ViP-Fluid: Visual Perception Driven Method for VR Fluid Rendering

Qixiang Ma*
Beihang University

Jian Wu†
Beihang University

Runze Fan‡
Beihang University

Guodong Sun§
Northwestern Polytechnical University

Xuehuai Shi¶
Nanjing University of Posts and Telecommunications

2024 IEEE International Symposium on Mixed and Augmented Reality (ISMAR) | 979-8-3315-1647-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISMAR62088.2024.00050

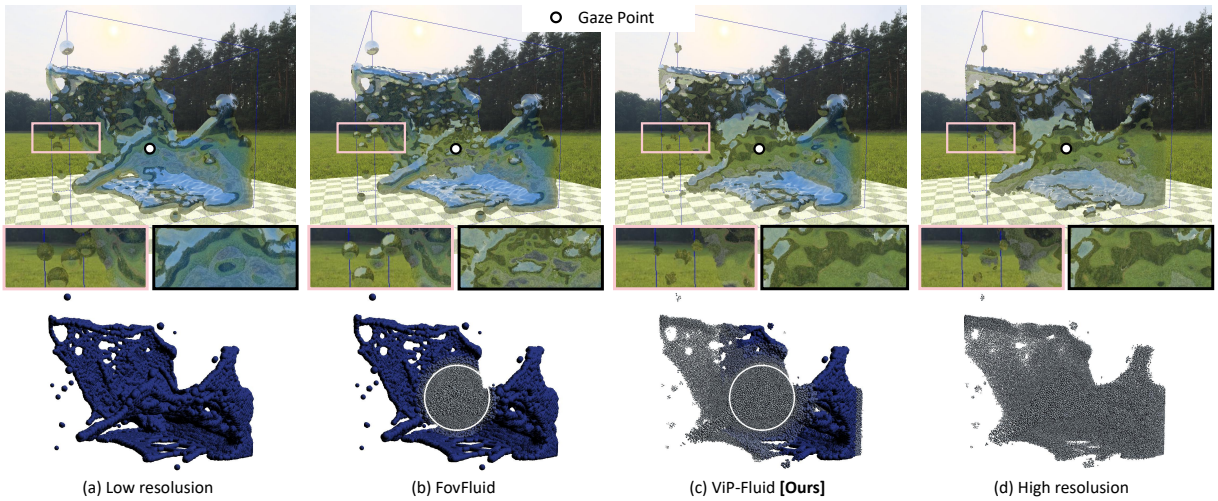


Figure 1: Comparison of fluid renderings for the *SingleDam* scene. Column (c) showcases our ViP-Fluid method alongside low (a) and high (d) resolution PBF[18] benchmarks, and the SOTA method FovFluid[42] (b). The first row depicts their renderings with zoom-in views of the visually salient (red box) and foveated areas (black box). The second row illustrates the granularity levels of particle swarms, transitioning from coarse (blue) to fine (white), as determined by the visual perception saliency model.

Abstract

The demand for fluid simulation and rendering in virtual reality (VR) is increasing. However, achieving high visual quality while maintaining real-time efficiency remains a challenge. Traditional foveated rendering methods balance the simulation quality in the foveated region but neglect the physical realism in the peripheral areas, and fail to account for the perceptual degradation caused by frame rate fluctuations during adaptive updates. To address these challenges, we propose a novel visual perception driven fluid rendering method ViP-Fluid, which further enhances rendering quality while balancing efficiency. Our approach employs a spatiotemporal saliency model for multi-granularity simulation and rendering of Lagrangian fluid systems, and introduces a Perception Threshold for Physical Process Elapsing (PTPE) metric, which guides our temporal acceleration strategy. Through a series of objective experiments, we demonstrate the advantages of our method in rendering quality and performance efficiency. ViP-Fluid demonstrates superior metrics not only in the foveated region but also in the salient and overall regions, achieving up to 2.15 times speed-up compared to the high-resolution Position Based Fluids (PBF) benchmark. Subsequent user experiments further validate the visual perception advantages of ViP-Fluid over both traditional and state-of-the-art methods, confirming the spatiotemporal fidelity of our acceleration strategy as well as a user preference for our approach.

*e-mail: sycamore_ma@buaa.edu.cn
†The corresponding author is Jian Wu, e-mail: lanayawj@buaa.edu.cn
‡e-mail: by2106131@buaa.edu.cn
§e-mail: gavinsun0921@mail.nwpu.edu.cn
¶e-mail: xuehuai@njupt.edu.cn

Index Terms: Virtual reality, fluid simulation, visual perception, foveated rendering, temporal acceleration.

1 Introduction

Fluid simulation and rendering in VR pose unique challenges, balancing high visual quality with real-time efficiency. Traditional methods like high-resolution Position-Based Fluids (PBF) [18] offer detailed visuals at the cost of computational complexity and reduced performance. In contrast, low-resolution simulations or foveated rendering methods like FovFluid [42] improve efficiency but often compromise quality, particularly in peripheral regions. FovFluid's limited two-level particle scale leads to significant spatiotemporal fluctuations during complex updates, overlooking potential visually salient details.

Many spatial and temporal fluid phenomena, such as splashes, ripples, fragmentation, turbulence, vortices, and water crowns, attract visual attention during simulation and rendering. In VR near-eye rendering tasks, improving the quality of these visual interest areas while balancing the efficiency in laminar or hidden areas is a valuable direction to explore. Our approach, ViP-Fluid, leverages multi-level granularity particle simulation, adjusting resolution based on visual perception and computational demands, and employs a spatiotemporal saliency model to enhance detail fidelity in both spatial and temporal aspects. To reduce negative effects in the temporal domain brought by adaptive updating, this method introduces a Perception Threshold for Physical Process Elapsing (PTPE) to assess these effects and calibrates a model through a pilot user study, which informs a temporal acceleration strategy to effectively balance quality and efficiency.

A series of objective experiments were conducted to evaluate the quality and performance of our method. Our experiments span across three fluid scenarios and two baselines, comparing five conditions. The results demonstrate the superior performance of ViP-Fluid across critical metrics in foveated, salient, and overall regions.

979-8-3315-1647-5/24/$31.00 ©2024 IEEE
DOI 10.1109/ISMAR62088.2024.00050

Our method achieves up to 2.15 times speed-up compared to the high-resolution PBF benchmark. Moreover, our acceleration efficiency effectively enhances the PTPE perception score. Subsequent user experiments further validate the visual perception advantages of ViP-Fluid over traditional state-of-the-art (SOTA) methods, confirming the spatiotemporal fidelity of our acceleration strategy. Furthermore, user preference for our approach is demonstrated to be favorable, reaching over 80% compared to FovFluid.

In summary, we present our main contributions as follows:

- We introduce a visual perception-driven saliency model that calculates the importance characteristics of Lagrangian fluid systems from spatial and temporal dimensions, guiding multi-granularity particle updates.
- We establish a PTPE threshold to assess temporal perception in physical simulations, calibrated through preliminary user experiments, and employed to direct temporal acceleration strategies in system simulations.
- We perform objective experiments and user studies to evaluate ViP-Fluid and compare it with other methods across different configurations, confirming the beneficial effects of our acceleration strategy on visual perception.

2 Related Works

Fluid simulation and rendering in VR, applicable to gaming, education, medicine, and engineering, has seen early prototypes that demonstrate its potential. For instance, Boettcher et al. [3] demonstrated fluid mechanics in VR, while Chen et al. [6] simulated acidic fluid splashes in labs. Yan et al. [44] use VR sketches to guide the generation of fluid shapes, while Deng et al. [9] explored free-hand interaction with fluids in VR. Other applications include water therapy [26], interactive gaming with real water and virtual feedback [8], and development of parallel accelerated fluid simulation systems [14, 45]. These initial implementations, whether through sparse simplification or small-scale simulation, have inspired advancements in VR fluid dynamics.

2.1 Non-data-driven Fluid Simulation

Traditional physics-based non-data-driven fluid simulation methods are categorized into three primary approaches: particle-based from a Lagrangian perspective [25, 40, 20], grid-based from an Eulerian perspective [33, 7], and the lattice Boltzmann method (LBM) [12] for complex flows like those in porous media. Deep learning or data-driven methods [5] are efficient in model testing but consume significant computational resources during data preparation and training. They achieve realistic visual effects but lack the physical accuracy and explainability of traditional models. Deep learning methods often require retraining for new scenes, reducing portability in VR applications. In VR simulations, particle-based methods such as Smoothed Particle Hydrodynamics (SPH) [15] and Position Based Fluids (PBF) [18] are prevalent due to their suitability for real-time rendering with Screen Space Fluid (SSF) algorithms [39, 38, 22]. To optimize the efficiency of fluid simulation for different scenarios, adaptive techniques adjust computational resources dynamically. For example, Two-Scale [32] and Multi-Scale [1] simulations vary resolution across scenes to match flow dynamics, while Temporal Blending [23] smooths SPH computations over time for enhanced realism. Innovations in adaptive fluid simulation, such as Pixar's mass-conserving method [13], have broad applications in virtual reality and gaming. FovFluids [42] exemplifies this with its two-scale approach, prioritizing high-resolution simulations in the gaze-focused areas and lower resolutions peripherally. In a hybrid Eulerian and Lagrangian perspective, several classic algorithms have been derived, such as the Material Point Method (MPM) [36], Particle-In-Cell (PIC) [16, 11], and Fluid-Implicit Particle (FLIP) [4]. Among them, several excellent adaptive methods have also emerged, such as [2, 21, 10]. They offer sophisticated tools for adapting fluid simulation.

2.2 Visual Perception Models

The adaptive methods discussed previously did not adequately account for the Human Visual System (HVS) perceptions, with FovFluids [42] being an exception focusing on real-time rendering. Visual perception models, integral to simulating HVS perceptions, use mathematical and computational approaches to capture and analyze processes related to light, color, contrast, space, and time. Models like the visual acuity decline [17, 43] and contrast sensitivity functions (CSF) [28, 24] address spatial eccentricity and contrast's impact on perception, influencing rendering quality based on proximity to the gaze point. Other studies [34, 31] have examined sensitivity changes due to relative motion, underscoring the increased perception sensitivity closer to gaze points. FovFluids leverages foveated rendering [41] to optimize rendering quality based on eccentricity, reducing computational loads in less sensitive peripheral areas. Despite these advancements, global adaptive simulation and inherent visual perception attributes of fluids remain underexploited. This paper integrates visual perception models into real-time fluid rendering, aiming to enhance realism and performance in VR applications by adapting to visually significant areas and viewer interactions.

3 Method

3.1 Rendering Pipeline

As depicted in Fig. 2, our ViP-Fluid rendering pipeline consists of three primary stages:

- **Visual Saliency Calculation:** Incorporating previous frame data on fluid particle attributes and gaze information derived from eye movement, we compute particle saliency using a visual perception model accounting for both spatial and temporal aspects. This generates a comprehensive saliency map, detailed further in subsection 3.2.
- **Multi-level Particles Update:** We enhance granularity in areas of high saliency by linking particle scale levels to their saliency values. Smaller particles in these areas ensure higher accuracy, while larger particles enhance efficiency. Particle sizes are dynamically adjusted through splitting or merging, as governed by the protocols in subsection 3.3.
- **Solver with Temporal Acceleration:** To accelerate fluid rendering in VR, we constructed velocity field grids and implemented a saliency-based activation mechanism for each particle. Activation sampling determines whether a particle is processed by the PBF solver to update its attributes on the grid or remains inactive, inheriting grid attributes directly and pausing splitting updates. This approach optimizes rendering speed and efficiency. Details on this strategy are in section 4.

After three stages, we generate a frame with particles of varied saliency granularities. These are rendered using the SSF algorithm and displayed on a VR Head-Mounted Display (HMD) integrated with an infrared eye movement tracking device for real-time binocular visualization and input of the next frame's gaze information.

3.2 Visual Perception Saliency of Particles

Contrasting the FovFluid [42] approach, our ViP-Fluid method comprehensively considers visual perception disparities stemming from gaze information, fluid dynamics, and changes in physical parameters. To address these disparities for multi-granularity rendering, we introduce a visual perception saliency model that integrates spatial and temporal components. This model guides subsequent granularity determination by defining visual saliency features for particles, as detailed in Equation 1. Here, the spatial feature f_S

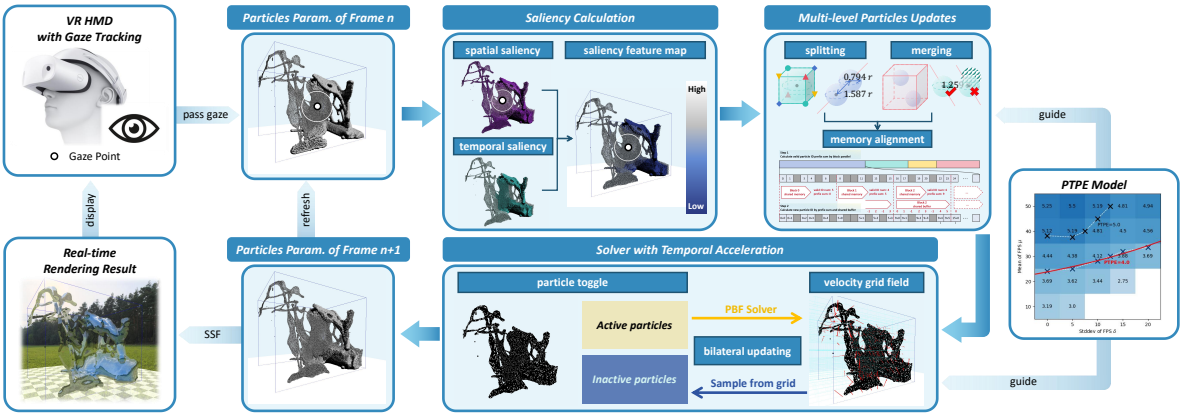
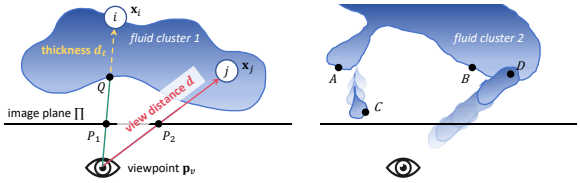


Figure 2: ViP-Fluid rendering pipeline.

and temporal feature f_T combine to calculate the spatio-temporal saliency of particle i:

$$F(i,\mathbf{p}_v,\mathbf{g},\Pi,\Phi) = \quad (1-\alpha)f_S+\alpha f_T \quad . \tag{1}$$

The function's input variables, in addition to some necessary physical parameters of particle i, also include viewpoint $\mathbf{p}_v$, gaze direction vector $\mathbf{g}$, image plane Π, and G-buffer Φ. In the following sections, we utilize the normal buffer component Φ_n and the depth buffer component Φ_z. Here, α represents the weight for the temporal saliency feature. Increasing α enhances temporal turbulence accuracy, while decreasing α highlights spatial details. It is adjustable (see subsection 5.1) based on user's scene requirements.



(a) The case of fluid cluster 1. (b) The case of fluid cluster 2.

Figure 3: Parameter illustration for f_a, f_m, and f_T with model explanations. (a) depicts particle view distance and thickness. (b) shows spatiotemporal saliency differences; manifold feature at point A exceeds B, while C and D show greater temporal saliency.

3.2.1 Spatial Saliency

The viewpoint's spatial relationship with the fluid and the fluid's manifold influence visual perception. We aim to enhance the saliency in visually acute regions, such as foveated areas or near the viewpoint, and where fluid manifold features like splashes or liquid bridges are prominent. Hence, spatial saliency involves two components: the visual acuity feature f_a and the fluid manifold feature f_m, as detailed in Equation 2:

$$f_S = f_a + f_m \quad . \tag{2}$$

Visual Acuity Features. Many SOTA foveated rendering works [42, 29] consider eccentricity to dictate rendering quality. Eccentricity on a 2D image plane dictates the granularity for each pixel or projected object. However, in a 3D fluid system, traditional 2D eccentricity fails to account for perspective-related variations in particle saliency. Distant particles require less granularity due to reduced perceptual significance, while overly coarse granularity near the viewpoint results in visible roughness. To dynamically adapt to these spatial variations, we introduce a trim factor γ_d in the visual acuity feature f_a, influenced by the optic axis depth of fluid particles, detailed in Equation 3:

$$f_a(i,\mathbf{p}_v,\mathbf{g},\Pi) = \quad clamp\left(\frac{e(\mathbf{x}_i,\mathbf{p}_v,\mathbf{g})}{\gamma_d},0,1\right) \quad , \tag{3}$$

where $e(\cdot)$ is the eccentricity function defined in [35], receiving the position $\mathbf{x}_i$ of particle i, and ensuring a smooth transition from high saliency inside the foveated area to lower saliency in the peripheral area. The denominator γ_d is presented in Equation 4. As the visual distance becomes closer, the salience of the particles also increases and the particle size becomes finer.

$$\gamma_d(i,\mathbf{p}_v,\Pi) = \quad \frac{d}{\|\mathbf{ints}_\Pi(\mathbf{x}_i-\mathbf{p}_v)-\mathbf{p}_v\|} \quad . \tag{4}$$

Here, d represents the view distance from the viewpoint to the particle, as shown in Figure 3. The operator $\mathbf{ints}_\Pi(\cdot)$ denotes the intersection of a vector with the image plane Π. $\mathbf{ints}_\Pi(\mathbf{x}_i-\mathbf{p}_v)$ represents the intersection of the viewpoint-particle vector with the image plane, as indicated by points P_1 and P_2 in the figure.

Manifold Features. We aim to enhance the visual saliency of particles in areas with significant free surface changes, such as splashes, liquid bridges, ripples, and wave edges. In computer graphics, the post-rendering normal vector buffer serves to indicate surface variability. Areas with notable normal vector changes suggest higher manifold detail in fluid particles. Additionally, the visual saliency of particles distant from the viewpoint should be influenced by their viewing distance. Consequently, we introduce a denominator γ_d in manifold feature f_m to adjust saliency based on distance, detailed in Equation 5:

$$f_m(i,\Phi) = \|\nabla\Phi_n(i)\|/\gamma_d \quad , \tag{5}$$

where $\Phi_n(i)$ represents the normal vector at the point corresponding to particle i projected onto the screen's normal vector buffer. $\nabla\Phi_n(i)$ denotes the gradient in its normal buffer neighborhood, which is calculated using a discrete method that examines differences within a neighborhood of five times the particle's radius. The manifold features are illustrated in Figure 3; under the same viewing distance, the spatial variation of the normal vectors around point A is significantly more pronounced compared to point B, which exhibits smoother normal vectors. This reflects a greater manifold feature at point A than that at B.

3.2.2 Temporal Saliency

Inspired by the results verified in CSF [27], where high-frequency objects in a low-frequency spatiotemporal background lead to a surge in visual sensitivity, we consider temporal saliency from two perspectives: temporal changes in physical quantities and particle motion. In PBF, the density ρ_i of particle i is a crucial constraint and process physical quantity that determines the spatial position of the particle in the next frame. Similarly, the particle's velocity $\mathbf{v}_i$ is an essential physical quantity that dictates the advective dynamic behavior of the particle. The more drastic their changes, the more significant the particle's temporal behavior is reflected. Con-

sequently, we define temporal saliency as shown in Equation 6:

$$f_T(i,\Pi,\Phi) = \left(\frac{\|\nabla\rho_i(2h_i)\|}{\rho_i+\varepsilon} + \frac{\|\nabla\mathbf{v}_i(2h_i)\|}{\|\mathbf{v}_i+\varepsilon\|}\right)/\gamma_t \quad , \tag{6}$$

where the gradient is computed within twice the support radius h_i of particle i as described in Equation 5, with a small value ε used to avoid division by zero. This saliency component is modified by a denominator γ_t, influenced by the thickness d_t illustrated in Figure 3 (see the yellow line segment in case 1). The trim factor γ_t is derived using Fresnel's transmission law [37], as detailed in Equation 7:

$$\gamma_t = max\left(e^{\frac{1}{2}d_t}, 5\right) \quad . \tag{7}$$

As the thickness of the fluid cluster increases, the visibility of particles deviating from the viewpoint diminishes exponentially due to occlusion caused by microcluster thickness. The thickness d_t can be computed via Equation 8:

$$d_t(\mathbf{x}_i,\Pi,\Phi) = \Phi_z(i) - \mathbf{x}_i.z - r_i \quad , \tag{8}$$

where $\Phi_z(i)$ represents the depth value of the projection point of particle i on the depth buffer (shown as point Q in Figure 3, with its depth value stored in the buffer at P_1 on the image plane Π), corresponding to the length of the green line segment. $\mathbf{x}_i.z$ represents the depth value of particle i, and r_i represents the radius of particle i. Figure 3 illustrates temporal saliency concept in case 2, where in fluid cluster 2 serving as the background, point C is in a high-density change area within a low-density background, potentially leading to complex temporal behaviors in subsequent frames. Point D is in a high-velocity change area within a low-velocity background, bringing significant visual stimuli.

3.2.3 Saliency Map

Based on the saliency features computed in Equation 1, we normalize these to derive each particle's saliency value s, which guides the new scaling level l as detailed in Equation 9:

$$l_i = \lfloor N(1-s_i) \rfloor, \tag{9}$$

where N is the number of granularity levels in the system. Higher N improves rendering quality in spatiotemporal salient regions but increases particle count, affecting performance. Lower N can cause fragmented results. It is adjustable (see subsection 5.1) based on user's computational resources, balancing efficiency with quality. We then update particles' attributes according to new l, as discussed in subsection 3.3. Figure 4 illustrates the spatial and temporal saliency maps as well as their ablation.

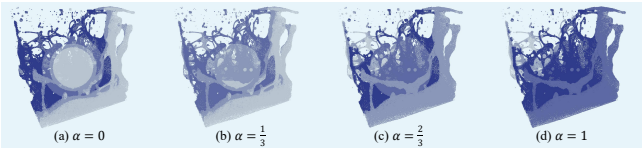


Figure 4: Ablation saliency maps of spatial and temporal features under different α. (a) shows the complete spatial saliency feature map, and (d) shows the complete temporal saliency feature map.

3.3 Legality Updates for Multi-level Particles

To simulate fluids in various salience areas using granular particles, we categorize each particle's scale level l into N levels. As l increases from 0 to $N-1$, the particle radius escalates accordingly. In our granularity scheme, each increment in level doubles the particle's volume and mass. Consequently, to maintain volume conservation, the particle radius must increase by $\sqrt[3]{2}$ each time l increments by one. At the base level $l = 0$, the particle radius is set at r_*, and its mass at m_*. Thus, the radius of particle i at any given level is defined in Equation 10:

$$r_i = \sqrt[3]{2^{l_i}} r_*. \tag{10}$$

Also, the support radius h_i of particle i will be updated in each frame according to the rules described in [1], similar to Equation 10. With the new l calculated from Equation 9, attributes of multi-level particles are updated by splitting particles that transitioned to lower levels and merging those that moved to higher levels.

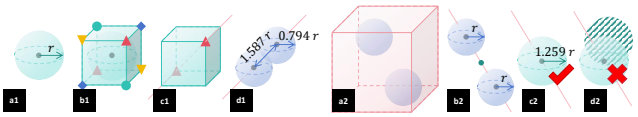


Figure 5: Illustration of the splitting and merging process. (a1) The particle to be split. (b1) Circumscribing cube of the particle. (c1) Sample diagonal line for splitting. (d1) Resulting particles after the split. (a2) Identification of neighboring agent particles with the same level and mass via the neighborhood grid (shown in red). (b2) Particles targeted for merging, with the target mass center marked in green. (c2) Successfully merged particle. (d2) Invalid merged particle overlapping with another larger particle (in green stripes).

3.3.1 Splitting

Algorithm 1: Split particles transitioned to lower levels.

Input: the pre-frame particles set $\mathbf{P}^n$ at the n^{th} frame
Output: the set $\mathbf{P}^n$ after the split update

foreach *particle* $i \in \mathbf{P}^n$ **do**
  if $l_i < m_i/m_*$ **then**
    $k = m_i/m_* - l_i$; `// the level difference`
    Randomly select 2^{k-1} diagonals of the circumscribing cube of particle i ;
    Split along these directions to generate 2^k particles set $\mathbf{i}$;
    $\mathbf{P}^n_{l_i+k} \setminus \{i\}$, $\mathbf{P}^n_{l_i} \leftarrow \mathbf{P}^n_{l_i} \cup \mathbf{i}$;
return $\mathbf{P}^n$;

The main logic of particle splitting is outlined in algorithm 1. Initially, a parallel traversal is performed for each particle i. If a particle requires splitting, the level difference k from current to the target level is calculated. For example, if $k = 2$, the particle undergoes two splits, resulting in $2^k = 4$ new particles. Each split divides a particle into two, maintaining the original center of mass and reducing the radius to 0.794 as Equation 10. To adhere to physical laws and prevent new particles from deviating from the system or overlapping, which could cause excessive repulsion, the distance between the new particles is rearranged to 1.587 times their pre-split radius. The central connecting axes of these particles are randomly arranged along a diagonal line of the circumscribing cube of the original particle, ensuring isotropy and eliminating artifacts. If sufficient splitting occurs, the outer surface of the new particle remains tightly fitted. Figure 5 illustrates the splitting process for $k = 1$; however, for $k > 1$, this algorithm selects 2^{k-1} diagonals for simultaneous splitting.

3.3.2 Merging

The main logic of particle merging is outlined in algorithm 2. Firstly, we perform parallel traversal of each particle i. If merging is necessary, we identify neighboring particles with the same level and mass within the hash grid, ensuring their support radii intersect. Merging is executed for groups of particles, multiplying their quantity by 2^k. The radius and mass of the new particle are recalculated based on Equation 10, positioning it at the combined center of mass. Mergers are rolled back if the new particle has previously been marked as small over the past k frames or if it overlaps with larger particles by support radius, to prevent oscillation and excessive repulsion. Figure 5 demonstrates the merger when $k = 1$. During the splitting and merging processes, substantial memory fragmentation occur. To optimize the efficiency of memory transfers to the GPU, we have implemented a parallel memory

Algorithm 2: Merge particles transitioned to higher levels.

Input: the pre-frame particles set $\mathbf{P}^n$ at the n^{th} frame
Output: the set $\mathbf{P}^n$ after the merge update

foreach *particle* $i \in \mathbf{P}^n$ **do**
  if $l_i > m_i/m_*$ **then**
    $k = l_i - m_i/m_*$; // the level difference
    // find by cuda hash grid
    $\mathbf{j} = findSameNeighbors(i, l_i, m_i, k)$;
    $i^* = merge(i, \mathbf{j})$;
    for $k' : 1 \to 3$ **do**
      if $l_i^{n-k'} - l_i^n > k'$ **or** $\forall_{l_j = l_{i^*}} j \in h(i^*)$ **then** // is invalid merge
        continue ;
      $\mathbf{P}^n_{l_i - k} \setminus (\{i\} \cup \mathbf{j})$, $\mathbf{P}^n_{l_i} \leftarrow \mathbf{P}^n_{l_i} \cup \{i^*\}$;
return $\mathbf{P}^n$;

alignment strategy, detailed in the supplementary materials.

4 Temporal Acceleration

Temporal acceleration enhances efficiency by bypassing computation for less significant particles, allowing them to inherit adjacent spatiotemporal information. This method also pauses their splitting updates to preserve physical properties and control particle counts, thus reducing computational load.

4.1 Perception Threshold for Physical Process Elapsing

The temporal acceleration strategy, detailed in subsection 4.3, mitigates frame rate fluctuations and reduces computation timing in non-critical areas. Before its explanation, we introduce PTPE to determine the optimal timing for implementing this strategy. PTPE evaluates distortions in VR physics simulations caused by temporal fluctuations experienced by users. As demonstrated in Equation 11, activating the acceleration, indicated by the switch tolerance τ, depends on the average frame rate μ and its standard deviation δ. Higher δ values indicate more pronounced fluctuations, increasing the risk of distortions, while higher μ values smooth the process and reduce perceived distortions. We calibrate this model's parameters through pilot user study, enhancing the precision in timing acceleration activation or deactivation. For instance, as outlined in algorithm 3, the strategy is activated when $\tau < 0$.

$$\tau = \mu - a \cdot \exp(b\delta) \quad . \tag{11}$$

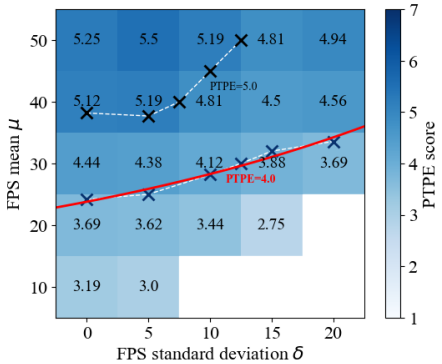


Figure 6: PTPE heatmap of FPS mean and standard deviation.

4.2 Pilot User Study

Study Setting. The *SingleDam* scene was rendered with the highest granularity, adhering to stringent spatial quality standards. The program dynamically refreshed the buffer based on specified frame rates μ and FPS standard deviations δ. Study involved 21 parameter configurations selected from the combinations of frame rates ranging from 10 to 50 (in increments of 10) and standard deviations from 0 to 20 (in increments of 5). Sixteen participants, aged between 20 and 32, observed each simulation setting for 10 seconds. After observing, they rated their satisfaction on a 7-point Likert scale, focusing on real-time step speed and smoothness.

Results. Average PTPE scores for various configurations are shown in Figure 6. Bilinear interpolation was applied to these scores at 4.0 and 5.0, forming corresponding contours. Notably, PTPE@4.0 contours, which represent neutral standards best, were used for exponential regression (Equation 11), resulting in $a = 23.68$ and $b = 0.02$. Consequently, the switching tolerance formula under PTPE@4.0 is represented as $\tau = \mu - 23.68 \cdot \exp(0.02\delta)$, optimizing acceleration strategy control in applications.

Algorithm 3: ViP-Fluid solver with temporal acceleration.

Input: the pre-frame particles set $\mathbf{P}^n$, velocity grid map $\mathbf{V}^n$, kinetic energy grid map $\mathbf{E}_k^n$, and PTPE tolerance τ
Output: the post-frame particles set $\mathbf{P}^{n+1}$, the post-frame velocity buffer $\mathbf{V}^{n+1}$, and the post-frame kinetic energy grid map $\mathbf{E}_k^{n+1}$

if $\tau \geq 0$ **then**
  foreach *particle* $i \in \mathbf{P}^n$ **do**
    $\mathbf{j} = findNeighbors(i)$, $i^* = PBF(i, \mathbf{j})$;
    $\mathbf{V}^{n+1} \leftarrow downsample(\mathbf{v}_{i^*}, \mathbf{V}^n)$;
    $\mathbf{E}_k^{n+1} \leftarrow downsample(m_{i^*}, \mathbf{v}_{i^*}, \mathbf{E}_k^n)$;
    $\mathbf{P}^{n+1} \leftarrow \mathbf{P}^n \setminus \{i\} \cup \{i^*\}$;
  return $\mathbf{P}^{n+1}$, $\mathbf{V}^{n+1}$, $\mathbf{E}_k^{n+1}$;
$\beta = exp(2b\tau)$, $p_i = \beta + (1-\beta) * normalize(F_{pim}(i))$;
sample $\mathbf{P}_{active}$ from $\mathbf{P}^n$ by probability p_i ;
foreach *particle* $i \in \mathbf{P}^n$ **do**
  if $i \in \mathbf{P}_{active}$ **then**
    $\mathbf{j} = findNeighbors(i)$, $i^* = PBF(i, \mathbf{j})$;
    $\mathbf{V}^{n+1} \leftarrow downsample(\mathbf{v}_{i^*}, \mathbf{V}^n)$;
  else
    $i^* = upsample(\mathbf{V}^{n+1})$;
    Pause the splitting behavior of i^* for the next frame.
  $\mathbf{E}_k^{n+1} \leftarrow downsample(m_{i^*}, \mathbf{v}_{i^*}, \mathbf{E}_k^n)$;
  $\mathbf{P}^{n+1} \leftarrow \mathbf{P}^n \setminus \{i\} \cup \{i^*\}$;
return $\mathbf{P}^{n+1}$, $\mathbf{V}^{n+1}$, $\mathbf{E}_k^{n+1}$;

4.3 Acceleration Strategy

we segment the scene into a $1:1000$ scale space based on the finest granularity of particles, typically forming a uniform $10 \times 10 \times 10$ grid to delineate the 3D space. Each grid cell contains the average velocity $\mathbf{V}(l, \mathbf{x})$ at various levels and the kinetic energy $\mathbf{E}_k(\mathbf{x})$ specific to that grid. Leveraging the visual perception saliency s from subsection 3.2, we evaluate the importance of physical attributes using the function $F_{pim}(\cdot)$, outlined in Equation 12. This function correlates a particle's physical importance directly with its mass, suggesting that heavier particles are of higher physical significance weight. Additionally, a particle's physical importance escalates more rapidly as its visual perceptual saliency increases. This assessment involves two main components: intra-level variation f_{in} and cross-level variation f_{cr}:

$$F_{pim}(i, \mathbf{V}, \mathbf{E}_k) = \frac{m_i}{m_*} \cdot \exp(s_i) \cdot (f_{in} + f_{cr}) \quad . \tag{12}$$

At the intra-level of the particle, minimizing simulation errors is critical, particularly in low-velocity backgrounds where significant velocity variations occur (Equation 13). This needs to generate high sampling probabilities in these areas. Moreover, when a cross-level particle displays inconsistent flow phenomena within the same region (Equation 14), it becomes essential to prioritize its sampling and compute an accurate solution to avoid artifact turbulence.

$$f_{in}(i, \mathbf{V}) = \frac{\|\nabla \mathbf{V}(l_i, \mathbf{x}_i)\|}{\|\mathbf{V}(l_i, \mathbf{x}_i)\| + \varepsilon} + \frac{\|\nabla \mathbf{V}(l_i, \mathbf{x}_i)\|}{\|\sum_{j=0}^{N-1} \mathbf{V}(j, \mathbf{x}_i)\| + \varepsilon} \quad . \tag{13}$$

$$f_{cr}(i,\mathbf{V},\mathbf{E}_k) = \quad \|\frac{\partial \mathbf{V}(l_i,\mathbf{x}_i)}{\partial l_i}\| + \frac{\|\nabla \mathbf{E}_k(\mathbf{x}_i)\|}{\mathbf{E}_k(\mathbf{x}_i)+\varepsilon} \quad . \tag{14}$$

The activation sampling probability p_i for particle i is governed by F_{pim} as specified in algorithm 3, with scaling confined between $[\beta, 1]$. Using p_i, we determine whether to activate particle i (activation toggle). If activated, the particle is processed through the PBF solver and its physics properties are updated on the grid for the next frame. If inactive, we directly sample its physical properties from the grid, pausing splitting updates to save computation time and ensure global physical accuracy. Here, $\beta = \exp(2b\tau)$. As τ becomes negative and the acceleration strategy activates, p_i's range compresses, reducing memory transfer, updates, and computational load. For example, $\tau = -1.0$ results in $\beta = 0.96$ and $\tau = -10.0$ results in $\beta = 0.67$, optimizing the sampling probability.

5 Experiments

We evaluated our method across three fluid scenes: *SingleDam*, *JetImpact*, and *Faucet*. The following sections compare it with SOTA methods, assessing quality and performance. Additionally, we conducted two user studies to confirm our method's benefits.

5.1 Implementation

Our approach utilizes an OpenGL-CUDA (C++) architecture and extends a single-granularity PBF solver [19] with multi-granular coupling based on [1]. Rendering is handled through SSF algorithm [39]. Experiments were performed on a PC with an RTX 3080 Ti GPU, Intel i7-10700KF CPU @ 3.80GHz, and 32GB RAM. The FOV angle is set to 15° as in [30]. The finest particles had a base radius $r_* = 0.075m$, calculated using a average viewing distance of 2 to 3 meters and $FPPD = 16.0$ [42]. The rest density was set to $\rho_0 = 1000$. We set α in Equation 1 to $\frac{2}{3}$, and N in Equation 9 to 4.

5.2 Results and Comparison

Figure 7 displays rendering results and visual perception saliency maps for three scenes, sampled at 50-frame intervals. Our method consistently delivers high-quality in foveal region while preserving detail in non-foveal areas like fragments and splashes. Efficiency is optimized by using larger particles deep inside fluid cluster or laminar regions. Notable phenomena include a liquid bridge in *SingleDam*, a turbulent ring in *JetImpact*, and a splash crown with ripples and a central dry spot in *Faucet* due to a high-speed impact.

Table 1: Quality comparison of three methods.

	method	LoRes	FovFluid[42]	Ours	Ours-TSp
SingleDam	PSNR@fov ↑	27.91	29.51	**29.88**	29.84
	PSNR@salient ↑	28.07	29.45	**30.16**	29.73
	PSNR@overall ↑	33.44	33.47	**34.19**	34.11
	SSIM-static ↑	0.72	0.73	**0.76**	**0.76**
	SSIM-cumul ↑	0.62	0.68	**0.71**	0.70
JetImpact	PSNR@fov ↑	29.28	**33.63**	33.04	32.42
	PSNR@salient ↑	30.46	30.70	**31.19**	31.05
	PSNR@overall ↑	32.60	32.93	**33.86**	33.49
	SSIM-static ↑	0.66	0.69	**0.78**	0.76
	SSIM-cumul ↑	0.29	0.53	**0.57**	0.42
Faucet	PSNR@fov ↑	31.89	32.86	**32.90**	32.72
	PSNR@salient ↑	29.19	29.24	**29.37**	29.35
	PSNR@overall ↑	34.03	34.18	**34.36**	34.33
	SSIM-static ↑	0.73	0.74	**0.76**	0.75
	SSIM-cumul ↑	0.34	0.47	**0.54**	0.51

5.2.1 Quality

We compared our methods, Ours and Ours-TSp (Time Speedup, with temporal acceleration), with PBF under low resolution (LoRes) and high resolution (HiRes, as groud truth, similar to the approach in [32, 21, 42]), and FovFluid. Our method focuses on fluid foveated rendering, where FovFluid is the SOTA method exclusively targeting this area. To ensure fairness, considering the fluctuating particle counts in Ours and FovFluid during execution, we standardized the baseline particle number n_* across all methods. This baseline represents the number of particles in each scene

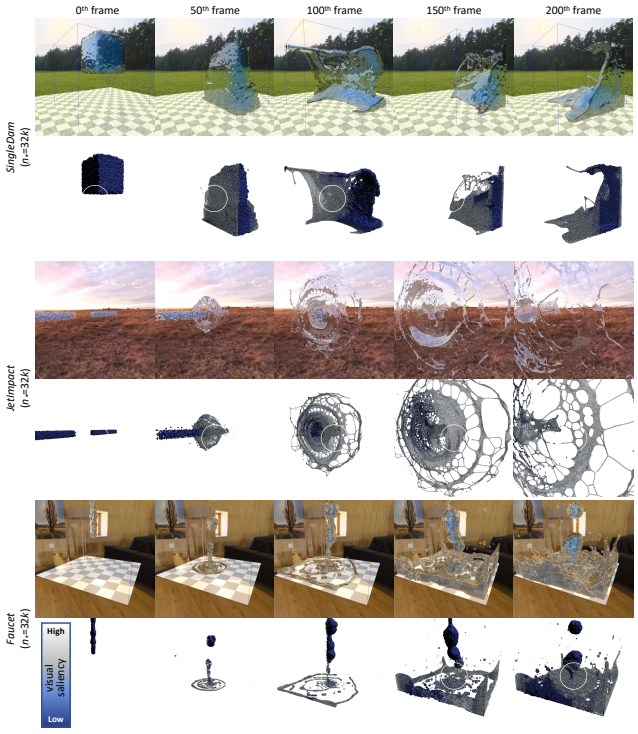


Figure 7: Rendered fluid animations using ViP-Fluid across three scenarios with a baseline of $n_* = 32k$. Each scenario features two rows: the top row shows the rendered outcomes and the bottom row illustrates the visual saliency feature maps of particle swarms.

simulated at the coarsest granularity. For instance, with $n_* = 16k$, our approach assumes all particles are at their coarsest, resulting in $16k$ particles, while in the scenario where every particle splits to the finest granularity, the count would reach $8n_* = 128k$. In contrast, Lo maintains a constant particle count at n_*, and Hi consistently uses $8n_*$. Figure 8 displays a detailed comparison across different methods for three scenes at $n_* = 32k$. Analogous to FovFluid, our method parallels HiRes in foveal detail. However, FovFluid, considering only two particle levels and transition areas, matches LoRes in peripheral quality. Conversely, our method retains finer details in visually salient regions outside the fovea, aligning better with human visual preferences. For instance, in the *SingleDam* scene, our method preserves edge details and tiny droplets; in the *Faucet* scene, it maintains ripples and a more realistically positioned dry spot. LoRes and FovFluid struggle to capture such details in non-foveal regions. Notably, in the Ours-TSp method, due to some particle properties being updated from the grid rather than solved, chaotic behavior emerges over time, with a minority of particles exhibiting convergent motion, leading to some loss of detail. For example, in the *JetImpact* scene, the vortex structure in Ours-TSp is less intact than in Ours, and in the *Faucet* scene, parts of the ripples in Ours-TSp coalesce into droplets, yet the overall physical realism remains plausible. Table 1 showcases each method's strengths and weaknesses through objective metrics, with no existing precedent for VR fluid simulation quality assessment. We evaluate rendering quality using the HiRes method as a benchmark against four other methods with a baseline particle count of $n_* = 32k$. Given the chaotic nature of Lagrangian fluid simulations, minor initial variations can lead to significantly diverse outcomes. To minimize the impact of system chaos, we designate a static checkpoint frame. Then we run each method from that point and analyze the rendering results five frames later across foveated, salient, and overall regions using PSNR. This approach provides a reasonable comparison basis. For structural similarity, we as-

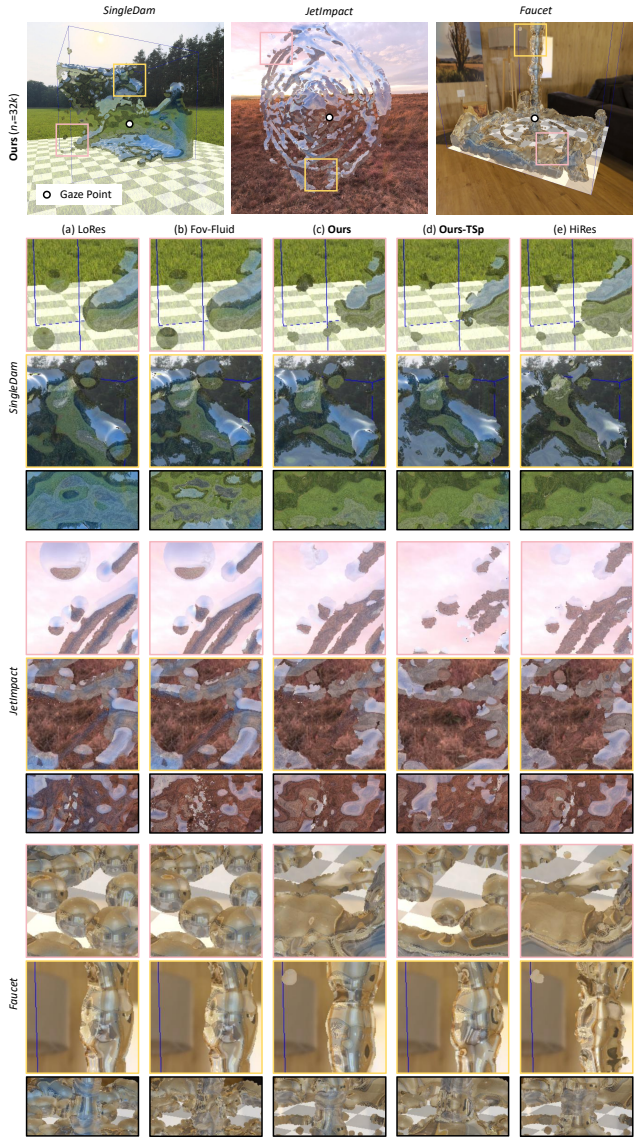


Figure 8: Quality comparison.

sess SSIM to gauge spatial fidelity using static checkpoint cases, and temporal fidelity by cumulatively averaging SSIM over the first 200 frames. Our method exhibits similar PSNR to FovFluid in the foveated region across three scenes. In other salient regions and overall, our method performs better than both the low-resolution PBF and FovFluid. Since the fluid background is not affected by rendering quality, the overall PSNR may be elevated due to blank areas, but the relative magnitudes among the methods remain consistent. The SSIM of our method is superior to the other methods, and the cumulative SSIM tends to decline due to the chaotic system. Except for the complex vortices in *JetImpact*, the accelerated strategies in other scenes are comparable to the naive strategy.

5.2.2 Performance

We evaluated each method's performance at various baseline granularities to assess VR simulation and rendering efficiency. Table 2 details runtime particle counts, timing, and speed-ups. We also present runtime metrics τ, modeled by PTPE@4.0 in subsection 4.2, to validate our acceleration strategy. Across different scenes and baselines, our method consistently showed slightly higher particle counts than FovFluid, which also explains enhancing detail in salient regions. Nevertheless, our method's speed-up aligns closely with FovFluid, particularly noticeable in high-load HiRes scenarios with large baselines. For instance, in *Faucet*@32k, speed-ups exceeded 2x, and despite the time spent managing vortex rings in *JetImpact*, a 1.34x speed-up was maintained. Our acceleration strategy also significantly improved PTPE perception, enhancing the users' perceptual quality. For instance, in the *SingleDam*@32k scene, after applying our acceleration strategy, τ increased from negative to 1.25, and improvements were also observed in other scenes and baselines.

5.3 User Study

To further validate the subjective visual perception of our method, we conducted a series of user studies. The host machine used in these studies matches the PC configuration in subsection 5.1, and is connected to an XR HMD with eye tracking for near-eye display.

5.3.1 Hypotheses

To address methodological differences, we propose the following hypotheses to guide user study designs and test our assumptions. **H1**: Our method exhibits superior performance compared to FovFluid in various scenes and baseline configurations. **H2**: Our method achieves visual effects comparable to HiRes in various scenes and baseline configurations. **H3**: Our acceleration strategy did not degrade visual perception of our naive strategy. **H4**: Users will prefer the acceleration strategy. **H5**: When faced with FovFluid and our method, users will prefer our method. **H6**: When faced with HiRes and our method, users will tend to choose HiRes.

5.3.2 User Study Design

Participants. We recruited 26 participants, with an age range of 16 to 35 years, including 10 females and 16 males. Among them, 5 participants had no experience using XR devices. All participants had normal or corrected-to-normal vision.

Conditions. The studies set up five comparison conditions, which are the low-resolution (LoRes) and high-resolution (HiRes) settings of the PBF solver, the classic FovFluid[42] method, and our method's naive strategy (Ours) and temporal speed-up strategy (Ours-TSp). The parameter settings for each method are consistent with those mentioned in subsection 5.1.

Task 1 (T1). The participants will be randomly assigned to two scenarios, and the simulation rendering granularity for the two scenarios will be randomly set to $n_* = 16k$ or $n_* = 32k$. In each scenario, the five methods will be rendered in a random order, with each method lasting for 30 seconds. Afterward, the participants will comprehensively report their sense of realism based on visual experience, including rendering efficiency, quality, and physical fidelity, using a 7-point Likert scale rating test. They will also complete a Cybersickness report based on spatial disorientation, unrealism, dizziness, and confusion using a 4-point Likert scale. After completing the report, participants will rest for 30 seconds before proceeding to the next round with a randomly selected method. Each two scenarios will have an additional 2-minute break in between, so each participant will report $2 \times 5 = 10$ subjective scores in total.

Task 2 (T2). Participants will be assigned randomly to scenarios with $n_* = 32k$. The five conditions will be paired into ten combinations, displayed in random order without regard to sequence within or between pairs. Each pair will be separated by a 1-minute interval, with each method shown for 30 seconds followed by a 5-second black screen. Participants will make forced choices on spatiotemporal quality and performance for each pair using the Two-Alternative Forced Choice (2AFC) method.

5.3.3 Results and Discussion

Realistic and Cybersickness Evaluation. The result distributions of the two reports in T1 are reflected in Figure 9. To validate

Table 2: Performance comparison of three methods.

	baseline benchmark	$n_* = 16k$ PBF-HiRes@128k					$n_* = 32k$ PBF-HiRes@256k				
	method	LoRes	FovFluid[42]	Ours	Ours-TSp	HiRes	LoRes	FovFluid[42]	Ours	Ours-TSp	HiRes
SingleDam	runtime num. ↓	16k	53.2±18.1k	72.6±27.7k	71.8±27.0k	128k	32k	80.6±42.6k	111.5±52.1k	109.8±48.6k	256k
	time (ms) ↓	18.1	30.6	31.2	30.9	36.6	19.8	31.3	39.7	33.4	52.1
	fps stddev. δ ↓	6.4	11.2	13.2	11.0	7.2	7.9	10.6	14.1	9.6	4.5
	PTPE@4.0 τ ↑	28.34	3.05	1.22	2.86	*-0.03*	22.77	2.68	*-6.21*	1.25	*-6.72*
	speed-up ↑	× 2.02	× 1.20	× 1.17	× 1.18		× 2.63	× 1.66	× 1.31	× 1.56	
JetImpact	runtime num. ↓	16k	51.2±18.8k	88.4±21.3k	84.8±18.1k	128k	32k	93.5±31.0k	159.6±38.0k	153.0±31.1k	256k
	time (ms) ↓	18.2	25.8	32.4	31.2	40.1	19.7	33.6	48.6	44.7	59.7
	fps stddev. δ ↓	7.6	9.6	8.8	8.6	6.3	8.6	9.1	7.3	6.8	4.8
	PTPE@4.0 τ ↑	27.38	10.07	2.63	3.93	*-1.92*	22.64	1.36	*-6.83*	*-4.76*	*-9.32*
	speed-up ↑	× 2.20	× 1.55	× 1.24	× 1.29		× 3.03	× 1.78	× 1.23	× 1.34	
Faucet	runtime num. ↓	16k	24.8±5.4k	38.5±4.5k	36.5±4.6k	128k	32k	44.7±10.3k	61.6±6.3k	62.0±5.9k	256k
	time (ms) ↓	17.6	20.3	22.2	20.5	31.3	18.9	20.8	23.7	22.7	48.8
	fps stddev. δ ↓	5.7	8.1	9.2	7.2	10.4	9.3	10.1	11.9	10.2	5.8
	PTPE@4.0 τ ↑	30.28	21.42	16.58	21.43	2.79	24.39	19.10	12.15	15.01	*-6.10*
	speed-up ↑	× 1.78	× 1.54	× 1.41	× 1.53		× 2.58	× 2.35	× 2.06	× 2.15	

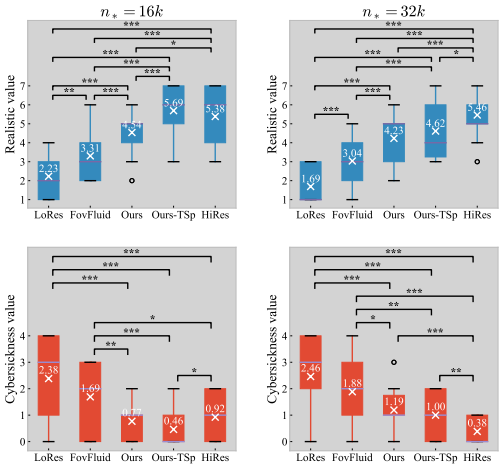


Figure 9: Realistic and cybersickness evaluation.

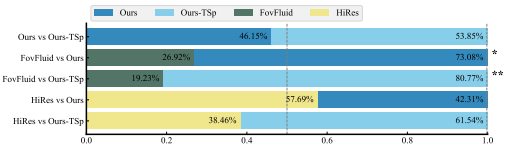


Figure 10: 2AFC results for chosen proportions of method groups.

hypotheses **H1**, **H2**, and **H3**, we conducted t-tests within the baselines to compare the statistical differences between the methods. The significant differences between conditions are marked with asterisks (*), with $p < 0.05^*$, $p < 0.01^{**}$, and $p < 0.001^{***}$. The mean report value for each condition is marked with an X. Our method (including Ours-TSp) shows significant differences in realism and cybersickness compared to FovFluid across different baselines, with an increase in the mean value. This confirms our hypothesis **H1**. Our naive method showed significant differences in realism compared to HiRes, but when $n_* = 16k$, our acceleration strategy mitigated the temporal significant difference, and even outperformed HiRes in terms of realism. However, when $n_* = 32k$, our acceleration strategy resulted in noticeable convergence phenomena, leading to slightly lower realism compared to HiRes, with the increase in cybersickness shown in Figure 9. Therefore, there is no conclusive evidence to fully support **H2**, but in low baseline scenarios, Ours-TSp performs comparably to or even better than HiRes. Apart from HiRes, our method outperforms all other methods across any baseline and scenario. For **H3**, neither set of figures shows a deterioration in visual perception metrics for Ours-TSp compared to Ours. In fact, for $n_* = 16k$, the acceleration strategy even enhanced temporal realism. Therefore, **H3** holds true.

User Preference. The T2 2AFC experiment results, displayed in Figure 10, included five groups relevant to this study, represented as selection rates. We conducted binomial tests to verify significant differences in user preferences, marking results with asterisks on the bars ($p < 0.05^*$, $p < 0.01^{**}$). In the first group, the accelerated strategy (Ours-TSp) was marginally favored over the naive (Ours), though not significantly enough to confirm hypothesis **H4**. However, this suggests that our acceleration strategy effectively preserves quality. Against FovFluid, our method, particularly Ours-TSp, was preferred by over 80% of users, significantly differing from FovFluid ($p = 0.029$, $p = 0.002$), thus supporting hypothesis **H5**. Our naive method approached almost around 50% preference, nearly equal to HiRes, but our accelerated strategy surpassed HiRes, highlighting the importance of temporal continuity in VR simulation. No significant difference confirmed hypothesis **H6**.

6 Conclusion, Limitations and Future Work

Conclusion. To reconcile quality with efficiency in VR fluid rendering, we introduced ViP-Fluid, driven by visual perception for multi-granularity particle simulation and rendering. This approach utilizes spatiotemporal visual saliency to guide particle dynamics and employs a PTPE-guided temporal acceleration strategy. Objective experiments showcased ViP-Fluid's superior performance in foveated, salient, and overall metrics, achieving up to 2.15 times speed-up over high-resolution PBF benchmarks. User studies further confirmed its visual advantages and high preference, underscoring the effectiveness and user satisfaction with our approach.

Limitations. Our method is suitable for simple boundary conditions and does not account for complex fluid-solid interactions or the physical interactions between humans and fluid systems, overlooking the influence of external object motion in complex scenarios. In terms of rendering, due to the use of screen space fluid, aspects such as lighting, refraction, and caustic effects are not thoroughly unbiased, leaving gaps in the analysis of user perception of luminance changes. Our method does not target large-scale scenarios like oceans. Increased particle numbers and environment scale need further study for macro-scale perceptual differences.

Future Work. Our method preserved more details in areas of visual perception interest and provided guidance for future XR simulation. This paper's simulation only considered pure fluid interaction or simple boundary coupling, leaving the challenge of fluid-solid coupling in VR simulation to be addressed. Future research directions can leverage the different visual perceptions brought by solids and scene movements to conduct more complex quality and efficiency balance studies. Similarly, temporal coupling with intelligent, data-driven methods can also be considered. VR applications can continuously explore large-scale open-water rendering and the physical interactions between humans or avatars and fluids.

Acknowledgments

This work is supported by the National Natural Science Foundation of China through Projects 61932003 and 62372026, Beijing Science and Technology Plan Project Z221100007722004, and the National Key R&D plan 2019YFC1521102.

REFERENCES

[1] B. Adams, M. Pauly, R. Keiser, and L. J. Guibas. Adaptively sampled particle fluids. In *ACM SIGGRAPH 2007 papers*, pp. 48–es. 2007. 2, 4, 6

[2] R. Ando, N. Thürey, and C. Wojtan. Highly adaptive liquid simulations on tetrahedral meshes. *ACM Transactions on Graphics (TOG)*, 32(4):1–10, 2013. 2

[3] K. E. Boettcher and A. S. Behr. Using virtual reality for teaching the derivation of conservation laws in fluid mechanics. *Int. J. Eng. Pedagog.*, 11(4):42–57, 2021. 2

[4] R. Bridson. *Fluid simulation for computer graphics*. AK Peters/CRC Press, 2015. 2

[5] Q. Chen, Y. Wang, H. Wang, and X. Yang. Data-driven simulation in fluids animation: A survey. *Virtual Reality & Intelligent Hardware*, 3(2):87–104, 2021. 2

[6] W. Chen, T. Sang, Y. Ma, Q. Chen, Y. Xiao, Z. Pan, and X. Yang. Real-time simulation of violent boiling in concentrated sulfuric acid dilution. *The Visual Computer*, 37:2631–2642, 2021. 2

[7] 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. 2

[8] R. Costa and J. Quarles. 3d interaction with virtual objects in real water. In *2019 11th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games)*, pp. 1–7. IEEE, 2019. 2

[9] H. Deng, J. Li, Y. Gao, X. Liang, H. Wu, and A. Hao. Phyvr: Physics-based multi-material and free-hand interaction in vr. In *2023 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 454–462. IEEE, 2023. 2

[10] F. Ferstl, R. Ando, C. Wojtan, R. Westermann, and N. Thuerey. Narrow band flip for liquid simulations. In *Computer Graphics Forum*, vol. 35, pp. 225–232. Wiley Online Library, 2016. 2

[11] C. Fu, Q. Guo, T. Gast, C. Jiang, and J. Teran. A polynomial particle-in-cell method. *ACM Transactions on Graphics (TOG)*, 36(6):1–12, 2017. 2

[12] Y. Han and P. A. Cundall. Lbm–dem modeling of fluid–solid interaction in porous media. *International Journal for Numerical and Analytical Methods in Geomechanics*, 37(10):1391–1407, 2013. 2

[13] C. J. Horvath and B. Solenthaler. Mass preserving multi-scale sph. *Pixar Technical Memo 13–04, Pixar Animation Studios*, 2013. 2

[14] C. Huang, J. Zhu, H. Sun, and E. Wu. Parallel-optimizing sph fluid simulation for realistic vr environments. *Computer Animation and Virtual Worlds*, 26(1):43–54, 2015. 2

[15] M. Ihmsen, J. Orthmann, B. Solenthaler, A. Kolb, and M. Teschner. Sph fluids in computer graphics. 2014. 2

[16] C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin. The affine particle-in-cell method. *ACM Transactions on Graphics (TOG)*, 34(4):1–10, 2015. 2

[17] L. C. Loschky, G. W. McConkie, J. Yang, and M. E. Miller. Perceptual effects of a gaze-contingent multi-resolution display based on a model of visual sensitivity. In *the ARL Federated Laboratory 5th Annual Symposium-ADID Consortium Proceedings*, pp. 53–58, 2001. 2

[18] M. Macklin and M. Müller. Position based fluids. *ACM Transactions on Graphics (TOG)*, 32(4):1–12, 2013. 1, 2

[19] naeioi. Pbf-cuda. https://github.com/naeioi/PBF-CUDA, 2022. 6

[20] X. Nie, Y. Hu, X. Shen, and Z. Su. Reconstructing and editing fluids using the adaptive multilayer external force guiding model. *Science China Information Sciences*, 65(11):212102, 2022. 2

[21] M. B. Nielsen and R. Bridson. Spatially adaptive flip fluid simulations in bifrost. In *ACM SIGGRAPH 2016 Talks*, pp. 1–2. 2016. 2, 6

[22] F. Oliveira and A. Paiva. Narrow-band screen-space fluid rendering. In *Computer Graphics Forum*, vol. 41, pp. 82–93. Wiley Online Library, 2022. 2

[23] J. Orthmann and A. Kolb. Temporal blending for adaptive sph. In *Computer Graphics Forum*, vol. 31, pp. 2436–2449. Wiley Online Library, 2012. 2

[24] S. N. Pattanaik, J. A. Ferwerda, M. D. Fairchild, and D. P. Greenberg. A multiscale model of adaptation and spatial vision for realistic image display. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pp. 287–298, 1998. 2

[25] S. Premžoe, T. Tasdizen, J. Bigler, A. Lefohn, and R. T. Whitaker. Particle-based simulation of fluids. In *Computer Graphics Forum*, vol. 22, pp. 401–410. Wiley Online Library, 2003. 2

[26] J. Quarles. Shark punch: A virtual reality game for aquatic rehabilitation. In *2015 IEEE Virtual Reality (VR)*, pp. 265–266. IEEE, 2015. 2

[27] M. Ramasubramanian, S. N. Pattanaik, and D. P. Greenberg. A perceptually based physical error metric for realistic image synthesis. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pp. 73–82, 1999. 3

[28] J. G. Robson. Spatial and temporal contrast-sensitivity functions of the visual system. *Josa*, 56(8):1141–1142, 1966. 2

[29] X. Shi, L. Wang, X. Wei, and L.-Q. Yan. Foveated photon mapping. *IEEE Transactions on Visualization and Computer Graphics*, 27(11):4183–4193, 2021. 3

[30] X. Shi, L. Wang, J. Wu, R. Fan, and A. Hao. Foveated stochastic lightcuts. *IEEE Transactions on Visualization and Computer Graphics*, 28(11):3684–3693, 2022. 6

[31] X. Shi, L. Wang, J. Wu, W. Ke, and C.-T. Lam. Locomotion-aware foveated rendering. In *2023 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*, pp. 471–481. IEEE, 2023. 2

[32] B. Solenthaler and M. Gross. Two-scale particle simulation. In *ACM SIGGRAPH 2011 papers*, pp. 1–8. 2011. 2, 6

[33] J. Stam. Stable fluids. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pp. 779–786. 2023. 2

[34] M. Stengel, S. Grogorick, M. Eisemann, and M. Magnor. Adaptive image-space sampling for gaze-contingent real-time rendering. In *Computer Graphics Forum*, vol. 35, pp. 129–139. Wiley Online Library, 2016. 2

[35] M. Stengel, S. Grogorick, M. Eisemann, and M. Magnor. Adaptive image-space sampling for gaze-contingent real-time rendering. Poster @ German Conference on Pattern Recognition 2016, Sep 2016. 3

[36] A. Stomakhin, C. Schroeder, L. Chai, J. Teran, and A. Selle. A material point method for snow simulation. *ACM Transactions on Graphics (TOG)*, 32(4):1–10, 2013. 2

[37] F. R. Tangherlini. Particle approach to the fresnel coefficients. *Physical Review A*, 12(1):139, 1975. 4

[38] N. Truong and C. Yuksel. A narrow-range filter for screen-space fluid rendering. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 1(1):1–15, 2018. 2

[39] W. J. van der Laan, S. Green, and M. Sainz. Screen space fluid rendering with curvature flow. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, pp. 91–98, 2009. 2, 6

[40] D. Violeau and R. Issa. Numerical modelling of complex turbulent free-surface flows with the sph method: an overview. *International Journal for Numerical Methods in Fluids*, 53(2):277–304, 2007. 2

[41] L. Wang, X. Shi, and Y. Liu. Foveated rendering: A state-of-the-art survey. *Computational Visual Media*, 9(2):195–228, 2023. 2

[42] Y. Wang, Y. Zhang, X. Yang, H. Wang, D. Liu, and X. Yang. Foveated fluid animation in virtual reality. In *2024 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*, pp. 535–545. IEEE, 2024. 1, 2, 3, 6, 7, 8

[43] F. W. Weymouth. Visual sensory units and the minimal angle of resolution. *American journal of ophthalmology*, 1958. 2

[44] G. Yan, Z. Chen, J. Yang, and H. Wang. Interactive liquid splash modeling by user sketches. *ACM Transactions on Graphics (TOG)*, 39(6):1–13, 2020. 2

[45] F. Zhang, Q. Wei, and L. Xu. An fast simulation tool for fluid animation in vr application based on gpus. *Multimedia Tools and Applications*, 79:16683–16706, 2020. 2