely user-controlled and not a result of physical simulation, virtual particles may still penetrate the hand model when they are driven to move too fast in a single MPM time step. To address this issue, an anti-penetration method for the particle hand model can be implemented.

Initially, in the MPM projection phase, we project hand particles onto an auxiliary grid system, specifically designed to store the number of the hand particles. Prior to executing the particle update

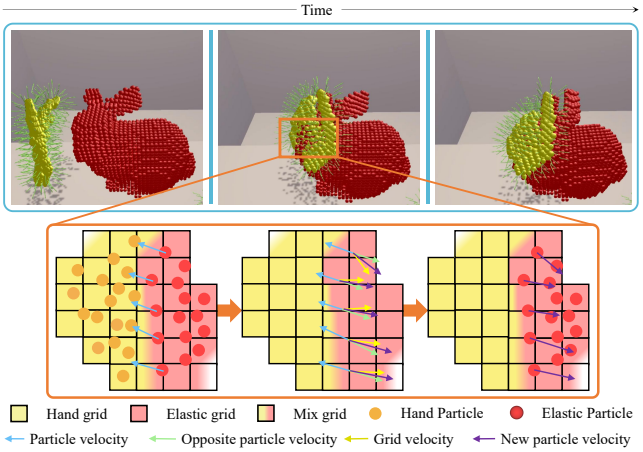

Figure 4: The anti-penetration process during the interaction between the hand and the elastic object.

procedure, we iterate through all grid, and if the number of hand particles on a cell surpasses a threshold min_d, the cell is marked as a hand model cell. For cells marked as part of the hand model, we analyze the directions of 27 neighboring cells centered around it and weight them by number of hand particles, determine the direction with the lowest number of hand particles, and create a temporary grid velocity in this direction to simulate a reactive force, as depicted in Fig. 4. During advection, for particles situated in hand-marked cells, we invert their original velocities along and incorporate the additional generated grid velocity weighted by the number of hand particles on each cell. Ultimately, this approach propels virtual particles away from the hand model, effectively preventing penetration.

Algorithm 1 Penetration Prevention

Input: Number of Hand Particles $cell_n$, Position of Particles x, Velocity of Cell $cell_v$
Output: Velocity of Particles v

for cell i in all cells **do**
  if $cell_n[i] > min_d$ **then**
    $normal = \overrightarrow{0}$
    for cell j in neighbor of cell i **do**
      $direct = x_i - x_j$
      $normal \mathrel{+}= cell_n[j] \cdot direct$
    end for
    $cell_v[i] \mathrel{+}= normal \cdot k$
  end if
end for
for p in all particles **do**
  $base = int\left(x_p/dx\right)$
  if $cell_n[base] > min_d$ **then**
    $v_h = \overrightarrow{0}$
    for cell j in neighbor of cell $base$ **do**
      $v_h \mathrel{+}= cell_n[j] \cdot cell_v[j]$
    end for
    $v_p = -v_p + v_h$
  end if
end for

The process of updating velocities of virtual particles is shown in Alg. 1, where $\overrightarrow{0}$ represents the zero vector, w_B represents the B-spline weight and $base$ represents the cell that the particle p stay in. We use min_d to examine the cells and particles that in the hand model. $normal$ represents the direction with the lowest number of hand particles, k represents the factor to control the magnitude of the anti-penetration force, where k is set to 2.5. And v_h represents the velocity of hand particle recorded on the cell that is inside the

hand model.

4 Real-time Rendering

4.1 Anisotropic Surface Reconstruction

Our system comprises various material objects composed of particles, each exhibiting distinct levels of transparency and deformation. To improve the visual display, we adapt distinct surface reconstruction and rendering techniques for different materials.

Since rigid objects do not undergo deformation and the surfaces of hand models are automatically tracked by the device, surface reconstruction is not necessary. Instead, we directly render these objects using the original mesh models, incorporating material textures and shaders.

Conversely, elastic objects exhibit relatively minor deformation while generally maintaining stable surfaces. To enhance the visual realism of the scene, we apply the marching cubes algorithm for surface reconstruction of elastic particles and recalculate texture coordinates on the reconstructed mesh. MPM algorithm projects particle mass to the grid, enabling the distribution of particles across the grid to be derived by dividing grid mass by individual particle mass. We apply a threshold value to differentiate between object interior and exterior. Cells with counts below the threshold are classified as external, while those surpassing it are labeled internal. In practice, this method has demonstrated efficiency in outlining fundamental geometric shapes, yielding substantial savings in time and space resources. This scheme propel real-time rendering acceleration.

Lastly, for fluid and plastic materials that experience significant deformations and display unstable surfaces, we employ screen-space surface reconstruction. In the following portion of this section, we will focus on the optimization techniques incorporated into our system for anisotropic screen-space surface reconstruction.

4.2 Particle Refinement

To reconstruct a smooth surface in real time, we initially calculate the depth texture of the particles. Following the initial setup, we refine the isosurface by applying bilinear smoothing, after which we compute the normals on this newly smoothed depth isosurface. To further enhance the quality of the surface, we undertake a process of particle anisotropic mapping. This involves conducting a principal component analysis on the neighborhood density distributions of the particles, as suggested by Yu et al. (2010) [30]. Implementing this technique necessitates a search for neighboring particles, which can significantly increase the time overhead. To streamline this process and improve efficiency, we bypass the need for an explicit neighbor particle search by obtaining the neighborhood particle distribution directly from the Eulerian grid. This strategic utilization of Material Point Method (MPM) grid not only bolsters the accuracy of our surface reconstruction but also maintains a high level of time efficiency.

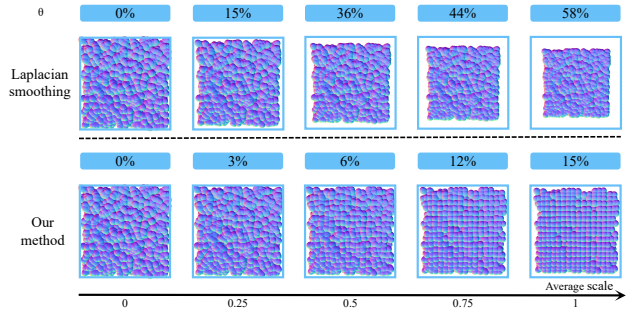

Figure 5: The top row shows the different degrees of the Laplacian smoothing [30], and the bottom row shows the result of our method, where theta is the degree of volume compression as Eq. 11.

The Laplacian smoothing method is commonly employed in particle refinement. This technique allows the target particle to acquire the distances and directions of neighboring particles, enabling it to determine the averaging direction, thus effectively reducing surface noise. Since the MPM is a hybrid Lagrangian-Eulerian method that obviates the need for neighbor search by projecting particles onto a grid and gathering data from the grid, our method comprises two simplified steps. First, particles project their numbers and directions onto the grid. And then the target particle collects the number and directions of neighboring particles from the grid. As these cells have unique weights and directions relative to the target particle, we can calculate the suitable direction for the new position of the particle to be averaged. These steps do not significantly increase computation and memory requirements, as both projection and collection can be integrated into the original MPM solver.

In our scheme, for each particle j, we directly collect the number of each neighboring cell i based on the particle's original position x_j. Then, the number of each cell i is combined with the B-spline weight w_{ij} assigned to the cell, resulting in the final weight $\widetilde{w}_{ij}$, the equation is shown as follows:

$$\widetilde{w}_{ij} = \rho_i + k_w \cdot w_{ij}, \tag{9}$$

where k_w represents the B-spline weight factor.

However, this method still remains a problem: the number of these cells distributes the weight of the same particle, thereby diminishing the particle's movement towards the average position. As a result, there is a minimal difference between the weighted average position and the original position, leading to insufficient particle position smoothing. To tackle this problem, we introduce a factor k_a, which scales the distance of particle movement from the original position to the new position. The averaged position of particle j is computed as follows:

$$\bar{\mathbf{x}}_j = \mathbf{x}_j + k_a \frac{\sum_i \widetilde{w}_{i,j} \left(\mathbf{x}_i - \mathbf{x}_j\right)}{\sum_i \widetilde{w}_{i,j}}, \tag{10}$$

where $\bar{x}_j$ denotes the weighted average position of particle j and $\widetilde{w}_{i,j}$ represents the weight of particle j on cell i, calculated in Eq. 9. Setting k_a excessively large can cause problems if the new particle position extends beyond the original neighborhood cell area, potentially impacting subsequent calculations of the anisotropic matrix. Through experimentation, we found that $k_a = 2.3$ is an appropriate value for a $64 \times 64 \times 64$ grid environment.

Our approach also provides the benefit of preserving the volume of the reconstructed surface without considerable compression. This is accomplished by managing the distance of particle movement to the new positions, effectively preventing particles from surpassing the original neighborhood cell area. It guarantees improved volume preservation of the fluid while maintaining a smooth surface. A comparison of the outcomes obtained using our particle position averaging method is depicted in Fig. 5. We compute the volume compression ratio θ as follows:

$$\theta = \frac{\mathbf{V_1}}{\mathbf{V_0}}, \tag{11}$$

where $\mathbf{V_0}$ represents the original volume of the particles and the $\mathbf{V_1}$ represents the volume of the particles before averaging.

Furthermore, to prevent excessively large particle position averaging that may result in discontinuous particle motion, we introduce a blending technique that merges the average position with the original position. The equation is shown as follows:

$$\widetilde{\mathbf{x}}_j = (1-\lambda)\,\mathbf{x_j} + \lambda \bar{\mathbf{x}}_j, \tag{12}$$

where $\bar{x}_j$ denotes the weighted average position of particle j, calculated in Eq. 10, and λ is a factor that smooths the movement of the particle. The figures in the bottom row of Fig. 5 display our method with different λ values, which are 0.2, 0.7, and 1 from left to right,

respectively. We set the value of λ to 0.7. By blending the average position and the original position, we attain a smoother transition between the two, effectively minimizing the discontinuity in particle motion. Compared to the original method [30], the choice of λ enables us to control the balance between the average position and the original position, ensuring more natural and continuous movement of the particles.

4.2.1 Hybrid Particle-Grid Anisotropy

In surface reconstruction, the rotation and stretching of particles based on their density distribution can significantly improve the quality and smoothness of the reconstructed surface. We utilize the distribution of the surrounding cells and B-spline weights to determine the anisotropic matrix G, which serves as a guide for the rotation and stretching of particles during surface reconstruction.

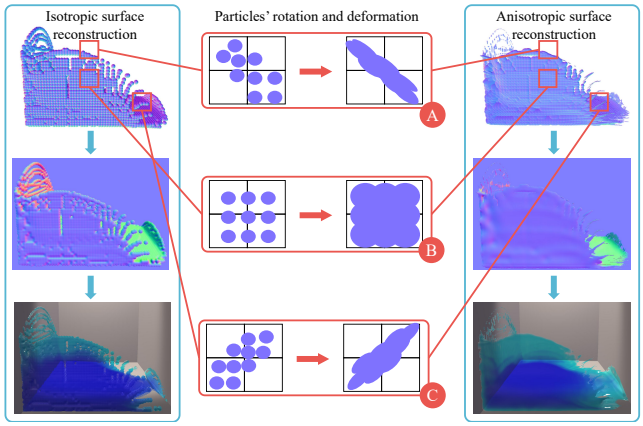

Figure 6: The left figures and right figures show the results of isotropic and anisotropic surface reconstruction, and the middle figures show the specific anisotropic process.

Based on the weighted average positions of the particles obtained in the previous section, for each particle j, we gather the position x_i of each neighboring cell i around the particle's average position. Subsequently, we calculate the covariance matrix of these cells' positions with the average position $\bar{x}_j$ of the particle and weight them using the method described in Eq. 9. Through this process, we can obtain the original covariance matrix G_j of particle j. The equation is shown as follows:

$$\mathbf{G}_j = \frac{\sum_i \widetilde{w}_{i,j} \left(\mathbf{x}_i - \bar{\mathbf{x}}_j\right) \left(\mathbf{x}_i - \bar{\mathbf{x}}_j\right)^T}{\sum_i \widetilde{w}_{i,j}}, \tag{13}$$

where $\widetilde{w}_{i,j}$ represents the weight of particle j on cell i, x_i denotes the position of cell i, and $\bar{x}_j$ represents the weighted average position of particle j.

Next, we perform singular value decomposition on the covariance matrix $\mathbf{G}$ to determine the direction and degree of particle stretching or compression, which are shown as:

$$\mathbf{G}_j = \mathbf{U}\sum\mathbf{W}, \tag{14}$$

$$\sum = diag\left(\sigma_1, \sigma_2, \sigma_3 ..... \sigma_n\right), \tag{15}$$

where the singular values, represented by the elements on the main diagonal of the matrix $\sum$, indicate the extent to which the particle stretches in each direction, and σ_i represents the i-th singular value of the covariance matrix $\mathbf{G}$.

To ensure these values remain as close to 0.1 as possible and avoid significant gaps between them, we apply two corrections:

$$\widetilde{\sigma}_i = max\left(\sigma_{max}, min\left(\sigma_{min}, \sigma_i\right)\right), \tag{16}$$

$$\widetilde{\sigma}_i = max\left(\widetilde{\sigma}_{max}/k, \sigma_i\right), \tag{17}$$

$$\widetilde{\sum} = diag\left(\widetilde{\sigma}_1, \widetilde{\sigma}_2, \widetilde{\sigma}_3 ..... \widetilde{\sigma}_n\right), \tag{18}$$

$$\widetilde{\mathbf{G}}_j = \mathbf{U}\widetilde{\sum}\mathbf{W}, \tag{19}$$

where the first correction, given by Eq.16, sets the maximum singular value σ_{max} to 0.15 and the minimum singular value σ_{min} to 0.05. The second correction, detailed in Eq. 17, involves setting the value of k to 3, while $\widetilde{\sigma}_{max}$ represents the maximum singular value in the matrix $\sum$, with a different meaning from that in Eq. 16. By applying these two corrections to all elements in the matrix $\sum$, we obtain the revised singular value matrix $\widetilde{\sum}$, as shown in Eq. 18. Finally, we multiply the new singular value matrix $\widetilde{\sum}$ by the matrices $\mathbf{U}$ and $\mathbf{W}$ to get the final anisotropic matrix $\widetilde{\mathbf{G}}$, as presented in Eq. 19.

By utilizing the anisotropic matrix $\widetilde{\mathbf{G}}$, we can incorporate anisotropic particles into the process of extracting depth texture and thickness texture for screen-space-based surface reconstruction. As demonstrated in Fig. 6, the left figures show the results of isotropic surface reconstruction, in which the particles are drawn as spheres. Meanwhile, the right figures show the results of anisotropic surface reconstruction, in which the particles are drawn as ellipsoids. The middle figures display the deformation of a particle from a sphere to an ellipsoid.

In Fig.6(B), if the particles are distributed uniformly around the target particle, then the target particle will uniformly extend in all directions, forming a flat shape. In Fig.6(A) and (C), if the particles around the target particle are distributed in only one direction, the target particle will stretch in the same direction and compress along the normal to that direction, forming a thin ellipsoid. Additionally, if the direction is not parallel to the axis, the anisotropic matrix G will rotate the axis of the particle in that direction. These figures showcase the improved visual quality achieved through the utilization of anisotropic particles in the reconstruction process.

During implementation, fluid particles are stored in the depth texture using the vertex shader [27]. Our anisotropic matrix and singular values are employed to transform the particles from their original sphere shape to an ellipsoid shape, following the technique introduced by Sigg et al. [24] for ellipsoid drawing. Subsequently, we perform depth texture smoothing and normal reconstruction to obtain the reconstructed surface.

5 Experiment and Validation

5.1 Experimental Settings

The experimental scenes are executed on a high-performance system, specifically an Intel i7 with 2.6GHz and an NVIDIA RTX 3070 graphics card. We employ *Leap Motion* to accurately capture and track the fine movements of hands and fingers, and a *VIVE PRO* is used to display VR scenes. The virtual environment used in this experiment is developed using *Unity 3D*, a widely-used game development engine known for its versatility and efficiency. The implementation of our physics-based solver uses the Taichi framework [15].

5.2 Feasibility Study

Simple test scene. Initially, we devised several scenarios to assess the HCI capabilities of distinct material objects. The experimental scenes illustrated in the punch and slap sections of Fig. 7 are tailored to gauge the impact of robust hand motions and forceful interactions. On the other hand, the scenarios portrayed in the point and grasp sections are curated to appraise delicate and intricate fingertip interactions. The synergy between our HCI and physical simulation elements yielded promising results.

Water controlling scene. As shown in Fig. 9, we designed a scene featuring free surface water manipulated by hand gestures. This scene comprised 60,000 fluid particles, and the fluid surface was reconstructed using our hybrid particle-grid anisotropy method. Initially, gravity was not present, and users could employ their

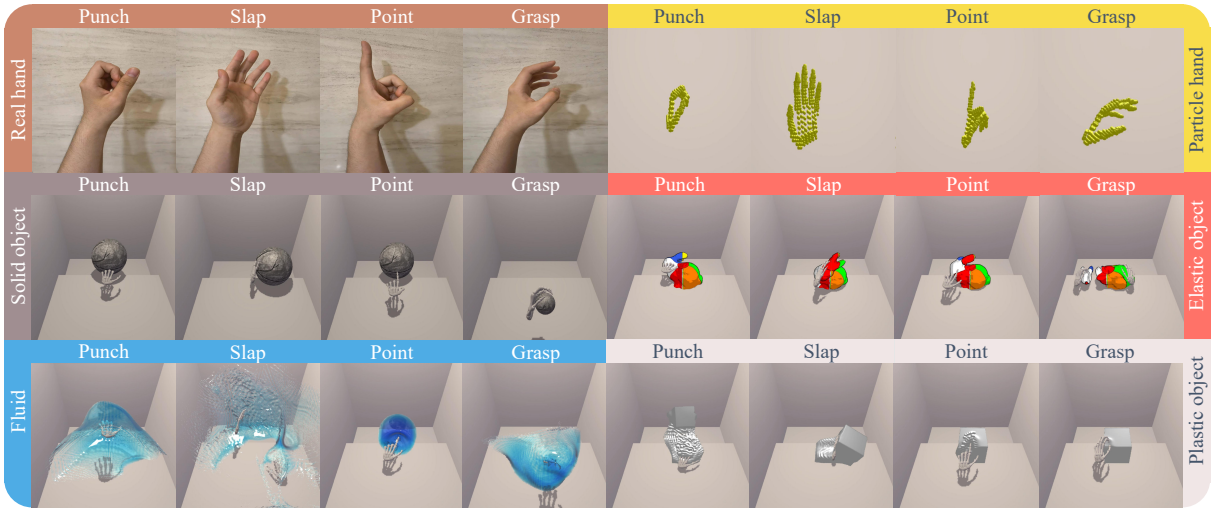

Figure 7: Interaction phenomenons between particle hand and different material objects.

Table 2: The benchmarks of experiment scenes.

Scene	Particle		FPS	Mem(GB)
	Type	Number		
Simple test	fluid	36,402	93.5	2.56
	elastic bunny	2,041	130.2	2.35
	plastic cube	8,000	112.7	2.44
	solid sphere	4,964	69.2	2.37
	hand	2,306	141.3	2.32
Water controlling	fluid	60,000	96.7	2.63
	hand	2,306		
Basketball shooting	elastic ball	4,950	122.3	2.38
	hand	2,306		
Firehose spraying	fluid	10,000	127.0	2.47
Teaser scene	elastic ball	9,461	81.7	2.70
	fluid	60,000		
	hand	2,306		

hands to exert external force, enabling a magical water control performance.

Basketball shooting scene. The virtual scene depicted in Fig. 8 comprised a basketball court, a basket, and a basketball composed of elastic particles with textures attached to the virtual objects. Participants were provided with a pair of avatar hands that mimic their own hand gestures. Their task was to control the virtual hand, grab the basketball, and then shoot it into the basket.

Firehose spraying scene. This scene shows a water firehose, a virtual hand model representing the right hand, four transparent containers, and four progress bars, as depicted in Fig. 10. Participants were asked to control the water firehose using their left hand, with the spray direction determined by the orientation of their left palm. The right hand was responsible for controlling the water spray switch and selecting the water color. To initiate the water spray, participants opened all five fingers of their right hand, then made a clenched fist to start, and moved the right hand in different directions to change the water color. The progress bars displayed the injection progress for each color, and participants need to fill all four progress bars.

The benchmarks of all scenes are presented in Table 2, where the scenarios in the top four lines of the **Simple Test** include the consumption of the fifth line of hand. Remarkably, all scene frame rates are consistently above 60, demonstrating real-time performance.

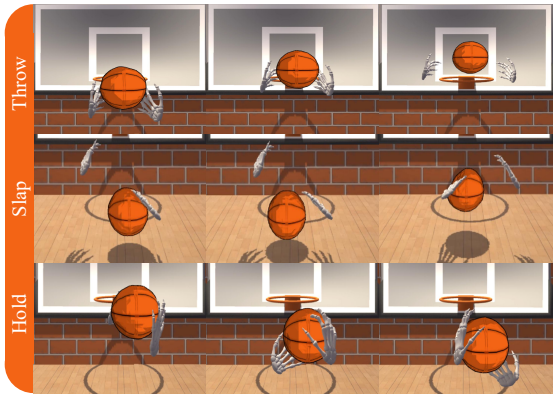

Figure 8: Basketball shooting scene. The basketball is composed of plastic particles.

5.3 Comparisons

We have conducted a comparison of the voxelization schemes presented in table 3. In the table, the four rows of data, H1, H2, H3, and H4, represent four different voxelization schemes for the index finger bone, while the columns B1, B2, B3, and B4 represent the four phalanxes from farthest to closest to the fingertip.

Table 3: The particle number of index finger bones.

Hand	Particle number					FPS
	B1	B2	B3	B4	Sum	
H1	28	24	19	12	83	192
H2	154	127	108	57	446	73
H3	81	85	49	5	220	152
H4	27	47	69	72	215	141

H1 denotes a very low voxelization density for each phalanx. In practice, this leaded to frequent penetrations of the hand model, and the extremely sparse distribution of particles made hand model nearly imperceptible during interactions with virtual objects. H2 denotes a very high voxelization density for each phalanx. Although this hand model exhibited good physical interaction, it significantly compromised system efficiency. H3 represents voxelization with a similar density for each phalanx, resulting in more particles in the phalanges further away from the fingertip due to their larger size, while the phalanges near the tip contains only a few particles. This hand model was capable of actions such as slapping, pushing, and pressing, but struggled with tasks requiring fingertip force, such as holding and pulling. In contrast, H4 represents a medium density of

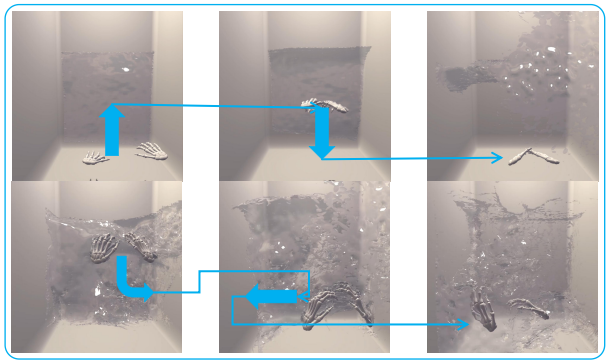

Figure 9: Water moving with hand control. The blue arrows indicate the hands moving directions.

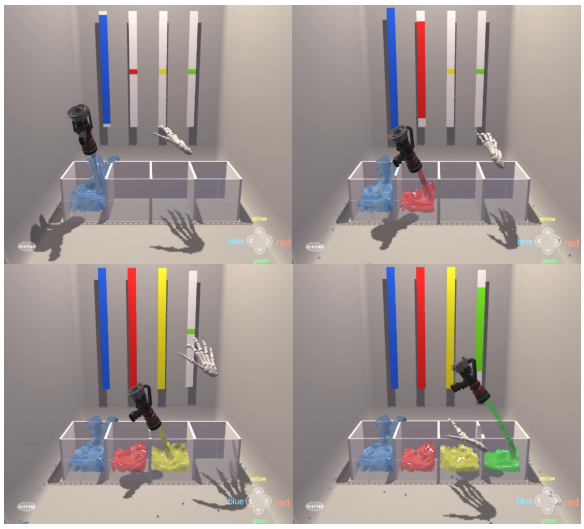

Figure 10: Firehose spraying scene. The left hand controls the firehose to spray four different colors of water.

voxelization for each phalanx, with a higher density at the fingertips. This hand model demonstrated excellent performance in holding and pulling actions, combining the advantages of the previously mentioned schemes. So we adopt the voxelization scheme H4.

We have also conducted user comparison tests and mathematical analyses to assess the naturalness, realism, and diversity of our system in comparison with HPL (Hand Physics Lab)[1]. The study included 20 participants (10 males and 10 females) between the ages of 20-25 and half of the participants had prior experience with VR/AR equipment. Initial analysis using the Shapiro Wilk test obtained significant differences in naturalness and diversity, while realism exhibited no significant distinction. So we used non-parametric tests for further exploration. For the non-parametric assessment, we have employed the Mann-Whitney test. In terms of realism ($U = 34$, $z = -0.904$, $p = 0.366$), our system aligned well with Hand Physics Lab (HPL). Conversely, both naturalness ($U = 22.500$, $z = -1.965$, $p = 0.049$) and diversity ($U = 24.000$, $z = -2.000$, $p = 0.045$) demonstrated notable discrepancies when compared to HPL. An examination of the median differences unveiled that HPL's naturalness (median: 3.000) and diversity (median: 3.250) were significantly lower than our results. Our system achieved a naturalness median of 4.000 and a diversity median of 4.400. The results indicate that our system matches HPL in realism, but notably surpasses HPL in terms of both diversity and naturalness.

[1]https://www.holonautic.com/hand-physics-lab

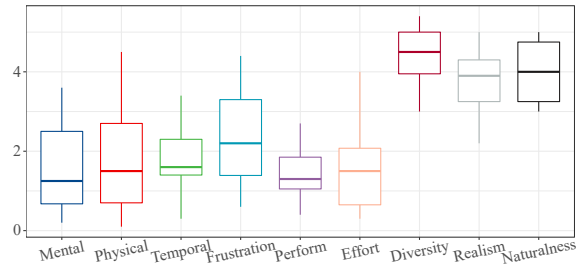

Figure 11: The box-plot of NASA-TLX scale scores and other items.

5.4 User Study

Furthermore, we have conducted user experiments with our system, including full NASA-TLX scale scores and some other metrics. Participants were the same as in the user experiment in Section 5.3, and they were tasked with completing a set of predefined tasks using our framework. Each participant performed the same experiments in **Basketball shooting scene** and **Firehose spraying scene** three times to ensure consistency and reliability of the results.

To evaluate the user experience and effectiveness of PhyVR, we administered a questionnaire that included six rating items: naturalness, usefulness, diversity, realism of the fluid, realism of the elastic object, and ease of learning. Participants were asked to rate each item on a Likert scale ranging from 1 (very poor) to 5 (very good). Fig. 11 provides a visual representation of the results, plotting the mean raw NASA-TLX scale scores and additional questions across all participants. This box plot is divided into two sections: the five columns on the left side assess the participants' burden during the experiment (with lower scores being preferable), while the three columns on the right side evaluate the effectiveness of our system (with higher scores being preferable). According to the questionnaire results, most participants reported low levels of mental, physical, temporal, and effort demands. However, while most participants experienced low frustration levels, a few reported high levels of frustration. This discrepancy was likely due to the limited gesture recognition area of the Leap Motion system. Despite this, the majority of participants rated highly in categories such as diversity, realism, and naturalness.

6 Conclusions

Our approach, PhyVR, integrates a physics engine that simulates lifelike interactions between the user's actual hand and virtual objects. This unique combination delivers an immersive and realistic VR experience. We utilize a unified particle system to seamlessly integrate virtual and real interactions, thereby enhancing the accuracy of physics-based calculations. Furthermore, we improve the fluid surface reconstruction method by introducing a novel hybrid particle-grid anisotropy. This advancement propels real-time rendering techniques in HCI to new heights. Our framework allows users to engage in a variety of physical simulations within VR environments, opening a new realm of possibilities for the fusion of VR and physical simulation. In this study, we offer a comprehensive comparison and evaluation of PhyVR. Experimental results demonstrate PhyVR's effectiveness in maintaining a strong sense of presence and delivering high-quality interaction performance, even in complex VR environments. These findings carry significant implications for the design and development of VR applications, especially those aiming to achieve realistic and immersive HCI.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No. 62002010, 62272019), Beijing Advanced Innovation Center for Biomedical Engineering under Grant (ZF138G1714), CAMS Innovation Fund for Medical Sciences (CIFMS) under Grant (2019-I2M-5-016), and Beijing Science and Technology Project (No. Z221100007722001).

REFERENCES

[1] A. D. Blaga, M. Frutos-Pascual, and C. Creed. Freehand grasping: An analysis of grasping for docking tasks in virtual reality. *IEEE Virtual Reality and 3D User Interfaces (VR)*, pp. 749–758, 2021.

[2] T. Brochu and T. Brochu. Robust topological operations for dynamic explicit surfaces. *SIAM Journal on Scientific Computing*, 31(4):2472–2493, 2009.

[3] V. Buchmann, S. Violich, M. Billinghurst, and A. Cockburn. Fingartips: gesture based direct manipulation in augmented reality. *ACM GRAPHITE*, pp. 212–221, 2004.

[4] N. Chentanez and M. Müller. Real-time eulerian water simulation using a restricted tall cell grid. In *ACM Siggraph 2011 Papers*, pp. 1–10. 2011.

[5] R. H. Durisen and J. E. Tohline. A numerical study of the fission hypothesis for rotating polytropes. *Space Science Reviews*, 27:267–273, 1980.

[6] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics*, 183(1):83–116, 2002.

[7] D. Enright, S. Marschner, and R. Fedkiw. Animation and rendering of complex water surfaces. *ACM Transactions on Graphics (TOG)*, 21(3):736–744, 2002.

[8] S. Eroglu, S. Gebhardt, P. Schmitz, D. Rausch, and T. W. Kuhlen. Fluid sketching—immersive sketching based on fluid flow. In *IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 475–482, 2018.

[9] Y. Gao, Q. Zhang, S. Li, A. Hao, and H. Qin. Accelerating liquid simulation with an improved data-driven method. In *Computer Graphics Forum*, vol. 39, pp. 180–191, 2020.

[10] C. Garre, F. Hernández, and A. Gracia. Interactive simulation of a deformable hand for haptic rendering. *IEEE World Haptics Conference(WHC)*, pp. 239–244, 2011.

[11] J. Guna, G. Jakus, M. Pogacnik, S. Tomazic, and J. Sodnik. An analysis of the precision and reliability of the leap motion sensor and its suitability for static and dynamic tracking. *Sensors*, 14(2):3702–3720, 2014.

[12] C. W. Hirt and B. D. Nichols. Volume of fluid(vof) method for the dynamics of free boundaries. *Journal of computational physics*, 39(1):201–225, 1981.

[13] M. Höll, M. Oberweger, C. Arth, and V. Lepetit. Efficient physics-based implementation for realistic hand-object interaction in virtual reality. *IEEE Virtual Reality and 3D User Interfaces (VR)*, pp. 175–182, 2018.

[14] Y. Hu, Y. Fang, Z. Ge, Z. Qu, Y. Zhu, A. Pradhana, and C. Jiang. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics (TOG)*, 37(4):150:1–150:14, 2018.

[15] Y. Hu, T.-M. Li, L. Anderson, L. Anderson, J. Ragan-Kelley, and F. Durand. Taichi: A language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics (TOG)*, 38(6):1–16, 2019.

[16] J. Jacobs and B. Froehlich. A soft hand model for physically-based manipulation of virtual objects. *IEEE Virtual Reality and 3D User Interfaces (VR)*, pp. 11–18, 2021.

[17] P. Molchanov, X. Yang, and S. Gupta. Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4207–4215, 2016.

[18] J. Monaghan. Smoothed particle hydrodynamics. *Annual Review of Astronomy Astrophysics*, pp. 543–548, 1992.

[19] M. Müller, S. Schirm, and S. Duthaler. Screen space meshes. *ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 9–15, 2007.

[20] K. Nasim and Y. J. Kim. Physics-based interactive virtual grasping. *HCI Korea (HCIK '16)*, pp. 114–120, 2016.

[21] S. Osher and R. Fedkiw. Level set methods and dynamic implicit surfaces. *Springer Verlag*, 2002.

[22] J. Pan, L. Zhang, P. Yu, Y. Shen, H. Wang, H. Hao, and H. Qin. Real-time vr simulation of laparoscopic cholecystectomy based on parallel position-based dynamics in gpu. In *IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 548–556, 2020.

[23] A. G. Perez, G. Cirio, F. Hernandez, C. Garre, and M. A. Otaduy. Strain limiting for soft finger contact simulation. *IEEE World Haptics Conference (WHC)*, pp. 79–84, 2013.

[24] C. Sigg, T. Weyrich, M. Botsch, and M. Gross. Gpu based ray-casting of quadratic surfaces. *Eurographics Symposium on Point-Based Graphics*, pp. 59–65, 2006.

[25] J. Strain. A fast semi-lagrangian contouring method for moving interfaces. *Journal of Computational Physics*, 170(1):373–394, 2001.

[26] D. Sulsky, S.-J. Zhou, and H. L. Schreyer. Application of a particle-in-cell method to solid mechanics. *Computer Physics Communications*, 87(1):236–252, 1995.

[27] W. J. van der Laan, S. Green, and M. Sainz. Screen space fluid rendering with curvature flow. *symposium on Interactive 3D graphics and games*, pp. 91–98, 2009.

[28] X. Xiao, S. Zhao, Y. Meng, L. Soghier, X. Zhang, and J. Hahn. A physics-based virtual reality simulation framework for neonatal endotracheal intubation. In *IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 557–565, 2020.

[29] D. Yima, G. N. Loison, F. H. Fard, E. Chan, A. McAllister, and F. Maurer. Gesture-driven interactions on a virtual hologram in mixed reality. *ACM ISS Companion*, pp. 55–61, 2016.

[30] J. Yu and G. Turk. Reconstructing surfaces of particle-based fluids using anisotropic kernels. *ACM SIGGRAPH/Eurographics symposium on Computer animation*, 32(1):1–12, 2010.