



清华大学

综合论文训练

基于 GPU 的大规模流体粒子 3D 实 景可视化方法研究

系 别 : 未央书院

专 业 : 数理基础科学 + 土木水利与海洋工程

姓 名 : 卢子期

指导教师 : 徐文杰 副教授

二〇二五年六月

关于论文使用授权的说明

本人完全了解清华大学有关保留、使用综合论文训练论文的规定，即：学校有权保留论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。

作者签名：尹子期

日期：2025.6.13

导师签名：徐进

日期：2025.6.13

摘要

随着数值仿真研究的发展，岩土、水利等领域对流体仿真规模的需求不断提高，这些领域需要处理大规模、高精度的流体仿真数据，但传统可视化技术存在效率低、直观性真实程度差等瓶颈。当下新兴技术如 VR/AR 的广阔应用前景也对实景流体可视化技术的发展提出了要求。

因此，本研究聚焦于大规模流体粒子数据的 3D 实景可视化技术，实现了亿级粒子数据的快速、可交互级别的可视化。本研究首先通过对输入数据特性的深入观察，将连续细节层次技术引入到流体粒子数据领域以减少性能的浪费；同时通过对硬件性能的监测，智能提前加载流体动画的未来一定帧以提高视频渲染的流畅度。数据以研究的核心部分基于 VTK 的 OpenGL 扩展中的 vtkOpenGLFluidMapper 类进行开发，运用粒子数据的连续细节层次技术和未来帧提前加载传输优化技术，开发了一种高效的流体粒子数据实景可视化方法，并实现了一个原型系统以验证算法的可行性和优越性。研究成功在不对视觉效果妥协的前提下提高了实景可视化的效率。

关键词：真实性渲染；流体可视化；连续细节层次；超大规模流体仿真

Abstract

With the progress of computer-aided simulation research, there's an increasing demand for large-scale 3D fluid simulation in geotechnical and hydraulic fields. These fields require processing large-scale, high-precision fluid simulation data. However, traditional visualization technologies have limitations including low efficiency, poor reality and lack of intuitiveness. At the same time, the promising application prospect of currently popping technologies such as Virtual Reality and Augmented Reality require development of photo-realistic scene fluid visualization.

Therefore, this study concentrates on the photo-realistic scene visualization of large-scale fluid particle data, aiming to rapidly and interactively visualize particle data at around hundred-million level. This study first introduces the continuous level of detail technique into the field of fluid particle data through in-depth observation of input data characteristics to reduce performance waste; At the same time, by monitoring hardware performance, intelligent preloading of future frames of fluid animation can improve the smoothness of video rendering. The core of this study is to develop on the vtkOpenGLFluidMapper class in VTK's OpenGL extension. By applying the continuous level-of-detail technology and the future frame pre-loading and transmission optimization technique, it establishes an efficient fluid particle data scene visualization method, and implemented a prototype system to validate the feasibility and superiority of the algorithm. The research improves the efficiency of authenticity visualization without significantly compromising visual effects.

Keywords: Photorealistic Rendering; Fluid Visualisation; Continuous Level-of-Details; Large-scale Fluid Simulation

目 录

第 1 章 绪 论.....	1
1.1 课题研究背景	1
1.2 课题研究意义	2
1.3 国内外研究现状	3
1.3.1 表面重建	3
1.3.2 屏幕空间法	4
1.4 技术路线	6
1.5 论文组织架构	6
第 2 章 基本概念阐述.....	8
2.1 真实性渲染	8
2.2 OpenGL 规范	12
2.2.1 OpenGL 规范简述.....	13
2.3 GPU 通用计算	14
2.4 The Visualization Toolkit (VTK).....	15
2.4.1 VTK 的介绍	16
2.4.2 Paraview 的介绍	17
2.4.3 VTK OpenGL 扩展的介绍	17
2.4.4 vtkOpenGLFluidMapper 的介绍	17
第 3 章 大规模流体渲染算法开发.....	21
3.1 核心技术解析	21
3.1.1 continuous Level of Details (cLoD) 技术	21
3.1.2 未来帧提前加载传输优化技术	23
3.2 算法输入与输出	24
3.3 算法设计流程图	25
3.4 算法的取舍与变体	25
3.5 原型开发	27
第 4 章 案例与测试.....	30
4.1 验证性案例	30
4.1.1 实验设置	30
4.1.2 表现对比	30

4.2 实际案例与效果、性能对比	34
第 5 章 总结与展望.....	37
5.1 本文总结	37
5.2 未来展望	38
参考文献.....	39
附录 A 外文资料的书面翻译	42
致 谢.....	77
声 明.....	78

插图清单

图 1.1 技术路线.....	7
图 2.1 Disney Principled BRDF 模型参数对渲染结果的影响 ^[25]	10
图 2.2 屏幕空间渲染技术管线 ^[17]	18
图 3.1 算法流程图.....	26
图 4.1 小球入水案例与 vtkOpenGLFluidMapper 对比	31
图 4.2 小球入水案例与 Paraview 对比.....	32
图 4.3 溃坝案例对比.....	33
图 4.4 三峡箭穿洞案例对比.....	35
图 4.5 三峡箭穿洞案例对比第二视角.....	36

附表清单

表 2.1 vtkOpenGLFluidMapper 中定制渲染输出的选项 ^[46]	20
表 4.1 本研究测试使用的计算机配置表.....	30
表 4.2 小球入水案例性能对比.....	32
表 4.3 溃坝案例性能对比.....	34
表 4.4 三峡箭穿洞案例性能对比.....	34

符号和缩略语说明

SPH	光滑粒子动力学 (Smoothed Particle Hydrodynamics)
CPU	中央处理器 (Central Processing Unit)
GPU	图形处理器 (Graphic Processing Unit)
GP GPU	通用并行计算 (General-Purpose com-putation on GPU)
FPS	每秒帧数 (Frames Per Second)
LOD	细节层次 (Level-Of-Details)
cLOD	连续细节层次 (continuous Level-Of-Details)
OpenGL	开放图形库 (Open Graphics Library)
VTK	视觉化工具函式库 (the Visualization Toolkit)
PBR	基于物理的渲染 (Physically-Based Rendering)
BRDF	双向反射分布函数 (Bidirectional Reflectance Function)
PCIe	高速串行计算机扩展总线标准 (Peripheral Component Interconnect express)

第1章 绪论

1.1 课题研究背景

随着人类社会的发展，岩土、水利、汽车、工业等诸多领域对3D流体仿真的需求与日俱增。

在岩土工程领域，流体仿真技术可对地下水渗流等岩土体内部流体行为进行数值模拟，预测土体内流体运动特征及结果，在工程方案验证及优化、岩土灾害预警、防治等领域具有重要意义，得到了较为广泛的应用，助力优化工程方案，同时为灾害预防提供科学依据，有效降低因地下水流动异常或岩土体内部流体压力变化引发的工程风险。

在水利工程领域，流体仿真技术被广泛应用于流域演进模拟等方面，在工程方案设计及验证、流域灾害风险评估、水力学研究等领域发挥了重要作用。该技术可以预测水体的流动路径、流速、流量变化等等物理量的大小和作用范围，为防洪减灾决策提供关键数据支持，还能使得水利工程设施设计能够保证未来的安全运行。

在防洪减灾的迫切需求下，研究者需要研究超大尺度的地质灾害，其过程往往极为复杂，受到多种因素的综合影响。开展相关物理试验不仅面临着技术上的巨大难题，如难以精确复现的土体应力应变条件、无法精准测量的地质结构和各种边界情况等，而且面临着高昂的人力物力成本挑战。而数值模拟作为一种计算机技术手段，相较之下有更加低廉的成本，更加灵活的边界设置，和更加可变的模拟尺度，因此能够成为对各种大尺度动力学过程进行深入研究分析的重要途径。

然而，流体仿真结果数据是由空间中的点位置及其影响范围或预先划分的体积网格与网格内部物理参数构成的抽象数字集合，这些数据虽然方便计算机处理，但是难以直观地看到和感受。为了将这些抽象的数据转化为视觉信息，以便于相关群众得到直观了解，也有助于研究人员进行深入分析和准确解释，可视化技术应运而生，成为流体仿真后处理环节中不可或缺的关键环节。经过多年的发展，可视化技术也逐渐发展出多个不同的类型。

目前在具有可视化功能的软件中，数字内容创作软件凭借其先进的算法和强大的光学模拟引擎，能够得到极其真实的图像，从而提供了高质量的视觉体验。但这种高精度的渲染往往需要消耗大量的计算资源，难以满足实时交互和快速分析的需求。而科学计算及可视化软件虽然能够通过将物理量范围映射到色彩空间、从流场提取流线、流管等可视化方式，实现实时交互的效果，帮助研究人员快速把

握模拟结果的基本特征，但其呈现方式相对较为单一，通常缺乏真实感渲染功能，导致普通用户在直观感受仿真结果时存在一定困难。

对于大规模、高精度的流体仿真，由于其数据量庞大和计算复杂度高，如果渲染这样大规模场景以提供后续的分析，当前技术通常耗时较长，无法满足实时的标准。因此，如何提高流体技术的效率，使其既能满足大规模、高精度仿真的需求，又能实现快速可视化以支持流畅的交互，成为了当前流体渲染领域亟待解决的重要问题。

1.2 课题研究意义

随着计算机性能的进步，当下流体仿真在多个领域应用广泛，仿真产生的海量的流体数据，往往达到千万级甚至亿级粒子规模。然而，传统可视化技术在处理大规模流体数据时存在诸多瓶颈，难以满足当前对大规模流体数据进行交互式预览的迫切需求。本研究聚焦于大规模流体数据的可视化，致力于实现的千万级粒子数据的快速渲染算法对推动相关领域的技术进步和实际应用具有极为重要的意义。

在基础研究层面，大规模流体数据可视化技术的研究将拓展计算机图形学的边界。目前，图形渲染技术在面对大规模数据时仍有诸多局限，如数据加载速度慢、渲染效率低等。本研究通过开发高效的可视化算法技术，提升大规模流体数据的可视化效率，减少数据处理和渲染的时间成本，从而在一定程度上扩展了流体数据可视化技术的应用范畴，为流体数据可视化技术的进一步发展的方向如应用到移动设备、VR/AR 设备上等增添了可行性。

此外，本研究还将提升工程设计和决策的水平。借助该技术，研究人员和工程技术人员及项目管理人员能够在短时间内获取流体仿真结果的直观信息，形成对于流体运动、灾害风险的直观认识，从而为工程设计和决策提供更加科学、准确的依据。

综上所述，本研究对于提高大规模粒子数据的可视化效率、推动相关领域的技术进步和实际应用均具有重要的理论意义和实践价值。它不仅将直接推动计算机图形学的发展，还将为岩土、水利、汽车、工业等领域解决复杂的工程问题提供有力的技术支持，助力行业的数字化升级。

1.3 国内外研究现状

流体仿真是计算机数值模拟的一个重要部分，多年的研究使得这个领域发展出了不计其数的算法和变体，但大都基于拉格朗日视角或欧拉视角或其混合方法。以下是几种经典算法：

- 欧拉视角：有限体积法（FVM）^[1]，有限差分法（FDM）^[2]，有限元法（FEM）^[3]。
- 拉格朗日视角：光滑粒子流体动力学（SPH）^[4]，半隐式运动粒子法（MPS）^[5]。
- 混合视角：质点网格法（PIC）^[6]，FLIP 法^[7]。
- 介观尺度：格子玻尔兹曼方法（LBM）^[8]。

以上几种算法及其变种几乎能覆盖所有流体仿真的算法。考虑到后续可视化的需求，在这几种算法中仿真得到结果的数据格式基本可以分为两大类型：网格型以及粒子型。网格型的数据不难被用于可视化，因渲染光栅化最初设计出来便是为网格数据量身打造；因此，基于无网格方法生成的粒子数据在后处理中面临着数据读取效率低、渲染技术不匹配等多种问题，快速渲染及实时交互难度较高、效率较低，后处理对仿真结果的呈现能力较差。因此，对粒子数据的后处理方式进行相应研究，具有较高的研究价值。目前，对于粒子数据的处理适配技术主要分为表面重建技术及屏幕空间方法。其中表面重建技术是指从粒子数据中重建出网格型的流体表面数据，回归到网格数据的可视化流程；屏幕空间方法则是跳过网格的相关渲染流程，利用泼溅法（Splatting）渲染为二维图像后再进行后续操作。

1.3.1 表面重建

在流体表面重建算法领域，所有算法的目的都是直接从流体在三维空间中的位置得到表征流体表面的三角形网格模型，该领域的研究者们也长期致力于探索如何借助有限数量的粒子重建出高质量的流体表面。目前，相关研究主要聚焦于粒子采样策略与模糊化算法的优化与创新^[9]，提升技术对不同类型流体的适应性，并提高重建后表面的真实性可信度。表面重建的经典算法是行进立方体算法（Marching Cubes）^[10]。Marching Cubes 算法虽然在表面重建方面非常出色，但其计算量较大，时间复杂度相对较高。对于静态的三维数据来说，这种计算量是可以接受的，因为一旦网格被重建完成，单个模型就可以被多次用于后续分析渲染操作。但对于动态流体模拟而言，流体表面是每时每刻都在变化的，这就需要算法为每一帧都重新构建流体表面，这种频繁的重建操作有较大的延迟，很难保证在很短的时间内完成计算，从而影响可视化的流畅度。

另一种方法是 Metaballs^[11-12]。Bruckner^[11]聚焦于分子动力学模拟产生的大规模时间依赖数据集的可视化问题，提出了一种基于高斯模型的动态分子表面可视

化方法，从而高效可视化由数百万个原子组成的动态分子表面。该方法完全在图像空间操作，利用可见性信息跳过不必要计算，构造基于原子影响球的列表并进行可见性排序和表面相交测试，采用基于距离函数的球面追踪方法及新的距离估计方法确保快速准确找到交点。这样的方法只依赖于粒子位置，因此被应用于流体渲染^[12]。

Xiangyun Xiao^[13]结合粒子溅射、光线投射和主成分分析(PCA)技术，在GPU上实现高效的提取表面算法。算法通过粒子溅射确定光线投射的入口点，利用SPH密度估计进行等值面提取，并采用PCA估计表面法线方向，以生成平滑且保细节的流体表面。实验表明，该方法能处理数百万粒子，在不同分辨率下均能保持实时渲染性能。

俞铭琪^[14]结合明暗度恢复形状(SFS)和Stokes波模型，首先计算流体表面的法向量，再确定流体表面高度，最后通过二维正态分布曲面对重建表面进行拟合，并生成水花飞溅效果以增强真实感。实验结果表明，该方法的效率能够满足实时性要求，同时重建的流体具有较强的真实感。

刘艳^[15]提出了一种基于八叉树的自适应流体表面重建方法，实验验证表明，该方法能有效重建流体表面，捕获流体细节特征，并适用于大规模粒子流体场景；该方法的贡献在于通过自适应剖分和局部距离场计算，显著降低了内存消耗和计算复杂度。

宋吉虎^[9]运用双向光线步进算法重建流体表面深度和法线信息，并基于延迟渲染管线进行渲染。

何学洋^[16]提出了基于屏幕空间的流体表面重建方法，通过考虑粒子邻域方向上的密度分布不均的情况，使用基于邻域的各向异性椭球进行流体表面重建，提升了表面平滑程度，实验表明改进的流体表面重建方法降低了局部法线的标准差，提升了渲染速度和表面平滑程度。

1.3.2 屏幕空间法

屏幕空间法以van der Laan的工作^[17]为基石。该文献关注基于粒子的流体的实时渲染问题，成功提出一种简单易实现、具有实时性能且能在速度与质量间完成灵活权衡的渲染方法。针对游戏等交互场景中SPH得到的基于粒子的流体模拟方法面临的表面提取难题，论文绕过这一步骤，提出基于屏幕空间的流体渲染方案，采用曲率流方法平滑流体表面，防止流体呈现果冻状外观，同时利用Perlin噪声为流体表面增添细节，模拟泡沫效果。有多项工作基于此进行了更深入的探索。

Nghia Truong^[18]开发了一种简单、高效的滤波器，以改善流体表面的平滑度，同时保持了流体原有的边界。具体来说，作者提出了一种窄范围滤波器，通过限

制深度范围来在范围内平滑深度图，并巧妙处理范围外的深度像素。该方法还结合了偏置校正方法和动态范围调整，以适应不同相机角度和不同斜率的流体表面。

Felipe Oliveira^[19]开发一种新的屏幕空间流体渲染技术——仅对液体表面的一条窄条带上的流体粒子进行粒子过滤，以提供平滑的液体表面和体渲染效果。作者引入了一种新的边界检测方法，允许用户从自由表面来生成粒子层，并通过粒子层的剥离生成窄带。该方法简化了粒子计算，降低了内存消耗，并提高了渲染效率。实验表明，与其他屏幕空间渲染方法相比，该方法在性能上有了显著提升，且在使用不同的过滤器时均表现出色。

Yanrui Xu^[20]通过变换拉伸点精灵为各向异性的椭球，结合加权主成分分析后的粒子分布，以生成平滑流体表面，并与流行的屏幕空间滤波器结合处理深度信息。实验表明，该方法能有效解决之前方法中存在的锯齿边缘和表面不平整问题，同时保留高频细节。与各向同性的处理相比，各向异性算法使流体表面渲染更平滑，光照阴影更真实，尤其在流体-刚体耦合场景中效果显著。

Yalan Zhang^[21]结合相分数纹理区分不同性质的流体，实现复杂混相与分离效果的真实呈现，并将纹理计算优化方法融入实时渲染流程，最终提升多相流体模拟的视觉效果与实时渲染效率。

还有多位作者扩展了屏幕空间方法。

Caio José^[22]利用了类似的表面平滑、法线估计的方法提出了基于光线追踪的渲染解决方案。该方案能集成至任意光线追踪引擎，保留锐利特征，实现基于光线追踪的反射与折射，并以交互速率渲染大量粒子，如在 1920×1920 分辨率下 200 万粒子可达 2.7fps (frame per second, 每秒渲染帧数)。实验结果表明，与传统方法相比，该方法在渲染质量与性能间取得平衡，且适用于多种粒子基方法。然而，其局限性在于仅能渲染流体单层，无法呈现流体内部体积，且在动态场景中因需更新加速结构，性能有所下降。

Xi Duan^[23]提出一种基于相机视角的屏幕空间细节级别策略和虚拟相机姿态自适应滤波策略，以实现开放海域表面的快速渲染解决方案。研究方法结合了 Gerstner 波浪模型、屏幕空间自适应网格生成和自适应滤波策略等三种策略以加速渲染，同时利用 OpenGL 的硬件细分特性实现自适应细分。该方法具有极低的时间成本和相对较高的渲染质量，显著提高了虚拟相机运动的自由度，但只适用于海洋领域。

孙颖君^[24]通过融合游戏引擎和粒子系统，构建滑坡模拟可视化方法。论文还设计了优化方案，通过改进镜像粒子法和双边滤波算法等提升了可视化方法的真实感和效率。实验证明了研究所提出的方法的有效性和可行性，成功实现了滑坡

灾害的三维动态可视化，并通过原型系统对公众的开放测试验证了其科普应用价值。

总体来说，在屏幕空间流体渲染算法中，性能消耗主要受滤波器选择、粒子数量及屏幕分辨率等因素影响。优化工作多着眼于如何降低计算复杂度，同时减少滤波过程中的信息损失，以提升流体表面的真实程度。

1.4 技术路线

本研究的主要目的是开发一种高效的可视化算法，用于大规模（千万级）流体粒子数据的可交互级别（9fps+）的可视化。经过全面的调研和测试，本研究采取技术路线如图 1.1：

1.5 论文组织架构

本论文共分为五个章节。

第一章为引言，介绍了大规模流体粒子数据可视化的研究背景、意义，国内外研究现状，面临的挑战，本论文的主要工作、贡献和技术路线。

第二章为可视化的基本知识简介，包括了渲染的基本原理，常用的渲染技术与优缺点，本文采用的基础技术与开发的新技术。

第三章为算法的介绍，介绍了本研究设计的算法输入输出、流程图，以及考虑到不同应用情境下的取舍和可能变体，最终介绍了研究者用于验证和测试而开发的技术原型。

第四章为案例与测试，包括了两个验证性的案例与一个实际应用案例。验证性案例规模较小，并采用经典的实验设置，用于验证算法的可行性，以及与其他方法的视觉效果和性能对比。实际案例规模较大，采用了实际研究过程中使用到的数据和设置，用于验证算法的优越性。

第五章为本论文的总结，同时分析了本论文工作的优缺点以及未来可能的改进。

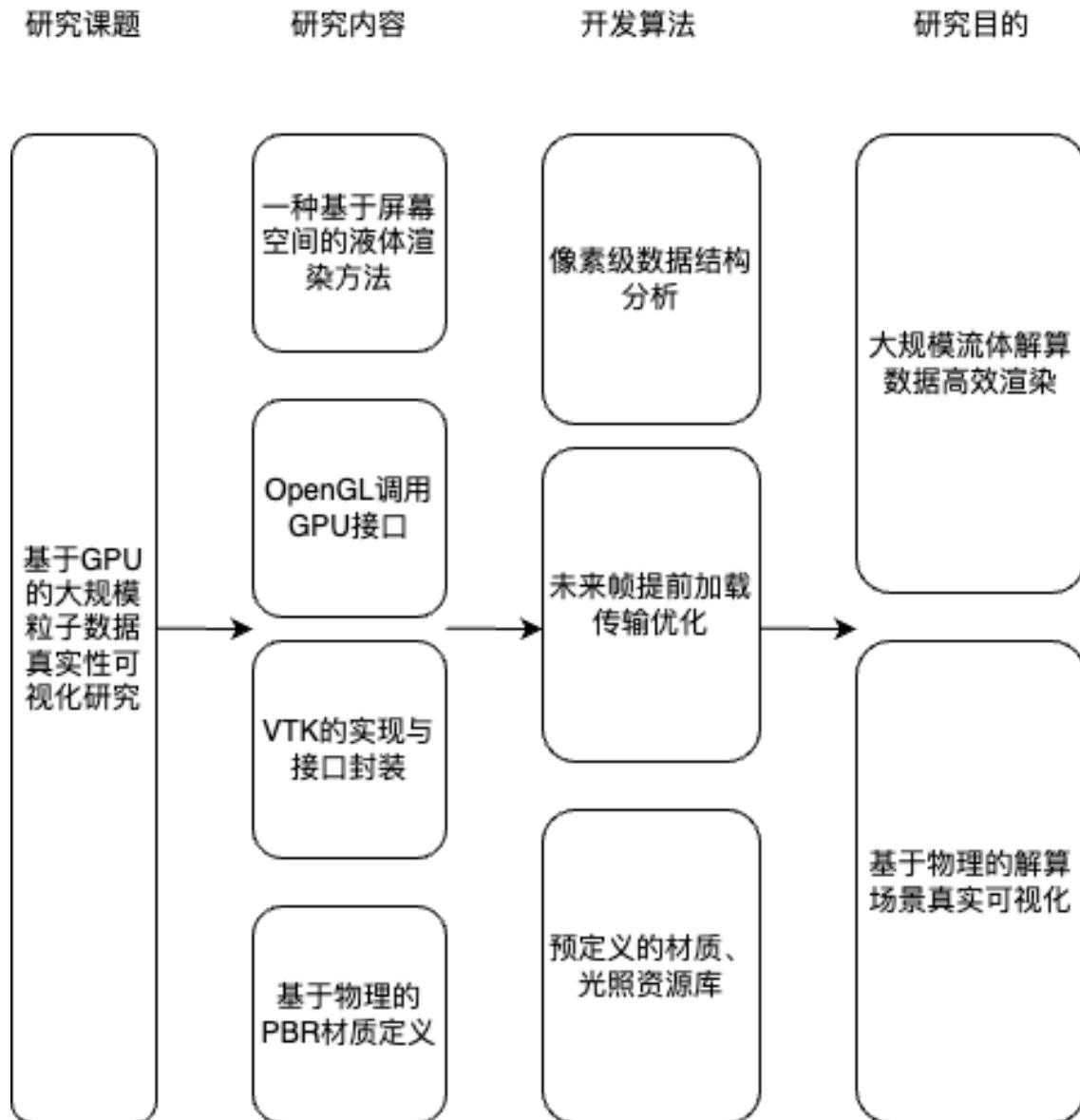


图 1.1 技术路线

第 2 章 基本概念阐述

本研究基于 VTK 的 OpenGL 扩展中的 `vtkOpenGLFluidMapper` 类进行开发，实现了对大规模流体粒子的高效真实性渲染。因此对 OpenGL 和 VTK 渲染作简要介绍。

2.1 真实性渲染

在计算机图形学中，渲染是指基于计算机图形学和光学的理论，通过模拟光在场景中的传播、与物体的相互作用以及最终在图像平面上的成像过程，将三维场景转换为二维图像。

渲染的第一步是场景建模，它包括三维模型构建和场景搭建。三维模型构建是使用专业的三维建模软件，如 Maya、3ds Max 等，通过创建几何体、调整顶点和边等操作，构建出物体的三维模型，确定物体在三维空间中的形状和位置。场景搭建是将多个三维模型放置到合适的场景中，确定它们的相对位置和关系，同时添加光源，设置光源的位置、强度、颜色等参数，以及添加相机并确定视角和视图范围。

在完成三维模型构建后，需要对三维模型进行几何变换及投影。几何变换是通过平移、旋转、缩放等对三维模型进行操作，使其达到合适的位置、方向和尺寸，以构建出预期的场景效果。投影变换则是将三维场景中的物体投影到二维平面上，常见的投影方式有透视投影和平行投影，透视投影模拟人眼观察效果，平行投影则使物体沿投影方向平行地投射到投影平面上，能保持物体的比例和形状。为使二维图像能够表现出三维物体的视觉效果，需要对其进行光照计算，即在场景中设置虚拟光源，并基于此对场景内光照效果进行模拟及计算。

为提高图像视觉效果，使其接近现实物体的视觉特征，可以设置模型的材质并且映射纹理。材质是为物体赋予不同的对光线的反应方式，如颜色、反射率、折射率、透明度、粗糙度等，这些反应方式会影响物体表面的视觉外观，使物体呈现出不同的质感和特性。纹理映射是将二维的纹理图像映射到三维模型的表面，使模型表面具有更加丰富的细节和真实的质感，如木材的纹理、布料的纹理等。

光线追踪与光栅化是目前比较普遍的两种渲染手段。光线追踪是模拟光线在场景中的传播路径，从相机发出光线，根据光的物理规律在场景中反射、折射，最终确定像素的颜色。这种方法能够产生非常逼真的渲染效果，如精确的阴影、反射和折射等，但计算成本较高，通常用于对图像质量要求较高的离线渲染。光线

追踪中可以使用 Optix 来渲染流体^[22]，但是效率过低，因此本研究采用光栅化渲染方法。光栅化是将三维模型的几何信息直接转换为二维图像上的像素，然后根据光照计算和材质纹理等确定每个像素的颜色值。光栅化算法效率较高，是实时渲染中常用的方法。

为进一步提高渲染效果，增加图像的真实性，渲染结果还需进行后期处理，如色彩校正、添加特效与图像合成。色彩校正是对渲染图像的对比度、饱和度、亮度进行调整以达到更美观的整体效果，使图像更加符合视觉审美和艺术要求。添加特效可以增强图像的真实感和艺术感，突出场景中的重点元素或营造出特定的氛围和风格，常见的特效有景深、运动模糊、辉光等。图像合成是将渲染好的图像与其他元素进行合成，如背景、前景、特效等，最终生成完整的图像或动画效果。整个渲染过程通过这些步骤的结合，将三维场景转换为具有真实感和艺术美感的二维图像。

随着硬件的发展和计算能力的增长，现代计算机的能力能够支持更加真实且符合物理的渲染，本研究也通过支持基于物理的渲染（Physical-Based Rendering, PBR）使最终图像效果更加真实。

在计算机图形学领域，基于物理的渲染技术在 2012 年取得了关键性进展。迪士尼动画工作室的 Brent Burley 在 Siggraph2012 上发表了题为《Physically-Based Shading at Disney》^[25] 的演讲，正式公开迪士尼原则 BRDF（Disney Principled BRDF）。这一创新性模型以极高的通用性，使用金属度、粗糙度等少量直观参数，将复杂材质表达出来，引发了电影和游戏行业的广泛关注，并使得以物理为基础的渲染技术正式进入大众视野。

Disney Principled BRDF 模型本质上是混合了金属和非金属的光照模型，它以金属度为指标线性插值金属和非金属的 BRDF 函数值。这一模型通过统一金属与非金属材质的表述，仅需少量参数即可涵盖自然界中绝大多数材质，并实现高度逼真的渲染品质。

在 PBR 的金属 / 粗糙度工作流中，固有色（baseColor）贴图同时包含金属和非金属的材质数据，包括金属的反射率值以及非金属的漫反射颜色。Disney Principled BRDF 的设计哲学是直观而非使用隐晦的物理参数，同时减少参数数量，将参数划分一个合理的值域，且在保证合理的参数组合时允许适当地超出范围。

该 BRDF 模型包含的各种参数以及各种参数对渲染结果的影响在图2.1中显示。

下文是 Disney Principled BRDF 的着色模型实现细节。

核心 BRDF 为微表面 Cook-Torrance BRDF 着色模型^[26]：

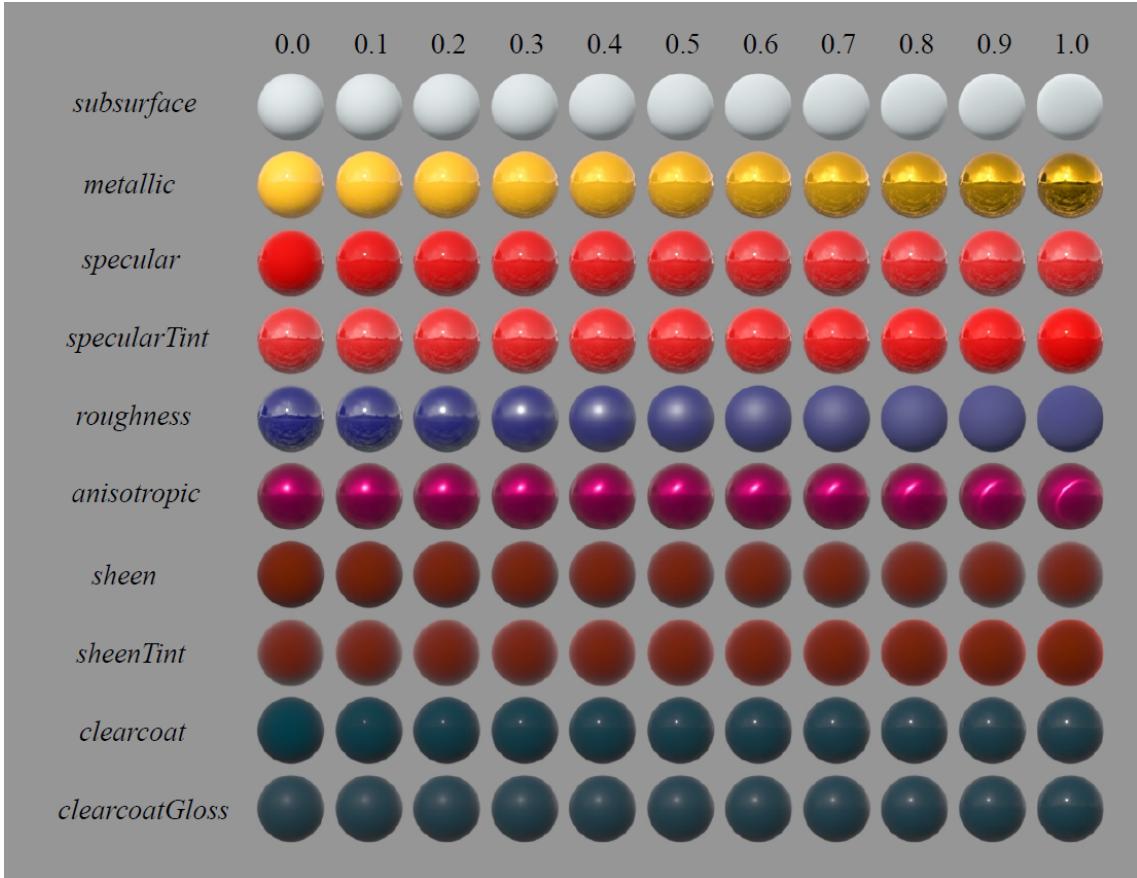


图 2.1 Disney Principled BRDF 模型参数对渲染结果的影响^[25]

$$f(\mathbf{l}, \mathbf{v}) = \text{diffuse} + \frac{D(\theta_h)F(\theta_d)G(\theta_l, \theta_v)}{4 \cos \theta_l \cos \theta_v} \quad (2.1)$$

其中向量 \mathbf{l}, \mathbf{v} 分别表示入射光和视角方向, \mathbf{h} 是 \mathbf{l} 与 \mathbf{v} 的平均向量, 向量 \mathbf{l} 与法线方向的夹角以及向量 \mathbf{v} 与法线方向的夹角分别记为 θ_l 和 θ_v , θ_h 同理。其中, θ_d 是向量 \mathbf{l} 与 \mathbf{v} 之间夹角的一半。在该研究提出的光照模型中, diffuse 表示漫反射函数, 描述物体表面的漫反射特性; D 代表微表面分布函数, 表征微表面朝向的分布情况; F 为菲涅尔系数, 反映物体表面光线的反射特性与入射角等参数之间的关系; G 则为阴影系数, 主要考虑了微表面排列以及光线遮挡等因素对光照效果产生的影响。

Disney 指出, 传统的 Lambertian 漫反射模型在模拟光滑表面边缘时通常过暗。为解决这一点, Disney 曾尝试引入菲涅尔因子, 但导致亮度进一步降低。因此通过深入分析 Merl 100 材料数据库中的材质, Disney 开发出新的基于经验的漫反射模型。该模型采用 Schlick Fresnel 近似, 并对掠射逆反射响应行为进行调整, 使其基于粗糙度参数动态变化而非固定为零。这种设计实现了光滑与粗糙表面漫反射之间的过渡, 公式表达为:

$$f_d = \frac{\text{baseColor}}{\pi} (1 + (F_{D90} - 1)(1 - \cos \theta_l)^5)(1 + (F_{D90} - 1)(1 - \cos \theta_v)^5) \quad (2.2)$$

$$F_{D90} = 0.5 + 2 \cos^2 \theta_d \text{roughness} \quad (2.3)$$

微表面分布函数为 GTR (Generalized-Trowbridge-Reitz) 函数:

$$D_{GTR} = \frac{c}{(\alpha^2 \cos^2 \theta_h + \sin^2 \theta_h)^\gamma} \quad (2.4)$$

其中 c 为缩放常数, α 为粗糙度参数, 取值为 roughness 的平方, 其值在 0 和 1 之间; $\alpha = 0$ 产生完全平滑的分布 (即 $\theta_h = 0$ 时的 delta 函数), $\alpha = 1$ 产生完全粗糙或均匀的分布。

文章作者采用了两个不同的 GTR 函数, 分别为 $\gamma = 1$ 和 $\gamma = 2$ 来拟合高光项。在 Disney Principled BRDF 模型中, 对于镜面反射部分的设计采用了两个固定参数的镜面反射波瓣, 即主波瓣和次级波瓣。这种双波瓣设计通过参数化的微表面分布模型, 实现了对复杂材质反射特性的高效近似。

在菲涅尔项的处理上, Disney 指出, Schlick Fresnel 近似在精度上已足够, 而与完整的菲涅尔方程相比, 其计算复杂度被大大降低。此外, 由于其他因素的影响, Schlick Fresnel 近似所引入的误差明显小于其他误差源产生的误差, 因此选择该近似。Schlick Fresnel 近似公式如下:

$$F_{\text{Schlick}}(\theta_d) = F_0 + (1 - F_0)(1 - \cos \theta_d)^5 \quad (2.5)$$

其中, F_0 是垂直入射时的镜面反射率, θ_d 为半向量 h 和视线向量 v 之间的夹角。

在 SIGGRAPH 2015^[27] 上, Disney 对菲涅尔项的处理进行了修正。Disney 指出, 当介质间的相对折射率接近 1 时, Schlick Fresnel 近似会产生较大的误差。为解决这一问题, Disney 建议在相对折射率接近 1 的情况下, 使用精确的菲涅尔方程来替代 Schlick 近似, 菲涅耳方程如下:

$$F(\theta_i, \eta) = \begin{cases} \frac{1}{2} \left[\left(\frac{\cos \theta_i - \eta \cos \theta_t}{\cos \theta_i + \eta \cos \theta_t} \right)^2 + \left(\frac{\cos \theta_t - \eta \cos \theta_i}{\cos \theta_t + \eta \cos \theta_i} \right)^2 \right] & \text{if } \cos^2 \theta_t > 0 \\ 1 & \text{otherwise} \end{cases} \quad (2.6)$$

在 Disney 的几何项 (Specular G) 处理中, 对于主镜面波瓣 (primary specular

lobe)，其方法基于 Walter 的近似理论，采用 Smith GGX 导出的几何项公式以模拟微表面阴影-遮蔽效应。为避免光泽表面的极端增益，Disney 对粗糙度参数 α 进行了重映射，具体为 $\alpha = (0.5 + \text{roughness}/2)^2$ ，将原始范围从 $[0, 1]$ 调整为 $[0.5, 1]$ 。这种调整使几何项随粗糙度的变化更加平滑，优化了参数的物理合理性，同时提升了美术工作流程的直观性。

对于次级波瓣 (secondary lobe)，Disney 未采用 Smith G 推导，而是直接使用固定粗糙度为 0.25 的 GGX 几何项。这一简化处理在清漆层模拟中被证明能有效平衡计算效率与视觉质量，确保次级反射的几何特性与预期的清漆层外观相符。具体的 Smith GGX 几何项公式如下：

$$G(\mathbf{l}, \mathbf{v}, \mathbf{h}) = G_{GGX}(\mathbf{l})G_{GGX}(\mathbf{v}) \quad (2.7)$$

$$G_{GGX}(\mathbf{v}) = \frac{2(\mathbf{n} \cdot \mathbf{v})}{(\mathbf{n} \cdot \mathbf{v}) + \sqrt{\alpha^2 + (1 - \alpha^2)(\mathbf{n} \cdot \mathbf{v})^2}} \quad (2.8)$$

$$\alpha = (0.5 + \text{roughness}/2)^2 \quad (2.9)$$

其中， θ_i 和 θ_o 分别为入射角与出射角， α 为重映射后的粗糙度参数。这种设计在保证物理精度的同时，显著简化了美术人员的参数调整流程。

Disney 在 SIGGRAPH 2015 上对几何项 (Specular G) 进行了修订^[27]。参考了 Heitz 在 2014 年的分析^[28]，决定淘汰对主镜面波瓣 (primary specular lobe) 的 Smith G 粗糙度的特殊重映射方法 (即之前将 α 从 $[0, 1]$ 重映射到 $[0.5, 1]$ 的做法)。转而采用了 Heitz 的各向异性几何项，这一方法能够更好地处理各向异性材质的微表面阴影和遮蔽效应，更符合物理原理，也无需对粗糙度参数进行特殊重映射，在此则不作介绍。对于次级波瓣 (secondary lobe)，考虑到其主要模拟清漆层反射，Disney 仍沿用固定粗糙度为 0.25 的 GGX G 项，以实现合理且良好的视觉效果，同时简化了计算。

本研究对除了水体以外的物体如地形、岩土块体的渲染使用了 PBR 渲染方法。同时对于没有自带贴图的物体的渲染使用 Poly Haven 提供的免费 PBR 贴图材质^[29]。

2.2 OpenGL 规范

本研究采用 OpenGL 语言为主要编程语言，因此下文对 OpenGL 基本内容与特点作简要介绍。

2.2.1 OpenGL 规范简述

OpenGL (Open Graphics Library, 开放图形库) 是一个各种编程语言均可使用、各种平台均可应用的应用程序编程接口 (API)^[30], 用于渲染 2D 和 3D 图形。它起源于 20 世纪 90 年代初, 由硅图公司 (Silicon Graphics, Inc., SGI) 基于其图形工作站所使用的 Iris GI 图形库开发, 并于 1992 年正式发布 1.0 版本。

OpenGL 应用可在 Windows、macOS、Linux 等多种操作系统上运行, 还可在这些不同平台之间轻松移植; OpenGL 可以充分利用图形硬件的并行计算能力, 实现高效的图形运算和渲染; OpenGL 能够绘制从简单图形比特到复杂 3D 场景的各种图形资源, 支持纹理映射、不同光照模型和材质等多种功能; OpenGL 具备灵活的可编程性, 可通过编写着色器程序来定制和用户需求相符合的效果; OpenGL 作为行业标准, 确保了所有开发者编写的代码在不同硬件软件之间都能运行, 这种兼容性使得开发者能够在各种设备上创建一致的图形体验^[16]。

2.2.1.1 OpenGL 相关工具

OpenGL 本质上是一个图形绘制的接口规范, 它仅规定了函数声明而无具体的实现代码, 类似于接口或抽象类。这种设计使得 OpenGL 本身不依赖于特定的硬件或软件平台, 其实际功能实现由显卡生产商负责提供, 例如 NVIDIA^[31] 和 AMD 等公司。它们根据 OpenGL 规范开发自己的图形驱动程序, 实现相应的 OpenGL 函数。这种机制允许生产商针对硬件进行优化, 从而提升图形处理的性能和效率。

OpenGL 的函数库体系较为庞大^[32], 包括核心库、实用库、辅助库、实用工具库、窗口库以及扩展函数库等^[33]。其中, 核心库 gl 是整个 OpenGL 体系的基础, 提供了最核心的图形绘制功能; 实用库 GLU 则封装了部分核心库功能, 提供接口, 使得使用更加简便; 辅助库 aux 和实用工具库 glut 提供了窗口管理和事件处理等额外的辅助功能, 其中功能更为强大的 glut 已经逐步取代了 aux 的地位; 窗口库提供了相应的接口, 适用于不同的操作系统; 扩展函数库可以让硬件厂商通过 OpenGL 的扩展机制, 根据自己硬件的特性来开发额外的功能。

考虑到如此复杂的函数库, 底层开发者们开发了 FreeGLUT, GLEW, GLAD, GLFW 等等高层次的工具包以辅助更多的开发者使用 OpenGL。本研究也受益于这些开源库, 使用 GLAD 进行了更加迅捷的软件开发, 因此在此稍作介绍。

GLUT (OpenGL Utility Toolkit)^[34]是一个提供跨平台窗口管理和事件处理功能的简化 OpenGL 程序开发的工具包。GLUT 的功能涵盖了很多方面, 比如窗口操作, 回调机制, 复杂的 3D 物体创建, 菜单管理和程序操作。然而, GLUT 的最后一个版本可以追溯到 1998 年, 已经相当陈旧, 且该项目已被废弃。

FreeGLUT 是一个开源的 GLUT 替代品，它完全兼容 GLUT 的 API，并在 GLUT 的基础上进行了改进和扩展。FreeGLUT 解决了 GLUT 的许多问题，如支持了 OpenGL 新版本、修复了 GLUT 的大量错误等。

GLFW^[35]是一个轻量级的开源库，专门为 OpenGL 的简洁开发而设计。它提供了一套简单的应用程序接口，用于快速创建窗口、管理 OpenGL 上下文、处理用户输入和事件等等。GLFW 的设计思路是保持轻量化，只关注窗口管理和事件处理等必要的功能，而不涉及图形绘制的具体实现。GLFW 支持多种操作系统，并且提供了对多窗口、多显示器、高 DPI 显示器的支持。它还支持键盘、鼠标、手柄等多种输入设备，并提供了轮询和回调两种事件处理机制。GLFW 的开源特性和 zlib/libpng 许可证使其成为各种商业和非商业项目的可行选择。

GLEW (OpenGL Extension Wrangler Library)^[36]是一个用于加载 OpenGL 扩展函数的库。在 OpenGL 2.0 及以后的版本中，许多功能都以扩展的形式存在——不同显卡厂商会发布针对同厂商显卡的扩展函数，这些函数往往具有独特的功能和极致的优化。GLEW 能够自动检测系统所支持的 OpenGL 扩展，并直接提供相应的函数接口，开发者无需手动查找和包含大量的头文件。

GLAD^[37]则是 GLEW 的升级版，它是一个支持多编程语言的 OpenGL 头文件生成器。与 GLEW 相比，GLAD 提供了更灵活的配置选项和更好的跨平台支持。GLAD 允许开发者根据自己的需求选择特定的 OpenGL 版本和扩展，生成定制化的头文件代码，有助于减少项目的依赖大小，提高加载效率。

考虑到 VTK 通过引入头文件实现与 GLAD 的兼容，无需进行其他操作，本研究使用 GLFW+GLAD 进行软件原型开发。

2.3 GPU 通用计算

本研究对常规图形处理技术的 GPU 计算方式进行优化，采用了基于计算着色器的 GPU 通用计算方式，有效提高了计算效率，对实现大规模流体数据渲染处理具有重要作用。

图形处理单元 (GPU) 最初设计是作为图形渲染领域的专用硬件以加速图形界面这种愈加流行的程序。而随着图形处理单元性能的不断提升，其强大的并行运算能力在非图形渲染领域逐渐得到应用，由此催生了 GPGPU (General-Purpose computation on GPU，通用并行计算) 技术。

GPGPU 技术通过运用图形处理器 (GPU) 的并行处理能力，来执行传统上由中央处理器 (CPU) 负责的，通常与图形处理无关的通用计算任务^[38]，如果数据处理的运算量远大于数据调度和传输的需求，应用 GPGPU 技术的程序将会在性能

上显著超越传统 CPU 应用程序^[39]。

计算着色器（Compute Shader）^[40]作为专为并行计算推出的着色器，在 OpenGL4.3（OpenGL ES 3.1）及之后版本引入，这为开发者提供了在 GPU 上执行通用计算任务的灵活高效手段，是实现 GPGPU 的关键技术之一。它是一种专门用于计算的着色器阶段。

计算着色器的执行模型可以归纳为以下几个步骤：第一步，内存空间等计算资源的分配与配置。接着对计算着色器程序进行编译，然后设定并开始计算着色器、指定线程的编号和布局，计算着色器程序中的指令在这之后由各线程独立执行，最后，根据需要来进行线程间的同步作业和数据的传输。

计算着色器与传统的顶点着色器（Vertex Shader）、片元着色器（Fragment Shader）等存在显著区别。传统着色器主要用于图形渲染管线的特定阶段，分别负责顶点变换和像素着色等任务，且其执行流程与图形渲染紧密相关。而独立于渲染管线、无需用户自定义输入输出、不参与图形的光栅化等处理过程的计算着色器，能够更充分地利用 GPU 的并行计算能力，专注于物理模拟、数据分析、图像处理等各种通用计算任务的执行。

计算着色器运行于 GPU 的独立部分，可创建和管理大量线程，并在这些线程间灵活地进行通信和协调。它充分利用了 GPU 的并行架构优势，可同时执行数千个线程，处理大型的计算任务，适用于深度学习、科学计算等需要大量重复计算的算法，从而显著提高了 GPU 的利用率。

为了充分发挥计算着色器的性能优势，开发者需精心设计算法和数据访问模式。应尽量减少全局内存访问，多利用局部内存或共享内存，以降低内存访问延迟；合理设置工作组大小，根据 GPU 硬件特性及任务需求，选择合适的工作组维度和规模，以提高线程执行效率和资源利用率；优化线程间的通信与同步机制，避免不必要的同步作业，减少线程等待时间；此外，还可采用分而治之等策略，将大规模问题分解为多个子问题并行处理，再综合各子问题的解得到最终结果。

2.4 The Visualization Toolkit (VTK)

数据可视化通过计算机图形学方法，以图像等形式清晰高效地传递和表达信息。随着数据表示方式的多样化及数据规模的不断增长，数据更不容易被人所理解，数据可视化的重要性日益凸显。多年来，数据可视化编程工具包相继出现，其中最具代表性的就是 VTK。

2.4.1 VTK 的介绍

VTK 是一种可获取源代码的、可无损移植到各大硬件平台的关注图形方面应用的计算机函数库。编程者们借助 VTK 可以真实地渲染多学科的科学数据，从而帮助大众明白那些原本以复杂、超大规模的、晦涩的数学形式呈现的科学概念或计算结果。

VTK 由一个 C++ 类基本库及多种编程语言接口层组成，它基于 OpenGL 三维函数库，采用面向对象的设计方法开发，让开发者不用关注图形开发过程中过于细节的一些问题，并提供了一些常用算法的封装（如表面重建中常用的 Marching Cubes 算法）^[41]，用户可直接使用 VTK 提供的 vtkMarchingCubes 类，无需重复编写 Marching Cubes 算法代码以对三维规则点阵数据进行表面重建。

VTK 有大量出色的机制，包括支持各种绘制方式，支持大量现有的图形学专用硬件和高效优化的软件；VTK 在面对超大规模数据时各种大小的内存都可以轻松驾驭，因为 VTK 被设计拥有流处理和缓存架构；VTK 的设备无关性确保了代码的良好可移植性，跨平台特性；支持多种数据类型，同时定义了自己的.vtp/.vtu 数据类型；VTK 在底层图形库（主要为 OpenGL）之上添加了一层抽象层，这一更高层次的抽象简化了可视化任务的创建；VTK 的核心数据模型具备表示几乎所有物理概念的能力，其基本数据结构特别适用于医学成像与工程工作；VTK 使用 MPI 支持了可扩展的分布式内存并行处理，此外还可以通过 vtkSMPTools 和 vtk-m 实现更细致粒度的并行处理。

VTK 采用了能够处理各种类型数据的管道执行机制，用户可根据各种原始数据类型、所用算法及预期结果设计流程。原始数据被转换为算法模块可以直接处理的数据格式，最终得到所需要的视觉化结果^[42]。所有类和算法模块均可扩展，用户可以在系统中集成自己开发的类或模块。这种开放且可扩展的特性是本研究得以顺利进行的保证。

利用 VTK 可视化数据的主要流程如下^[43]：

1. 根据原始数据类型选择相应的类。
2. 过滤器接收数据源中的数据并执行各种过滤操作。
3. 映射将处理后的数据转换为几何数据。
4. 角色在窗口中作为一个实体呈现可视化数据。
5. 角色通过 vtkRender 类在窗口中渲染图像以显示结果。vtkRender 包括多种渲染方法，并依赖相机 vtkCamera，光照 vtkLight 来得到最终图像。
6. vtkRenderWindow 渲染窗口用于展示渲染的图像，同时大多借用 vtkRenderWindowInteractor 窗口交互器和 vtkInteractorObserver（观察者）来实现交互

化窗口。

本研究也遵循了这个范式进行了软件原型的开发。

VTK 的源码是开放的，可从 VTK 官方网站免费获取。源码下载页面位于<https://vtk.org/download/>。

2.4.2 Paraview 的介绍

Paraview^[44]作为 Kitware 官方基于 VTK 开发的科学可视化软件，也支持了流体数据的可视化，因此本文稍作介绍，并作为成熟案例与本研究的方法进行对比。

在对高性能计算的支持方面，ParaView 具有极强的可扩展性，利用集群或超级计算机的聚合磁盘空间、处理能力和内存，进行大规模科学数据集的可视化和分析。其支持 MPI 的可执行文件家族可以在分布式内存计算机上并行运行，每个节点仅处理整个数据的一小部分。当结果几何形状较大时，众多结果会并行渲染和组合；当几何形状较小时，则可以串联方式处理。凭借 ParaView 的客户端/服务器架构，用户可以在集群或超级计算机上远程执行计算，然后使用本地个人电脑或笔记本电脑查看结果。

尽管有以上种种优点，Paraview 注重于大规模数据的科学可视化而非真实性可视化，其真实性可视化功能也并未对大规模数据的输入进行针对性优化，这为本研究的开展留出了空白。

2.4.3 VTK OpenGL 扩展的介绍

VTK 使用 vtkOpenGLHelper 这一个类来存储并管理着色器、OpenGL 程序、缓存等，为 OpenGL 开发提供了便利。

VTK 的开发者在 9.4.0 版本中引入了 GLAD 作为第三方扩展^[45]，因此开发者能够引入相关头文件（vtk_glad.h）即可在 VTK 的代码中使用 GLAD 库中的函数。

2.4.4 vtkOpenGLFluidMapper 的介绍

VTK 的开发者在 vtkOpenGLFluidMapper 类中用 OpenGL 代码实现了屏幕空间法的真实性流体渲染技术^[17]。作为本研究最直接的基石，有必要在此对其进行详细介绍。

在屏幕空间渲染技术中，首先采用点精灵（Point Sprite）方法渲染粒子场景，在两个二维纹理贴图中分别存储液体在屏幕空间中的深度信息和厚度信息。其中，深度纹理用于记录场景中各像素对应的液体表面点至摄像机的距离，而厚度纹理则用于存储各像素对应的累积厚度数据。随后，对两个纹理进行平滑处理可以减少噪声和锯齿效应，进而能从平滑后的深度纹理中重建高质量的表面法线向量。最

后利用之前的步骤获取的法线信息以及环境纹理，结合光学反射与折射原理（菲涅耳定律），对液体进行最终的着色计算，生成具有真实感的图像。流程图见2.2，以下详细介绍流程中的每一部分，包括算法以及 OpenGL 代码 GLSL 实现细节。

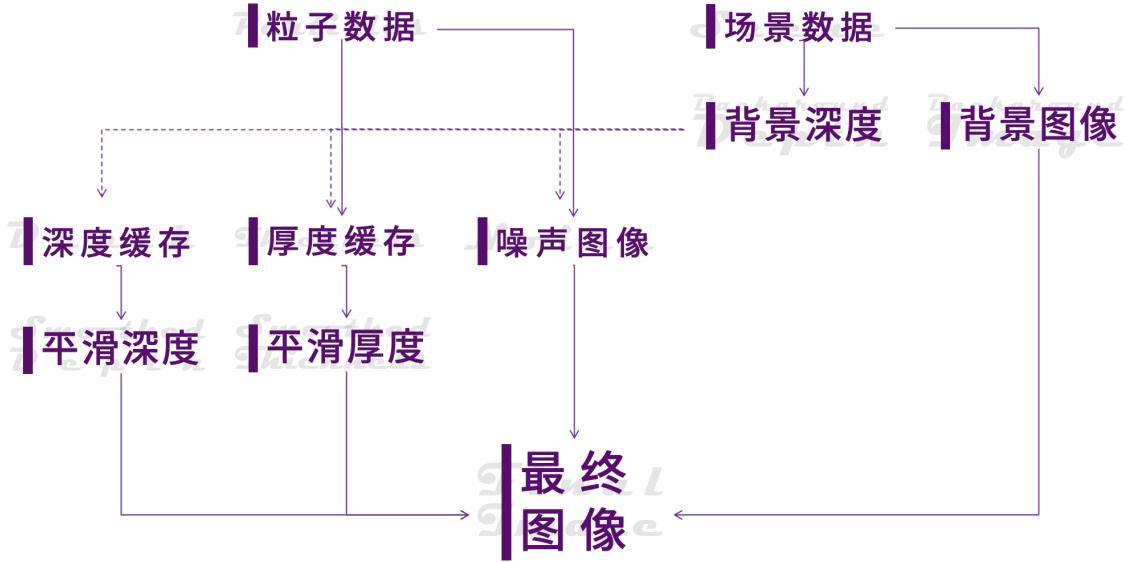


图 2.2 屏幕空间渲染技术管线^[17]

首先，渲染不包含流体的场景作为背景。这一步骤是通过传统的渲染方法将场景中的其他对象绘制到屏幕上，生成场景纹理。

接下来，使用 OpenGL 的点精灵 (Point Sprite) 技术渲染流体粒子，将流体的深度信息记录到深度纹理中。

点精灵是一种始终朝向摄像机的正方形，其大小可以动态设置。为了正确设置点精灵，算法需要根据世界空间的流体粒子大小（粒子半径）计算其在屏幕空间的投影大小。这一计算基于透视投影原理，通过三角形相似性确定世界空间半径到屏幕空间半径的映射关系即可。在片元着色器中，算法利用点精灵提供的，经过光栅化插值之后的点精灵纹理坐标变量 `gl_PointCoord`，将正方形点精灵转换为圆形，这是通过将与点精灵中心距离大于半径的像素消除做到的。为了获取点精灵中每个像素的相对位置，需要将 `gl_PointCoord` 坐标转换为以点精灵中心为原点的坐标系统。这种方式可以准确地计算每个像素的法线信息，并进一步确定像素在点精灵中的径向位置。对于超出设定半径阈值（例如 1.0）的像素，通过 `discard` 操作将其丢弃，从而实现点精灵的圆形外观。点精灵技术基于解析几何方法绘制球体，这种方法具有较高的精度，与传统的球体网格绘制方法相比，点精灵技术避免了因三角网格分辨率限制而导致的精度问题，同时显著减少了绘制调用和顶点处理的开销。

在渲染过程中，为了获取流体的深度信息，需要开启深度测试，并创建一个帧

缓冲对象。算法将流体粒子的深度数据渲染到帧缓冲对象的深度缓冲区中，也即深度纹理。获取标准坐标系的深度则可以通过计算流体粒子在摄像机空间中的位置，之后转换到裁剪空间并做透视线除法即可。

厚度纹理主要用于透明流体折射光线计算，对于不透光流体（如牛奶）则无需生成。而为了获取流体的厚度纹理，可采用 OpenGL 透明融合技术，OpenGL 的透明融合机制将每个流体粒子的厚度贡献值累积到颜色缓冲区中，而贡献值从其中心到边缘逐渐减少。算法并不计算球的实际厚度，而是将计算得到的法线的 z 分量作为厚度值，并乘以一个缩放系数。在具体设置上，在 CPU 端，关闭深度测试以避免深度竞争问题；开启透明融合，并设置融合函数为加法融合（additive blending），加法融合是直接相加原像素值和目标像素值。OpenGL 中还需要将 `glBlendFunc()` 的参数均设置为 `GL_ONE`，这样可以确保不同流体粒子的厚度值直接叠加，生成从摄像机视角观察到的流体厚度信息。

生成后的纹理如果未经过平滑，直接进行后续渲染操作得到的画面会有很强的粒子感，因此各种研究都在尝试使用各种滤波器来更好地平滑纹理，代码实现也支持了多种滤波器。

接下来的光照计算非常依赖表面的法线，因此有必要根据纹理来重建流体的法线。这里不必关注从设备标准空间 NDC 转换到摄像机空间的复杂数学细节，只需要了解可以通过逆投影矩阵得到摄像机空间的顶点坐标，然后再根据顶点坐标重建顶点法线。在计算法线向量时，本研究利用有限差分法计算沿着屏幕空间坐标轴方向的偏导数，进而通过叉乘操作得到法线向量。具体来说，通过对深度纹理在 x 方向（水平方向）和 y 方向（垂直方向）应用有限差分法，计算出两个偏导数向量再进行叉乘。这里的有限差分法是一种数值方法，用于近似计算函数的导数，在本研究的情况下，函数就是深度纹理在屏幕空间中的变化。

$$\frac{\partial f}{\partial s} \approx \frac{f(s + \Delta s) - f(s)}{\Delta s} \quad (2.10)$$

为了提高法线向量的准确性，本研究综合考虑了前向差分法和后向差分法的结果，选择了较小的差分值。这是因为较大的差分值可能出现在边界处，即背景或距离较远的流体深度值进行差分。这种情况下深度值不连续，从而导致错误的法线向量结果。代码实现对深度纹理图的周围四个像素分别进行了 x 方向和 y 方向的差分计算，以确保法线向量的准确性。这样得到的法线纹理可用于计算着色，即通过法线来计算液体表面的光线折射与反射方向，从而综合计算像素的着色。

本方法使用基于图像的光照（Image-Based Lighting, IBL）来计算反射颜色；若未开启 IBL，则将反射颜色默认设置为纯白色。本研究还使用 Beer-Lambert 定律

计算流体的透光率。

$$I(d) = I_0 \cdot e^{-kd} \quad (2.11)$$

公式 (2.11) 中, d 是厚度, I_0 是光照 RGB 向量, k 是光能衰减因子向量, 通常选择使用 `1.0 - liquidColor` 为默认值。Beer-Lambert 定律描述了光在流体中的吸收情况, 即透光率随光在流体中行进过的距离的增加呈指数衰减。厚度值从之前渲染的厚度纹理中采样得到。为了实现折射效果, 从背景纹理中采样并应用纹理坐标偏移, 生成折射效果。最终根据菲涅尔定律混合折射和反射光线以得到最终颜色。

在介绍完算法和实现之后, 值得补充的是该算法中一些参数的效果。当 `vtkOpenGLFluidMapper` 将 `vtkPolyData` 作为输入, 并支持标量颜色和标量可见性的同时, 在管道的不同阶段有许多选项可以定制流体渲染的效果。表2.1描述了一些关键参数和含义。

表 2.1 `vtkOpenGLFluidMapper` 中定制渲染输出的选项^[46]

参数	解释
<code>ParticleRadius</code>	渲染球体的半径 - 将值设置为大于粒子间距时, 可获得更平滑的表面
<code>DisplayMode</code>	选择将粒子渲染为不透明表面或半透明流体
<code>RefractiveIndex</code>	被渲染的流体的折射率
<code>RefractionScale</code>	调整通过体积的折射量
<code>AdditionalReflection</code>	为流体表面增加更多的反射光
<code>ScalarVisibility</code>	启用/禁用粒子的顶点颜色
<code>AttenuationColor</code>	在流体体积中呈指数级吸收的颜色
<code>AttenuationScale</code>	通过 <code>AttenuationColor</code> 定义的颜色的指数吸收因子
<code>SurfaceFilterMethod</code>	选择使用窄范围滤波器还是双边高斯滤波器来平滑深度表面
<code>SurfaceFilterRadius</code>	用于平滑深度表面的滤波半径
<code>SurfaceFilterIterations</code>	深度表面平滑滤波的迭代次数
<code>ThicknessAndVolumeColorFilterRadius</code>	用于过滤体积厚度和粒子颜色的滤波半径
<code>ThicknessAndVolumeColorFilterIterations</code>	用于体积厚度和颜色平滑滤波的迭代次数

第3章 大规模流体渲染算法开发

3.1 核心技术解析

在本研究中，为达成既定的研究目标，本研究成功开发了多种创新性技术。以下将对这些技术进行详细阐述。

3.1.1 continuous Level of Details (cLoD) 技术

本技术基于细节层次技术 (Level-of-Details, LoD)^[47] 开发，以下是该技术的介绍。

基于这样几个事实：

- 一个物体被渲染为图像的速度与物体建模表达方式的复杂度有关
- 人眼观察物体/相机观察世界具有透视关系——近大远小——物体移到远处会变小，导致细节不再重要
- 人眼观察世界时具有注意力机制，有一部分的细节会得到关注，需要精细的展现；而其他部分则在记忆编码中被简单地概括，其细节并不具有较高的优先度，可以允许较为粗糙的展现

本文可以通过在不必要的地方减少物体建模精度以达成在同样的视觉效果前提下提高渲染速度的结果。

LOD 技术根据物体在屏幕空间中的重要度以及物体在设备标准空间中所处的位置来分配算力，通过降低较远模型的细节程度从而提高渲染的效率。非常容易可以推导的，这样的技术有两个重要节点如下：

1. 如何判断模型的重要度以在世界中分配渲染算力资源
2. 如何得到不同细节程度的物体模型以用于渲染

一般来说，第一点比较容易，通常根据模型在屏幕上占据的像素大小/和相机的欧几里得空间距离来决定重要度；而对于第二个难点，近年来多种新型算法被研究出来用于得到不同细节程度的物体模型，如 Unreal Engine 虚幻引擎在 Siggraph 2021 中阐述的网格模型的无缝动态减面技术 Nanite，以及应用于点云模型的 clood 技术^[48]（本研究的灵感来源），和用于海洋波浪模拟的 LOD 技术^[23]。但是应用于流体粒子数据的 LOD 技术还是本研究首次引入并适配。

最先被开发出来的 LOD 技术是面向传统网格模型的。传统方法是由创作者根据需求分别编辑出各个 LOD 等级的网格模型，从低到高精细度和面数逐级降低，同时各等级模型之间渲染出来的效果不能让观众感到割裂，然后 LOD 系统根据计

算出来的重要度选择并切换相应 LOD 等级的模型。这样的技术首先需要大量的人工去编辑模型，并且最后的效果也非常依赖艺术家的审美以及经验，随机性较高且无法工业流水线化；同时不同等级 LOD 切换过程中可能出现视觉效果上的突变，不利于最终效果的呈现。

基于此，虚幻引擎 Unreal Engine 5 开发了 Nanite 技术。简要来说，对于如何得到不同细节程度的模型，Nanite 技术首先在 GPU 上将整个网格模型分为一簇一簇的三角簇来看待，每一簇最低为 128 个有公共边的三角形组合在一起，使用一定的算法保证每一簇尽可能地集中，簇与簇之间的公共边尽可能少且面积尽可能相等。然后则是减面过程，每个 LOD 等级先划分组，思想与划分簇类似，组的作用是在减面过程中锁住组与组之间的公共边，每个等级减完面之后则放开限制，重新划分组。这样的一系列减面过程得到了一个细节层级树，基于此，GPU 得以实现高速并行下细节水平的选择剔除工作，这样就使得剔除后，屏幕空间中的网格模型能尽可能保证每一个像素都得到足够的细节且不会有太多的算力损失。最后使用经典的重要度计算模型来计算应当切换到哪一层级，达成了几乎连续且实时的 LOD 切换。

同样的，基于点云的 LOD 技术也有类似的先划分点云为簇再进行剔除的技术，和计算根据 LOD 指数应当存在的粒子间隔，并剔除低于该间隔的粒子的技术。但是点云数据虽然粒子数极高，达到亿级甚至十亿的程度。但是扫描点云数据具有集中于物体表面、每个点有颜色数据、粒子间距密集等特点，和流体模拟数据的粒子数据有很大区别。流体数据的粒子存在于三维空间中的形式是填满物体内部空间的，且具有影响范围，并不是非常密集，同时存在如飞溅的水滴等离群的粒子。基于该先验，本研究综合了多种 LOD 技术的长处，针对流体数据和基于屏幕空间的渲染法定制了 cLOD 技术。

首先计算每个粒子的 cLOD 值，即 LOD 级别。但是不同的是，本文将取值很符合直觉地从正整数集延拓至正实数集。这样本文不需要对筛选、剔除粒子模型并且分配算力资源的算法采用特殊操作，直接采用连续的 LOD 选择即可。考虑到粒子进行 LOD 剔除操作后在屏幕空间法中会经历透视投影到屏幕上的操作，本研究提出如下计算 LOD 级别的公式：

$$lod = \frac{1000 \times \max\{1 - 0.7 \times \frac{d_c}{d_{max}}, 0.3\}}{d \times CLOD} \times \text{random}(Position) \quad (3.1)$$

其中：

- lod 为针对某一质点的 cLOD 级别
- d_c 为投影到屏幕上的点到屏幕中心点的距离

- d_{max} 为屏幕中心到边缘的最大距离，用于归一化 d_c
- $\max\{1 - 0.7 \times d_c, 0.3\}$ 则是将距离取值范围作了一个钳制到 [0.3,1]
- d 为设备标准空间中该点到相机中心的距离
- CLOD 为一个内部参数，以根据输入数据的范围来控制 cLOD 级别的数据分布，方便后续的筛选剔除
- $\text{random}(\mathbf{Position})$ 则是随机项，在大量数据中进行随机扰动，防止过于有规律的模式产生类似摩尔纹的幻觉（Illusion）

最终在 OpenGL 的计算着色器实现中，本研究考虑到实现效率与精度，使用以下方式来产生随机数：

$$\text{fract}(\sin(\mathbf{Position} \cdot \begin{bmatrix} 12.9898 \\ 78.233 \\ 54.143 \end{bmatrix})) \times 437586.5453) \quad (3.2)$$

其中 $\text{fract}()$ 函数为提取小数部分，函数值域为 [0,1]。在公式3.2计算后 LOD 值取值范围仅是正实数，因此还应当采用一个平滑的映射将值域映射到 [0,1] 上。这样的方式在最终的实践中被证实是在一般的数据范围中表现良好，能够产生随机的扰乱剔除。这样的方式产生的 cLOD 值分布可以被描述为从屏幕中间向外逐渐减小，从相机中心逐渐向远减少，这也符合人类观察世界的直觉——看不清远处的事物但是可以细致地观察近处的物体；更关注视野中样的物体而对视野边缘的物体一般不会投入太多的关注。

其次在计算完 cLOD 值后应当剔除某些粒子，在剔除粒子较少的情况下，因大量粒子均渲染到一个像素内，剔除对最终效果的影响微乎其微，本研究提出将控制权给到用户，使用用户输入的 cLOD 过滤值为阈值。而考虑到粒子数据的特殊性，本研究采用所有 LOD 级别随机选择粒子剔除的方法，剔除概率根据计算出的 cLOD 值和阈值之比，而非全部剔除。

本节所述技术虽然在本研究中使用 OpenGL 的计算着色器实现，但其他通用并行计算模型 GPGPU 技术均可实现本技术，通用性广泛。

3.1.2 未来帧提前加载传输优化技术

目前，流体动画渲染的时间成本主要可分为两部分，流体数据从磁盘加载至内存的时间以及渲染/可视化算法将流体数据转化为图像数据的时间。当粒子数较少时，前者可几乎忽略不计，后者时间占据大头；而当粒子数达到千万等级时，随着渲染算法的改进与时间的减少，前者在总渲染时间中所占的比例便变得举足轻重起来。本节所述技术便是为解决这种情况而开发。

事实上，这两类时间所涉及的计算机操作虽在一帧内有前后依赖关系，但是多帧的数据之间通常并无依赖关系，因此可以采用单指令流多数据流（Single Instruction Multiple Data, SIMD）的思想加速动画渲染。本文使用两个线程分别管理文件读取与渲染，在渲染的同时或程序运行但无操作时，利用后台进行数据读取。同时对于数据体积的限制，本研究提出先通过系统函数获取最大可用内存容量，再计算得出适宜的提前加载传输帧数，进行提前的加载；在已经加载的帧被渲染之后，算法丢弃该帧并读入新帧。

在对程序调试之后，研究者发现，原生的数据读取类在实践中只有约 8MB/s~14MB/s 的读取速度，远远低于 PCIe4.0 硬盘的最大设计速度 7000MB/s~8000MB/s，影响数据读取的另一因素 CPU 利用率也非常低，因此可以使用多线程并行读取多帧数据的设计来加速数据加载和渲染，或者使用 vtk 原生支持的 XML 并行 I/O 读取文件（但是这需要在存储数据时就存储 XML 类型的文件）。

3.2 算法输入与输出

本研究面向大规模流体粒子数据精心设计了渲染算法，其输入为包含丰富物理属性信息的三维空间粒子数据集。这些粒子数据以结构化的三维向量数组形式组织，每个粒子拥有以双精度浮点数记录的精确的三维空间笛卡尔坐标系下的位置坐标 (x, y, z) 以及附属的基于索引的速度、密度、颜色等物理属性信息，这些属性信息支持向量和标量两种形式。在计算机内存中，这些数据通过高效的存储结构和索引机制进行管理，以确保数据能够被快速访问和处理。

具体地说，一般的流体结算数据以 vtkDataSet 类存储为 vtk 文件或 vtkUnstructuredGrid 类存储为 vtu 文件。这样的文件经由 vtkDataSetReader/vtkXMLUnstructuredGridReader 读取后成为内存中的类，一般由 vtkdataArray 类及其子类进行数据管理（大多数为 vtkDoubleArray，以双精度浮点数存储），并且在空间位置数据之外有压强、速度、密度等等相应的 Tuple 数据。而考虑到后续只需要空间位置、影响范围的粒子半径这两种数据，且本算法对数据精度要求并不如数值模拟高，本算法对读入后的数据进行了过滤，将只包含空间位置的数据类作为输入传递到核心算法函数中，且将双精度浮点数转换为单精度浮点数，具体过程在后续文章中详解；同样地，也天然支持只有空间位置的单精度浮点数数组输入。

算法输出为二维图像。图像本质上是一个二维颜色向量数组，通过存储红色、绿色、蓝色颜色分量来表征颜色。同时算法支持将输入的时间序列输出为图像时间序列——也就是视频——来直观的显示流体的动态行为。

3.3 算法设计流程图

大规模流体渲染算法的设计流程图如图3.1所示，该流程图全面且系统地展示了本研究设计的从原始粒子数据输入到最终图像输出的完整路径。整个流程分为以下几个大阶段：

1. 数据预处理与参数初始化：对输入的粒子数据进行清洗和规范化处理，去除噪声和异常值，并根据流体的物理特性和可视化目标确定渲染参数，如相机位置、视角、光照条件，以及最重要的，对流体外观产生重要影响的各种 vtkOpenGLFluidMapper 参数等。同时，设定初始化连续细节层次（cLoD）相关的参数，为后续的细节层次调整做好准备。
2. 细节层次动态调整：基于 cLoD 技术，根据粒子在屏幕空间中的投影位置和距离相机的远近，实时计算每个粒子的细节层次值，并据此动态调整粒子的渲染细节。通过这种方式，在保证视觉连续性的前提下，实现渲染资源的合理分配，优化渲染性能。
3. 屏幕空间渲染阶段：首先通过点精灵渲染技术将流体粒子投影到屏幕空间，并通过不同的设置生成用于深度纹理和厚度纹理，然后平滑纹理、基于有限差分法的重建法线、基于物理计算光照，逐步渲染出流体的真实外观。在光照计算过程中，算法综合考虑了反射、折射以及光在水体中的吸收等物理现象，确保流体图像的真实感。
4. 后处理与图像合成：先对渲染得到的图像进行颜色校正等后处理操作。然后将流体图像与环境中的块体、地形等其他元素进行合成，得到最终的图像。

3.4 算法的取舍与变体

在算法的设计过程中，为平衡效率与质量，本研究对多个关键技术环节进行了深入分析，根据不同方式的效果和成本进行合理取舍，同时探索了多种变体方案以适应不同应用场景。

- **细节层次（LOD）的取舍与变体：**连续细节层次（cLoD）技术作为本研究的核心创新之一，摒弃了传统离散 LOD 技术因模型切换导致的视觉突变问题，通过将 LOD 值拓展至连续的正实数域，实现了粒子细节程度的平滑过渡。然而，cLoD 技术的细节层次计算方式对计算平台以及运行环境有一定需求，如 vtk 版本需要高于 9.4、OpenGL 版本需要高于 4.3 等等。针对这个问题，本研究提出一种更加通用的变体方案作为替代，将 GPU 上的计算与剔除环节替换为在 CPU 上使用 vtk 内置并行函数实现。虽然这样的方案降低了运行效率，但是提高了通用性，在所有平台上都能使用。本变体为不同硬

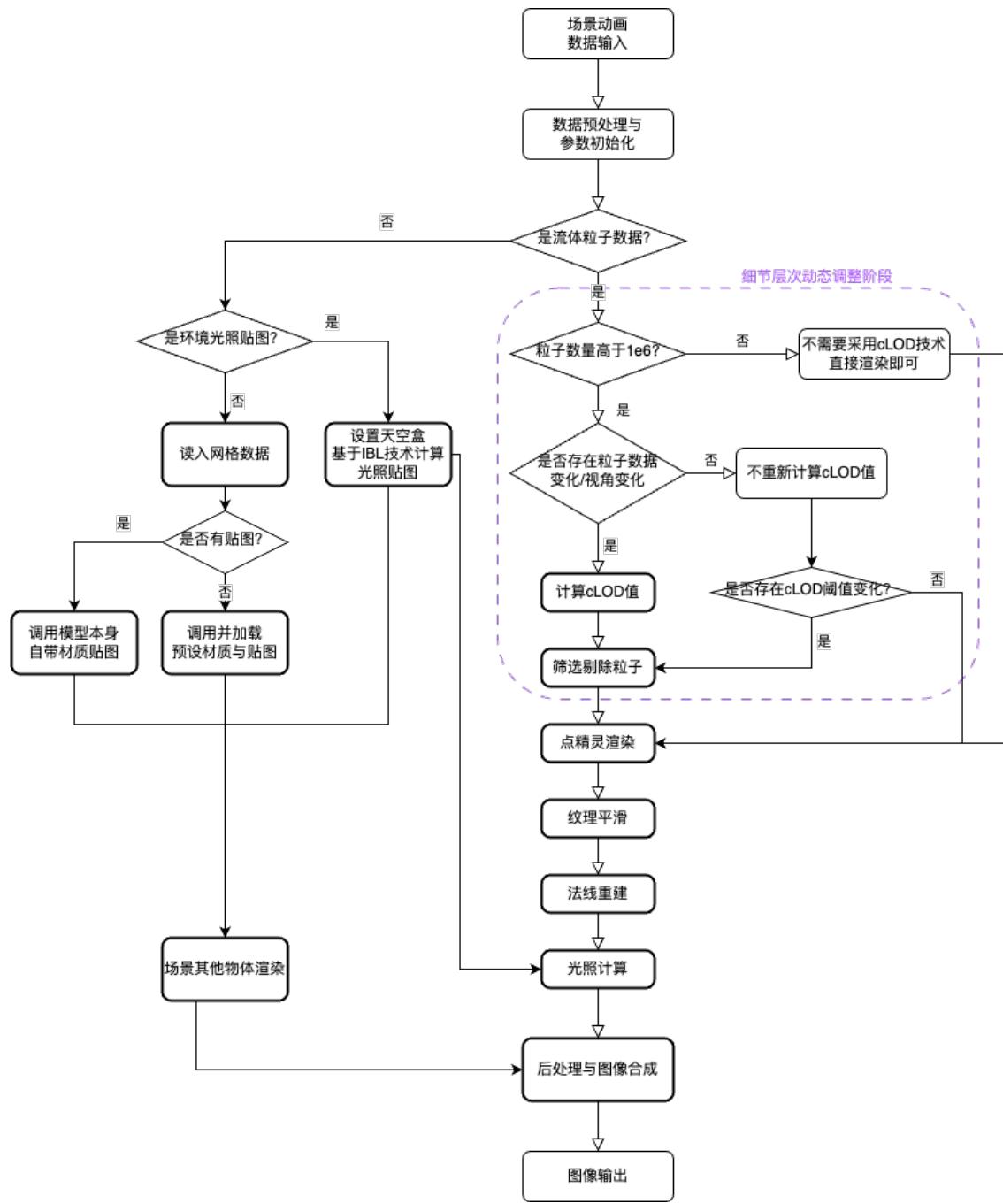


图 3.1 算法流程图

件条件提供了灵活的选择。

- **纹理滤波器的选择与优化：**在纹理平滑环节，本研究使用窄范围滤波器，但该滤波器有较高的计算复杂度。但是对于更加低性能的硬件平台，窄范围滤波器可能导致最终帧率过低。为此，本研究提出一种自适应滤波器切换策略，根据流体表面的局部复杂度和曲率，在双边高斯滤波器和窄范围滤波器之间动态切换，从而在效果和性能之间取得平衡。
- **光照模型的优化与简化策略：**基于图像的光照模型虽然能提供真实的光照环境模拟，但在光照计算复杂度、纹理资源、内存规模上要求较高。为此，本研究还支持了 vtk 的其他光照模型，如立方体贴图光照、基于设定光源的光照等。对于计算资源受限的场景，还可进一步简化为 Lambert 光照模型或 Blinn-Phong 光照模型，这些模型通过摈弃光照反射和折射等复杂项的计算，在降低了计算复杂度的同时，依然能够提供基本的光照效果以满足光照质量要求不高的需求。
- **粒子筛选与剔除的优化方法：**基于 cLoD 的随机剔除策略实现了较为均匀的粒子分布和合理的渲染负载控制。然而，该策略在某些复杂场景下可能导致保留过多边界情况的粒子，进而导致最终效果不佳。为解决这一问题，本研究提出了一种改进的基于视锥体和远近裁切平面剔除的粒子筛选方法，首先快速排除那些完全位于视锥体外的粒子，从而减少后续基于 cLoD 方法的剔除需要面对的数据量。对于过于接近摄像机的流体粒子（即在屏幕空间中占据过多像素的粒子），屏幕空间渲染算法中的平滑算法效果不佳，因此可以将近剔除面的阈值提高，通过剔除一部分的粒子来提升最终的视觉效果。

3.5 原型开发

为验证大规模流体渲染算法的有效性并展示其实际应用潜力，本研究开发了一个功能完善的原型系统。该原型系统采用 C++ 编程语言结合 OpenGL 图形 API 构建，并充分利用了 VTK 库的丰富功能，实现了从大规模流体粒子数据输入到输出的完整流程。该系统包括以下多个模块：

1. **粒子数据的加载与管理模块：**该模块负责从多种数据源高效加载流体粒子数据，并将其组织成适合渲染的内存结构。本研究设计了一种基于数据流的加载机制，通过多线程技术实现数据的异步加载，确保在数据加载过程中渲染线程能够持续运行，避免渲染过程因数据加载而中断，为流体的动态模拟和渲染提供了坚实的数据支持。
2. **连续细节层次 (cLoD) 技术的集成模块：**本文精心设计了 cLoD 技术的集成

方案，使其能够无缝融入整个渲染流程。本文在头文件中，通过调用 GLAD 提供的函数编写基于 OpenGL 规范的计算着色器代码在 GPU 上实现了高效的 cLoD 算法。本文能够实时计算每个粒子的细节层次值，并据此动态调整粒子的细节程度。为了提高 cLoD 计算的整体效率，本研究提出了一种视角敏感的 cLoD 调整策略，基于相机位置和角度来对粒子的筛选程度进行精细调整。具体来说，代码利用 vtk 的回调机制实现了对视角变换和 cLOD 阈值变化的监测，即每当需要重新计算 cLOD 值时才重新计算，而不是每帧都进行计算。

3. **未来帧提前加载传输优化模块：**针对大规模流体数据渲染过程中数据加载时间对渲染效率的影响，本文创新性地提出了未来帧提前加载传输优化技术。为了精确控制提前加载的帧数，本研究在模块中实现了一套基于系统内存监控的动态调整算法，实时获取硬件的可用内存容量，并根据流体数据的大小和渲染速率动态计算适宜的提前加载帧数。
4. **用户交互的参数界面：**为了增强应用整体的交互性，本文设计并实现了一个直观且功能丰富的用户可调节参数界面。该界面使用户能够在渲染过程中调整关键参数，以获得满足特定需求的可视化结果。界面主要由以下三个核心组件构成：
 - (a) 滑块组件作为用户界面中至关重要的一个部分，主要用于实时调节连续细节层次筛选粒子的阈值。在渲染大规模流体粒子数据时，合理设置 cLOD 阈值能够实现渲染质量和性能之间的动态平衡。具体而言，用户通过拖动滑块来选定 cLOD 阈值，进而调整流体粒子的细节程度。当阈值提高时，流体粒子数量增加，流体表面的微观特征得以精细呈现，适用于对流体局部细节进行深入研究的场景；而降低阈值则减少细节，提高渲染效率，利于快速获取流体整体运动状态。滑块的调节范围自动设定，覆盖了从展现流体粒子最精细结构到仅呈现大致轮廓的区间，满足不同需求。此外，滑块下方即时显示当前阈值，每次调整滑块后都调用回调函数，实时更新图像，让用户即刻看到参数变动对流体可视化效果的影响，极大提升了交互体验和工作效率。
 - (b) 播放按钮用于加载并持续渲染流体动画。点击播放按钮后，系统开始读取设定的流体动画序列，基于设定渲染参数渲染每一帧流体图像并显示。具体来说，系统预先加载流体动画各帧数据至内存缓冲区，渲染时依次从缓冲区读取数据，筛选粒子后由 GPU 渲染图像。用户可以随时点击播放按钮暂停或继续动画播放，灵活控制播放进度。此外，界面状

态栏实时显示当前播放帧数及总帧数。

- (c) 导出按钮为用户提供更进一步的功能，即把当前界面呈现的流体可视化内容保存为视频文件或图片文件，便于用户后续深入分析、分享展示或用作其他项目素材。点击导出按钮后，系统弹出对话框供用户指定保存路径及文件名，并提供多种常见视频格式选项，如 MP4、AVI 等。导出模块引用的编码库支持高效的压缩算法，在画质和文件体积之间达成平衡，方便存储与展示。

第4章 案例与测试

本章为对算法的测试以及搭建的案例，按照规模大小分为验证性案例和实际工程案例两部分。在介绍案例与测试之前，有必要介绍一下案例验证使用的计算机的性能与配置如表4.1：

表 4.1 本研究测试使用的计算机配置表

计算机部件	型号
中央处理器 (CPU)	英特尔®至强®处理器 E5-2696v3 Intel® Xeon® E5-2696v3
内存	DDR3 2133MHz ECC Memory, 96GB
图形处理器 (GPU)	NVIDIA® GeForce® RTX 2070 SUPER
硬盘	PCIe 4.0 SSD

4.1 验证性案例

本章将通过两个经典案例——“小球入水”和“溃坝实验”，对本文所提出的基于 GPU 的大规模粒子数据真实性可视化方法进行验证和测试。以下将分别介绍每个案例的实验设置以及详细的性能对比分析。

4.1.1 实验设置

通过与现实中小球入水实验视频的对比，小球入水案例主要用于验证算法在渲染流体时的准确性。

实验场景为复刻论文^[49]中拍摄的场景，因此大小为 $30 \times 50 \times 60\text{cm}^3$ ，小球入水速度约为 217cm/s ，小球半径为 1.27cm 。粒子数量为 459472。

溃坝实验旨在评估算法在模拟大规模流体动态变化过程中的性能表现，尤其是对流体快速流动和水滴飞溅的可视化效果。这一案例能够有效反映算法在处理大规模粒子数据时的稳定性、时间连续性和视觉真实性。模拟粒子数量为 1131712。

4.1.2 表现对比

将仿真得到的数据分别使用本研究的方法、VTK 默认渲染方法以及 Paraview 渲染这三种方法进行渲染，测量并计算性能（渲染时间和帧率）并对比，得到结果

如图4.1、图4.2及表4.2、表4.3。

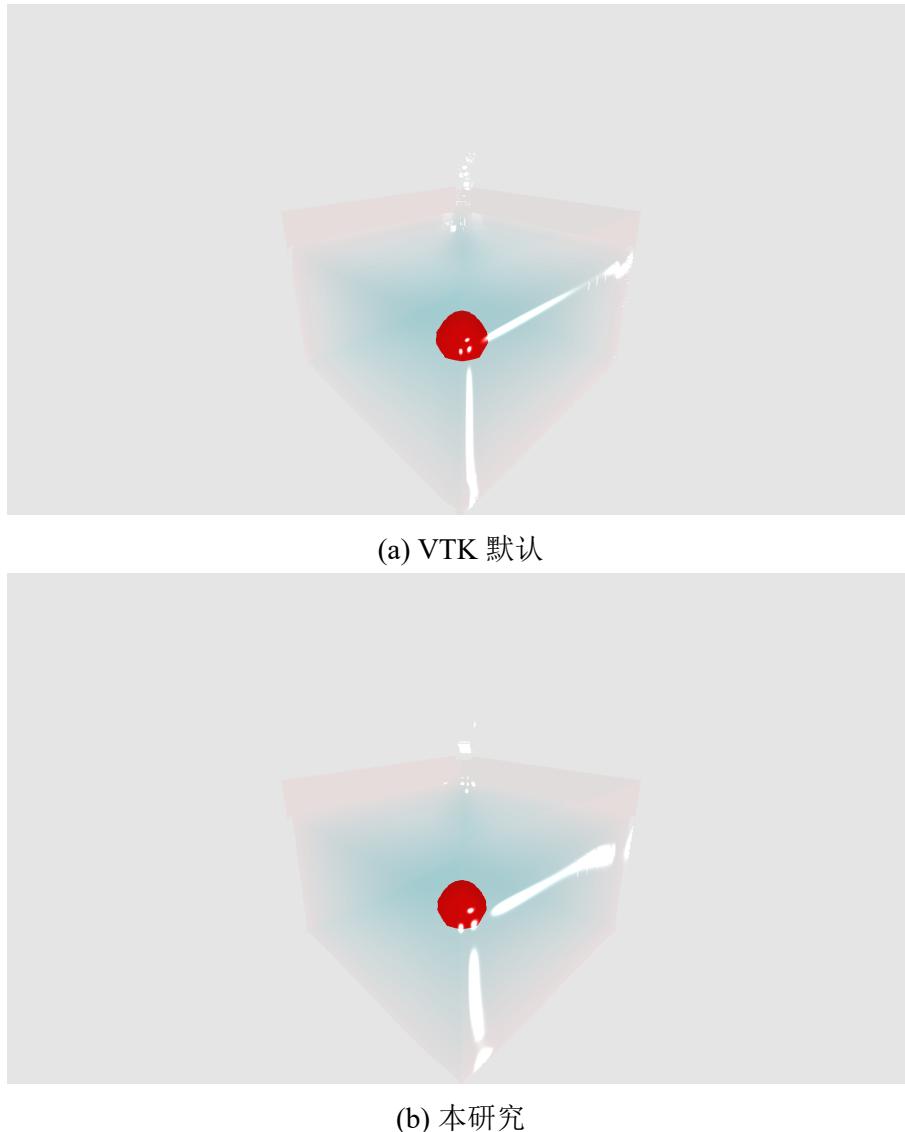
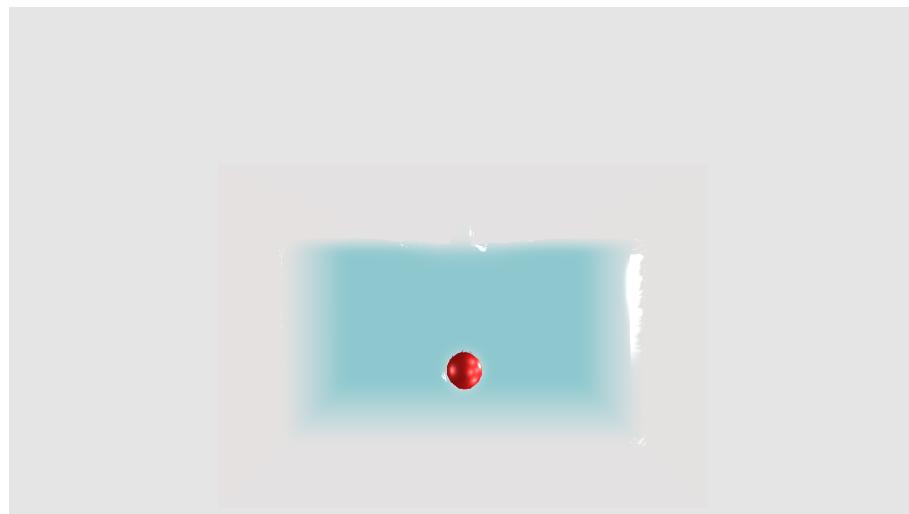


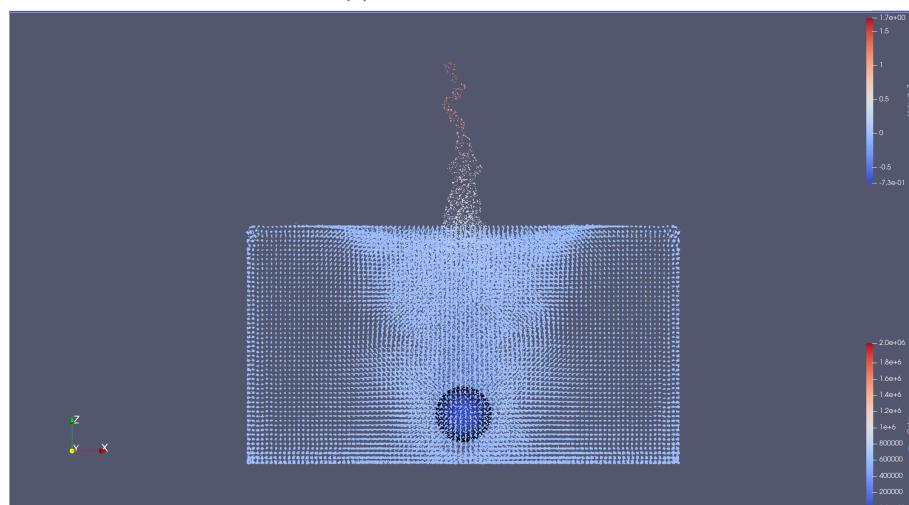
图 4.1 小球入水案例与 vtkOpenGLFluidMapper 对比

实验结果显示，小球入水产生的流体现象得到了可靠的可视化呈现。通过对不同时间步的渲染图像，用户可以清晰地观察到流体粒子在小球撞击作用下的动态响应。在渲染性能方面，本算法在配备 RTX 2070 Super GPU 的测试平台上，能够以平均 33.18 帧/秒（fps）的速度对整个场景进行实时渲染，充分证明了其在处理复杂流体与刚体相互作用场景时的高效性。

对于溃坝场景，本算法能够精确地呈现出溃坝瞬间流体的高速涌出、水波的快速扩散以及后续的波动衰减等复杂现象。渲染图像序列生动地再现了流体在不同时间阶段的行为，为流体动力学研究提供了直观的可视化支持。在性能方面，该案例在相同的硬件配置下，平均帧率为 48.12 fps，尽管粒子数量较“小球入水”案例翻倍，但本算法依然能够保持较为流畅的可视化结果。



(a) 本研究的切片图



(b) Paraview 渲染



(c) 小球入水时间序列图

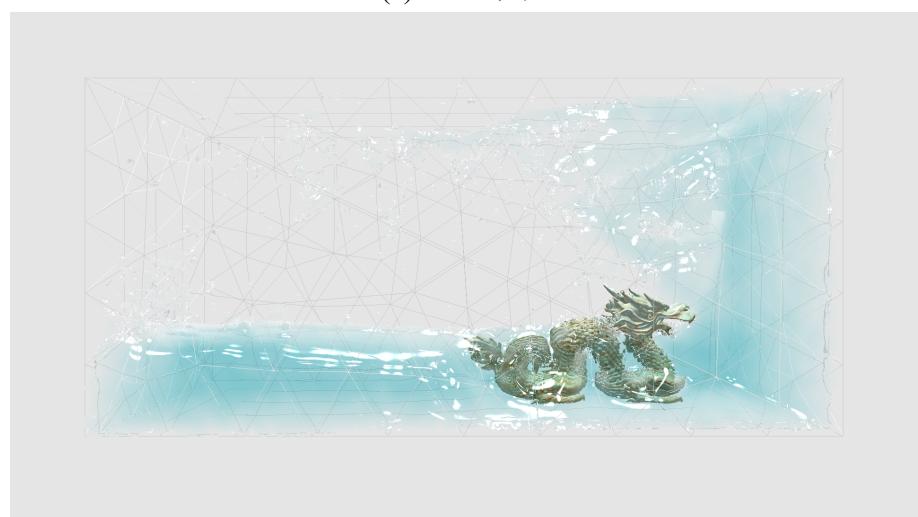
图 4.2 小球入水案例与 Paraview 对比

表 4.2 小球入水案例性能对比

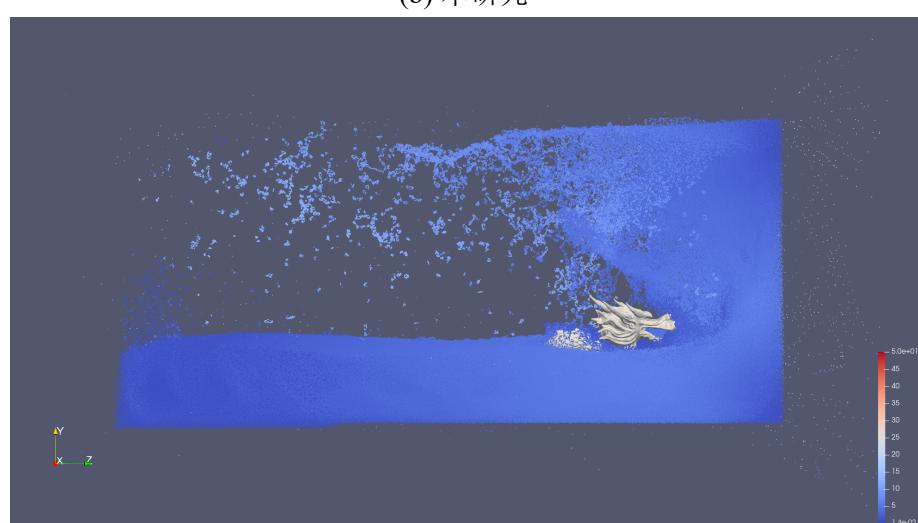
方法	计算着色器执行时间	渲染时间	渲染帧率
VTK 默认方法	N/A	401.099ms	2.5fps
本 研 究 方 法 (cLOD=0.6)	4ms	26.154ms	33.18fps



(a) VTK 默认



(b) 本研究



(c) Paraview 渲染

图 4.3 溃坝案例对比

表 4.3 溃坝案例性能对比

方法	计算着色器执行时间	渲染时间	渲染帧率
VTK 默认方法	N/A	701.587ms	1.425fps
本研究方法 (cLOD=0.3)	7ms	20.7819ms	48.119fps

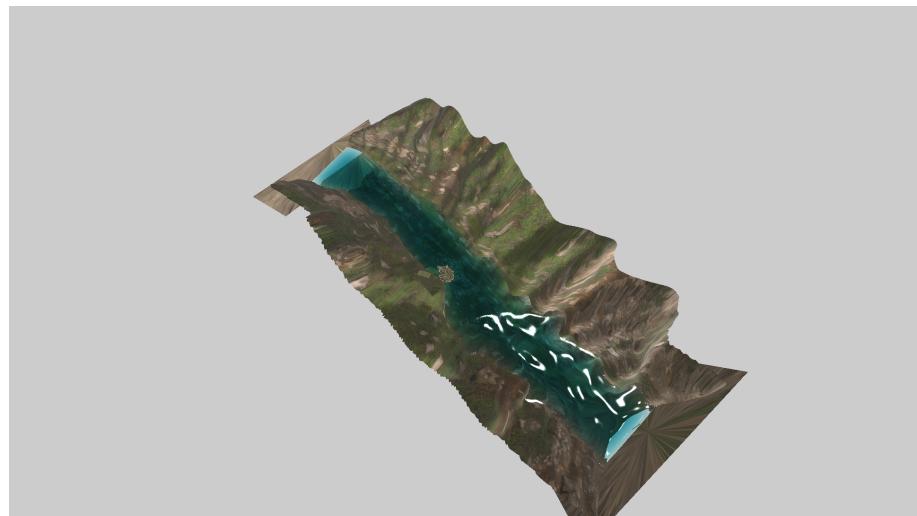
4.2 实际案例与效果、性能对比

实际案例选取三峡库区箭穿洞滑坡涌浪场景，流体模拟粒子数为 17988089。考虑到性能约束，案例中将 cLOD 值设置为 0.3。得到的效果如图4.5，性能对比如表4.4。

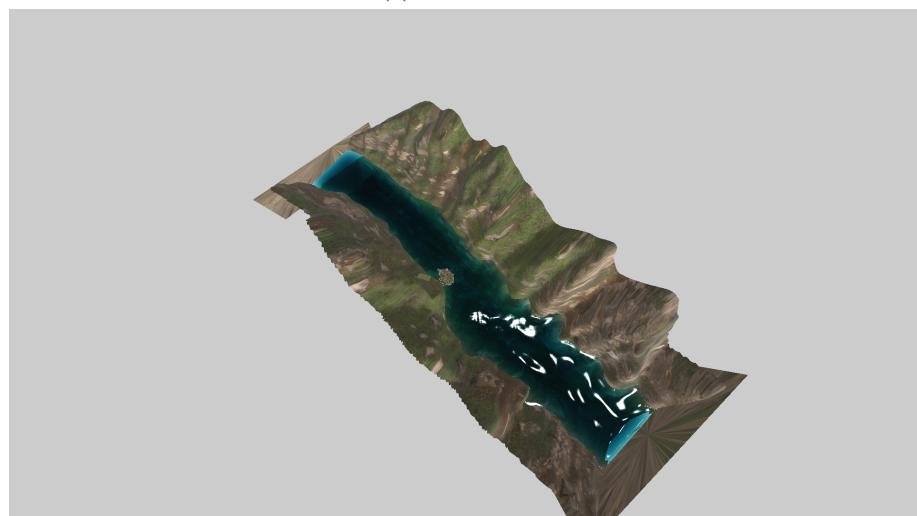
表 4.4 三峡箭穿洞案例性能对比

方法	计算着色器执行时间	渲染时间	渲染帧率
VTK 默认方法	N/A	621.473ms	1.425fps
本研究方法 (cLOD=0.3)	55ms	72.9964ms	7.813fps

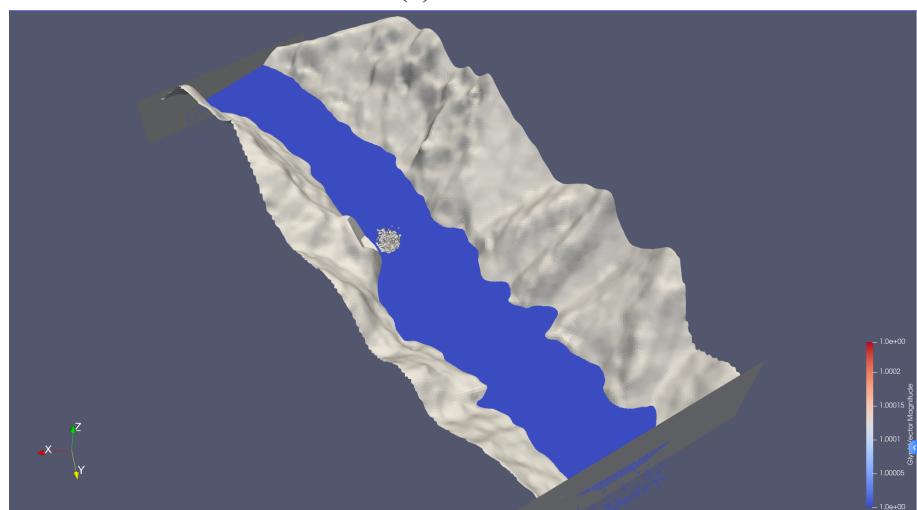
从表中可以看到，对于近两千万个粒子的渲染以传统的方法只有 1.425 帧每秒，这样的效率无法及时响应研究人员的鼠标拖动等指令，影响结果的分析；而应用了本研究的方法之后，渲染系统的效率加速了近乎 5 倍，达到了可交互的程度。



(a) VTK 默认

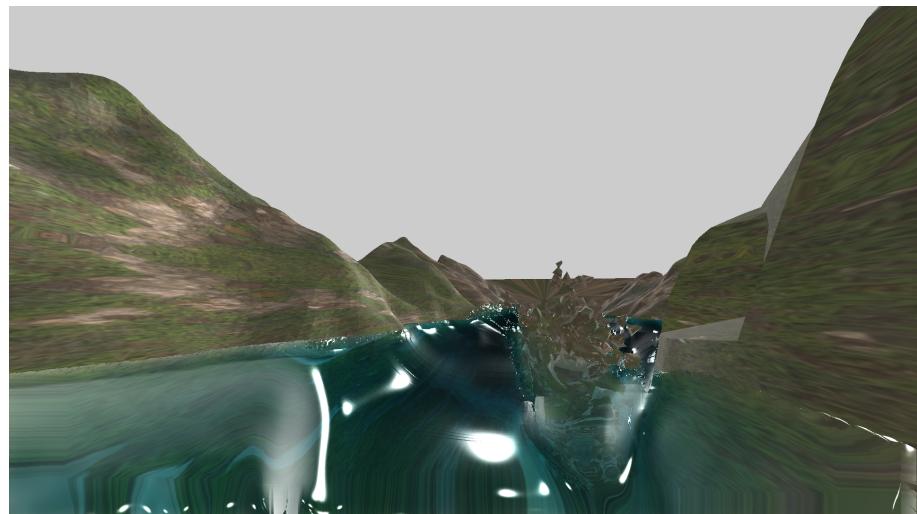


(b) 本研究

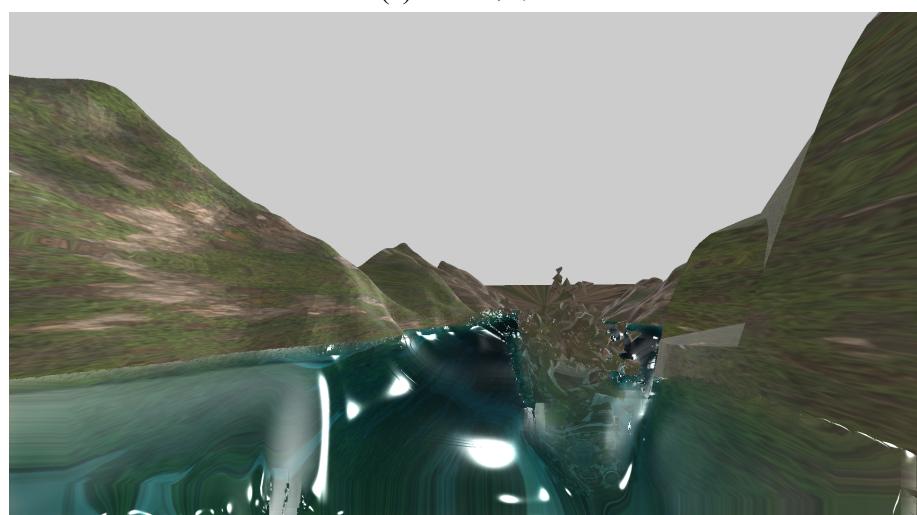


(c) Paraview 渲染

图 4.4 三峡箭穿洞案例对比



(a) VTK 默认



(b) 本研究

图 4.5 三峡箭穿洞案例对比第二视角

第 5 章 总结与展望

5.1 本文总结

本文针对大规模流体粒子数据的可视化难题，通过深入研究流体仿真数据的特点和现有可视化技术的局限性，通过对各种可视化技术的全面调研和思考，提出了一种基于 GPU 的大规模流体粒子真实性高效渲染方法，填补了大规模流体粒子数据真实可视化领域的空白。课题的提出源于岩土、水利、汽车、工业等领域对 3D 流体仿真需求。在这些领域中，流体仿真技术对于工程设计优化、灾害预防以及科学规律探究有着关键作用。

本研究的主要贡献包括以下几个方面：

1. 提出了一种基于 cLoD 技术的流体粒子数据可视化算法。该算法通过实时计算粒子的细节层次值和基于用户给定的阈值的筛选，实现了渲染资源的合理分配，在保证视觉连续性的前提下，有效提高了渲染效率。这使得用户能够根据实际需求灵活调整模型的细节程度，是大规模流体数据的高效可视化的一种创新解决方案。
2. 开发了未来帧提前加载传输优化技术。这项技术通过多线程机制，在渲染当前帧、等待未来帧的同时提前加载未来帧的数据。它能够显著减少数据加载时间对渲染效率的影响，提高了整个渲染流程的流畅度。
3. 设计并实现了一个功能完善的原型系统。该系统不仅支持用户通过滑块实时调整 cLoD 阈值，以平衡渲染质量和性能，还集成了播放按钮和导出按钮。播放按钮允许用户便捷地加载并持续渲染流体动画，直观地观察流体的动态演化过程；导出按钮则使用户能够将可视化结果保存为视频文件，方便后续分析和展示。这个原型系统为实际应用提供了一个高效、便捷的流体可视化工具。

综上所述，本研究为大规模流体粒子数据的真实性可视化提供了一种高效、稳定且具有广泛应用潜力的方法。随着计算机图形学、流体力学以及高性能计算技术的不断发展，预计流体可视化技术将在未来迎来更多的突破和创新，为科学研究、工程实践以及创意产业等领域做出更大的贡献。

5.2 未来展望

尽管本研究在大规模流体粒子数据可视化方面取得了一定成果，但仍存在一些值得改进的方向。未来的研究可以从以下几个方面展开：

1. 进一步提升算法的细节表现能力和物理真实感。通过引入更先进的光照模型以及粒子相互作用算法，增强流体表面的细节表现能力，如使用多层结构来提高波纹、水花等微观结构的真实感。还有液体与空气相互作用时产生的气泡/泡沫也需要全新的光照模型也是现有的算法不适用的部分。这将有助于更精确地还原流体的物理特性，为科学的研究和工程应用提供更具价值的可视化结果。
2. 拓展算法的应用领域和功能。例如，可以探索将算法应用于虚拟现实（VR）、增强现实（AR）等沉浸式环境中的可能性，为用户提供更加直观、沉浸的流体交互体验。
3. 采用更高效或质量更高的筛选算法。比如使用 KNN 算法计算临域中流体粒子的密集程度以针对每一个像素来自适应地更改筛选程度，或者使用光线投射算法的概念去判断哪些粒子会被渲染到同一个像素中，由此来限制每个像素所包含的粒子数目，更好地进行筛选。

参考文献

- [1] Eymard R, Gallouët T, Herbin R. Finite volume methods[M/OL]//Handbook of Numerical Analysis: Vol. 7 Solution of Equation in \mathbb{R}^n (Part 3), Techniques of Scientific Computing (Part 3). Elsevier, 2000: 713-1018. <https://www.sciencedirect.com/science/article/pii/S1570865900070058>. DOI: [https://doi.org/10.1016/S1570-8659\(00\)07005-8](https://doi.org/10.1016/S1570-8659(00)07005-8).
- [2] Zhou P b. Finite difference method[M/OL]//Numerical Analysis of Electromagnetic Fields. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993: 63-94. https://doi.org/10.1007/978-3-642-50319-1_3.
- [3] Bathe K J. Finite element method[M/OL]//Wiley Encyclopedia of Computer Science and Engineering. John Wiley & Sons, Ltd, 2008: 1-12. <https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470050118.ecse159>. DOI: <https://doi.org/10.1002/9780470050118.ecse159>.
- [4] Gingold R A, Monaghan J J. Smoothed particle hydrodynamics: theory and application to non-spherical stars[J]. Monthly notices of the royal astronomical society, 1977, 181(3): 375-389.
- [5] Koshizuka S, and Y O. Moving-particle semi-implicit method for fragmentation of incompressible fluid[J/OL]. Nuclear Science and Engineering, 1996, 123(3): 421-434. <https://doi.org/10.13182/NSE96-A24205>.
- [6] Tskhakaya D, Matyash K, Schneider R, et al. The particle-in-cell method[J/OL]. Contributions to Plasma Physics, 2007, 47(8-9): 563-594. <https://onlinelibrary.wiley.com/doi/abs/10.1002/ctpp.200710072>. DOI: <https://doi.org/10.1002/ctpp.200710072>.
- [7] Brackbill J, Ruppel H. Flip: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions[J/OL]. Journal of Computational Physics, 1986, 65(2): 314-343. <https://www.sciencedirect.com/science/article/pii/0021999186902111>. DOI: [https://doi.org/10.1016/0021-9991\(86\)90211-1](https://doi.org/10.1016/0021-9991(86)90211-1).
- [8] Chen S, Doolen G D. Lattice boltzmann method for fluid flows[J/OL]. Annual Review of Fluid Mechanics, 1998, 30(Volume 30, 1998): 329-364. <https://www.annualreviews.org/content/journals/10.1146/annurev.fluid.30.1.329>. DOI: <https://doi.org/10.1146/annurev.fluid.30.1.329>.
- [9] 宋吉虎. 基于粒子的实时流体模拟和渲染[D]. 吉林大学, 2023.
- [10] Lorensen W E, Cline H E. Marching cubes: A high resolution 3d surface construction algorithm [C/OL]//SIGGRAPH '87: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques. New York, NY, USA: Association for Computing Machinery, 1987: 163-169. <https://doi.org/10.1145/37401.37422>.
- [11] Bruckner S. Dynamic visibility-driven molecular surfaces[C]//Computer Graphics Forum: Vol. 38. Wiley Online Library, 2019: 317-329.
- [12] Lackner K. Integrating gpu-based fluid simulation with metaball rendering[D]. TU Wien, 2022.
- [13] Xiao X, Zhang S, Yang X. Real-time high-quality surface rendering for large scale particle-based fluids[C/OL]//I3D '17: Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. New York, NY, USA: Association for Computing Machinery, 2017. <https://doi.org/10.1145/3023368.3023377>.

- [14] 俞铭琪, 全红艳, 宋晓. 真实感流体实时重建[J/OL]. 计算机辅助设计与图形学学报, 2013, 25(5): 622-630. <https://www.jcad.cn/cn/article/id/b965946e-2114-4f4f-ac15-c0232c530322>.
- [15] 刘艳王新颖. 一种拉格朗日粒子流体的高效表面重建方法[J/OL]. 图学学报, 2016, 37(5): 607-613. DOI: <http://www.txxb.com.cn/CN/10.11996/JG.j.2095-302X.2016050607>.
- [16] 何学洋. 流体模拟与真实感渲染技术研究及应用[D]. 电子科技大学, 2024.
- [17] van der Laan W J, Green S, Sainz M. Screen space fluid rendering with curvature flow[C/OL]// I3D '09: Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games. New York, NY, USA: Association for Computing Machinery, 2009: 91-98. <https://doi.org/10.1145/1507149.1507164>.
- [18] Truong N, Yuksel C. A narrow-range filter for screen-space fluid rendering[J/OL]. Proc. ACM Comput. Graph. Interact. Tech., 2018, 1(1). <https://doi.org/10.1145/3203201>.
- [19] Oliveira F, Paiva A. Narrow-band screen-space fluid rendering[J/OL]. Computer Graphics Forum, 2022, 41(6): 82-93. <https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14510>. DOI: <https://doi.org/10.1111/cgf.14510>.
- [20] Xu Y, Xu Y, Xiong Y, et al. Anisotropic screen space rendering for particle-based fluid simulation[J/OL]. Computers & Graphics, 2023, 110: 118-124. <https://www.sciencedirect.com/science/article/pii/S0097849322002308>. DOI: <https://doi.org/10.1016/j.cag.2022.12.007>.
- [21] Zhang Y, Xu Y, Xu Y, et al. Real-time screen space rendering method for particle-based multiphase fluid simulation[J/OL]. Simulation Modelling Practice and Theory, 2024, 136: 103008. <https://www.sciencedirect.com/science/article/pii/S1569190X24001229>. DOI: <https://doi.org/10.1016/j.simpat.2024.103008>.
- [22] dos Santos Brito C J, B. Vieira e Silva A L, Teixeira J M, et al. Ray tracer based rendering solution for large scale fluid rendering[J/OL]. Computers & Graphics, 2018, 77: 65-79. <https://www.sciencedirect.com/science/article/pii/S0097849318301547>. DOI: <https://doi.org/10.1016/j.cag.2018.09.019>.
- [23] Duan X, Liu J, Wang X. Real-time wave simulation of large-scale open sea based on self-adaptive filtering and screen space level of detail[J/OL]. Journal of Marine Science and Engineering, 2024, 12(4). <https://www.mdpi.com/2077-1312/12/4/572>. DOI: 10.3390/jmse12040572.
- [24] 孙颖君. 基于游戏引擎的滑坡三维动态可视化方法研究[D]. 福建师范大学, 2022.
- [25] Burley B, Studios W D A. Physically-based shading at disney[C]//Acm Siggraph: Vol. 2012. vol. 2012, 2012: 1-7.
- [26] Cook R L, Torrance K E. A reflectance model for computer graphics[J]. ACM Transactions on Graphics (ToG), 1982, 1(1): 7-24.
- [27] Burley B. Extending the disney brdf to a bsdf with integrated subsurface scattering[J]. SIGGRAPH Course: Physically Based Shading in Theory and Practice. ACM, New York, NY, 2015, 19(7): 9.
- [28] Heitz E. Understanding the masking-shadowing function in microfacet-based brdfs[J]. Journal of Computer Graphics Techniques, 2014, 3(2): 32-91.

- [29] community project. P H. Poly haven _ the public 3d asset library[EB/OL]. <https://polyhaven.com/>.
- [30] Groups K. Opengl - the industry standard for high performance graphics[EB/OL]. <https://www.opengl.org/>.
- [31] NVIDIA. Opengl _ nvidia developer[EB/OL]. <https://developer.nvidia.com/opengl>.
- [32] 杜义君, 苏鸿根. 基于 Visual Basic 的 OpenGL 三维图形开发环境的构建及其应用[J]. 塔里木大学学报, 2009, 21(01): 35-39.
- [33] 万里驰骋. openGL 中的 gl,glu,glut - 万里驰骋 - 博客园[EB/OL]. <https://www.cnblogs.com/xieqianli/p/11398002.html>.
- [34] OpenGL. Glut - the opengl utility toolkit[EB/OL]. https://www.opengl.org/resources/libraries/glut/glut_downloads.php.
- [35] project G. An opengl library _ glfw[EB/OL]. <https://www.glfw.org/>.
- [36] Nigel Stewart M I, Magallon M. Glew_ the opengl extension wrangler library[EB/OL]. <https://glew.sourceforge.net/>.
- [37] Herberth D. Glad_ multi-language gl/gles/egl/glx/wgl loader-generator based on the official specs.[EB/OL]. <https://glad.dav1d.de/>.
- [38] 丁科, 谭营. GPU 通用计算及其在计算智能领域的应用[J]. 智能系统学报, 2015, 10(1): 1-11.
- [39] 姜波. 面向区间分析的神经网络算法及应用研究[D]. 大庆石油学院, 2010.
- [40] Groups K. Compute shader - opengl wiki[EB/OL]. https://www.khronos.org/opengl/wiki/Compute_Shader.
- [41] 胡刚. 基于 VTK 的地质体三维可视化研究[D]. 中南大学, 2011.
- [42] 郑雪虎. 基于医学图像的三维非均质生物组织建模理论及方法研究[D]. 河北工业大学, 2007.
- [43] 许庆功, 李昌华. VTK 框架结构与运行机制的探讨[J]. 洛阳理工学院学报 (自然科学版), 2008(01): 67-70.
- [44] Kitware I. Paraview - open-source, multi-platform data analysis and visualization application [EB/OL]. <https://www.paraview.org/>.
- [45] Kitware I. Vtk v9.4.0[EB/OL]. <https://www.kitware.com/vtk-v9-4-0/>.
- [46] Kitware I. Screen-space fluid rendering in vtk[EB/OL]. <https://www.kitware.com/screen-space-fluid-rendering-vtk/>.
- [47] Heok T K, Damant D. A review on level of detail[C/OL]//Proceedings. International Conference on Computer Graphics, Imaging and Visualization, 2004. CGIV 2004. 2004: 70-75. DOI: 10.1109/CGIV.2004.1323963.
- [48] Schütz M, Krösl K, Wimmer M. Real-time continuous level of detail rendering of point clouds [C/OL]//2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR). 2019: 103-110. DOI: 10.1109/VR.2019.8798284.
- [49] Aristoff J, Truscott T, Techet A, et al. The water entry of decelerating spheres[J/OL]. Physics of Fluids - PHYS FLUIDS, 2010, 22. DOI: 10.1063/1.3309454.

附录 A 外文资料的书面翻译

Ray Tracer based rendering solution for large scale fluid rendering

基于光线追踪的大规模流体渲染解决方案

目录

摘要	42
A.1 引言	43
A.2 最新技术	45
A.2.1 光滑粒子动力学方法	45
A.2.2 流体渲染	47
A.3 方法	48
A.3.1 平滑粒子流体动力学方法	48
A.3.2 光线追踪渲染解决方案	51
A.4 实现	55
A.4.1 DualSPHysics	56
A.4.2 OptiX	58
A.5 测试案例	59
A.6 结果	61
A.6.1 硬件和软件环境	61
A.6.2 模拟性能	61
A.6.3 渲染方法的应用	65
A.7 总结	67
A.7.1 未来工作	68
致谢	69
补充材料	69
参考文献	69

摘要

使用无网格方法的流体模拟已成为解决需要处理大变形的力学问题的可靠方式，并且在海军工程、机械工程、电影和游戏等许多应用中变得非常流行。当前

的挑战是实时模拟和渲染流体。本工作的贡献包括：SPH 方法的 GPU 实现，与参考的多核 CPU 实现相比，平均加速了 11.55 倍，能够以大约 3 帧每秒的速度模拟 100 万个粒子；此外，提出了一种基于光线追踪的渲染解决方案，可以用来渲染任何基于粒子的流体模拟方法，它能够以交互式速率渲染大量粒子，例如，在 1920 x 1920 像素分辨率下，以 2.7 帧每秒的速度渲染 200 万个粒子。

A.1 引言

海军和机械工程中的一些流体动力学问题需要以高数值精度进行模拟。这类模拟的经典方法是有限元方法（FEM），它能够有效处理绝大多数模拟问题，但在存在大变形和边界区域的情况下会变得效率低下。

为了克服这些挑战，可以使用无网格方法，如平滑粒子流体动力学（SPH）^[1] 和移动粒子半隐式（MPS）方法^[2]。这些技术可以使用离散数量的粒子系统高效地模拟流体，并在不需要使用网格的情况下求解纳维-斯托克斯运动方程。这使得在传统基于网格的方法变得非常复杂的情况下，这些方法具有高度的灵活性^[3]。根据 Marrone 等人^[4] 的说法，SPH 方法可以在不损失体积和不需要创建任何网格的情况下模拟剧烈冲击。

流体模拟在游戏产业中非常普遍。例如，Unity 游戏引擎拥有自己的基于网格的水模拟器^[5]，而 Unreal 引擎则使用粒子系统来模拟流体动力学^[6]。还有一些游戏更倾向于创建自己的流体模拟器，如 PixelJunk Shooter 2，它拥有自己的流体引擎来模拟不同密度的流体（如水和岩浆）以及流体之间的热力学^[7]。这些方法可以产生高质量的视觉结果，但在数值上与现实相去甚远。

在电影制作行业中，基于粒子的方法非常常见。例如，Houdini-FX^[8] 软件使用流体隐式粒子方法（FLIP）^[9] 来模拟水和粘性流体，这是一种更快的方法，但数值上不够精确。另一个例子是 RealFlow^[10] 软件，它提供了两种流体模拟方法：基于位置的动力学方法（PBD），这是一种更快的解决方案^[11]，以及 SPH，它速度慢但非常精确^[12]。

SPH 方法已在游戏^[13] 和电影^[14] 产业中得到应用，并且该方法已成为动画产业中最流行的基于粒子的方法之一^[15]。它展现了高度的实现灵活性和数值精度，为用户提供了高质量的体验，但同时也面临着为大量粒子实现实时解决方案的挑战^[14]。

在这些行业中，一个巨大的挑战是真实地渲染模拟出的流体。为了可视化模拟结果，有必要使用被识别为自由表面的粒子重建表面^[16]。渲染结果可以使用多种方法，如直接渲染^[17]、3D 标量场^[16]、体渲染^[18] 或屏幕空间方法^[13]。

这项工作专注于屏幕空间方法（Screen Space Approach），它面临两个主要挑战：找到最佳函数来创建平滑表面^[19]以及在流体表面创建逼真的照明效果^[20]。尽管最近有工作实现了实时高质量渲染，但这些方法仍然无法提供适用于所有方法的解决方案。例如，Xiao 等人的方法^[20]依赖于粒子的质量来重建表面，需要修改才能适用于无质量的方法，如移动粒子半隐式方法^[21]，而且屏幕空间方法尚未整合到全局照明算法中。

本工作的主要贡献包括：

1. 提供了一个 SPH GPU 实现，与并行 CPU 实现相比，平均加速比为 11.55。
2. 基于光线追踪的渲染解决方案，该方案：
 - (a) 可以集成到任何光线追踪引擎中；
 - (b) 保持了清晰的特征；
 - (c) 渲染基于光线追踪的反射和折射；
 - (d) 能够以交互式速率渲染大量粒子，例如，在 1920 x 1920 像素分辨率下，以 2.7 帧每秒的速度渲染粒子；
 - (e) 可以用于渲染任何基于粒子的方法，无需修改渲染的任何步骤。

尽管这种方法在性能上与 Xiao 等人^[20]的最新工作不匹配，但它仅依赖于使用粒子位置创建的球体，因此可以用于渲染任何基于粒子的方法，无需修改渲染的任何步骤，而最新技术则同时使用粒子质量和 SPH 公式来重建表面。

由于我们的方法能够以交互式速率渲染大量粒子，一个应用是将其用于增强现实或虚拟现实中，这需要达到交互式帧率，正如 Zhang 和 Liu^[22]的工作所示。他们的工作以 20 帧每秒的速度运行 10 万个粒子的模拟，但它仅使用 Marching Cubes 算法渲染表面。而我们的算法能够以 10 帧每秒的速度渲染这么多粒子，并采用基于光线追踪的透明渲染，因此可以产生更真实的渲染结果。

这种方法的另一个用途是作为预览可视化，如 Xiao 等人^[20]所述。像 3DS Max 和 Realflow 这样的软件可以生成更逼真的流体表面，但它们需要数分钟来处理大量粒子。因此，为了避免动画师在等待离线渲染提供最终图像时浪费大量时间，他/她可以使用我们提出的预览方法来可视化场景的预览。

下一节将讨论基于 SPH 流体模拟的最新技术以及如何渲染它们的结果。第 A.3 节描述了 SPH 技术及其控制方程。同时，本节还描述了在这项工作中用于渲染模拟结果的方法。之后，第 A.4 节详细说明了本工作中使用的实现和工具。第 A.5 节描述了本工作中进行的测试案例。第 A.6 节分析并讨论了关于 GPU 加速和渲染视觉质量及性能的所得结果。最后，在第 A.7 节中，将讨论结论，展示贡献，并探索未来的可能性和增强。

A.2 最新技术

本节介绍了基于粒子模拟的 SPH 方法和流体渲染技术的最新研究进展。

A.2.1 光滑粒子动力学方法

SPH 方法最初由^[23]和^[1]引入，用于模拟天文现象。从那时起，它已经被广泛扩展到模拟流体^[24]、^[25]甚至固体行为^[26]，主要关注点是那些可能限制基于网格方法模拟应用的方面，例如大变形。

一种对原始 SPH 方法的直接改进是应用于弱可压缩流体。在这类流体中，压力可以通过状态方程来计算。论文^[27-30]展示了弱可压缩（WCSPH）和真正不可压缩方法（ISPH）的实现之间的比较，在这些方法中应用了 Cummins 和 Rudman^[31]引入的技术。

Shadloo 等人^[28]指出，与 ISPH 相比，WCSPH 编程更为简便，并且粒子分布更加有序。由于这些原因，WCSPH 已成为使用 SPH 解决线性动量平衡方程的最常用方法。ISPH 在解决高雷诺数的湍流流体流动时具有更好的稳定性，而 WCSPH 则因密度变化大而更适合低湍流流动。ISPH 方法还提供了更准确的压力场计算，但需要求解线性系统，这导致大量粒子的计算成本高。因此，Violeau 和 Rogers^[32]指出，未来 WCSPH 将需要辅以更鲁棒的公式，以提高不可压缩方法的精度。

在真正的不可压缩方法中，密度是通过泊松方程来计算的。这个方程可以由一个稀疏线性系统来表示。根据参考文献^[33-36]，这类方法可以生成更精确的解，但需要更多的计算时间。

最初的（天体物理学）平滑粒子流体动力学（SPH）方法与新型基于粒子的流体模拟之间的主要区别在于边界条件的包含。像参考文献^[37-41]中提出的那样，有效率的方法来处理这些问题。另一种可能遭受不稳定问题边界是多相场景中两种或更多不同流体之间的界面。参考文献^[42-43]中的工作展示了如何处理这个问题。

除了存在于流体中的可压缩性因子和边界条件之外，还可以根据正在解决的问题类型引入一些其他元素来影响流体行为，例如粘度、是否存在湍流、平滑函数的类型等。一些研究根据正在关注的问题提出了调整粘度的结果^[44-48]。

Vieira-e-Silva 等人^[49]开发了一种仅依赖于 XSPH 公式来模拟粘度并避免粒子穿透问题（边界条件）^[50]的 SPH 公式。这种方法导致了一种小型 SPH 公式，具有相对较高的数值精度，由于其计算量小和使用 XSPH 方法容易控制粘度，因此可以用于交互式应用，从而形成了一种易于调节的方法，具有快速且计算量低的特点。

模拟湍流最常见的方法是将雷诺平均纳维-斯托克斯湍流（RANS）模型添加

到 SPH 方法中^[51]。值得注意的是，文献中发现的大多数无网格工作都涉及低雷诺数^[47,52-56]。最后这项工作讨论了这个值在文献中的常见用法。

平滑函数是 SPH 方法中的一个重要选择，因为它模拟了粒子之间如何根据它们之间的距离相互影响。为了解释平滑函数对无网格模拟的影响，一些研究^([47,57-61])讨论了在改变它们的平滑函数时行为的变化，评估了方法的准确性和稳定性。

不幸的是，与其他无网格方法类似，SPH 技术也存在稳定性问题。一些（基于粒子方法的主要问题）与边界处的数值误差有关，即在自由表面或与固体边界相互作用时^[62-67]。这些作品描述了每种特定的粒子方法为何会出现这些不稳定性问题。一些改进措施，有助于减轻不稳定性的影响，甚至提高模拟的准确性，已在^[34,38,68-70]的作品中实现。

另一个常见问题是与粒子之间的相互作用有关，更准确地说，当它们彼此靠得太近时会产生排斥应力，导致一种被称为拉伸不稳定的不稳定性^[71]。一些研究尝试通过其他方法^[48]来克服这个问题。

鉴于模拟的无网格特性，可以利用集群技术或通用图形处理器编程（GPGPU）技术来创建并行解决方案，以减少时间消耗^[72-76]。

在虚拟现实（VR）社区中，SPH 模拟已经被用于许多应用。Cirio 等人^[77]提出了一种六自由度（DoF）的触觉交互，使用 WCSPH 模拟流体，提供基于力的反馈，并能够实现每秒 90 帧（fps）的实时性能，对于 32,768 个粒子来说，这足以模拟煎饼面团或小碗中流体的行为。这项工作被扩展到处理熔化和冻结现象^[78]，以及与可变形和刚体的交互^[79]。

另一个针对虚拟现实（VR）社区的例子是 Pang 等人^[80]的工作，他们使用 PhysX 内置的基于 SPH 的流体求解器来模拟基于 VR 的手术模拟器中的出血效果，能够以 5000 个粒子实现 49 帧每秒（fps）的性能。此外，Wang 和 Wang^[81]提出了一种使用 SPH 和 FEM 与流体的触觉交互，以操作一艘用两只桨在流体中划行的独木舟。

对于游戏行业来说，使用 SPH 方法的一些工作专注于为交互式应用和实时应用模拟流体，始于 Müller 等人^[82]的工作。其他工作则专注于模拟具有不同属性和特征的流体，因此可以表示大多数流体类型和行为。如果应用程序将用于游戏目的，性能需要成为关注焦点，因此目标始终应该是至少接近实时运行的模拟；然而，流体还需要保持其物理属性，并尽可能连贯地呈现它们^[83-84]。

A.2.2 流体渲染

在 SPH 文献中，已经提出了许多方法来重建给定一组粒子的表面。一种可能的方法是使用 3D 标量场；通过计算一个核函数来定义液体表面，该函数将定义一个标量密度场。然后，使用 Marching Cubes 算法^[85]来生成这个 3D 场等值面的三角网格。标量场公式的选择是创建高质量表面的关键；最简单的选择是使用 *blobbies* 方法，也称为 *metaballs*^[86]，但这种方法可能会根据粒子分布创建表面凹陷。

使用基于彼此靠近的粒子的加权平均值的 3D 标量场，可以通过各向同性核^[87-89]或各向异性核^[16]来找到更平滑的表面，这可以在锐利特征和边缘处产生更好的视觉效果。但这种方法渲染流体需要几分钟，使得该渲染方法更适合离线应用。

渲染流体表面的第二种方法是采用显式方法，这种方法经常在欧拉背景下使用^[90]。在 SPH 文献中，可以找到一些使用这种方法的工作，例如^[91]，它利用粒子模拟的信息来改变表面位置。这些方法有很高的内存消耗，为了避免这个问题，可以使用点绘制^[92]和光线与 *metaballs* 等值面相交的方法^[93]。

A.2.2.1 相关工作

为了实时渲染流体粒子，van der Laan 等人^[13]提出了一种屏幕空间方法，该方法使用球体的位置作为基本图形，并将它们渲染为点精灵，以从深度图重建流体表面。一旦重建，表面可能看起来像果酱。为了克服这个问题，van der Laan 等人提出了对深度图进行平滑以创建更连贯的表面，并且可以为此目的使用许多算法：二项式滤波器^[94]、高斯滤波器^[13]、曲率流^[13,95]和后平滑滤波器^[19]。在表面平滑之后，使用基于菲涅尔的反射和折射以及基于厚度的颜色衰减来计算流体颜色。

van der Laan 等人^[13]的工作是本研究提出的基于屏幕空间方法的基础。但与^[13]中计算基于菲涅尔的折射和基于厚度的颜色衰减不同，本文是从光线追踪（Ray Tracing）的角度计算流体表面的最终颜色。

为了渲染喷雾和泡沫，Ihmsen 等人^[96]提出了一种后处理方法，该方法在预先渲染的流体体积中采样视线射线，并根据基于体积密度的函数减弱流体颜色。体积密度值决定了在体积半径内是否有任何散射粒子，并且对于喷雾和泡沫使用不同的半径。这种方法使用 Mental Ray^[97]实现，并提升了流体渲染的真实性。但是这种方法渲染单帧需要几秒钟，尤其是当粒子数量很大时。这与本文追求视觉质量和性能平衡的工作的目标不同。

van der Laan 等人^[13]的工作可以生成高质量的流体表面重建，但使用的是简单的照明模型。为了获得更真实的结果，可以像 Zirr 和 Dachsbaecher^[98]的工作那样

使用光线追踪的折射和反射。他们提出了一种使用光线投射折射和反射的视图自适应高分辨率体素化 SPH 粒子数据的方法，能够以交互速率提供高质量的渲染解决方案。尽管这种高视觉质量的结果能够在交互时间内渲染流体的内部体积，但这种方法并不十分稳定，因为渲染时间即使对于相同数量的粒子也会因流体表面配置的不同而急剧变化。

Xiao 等人^[20]提出了一种结合粒子溅射（Particle Splatting）、光线投射（Ray Casting）和表面法线估计的方法。为了重建流体表面，基于光线投射和使用依赖于质量的 SPH（平滑粒子流体动力学）公式计算的密度属性值构建了一个等值面。主成分分析（PCA）被用来计算表面法线，它对噪声具有鲁棒性，并且比其他法线估计方法更平滑。该方法最终使用 van der Laan 等人^[13]的方法结合基于光线投射的折射来确定流体的颜色。该方法完全在 GPU 上使用 CUDA 8.0 和 OpenGL 着色语言实现，能够以每秒 10 帧的速度在全高清分辨率（1920 x 1080）下渲染 2,027,776 个流体粒子。但与我们的工作不同的是，由于其依赖于质量的表面重建，这些方法无法在不做任何修改的情况下渲染无质量的方法，如 MPS。

A.3 方法

本节介绍了基于 Vieira-e-Silva 等人^[49]的工作的 SPH（平滑粒子流体动力学）方法，该方法在本研究中在 CPU 和 GPU 上进行了并行化处理，以及基于光线追踪的解决方案来渲染流体模拟结果。

A.3.1 平滑粒子流体动力学方法

SPH（Smoothed Particle Hydrodynamics，平滑粒子流体动力学）是一种拉格朗日方法，主要用于模拟流体动力学问题，解决纳维-斯托克斯方程：

$$\frac{d\mathbf{u}}{dt} = -\frac{1}{\rho} \nabla p + \frac{1}{\rho} \nabla \cdot \boldsymbol{\square} + \mathbf{F}_{ext} \quad (\text{A.1})$$

其中 \mathbf{u} 是流体的速度， t 是时间， ρ 是流体的密度， $\boldsymbol{\square}$ 是流体系统的压力， $\boldsymbol{\tau}$ 是偏应力张量， \mathbf{F}_{ext} 是流体系统中的外力函数。

在本工作中使用的核函数是大多数 SPH 公式中使用的核函数，这是一个三次样条核函数，如下所示：

$$W(r, h) = \begin{cases} \frac{2}{3} - r^2 + \frac{1}{2}r^5 & 0 \leq r \leq 1 \\ \frac{1}{6}(2-r)^3 & 1 < r \leq 2 \\ 0 & otherwise \end{cases} \quad (\text{A.2})$$

对于一维、二维、三维问题，其中的 α_d 分别为 $1/h, 5/7\pi h^2, 3/2\pi h^3$

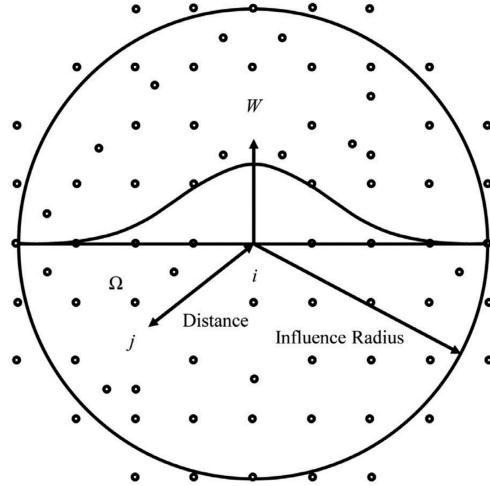


图 A.1 对二维问题的粒子近似

公式 (A.2) 类似于高斯函数，但其二阶导数有一些接近线性函数的结果，可能会导致横向模式不稳定性^[53]。

SPH 方法的第二部分是将连续的流体动力学问题近似为一系列粒子。一定量的流体被描述为有限数量的粒子。每个位于位置 x 的粒子具有速度 u 、质量 m 、密度 ρ 、粘度 μ 和一个影响半径 h ，这个半径描述了一个粒子对其邻居的影响^[50]，如图A.1所示。

一个粒子的影响半径定义了一个领域，这是一个影响区域（在 2D 中）或一个影响体积（在 3D 中）^[57]。假设在粒子 a 的领域内存在两个不同的粒子，那么离粒子 a 更近的一个粒子受到的影响比另一个粒子更大。可以为系统中的每个粒子分配不同影响半径，并且领域可以有不同的形状，正如 Liu 和 Liu^[57]所建议的。

要从单个粒子获取邻居，使用两个粒子之间的距离的几何比较。如果一对粒子之间的距离小于影响领域，那么这些粒子就是彼此的邻居，一个粒子的影响半径可以计算为 γdx ，其中 γ 是一个常数，通常选择在 1.0 和 1.3 之间， dx 是粒子的初始间距，根据 Monaghan 和 Kajtar 的工作^[41]。在这项工作中， γ 是 1.0，因为它是 DualSPHysics 代码中的默认值，并且它被用来保持一个小的影响半径值，这导致了少量的粒子邻居，从而创建了一个更快的方法计算，结果没有表现出任何明显的视觉不稳定性，此外，DualSPHysics 指南^[99]指出， γ 值大于 1.0 更适合于波动传播测试案例，这些案例在本工作中并不存在。

邻域搜索是一个潜在的耗时步骤，通常使用加速的空间访问结构进行优化，如均匀网格或八叉树，而不是简单的蛮力搜索^[100]。

在离散公式中，插值计算可以被定义为

$$f(x_i) = \sum_{j=1}^N \frac{m_j}{\rho_j} f(x_j) W(x - x_j, h) \quad (\text{A.3})$$

其中 N 是一个粒子的邻居数量， j 是邻居粒子的索引。

使用相同的方法，散度算子可以应用为：

$$\nabla \cdot f(x_i) = - \sum_{j=1}^N \frac{m_j}{\rho_j} f(x_j) \cdot \nabla W(x - x_j, h) \quad (\text{A.4})$$

而且对于梯度算子：

$$\nabla f(x_i) = \sum_{j=1}^N \frac{m_j}{\rho_j} f(x_j) \nabla W(x - x_j, h) \quad (\text{A.5})$$

这些导数可能会导致较大的数值误差。因此，为了克服这些限制，可以进行一些代数运算，并找到导数的稳定形式^[62]，如下所示：

$$\nabla \cdot f(x_i) = \sum_{j=1}^N m_j \left(\frac{f(x_j)}{\rho_j^2} + \frac{f(x_i)}{\rho_i^2} \right) \cdot \nabla W(x_i - x_j, h) \quad (\text{A.6})$$

纳维-斯托克斯方程描述了流体运动的三个主要组成部分：压力、粘度和外力。WCSPH（弱可压缩平滑粒子流体动力学）通过将流体视为一个弱可压缩系统来解决流体运动问题，这基于这样一个事实：每一个不可压缩流体都稍微具有可压缩性，因此，该方法模拟了一个准不可压缩方程来模拟仿真^[71]。

为了计算这些组成部分，第一步是确定粒子的密度，可以使用连续性方程^[57]来计算，如下所示：

$$\frac{d\rho_i}{dt} = \sum_j m_j (\mathbf{u}_i - \mathbf{u}_j) \nabla W_{ij} \quad (\text{A.7})$$

在计算了系统中粒子的密度之后，下一步是计算它们的压力。对于一个弱可压缩系统，每当必须强制执行低密度变化时，可以使用 Tait 方程^[49]，如下所示：

$$P_i = B \left[\frac{\rho_i^\gamma}{\rho_0} - 1 \right] \quad (\text{A.8})$$

其中 k_p 和 B 是压力常数， ρ_0 是流体的静止密度， γ 是一个常数，通常其值为 7^[101]。

压力力通常使用方程 (A.6) 的导数表达式来计算，结果为：

$$\frac{1}{\rho_i} \nabla P_i = - \sum_{j=1}^N m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} \right) \nabla W_{ij} \quad (\text{A.9})$$

这种方法确保了两个粒子之间的模块等价性，并保持了线性和角动量的守恒，从

而使得模拟更加稳定^[50]。

在系统中模拟粘度的一个简单方法是使用 XSPH (扩展平滑粒子流体动力学) 方法。这种公式在计算上比其他方法更便宜，也更容易调整，因为它只使用一个可调参数^[101]。这种方法通过降低粒子速度^[101]，迫使彼此靠近的粒子以接近的速度移动，从而大致保持角动量和线动量的守恒，使用的是：

$$\mathbf{u}'_i = \mathbf{u}_i + \epsilon \sum_{j=1} m_b \frac{\mathbf{u}_i - \mathbf{u}_j}{\bar{\rho}_j} W_{ij} \quad (\text{A.10})$$

其中 ϵ 是 XSPH 方法的可调参数。

最后，纳维-斯托克斯控制方程中的最终项与作用于系统的外力有关，通常由重力表示。粒子的新速度和位置是使用简单的一阶欧拉时间积分^[101]来计算的，描述如下：

$$\mathbf{u}_i^{t+1} = \mathbf{u}_i^t + \mathbf{a}_i^t t \quad (\text{A.11})$$

和

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{u}_i^{t+1} t \quad (\text{A.12})$$

其中 a_i 是粒子 i 的加速度。

在这项工作中，使用了 Vieira-e-Silva 等人^[49]提出的 WCSPH (弱可压缩平滑粒子流体动力学) 方法，它使用 Tait 方程 (A.8) 来计算压力。压力是通过方程 (A.9) 计算的，XSPH 作为粘度因子和边界条件。

实现的 SPH 方法流程可以在图A.2中找到。

A.3.2 光线追踪渲染解决方案

为了获得更真实的流体渲染效果，文章开发了一种基于光线追踪的渲染解决方案，可以总结为四个步骤。首先，场景中的每个粒子被渲染为一个球体，并将深度图存储到一个缓冲区中，然后，使用 van der Laan 等人^[13]提出的方法对深度图进行平滑处理。利用深度差异计算每个表面点的法线，最后，使用流体颜色以及基于光线投射的反射和折射的结果计算最终颜色。图A.3展示了所提出方法的概览，以下将详细解释渲染解决方案的每一步。

A.3.2.1 流体表面重建

c 在所提出的方法中，每个粒子被表示为一个半径等于初始间距距离的球体。这个半径值与 SPH 方法的平滑长度相同，能够避免在初始表面重建时出现孔洞，而且视觉结果表明，在模拟过程中避免了孔洞的出现。

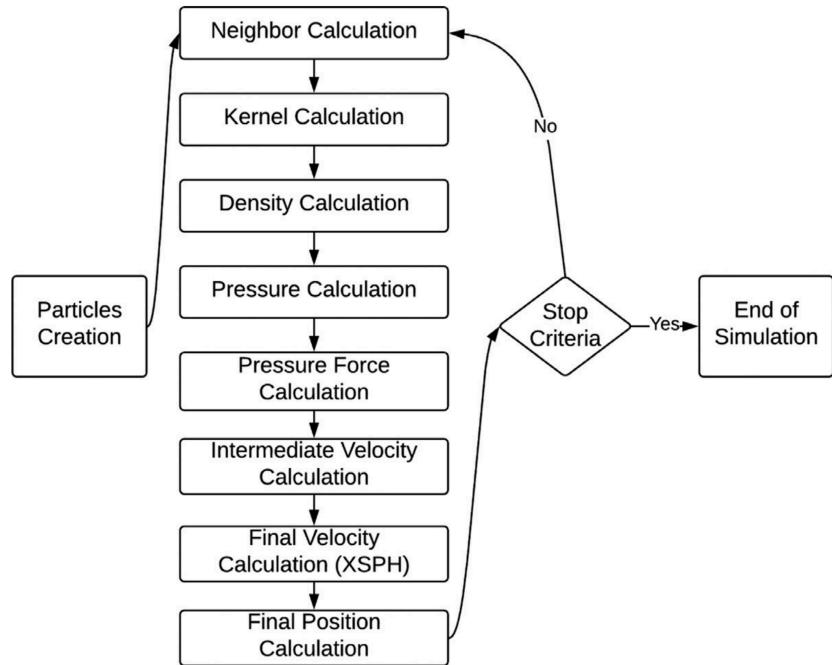


图 A.2 平滑粒子流体动力学模拟流程

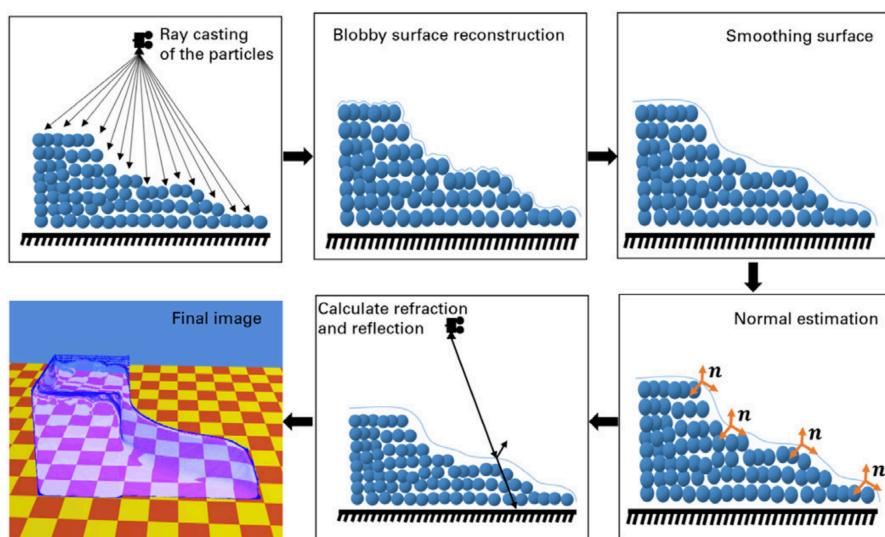


图 A.3 渲染方法总览

使用光线追踪器渲染球体后，得到的表面呈现出类似泡泡或果冻的外观。为了在不需要创建除流体粒子以外的任何其他结构的情况下创建一个更真实的表面，对表面应用了模糊算法，以最小化表面接近点之间的差异。

为了实现这个目的，使用了屏幕空间方法^[13]。第一步是创建场景的深度图；这个深度图是使用光线投射程序创建的，它保留了每个像素的最近命中值，这一步的结果可以在图A.4中看到。在创建了场景的深度图之后，使用平滑算法来为流体表面创建一个更真实的外观。基于这个新的深度图计算表面的法线，有了这个新的表面，渲染过程可以继续进行。

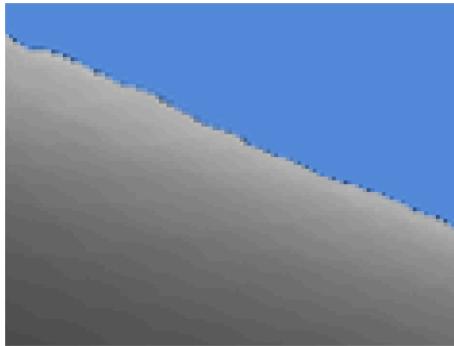


图 A.4 通过光线投射（Ray Casting）程序创建深度图

A.3.2.2 屏幕空间曲率流

为了获得更美观的结果，^[13]提出了一种基于^[102]的曲率流方法，该方法平滑了粒子间曲率的突然变化。由于视点是固定的，可以通过按曲率比例移动深度值 z 来应用平滑效果，定义如下：

$$\frac{\partial z}{\partial t} = H \quad (\text{A.13})$$

其中 t 是平滑时间步长， H 是平均曲率。

平均曲率定义为表面单位法线 n 的散度，如下所示：

$$2H = \nabla \cdot \hat{n} \quad (\text{A.14})$$

在视图空间中，一个点 P 在 V_x 和 V_y 上被映射到深度图中的一个值，通过反转投影变换来实现，定义如下：

$$\mathbf{P}(x, y) = \begin{bmatrix} (\frac{2x}{V_x} - 1)/F_x \\ (\frac{2y}{V_y} - 1)/F_y \\ 1 \end{bmatrix} z(x, y) = \begin{bmatrix} W_x \\ W_y \\ 1 \end{bmatrix} z(x, y) \quad (\text{A.15})$$

其中 V_x 和 V_y 是视口的尺寸, F_x 和 F_y 是在 x 和 y 方向上的焦距。

表面一个点的法线是通过在 x 和 y 方向上对 \mathbf{P} 的导数进行叉积计算得出的, 如方程 (A.16) 所示:

$$\mathbf{n}(x, y) = \frac{\partial \mathbf{P}}{\partial x} \times \frac{\partial \mathbf{P}}{\partial y} = \begin{bmatrix} C_x z + W_x \frac{\partial z}{\partial x} \\ W_y \frac{\partial z}{\partial x} \\ \frac{\partial z}{\partial x} \end{bmatrix} \times \begin{bmatrix} W_y \frac{\partial z}{\partial y} \\ C_y z + W_y \frac{\partial z}{\partial y} \\ \frac{\partial z}{\partial y} \end{bmatrix} \approx \begin{bmatrix} -C_y \frac{\partial z}{\partial x} \\ -C_x \frac{\partial z}{\partial y} \\ C_x C_y z \end{bmatrix} z \quad (\text{A.16})$$

其中 $C_x = \frac{2}{V_x F_x}$, $C_y = \frac{2}{V_y F_y}$, 并且为了简化计算, 忽略 W_x 和 W_y 这两个项, 因为它们对计算的贡献较小。

单位法线是通过以下方式计算的:

$$\hat{\mathbf{n}} = \frac{\mathbf{n}(x, y)}{||\mathbf{n}(x, y)||} = \frac{(-C_y \frac{\partial z}{\partial x}, -C_x \frac{\partial z}{\partial y}, C_x C_y z)^T}{\sqrt{D}} \quad (\text{A.17})$$

其中 $D = C_y^2 \left(\frac{\partial z}{\partial x} \right)^2 + C_x^2 \left(\frac{\partial z}{\partial y} \right)^2 + C_x^2 C_y^2 z^2$, 并将其代入方程 (A.13) 中, 以便以一种可以求导的方式表达 H 。

z 方向的散度分量总是零, 因为 z 是 x 和 y 的函数, 在 x 和 y 保持不变时, z 也保持不变。因此:

$$2H = \frac{\partial \hat{n}_x}{\partial x} + \frac{\partial \hat{n}_y}{\partial y} = \frac{C_y E_x + C_x E_y}{D^{3/2}} \quad (\text{A.18})$$

其中

$$E_x = \frac{1}{2} \frac{\partial z}{\partial x} \frac{\partial D}{\partial x} - \frac{\partial^2 z}{\partial x^2} D \quad (\text{A.19})$$

和

$$E_y = \frac{1}{2} \frac{\partial z}{\partial y} \frac{\partial D}{\partial y} - \frac{\partial^2 z}{\partial y^2} D \quad (\text{A.20})$$

使用简单的欧拉积分法对方程 (A.13) 进行时间积分, 以在每次迭代中改变深度值, 而 z 方向的导数是使用有限差分法计算的。为了创建一个更平滑的表面, 可以增加迭代次数, 这可能会导致计算时间增加。平滑处理的结果可以在图A.5中看到。

A.3.2.3 法线估计

某个点的视空间法线是通过深度图的有限差分来确定的。这种方法可能会在流体轮廓附近产生伪影, 因此, 在这种情况下, 差分是在相反方向计算的, 这可以通过最小的有限差分来检测^[13]。这种估计的结果可以在图A.6中看到。

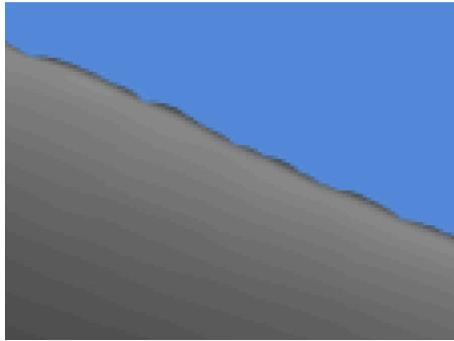


图 A.5 深度图模糊结果

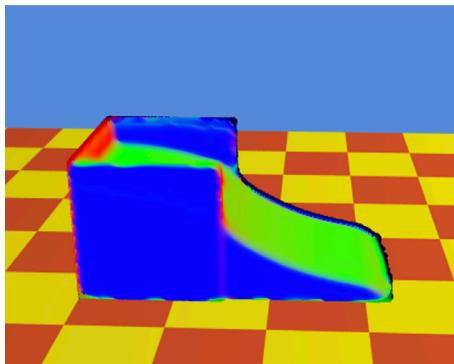


图 A.6 法线估计结果

A.3.2.4 渲染

在重建流体表面之后，流体被渲染为一个透明表面，光线的折射和反射都按照以下方式计算：

$$\mathbf{r} = 2(\mathbf{n} \cdot \mathbf{l})\mathbf{n} - \mathbf{l} \quad (\text{A.21})$$

和

$$R(\theta) = R_0 + (1 - R_0)(1 - \cos \theta)^5 \quad (\text{A.22})$$

其中 $R_0 = \left(\frac{n-1}{n+1}\right)^2$ 。为了实现更高的性能，只计算了单层折射，这不是一个物理上精确的模型，但可以提供高质量的可视化^[20]。最终结果可以在图A.7中看到。

A.4 实现

为了实现高性能，SPH 工具是在开源的 DualSPHysics 代码^[103]的基础上实现的，并且进行了几项修改以模拟由 Vieira-e-Silva 等人^[49]开发的 WCSPH 方法。为了实现基于光线追踪的流体渲染，使用了 NVIDIA 的 OptiX 光线追踪引擎^[104]，该引擎完全在 GPU 上使用 CUDA 编程模型^[105]进行光线追踪处理。OptiX 支持 Whitted 简单模型^[106]中解释的主要和次要光线，并支持反射、折射和阴影。

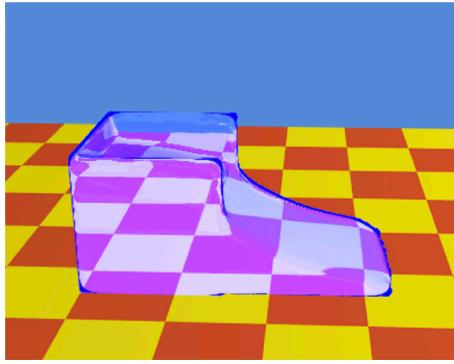


图 A.7 最终渲染结果

A.4.1 DualSPHysics

DualSPHysics 是一个开源项目, 其创建目的是为了鼓励其他研究人员研究 SPH (平滑粒子流体动力学), 并且它拥有由自由软件基金会发布的 GNU 通用公共许可证^[103]。代码适用于基于 CPU 和 GPU 的 SPH 模拟, 能够以数值稳定性和准确性计算流体行为, 并且已经被许多应用所使用^[107-109]。代码是用 C++ 编写的, 使用 OpenMP^[110]进行并行 CPU 实现, 以及 CUDA^[105]进行 GPU 实现。作者声称, 在有 300 万个粒子的场景中, GPU 实现相比于并行 CPU 实现的速度提升了高达 24 倍^[103]。

代码以集成的形式创建, 使得同一个应用程序可以在 CPU 或 GPU 上运行。主代码对于两种实现都有一个共同的核心, 只有少量的编码差异和特定的优化。例如, 所有的粒子交互循环都是使用一个线程来计算每个粒子的邻域施加的力, 但 CPU 代码使用 OpenMP 来并行化这个交互, 而 GPU 版本使用 CUDA。

DualSPHysics 实现的 SPH 代码可以分为三个主要阶段^[103]: (1) 邻域组织, (2) 粒子相互作用, 以及 (3) 时间积分。

在第一阶段, 使用单元链表 (CLL) 方法^[111]计算粒子的邻域。在 CLL 中, 域被划分为正方形单元 (2D) 或立方体单元 (3D), 边长是影响半径 (h) 的两倍, 粒子根据它们所属的单元被存储在列表中。

为了创建一个粒子的邻域, 只考虑相邻单元中的粒子作为潜在的邻居。值得注意的是, 并没有创建一个邻居列表, 而是根据粒子所属的单元对粒子列表进行重新排序。这种方法比创建一个真正的邻居粒子列表更快, 且消耗的内存更少^[111]。在第一阶段结束时, 使用粒子列表对每个数组进行重新排序。

第二阶段通过解决动量和连续性方程来计算相邻粒子之间的相互作用。如果两个粒子之间的距离小于 $2h$, 则它们之间会发生相互作用。

使用第二阶段的结果, 计算时间积分。在这个阶段, 计算新粒子的密度、速度和位置, 可以计算新的的时间步长, 将粒子信息存储在硬盘上, 并对数组进行排

序，使得同一单元内的粒子彼此靠近。

输入数据由一个名为 GenCase^[103]的工具读取，它读取一个包含模拟几何形状（流体和边界）的.xml 文件以及几个常数，如流体的参考密度、Tait 方程中使用的 gamma 值和重力值。可以使用 GenCase 创建多种几何形状，如代码A.1所示，它显示了创建一个球体所需的 XML 代码。初始粒子间距也由输入的.xml 文件定义。

Listing A.1 在输入文件中，通过其半径和中心点定义球体

```
<drawsphere radius="0.2">  
  
<point x="0.2" y="0.5" z="0.6" />  
  
</ drawsphere>
```

在每个时间步长之后，粒子的位置被存储到一个.xyz 文件中，该文件包含粒子位置的列表，或者存储到一个.vtk 文件中，后者还存储了每个粒子的密度、速度和压力。

A.4.1.1 对 DualSPHysics 的修改

在本节中，将解释对 DualSPHysics 代码所做的修改，以实现 Vieira-e-Silva 等人^[49]提出的 WCSPH 方法。

WCSPH。 WCSPH 方法已在 DualSPHysics v4.0^[103]的开源代码中实现，它使用网格来计算粒子的邻域，并使用 CUDA 来加速计算，能够模拟多达数百万粒子。

为了实现 WCSPH，对 DualSPHysics 代码的两个部分进行了修改：(1) 欧拉积分和 (2) XSPH 计算。

欧拉积分。 欧拉积分用于计算流体粒子的新位置，如方程 (A.12) 所示。粒子的位置和速度存储在数组中，这些数组已经由原始的 DualSPHysics 实现使用。在每个时间步长之后，粒子网格位置，由 Cell 表示，会更新，如代码A.2所示。第 3 行显示了使用欧拉积分计算新位置，而新的网格位置在第 8 行计算。

Listing A.2 欧拉积分在 GPU 上的伪代码，其中 CTE 表示常数

```
If (ParticleID = Fluid) // Fluid article  
    // Position update - Euler integration  
    ParticlePosition += Velrhop1 * dt;
```

```

dPosition = ParticlePosition - CTE.MinimumPosition;

// Stores cell and checks
Cell = dPosition / CTE.CellSize;
endif

```

XSPH 计算。 为了计算 XSPH，使用一个 CUDA 核心计算方程 (A.10) 的求和，该核心通过粒子邻域进行交互，并将结果存储。随后，另一个 CUDA 核心计算最终速度，如方程 (A.11) 所示，并将结果存储到速度数组中，如代码 A.3 所示。如果 ParticleID 是流体粒子，使用 XSPH 结果更新速度 Velhopnew，如第 5 行所示。

Listing A.3 XSPH (扩展平滑粒子流体动力学) 的 GPU 伪代码

```

If (ParticleID = Boundary) // Boundary particle
    Velhopnew = (0, velhop1.density);
Else // Fluid particle
    // Velocity update using the XSPH
    Velhopnew = Velhop - XSPH * Epsilon;
endif

```

A.4.2 OptiX

OptiX^[104]是由 NVIDIA 开发的一种通用光线追踪引擎，它提供了一个可编程的光线追踪流水线，具有轻量级的场景表示，并且可以用于渲染、人工智能、动画和科学可视化等领域。

OptiX 的主要特点包括：(a) 它专门针对光线追踪算法，并不嵌入到某些特定的流水线中，(b) 它可以用于基于渲染和非渲染的应用，(c) 该引擎抽象了光线批处理、重排序和加速结构的创建，(d) 该引擎调整使用的硬件的执行，(e) 它提供了一个灵活的节点图，允许组织场景以达到更高的性能结果。

使用 OptiX 的应用程序被组织成一系列程序，可以是八种类型：光线生成、相交、包围盒、最近命中、任意命中、未命中、异常和选择器访问。光线生成程序是光线追踪流水线的入口点，执行像素操作。这些操作可以是追踪相机的像素、计算烘焙照明，甚至是缓冲区执行像素操作。相交程序被创建来执行光线-几何体相交测试，它们也可以根据命中位置计算任何操作，为了提高相交测试的性能，测试是使用几何体的包围盒完成的，包围盒由包围盒程序计算。

最近命中程序在计算出最近相交时执行操作。这些操作通常是着色并将结果存储在缓冲区中，或者向场景中投射新的光线。

如果需要对每个光线-物体相交进行一些计算，则使用任意命中程序，对于没有相交的情况，调用未命中程序。最后，当系统发现异常条件，如没有可用内存或索引超出范围时，调用异常程序。选择器访问程序对节点图执行操作。

场景被表示为一个轻量级的图，可以有效地控制光线穿过场景。场景图由四个主要节点组成，组节点聚集其他节点，并与加速结构相关联。几何组节点包含物体的几何体和材质，而变换节点有一个 4×3 矩阵，对几何体执行仿射变换。最后，选择器节点用于对其子节点执行任何操作。

A.4.2.1 渲染实现

基于光线追踪的渲染实现由三个主要的光线生成程序组成。第一个是深度图的创建，这是通过从针孔相机向包含一系列代表流体粒子的球体的场景中投射光线来完成的。每个光线最近命中的值被存储在 `rtBuffer` 中。

为了创建一个更真实的流体表面，使用第 A.3.2.2 节中解释的技术对深度图进行平滑处理，并且为了达到高质量的结果，平滑处理执行了 50 次迭代。尽管使用了光线生成程序，但该程序并没有在场景中投射任何光线，而是对每个像素进行处理。

最后，最终的光线生成程序使用平滑深度图来计算这些点的法线，计算流体的最终颜色，并且它还追踪次级光线来计算折射和反射颜色。为了实现高性能和更真实的视觉质量，次级光线不击中流体，因此流体内部的粒子不能被可视化。

代码中使用了三个几何组，一个包含场景中的所有元素（背景 + 流体），第二个只包含流体的球体粒子，最后一个包含背景元素。第一组用于着色流体和背景，流体组用于创建深度图，背景组用于在光线击中流体后追踪场景的次级光线。这种划分是正确渲染流体所必需的，但它会占用大量内存。这样的划分之所以可能实现，是因为 OptiX 的节点表示是轻量级的。

我们使用了线性边界体积层次结构 (LBVH)^[112] 作为加速结构，它专注于构建速度。由于流体粒子的位置不断变化，这种结构是最佳可用选项。

A.5 测试案例

为了测试渲染性能和视觉质量，我们进行了两个测试案例。首先，我们构建了一个 3D 溃坝案例。大坝被构建为一个 3D 矩形域，在模拟开始时立即崩塌。流体柱的高度 H 为 0.3 米，长度 L 为 0.3 米，初始深度 D 为 0.3 米。域的底部宽度 L_b

为 1.2 米。这个测试案例的初始配置可以在图A.8中可视化。

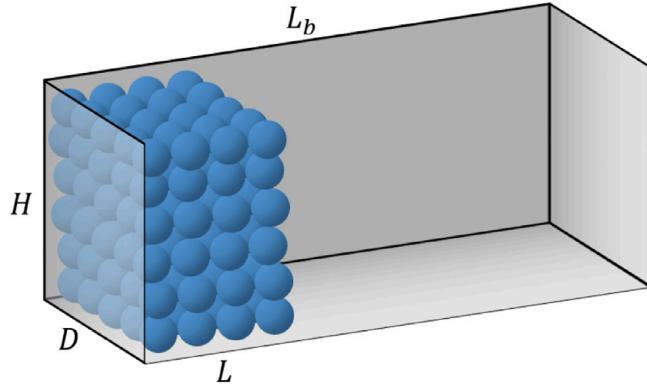


图 A.8 3D 漫坝的初始配置

此外，还构建了一个水滴场景，流体以 3D 矩形配置存在，初始高度 d 为 0.2 米，初始宽度 l 为 0.2 米，初始深度 d 为 0.2 米。水从高度 H_b 为 1 米处落下，初始速度为零，落入一个受限的 3D 矩形配置的流体中，其高度 H 为 0.5 米，长度 L 为 0.5 米，深度 D 为 0.5 米。这个测试案例的初始配置显示在图A.9中。

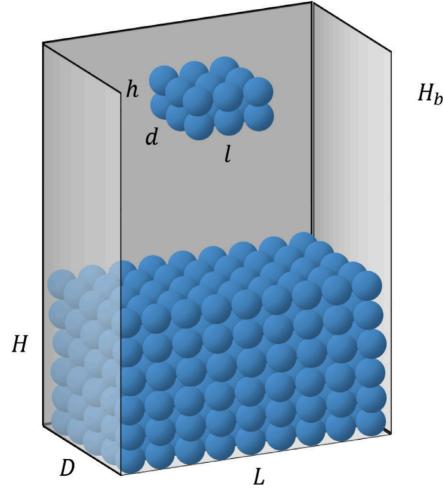


图 A.9 3D 的水滴案例初始配置

3D 漫坝由 100 万个流体粒子组成，水滴场景包含 50 万个流体粒子。

类似于 Xiao 等人^[20]的工作，为了分析算法的性能，进行了几项测试，改变了屏幕分辨率值（1024 x 1024、1440 x 1440 和 1920 x 1920），粒子数量（75 万、100 万和 200 万），并且还测试了静态和动态场景。在动态场景中，流体粒子改变了它们的位置，对于每一帧，必须重建加速结构。

这项工作中使用的粒子数量被选择为我们可以测量大规模场景的模拟和渲染性能，这些场景在文献中通常在 10^5 到 10^7 的数量级之间，如参考文献^[14,113-114]中所示。

测试使用了高达 200 万个粒子，以保持性能在交互水平。尽管对什么是被认为是交互式帧率的定义相当广泛，并且在不同的工作中有所不同，这项工作根据 Overby 等人^[115]、Wald 等人^[116]和 Kniss 等人^[117]的工作，将交互式速率视为高于 2 fps。

A.6 结果

在这一部分中，我们将展示并分析基于光线追踪的渲染解决方案的结果，涉及视觉质量和性能（帧率）。这些结果将与 Xiao 等人^[20]的工作中发现的结果进行比较，他们的作品能够在大约 200 万个粒子和 1920 x 1920 的分辨率下达到 8 帧每秒的性能。

A.6.1 硬件和软件环境

用于运行测试案例的计算机配备了一个 Intel Core i7-4790L @ 4.00 GHz 的 CPU，安装了 32 GB 的 RAM，并且搭载了 Windows 10 64 位操作系统（x64）。使用的 GPU 是 NVIDIA GeForce GTX 960，配备了 4 GB 的 RAM，并安装了 CUDA 7.5。

A.6.2 模拟性能

粒子数	CPU (ms)	GPU (ms)	加速比
100k	344	37	9.2
500k	2034	171	11.8
750k	3313	261	12.7
1M	4446	356	12.5

表 A.1 CPU 和 GPU 的计算时间以及每个测试案例各自的加速比。

表A.1显示了使用 OpenMP 的并行 CPU 实现和使用 CUDA 的并行 GPU 实现在五种不同粒子数量下的模拟时间。表中的时间代表了计算粒子相互作用、新粒子位置以及根据它们所属单元排序数组的计算时间。

从上述结果中，可以注意到 GPU 版本与并行 CPU 实现相比，平均加速比为 11.55。

模拟方法的实现可以分为六个步骤：密度计算、压力力计算、中间速度计算、XSPH、新位置计算和数组排序。如果比较每个步骤的时间，可以注意到密度计算、压力力计算和 XSPH 是最耗时的，每个步骤在 GPU 和 CPU 中大约消耗了处理时

间的三分之一，如图A.10所示。中间速度计算和新位置计算的时间没有出现在图表中，因为两者的计算时间都不到总计算时间的1%。



图 A.10 CPU 和 GPU 版本的性能时间分析

此外，重新排序计算占据了1%的计算时间，在GPU上的耗时比CPU多。然而，当粒子数量从10万增加到100万时，在GPU版本中，计算时间从1毫秒增加到3毫秒；在CPU版本中，计算时间从2毫秒增加到39毫秒。

A.6.2.1 渲染视觉质量

从视觉上看，这项技术可以通过基于光线追踪的反射和折射来呈现高质量的结果，如图A.11和图A.12所示。然而，该方法只能渲染流体的单层，这在物理上是不准确的，也无法可视化流体的内部体积。

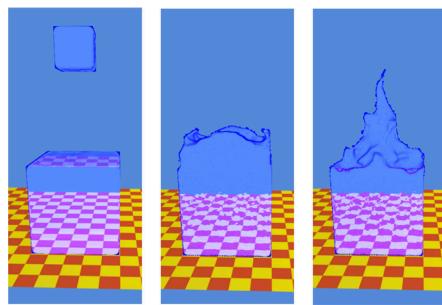


图 A.11 水掉落场景的渲染结果

平滑方法还能够保留尖锐特征，如立方体的角落，如图A.11所示，并且渲染方法可以集成到许多基于粒子的方法中，因为它只需要粒子位置作为输入，不需要提供模拟方法的信息，如密度或等值线。

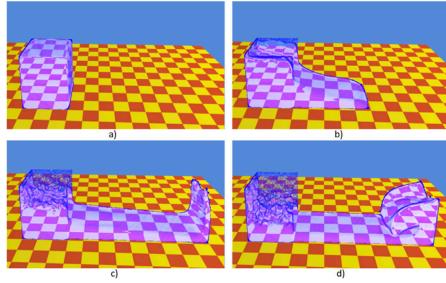


图 A.12 溃坝的渲染结果

A.6.2.2 性能

为了在视觉质量与高计算性能之间取得最佳平衡，我们比较了不同数量的平滑迭代在视觉质量与帧率（fps）方面的表现，使用的是滴水测试案例，结果可以在图A.13和图A.14中找到。

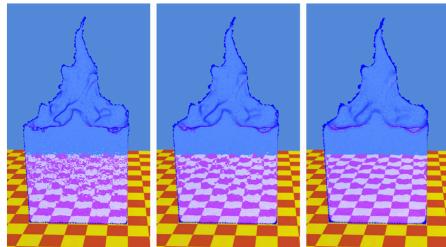


图 A.13 使用不同数量的平滑迭代进行渲染的结果：25 次（左），50 次（中）和 100 次（右）。

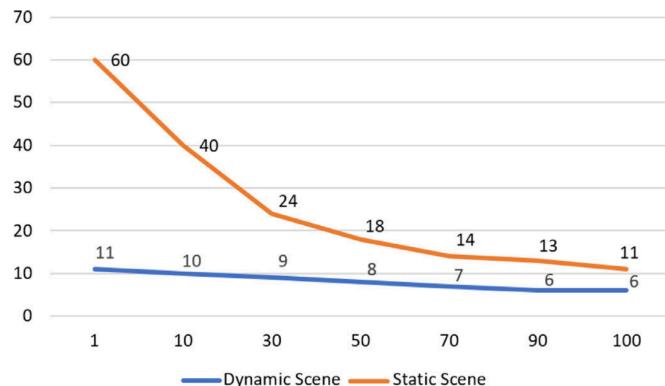


图 A.14 根据平滑迭代次数变化的帧率（fps）。测试是在 500,000 个粒子和 1000 x 1000 像素的图像分辨率下进行的

随着平滑迭代次数的增加，流体表面趋于更加平滑，这使得折射能够清晰地被可视化，但计算性能较低。通过比较图A.13（左）和图A.13（右），可以发现第一个图比第二个图结果更嘈杂，但性能（帧率）更好。

平滑迭代次数较少时产生的噪声会导致表面出现大量凸起和表面法线的误计算，而表面法线被用来计算折射光线。

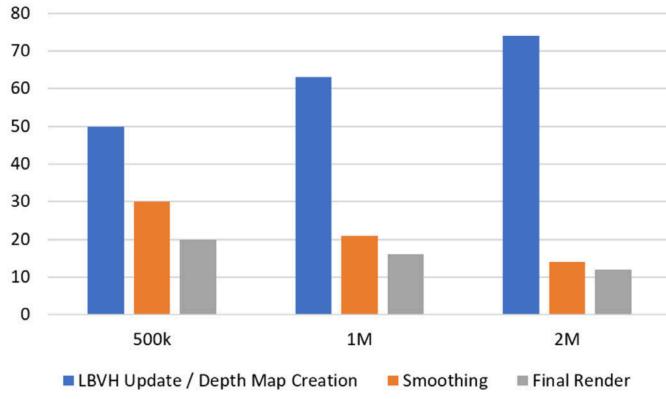


图 A.15 渲染百分比时间与粒子数量之间的比较

随着迭代次数的增加，无论是静态场景还是动态场景，性能（帧率）都会下降，如图A.14所示。从图表中可以注意到静态和动态场景之间存在很大的性能差异。这是因为每帧都需要更新加速结构。此外，对于相同的配置，静态场景的性能下降斜率比动态场景更大，这表明平滑过程对时间消耗的贡献不大，性能差异主要依赖于 LBVH（线性边界体积层次结构）的更新。

为了验证这个假设，进行了渲染百分比时间和粒子数量之间的比较，揭示了 LBVH 更新和深度图创建消耗了大部分计算时间，如图A.15所示。LBVH 更新和深度图创建是一起计算的，因为 OptiX 在第一个程序（深度图创建）的同时更新加速结构。

为了在视觉质量和性能之间取得最佳平衡，经验性地选择了 50 次平滑迭代，这产生了一个低噪声水平的流体表面和良好的透明度质量，能够在静态场景中以 1000×1000 分辨率、18 帧每秒的速度渲染由 50 万个粒子组成的流体，在动态场景中以 8 帧每秒的速度渲染，这是 Xiao 等人^[20]工作中使用的分辨率之一。如果应用不需要交互式性能率，可以使用较高数量的平滑迭代来创建更真实的表面。

为了分析渲染性能，进行了多项测试，变化图像分辨率、粒子数量和场景类型（静态或动态），如表A.2所示。随着分辨率值或粒子数量的增加，系统的性能会降低。但即使是需要更新加速结构的动态场景，渲染解决方案也能以交互式速率创建最终图像。

	750k 静态	750k 动态	1M 静态	1M 动态	2M 静态	2M 动态
1024×1024	23	5	17	4	18	3
1440×1440	17	5	17	4	12	2
1920×1920	11	4	8	3	8	2

表 A.2 静态和动态场景下，给定图像分辨率和粒子数量的帧率 (fps) 变化。

A.6.2.3 和相关工作的比较

我们所提出的方法以 van der Laan^[13]的工作为基础，使用平滑过程进行表面重建以创建一个逼真的表面，但与最新技术的工作不同，表面深度是利用光线投射的命中距离信息计算的，而反射和折射都是使用光线追踪方法而不是基于菲涅尔的计算来计算的。这样，渲染结果比 van der Laan^[13]的工作更具物理基础，但计算成本更高。

Zirr 和 Dachsbacher^[98]的工作能够渲染流体体积，而我们的方法只能渲染单层。在性能方面，这两项工作都能达到交互式速率，具体取决于粒子数量和屏幕分辨率。例如，他们的方法能够在 1920 x 1080 的屏幕分辨率下以大约 4 帧每秒的速度渲染 200 万个粒子，而我们的工作在 1920 x 1920 的屏幕分辨率下以 2 帧每秒的速度渲染相同数量的流体。此外，这两项工作都只能渲染单一流体。

与 Ihmsen 等人^[96]的工作相比，我们所提出的方法不能渲染泡沫和喷雾。与他们的工作不同，他们以离线方式渲染无光线折射的流体（用时 27 秒渲染单帧画面，包含 200 万个粒子），我们的方法以交互式速率渲染基于光线追踪的拥有透明度的流体。

最后，与 Xiao 等人^[20]的工作相比，我们的平滑方法也能保留尖锐特征，如图A.11所示，并且如 van der Laan 等人^[13]所述，这种平滑方法比双边高斯平滑和高斯模糊产生更好的结果。

对于 1920 x 1920 的屏幕分辨率和包含 200 万个粒子的案例，我们的方法以 2 帧每秒的速度进行渲染，而他们的工作大约有 8 帧每秒的性能。然而，最新技术的工作依赖于粒子质量来重建等值面，而我们的方法仅依赖于使用粒子位置创建的球体，因此它可以集成到许多基于粒子的方法中，如 SPH、PIC^[118]和无质量方法 MPS^[64]，无需修改任何渲染步骤。

A.6.3 渲染方法的应用

一种应用是使用我们的方法在增强现实或虚拟现实中渲染流体，这需要交互式性能。Huang 等人^[119]提出了一个基于 GPU 的框架，将 SPH 方法与真实的虚拟现实环境结合使用，降低了 SPH 方法的计算成本，并且实现了基于直方图金字塔的并行版本的步进立方体技术来重建流体表面，并以透明外观渲染最终结果。这种方法能够以基本的基于着色器的透明度在 9 帧每秒的速度渲染 60k 个流体粒子，而本文提出的方法能够以 3 帧每秒的速度渲染 100 万流体粒子，并使用更真实的基于光线追踪的透明度模型。

Tse 等人^[120]创建了一个触觉接口，以协助本科牙科学生的培训，并使用 SPH

方法模拟牙科填充。这项工作能够以 450Hz 的速率渲染多达 7k 粒子，但仅将粒子显示为球体，没有表面重建或透明外观。Zhang 和 Liu^[22]提出了一种多相流体的触觉交互，模拟并渲染基于粒子的流体，使用步进立方体算法以 20 帧每秒的速度渲染 10 万个粒子，渲染出不透明的表面。我们的方法可以用于这个应用，能够以 10 帧每秒的速度渲染 10 万个粒子，并且使用基于光线追踪的透明渲染解决方案。尽管我们的工作与 Zhang 和 Liu^[22]的工作以及 Tse 等人^[120]的工作的焦点不同，但这种比较表明，我们的渲染解决方案可以集成到依赖于交互式流体渲染的触觉解决方案中。

在 Pang 等人^[80]的工作中，使用了内置的基于 PhysX 的 SPH 流体求解器来模拟出血现象，使用步进立方体算法重建表面并以不透明渲染可视化网格。与其重建网格来可视化结果，我们所提出的解决方案可以用来仅用粒子位置以交互式速率渲染表面。

这种方法的另一个用途是作为预览可视化，如 Xiao 等人^[20]所述。像 3DS Max、Maya、Blender 和 Realflow 这样的商业软件可以生成更真实的流体表面，但处理大量粒子的过程可能需要数分钟。使用我们所提出的方法，与离线渲染相比，来获取场景的预览可以节省大量时间。

为了验证这种应用，我们使用 3DS Max 2019 和 Realflow 10 在视觉质量和性能方面进行了评估，使用的是 75 万、100 万和 200 万个粒子的 3D 溃坝测试案例，图像分辨率为 1024 x 1024。3DS Max 2019 有一个流体模拟模块，粒子的渲染由 Arnold Render 执行，最大折射深度光线值为 3，最大反射深度光线值为 1，没有阴影，每个像素样本数（spp）为 1。

Realflow 使用 SPH 方法^[12]来模拟流体，并且有一个由 Maxwell Render 执行的渲染模块，该模块不向用户提供太多的照明控制，如最大深度光线和每个像素样本数，但有一系列预存储的材质，测试是使用没有阴影的水材料进行的。

这两种软件和所提出的解决方案的表现可以在表A.16中找到。3DS Max 和 Realflow 的渲染性能涉及表面重建，将流体粒子转化为表面网格，以及照明算法。从视觉上看，商业软件能够比我们的方法创建更真实的图像，如图A.16所示，但计算成本要高得多。结果证实，我们所提出的渲染方法可以用作预览渲染，但也表明这些商业软件的渲染性能不能用于实时或交互式应用，而我们的解决方案具有适用于那种应用的兼容性能。

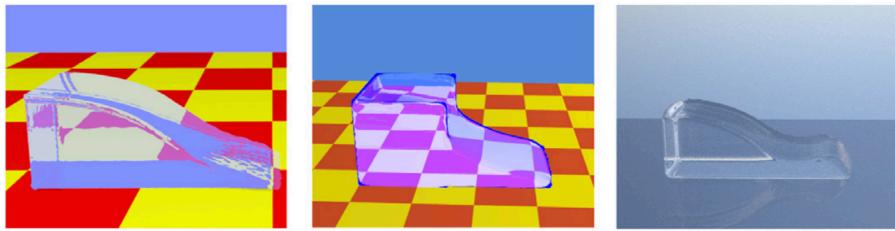


图 A.16 3DS Max 2019 (左) 的渲染结果、我们的方法 (中) 以及 Realflow (右) 对于拥有 100 万个粒子的流体。由于每个商业软件都使用不同的基于粒子的方法来模拟流体，输入粒子的位置可能彼此不同

渲染方法	粒子数量		
	750k	1M	2M
我们提出的方法	0.2	0.25	0.33
3DS Max 2019	74	100	211
Realflow 10	17	17	20

表 A.3 与商业软件相比，我们的方法在不同粒子数量下渲染时间

A.7 总结

本工作的第一个贡献是实现了由 Vieira-e-Silva 等人^[49]开发的 SPH 方法的 GPU 版本。这个实现使用了 DualSPHysics 开源代码^[103]，与并行 CPU 实现相比，平均加速比为 11.55，并且能够以大约 3 帧每秒（仅模拟）模拟高达 100 万个粒子。

此外，我们提出了一种基于光线追踪的渲染解决方案，能够交互式地渲染大量粒子，例如，在 1920 x 1920 像素分辨率下，以 3 帧每秒（仅渲染）的速度渲染 100 万个粒子。我们所提出的渲染解决方案以粒子位置为输入，可以与任何基于粒子的流体模拟方法一起使用。流体表面是使用 van der Laan^[13]提出的平滑方法重建的，它能够渲染环境反射和折射，但无法渲染流体的内部体积。

模拟和渲染都具有交互式性能，可以用于渲染预览或虚拟现实和增强现实解决方案中的应用。但是，与 Xiao 等人^[20]的工作相比，我们所提出的解决方案未能超越最新的性能结果，但它更容易集成到任何基于粒子的流体模拟方法中。同样，与 Zhang 和 Liu^[22]的工作相比，我们所提出的解决方案能够在 1024 x 768 像素分辨率下以 10 帧每秒的速度模拟和渲染 10 万个粒子，并使用基于光线追踪的透明渲染，而 Zhang 和 Liu^[22]的工作以 20 帧每秒的速度模拟和渲染这些粒子，但采用的是不透明的渲染。

A.7.1 未来工作

SPH 方法可以扩展到使用线性系统来模拟不可压缩流，以解决压力分量问题，它的计算成本很高，并且可以利用并行实现^[34]。此外，还可以改进该方法以解决湍流问题，以便在工程应用中使用，如船舶动力学和洪水^[121]。

渲染可以通过更精确的模糊函数来改进，例如 Reichl 等人^[19]工作中提出的那个。此外，解决方案可以通过类似于 Zirr 和 Dachsbacher^[98]的工作来改进，以渲染流体内部体积，或者使用 van der Laan 等人^[13]提出的流体厚度。后者使用累加混合在像素中累积厚度，减弱流体颜色和透明度，然后在相机前的大量流体上创建更深的颜色。这些方法具有高质量的视觉效果，并且比计算光线与流体中每个粒子的相互作用更快。

为了渲染喷雾、泡沫和气泡，可以采用类似于 Ihmsen 等人^[96]的工作的方法，该方法根据扩散粒子的数量减弱流体颜色。此外，本方法可以扩展到渲染多相流体，如 dos Santos Brito 等人^[49]的工作，它在不同的缓冲区中渲染每种流体并将它们混合以创建最终图像。

最后，可以将完整的光线追踪算法集成到解决方案中，来能够渲染具有全局照明但没有在现实世界水照明中非常常见的焦散效应的场景。为了创建这种效果，可以使用光子映射焦散计算，它创建高质量图像但计算成本高^[122]，或者使用基于图像的方法，该方法生成焦散纹理图，并使用该图在非光泽表面上实时创建焦散效果，但照明误差高^[123]。

要在 VR 和 AR 应用中广泛使用，最好是渲染方法具有 30 帧每秒或更高的性能；我们的解决方案能够以 15 次平滑迭代在高清分辨率下为 50k 粒子实现这一速率。为了达到这个速率，一个简单的解决方案是使用少量的粒子，这会导致模拟的数值精度降低，但对于大多数 VR/AR 应用来说是合适的^[78,81]。另一个解决方案是计算流体自由表面^[34,124]，并使用这些粒子作为渲染的输入。由于解决方案不渲染流体内部粒子，表面重建应保持其一致性。

提高渲染方法性能的另一种方法是使用多 GPU 设置，并将流体域平均分配给每个 GPU 进行渲染。多 GPU 方法已用于模拟，如 Rustico 等人^[125]和 Verma 等人^[126]的工作所示，但它也可以用于渲染，能够以更好的性能渲染更多粒子。另一种可能性是使用类似于 Zhang 等人^[127]的多分辨率方法，能够减少渲染粒子的数量并提高方法性能，并且由于我们所提出的方法是不渲染内部流体，这可能不会降低视觉质量。

致谢

作者们对巴西国家科技发展委员会（CNPq, Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil,) 表示感谢，该机构支持本研究，项目编号为456332/2013-8，同时感谢 Caio Brito 的博士学位项目，项目编号为 140905/2018-9。

补充材料

与本文相关的补充材料可以在在线版本中找到，网址为doi:10.1016/j.cag.2018.09.019。

参考文献

- [1] Gingold R A, Monaghan J J. Smoothed particle hydrodynamics: theory and application to non-spherical stars[J]. Monthly notices of the royal astronomical society, 1977, 181(3): 375-389.
- [2] Koshizuka S, Nobe A, Oka Y. Numerical analysis of breaking waves using the moving particle semi-implicit method[J]. International journal for numerical methods in fluids, 1998, 26(7): 751-769.
- [3] Rogers B D, Dalrymple R A, Gesteira M, et al. Smoothed particle hydrodynamics for naval hydrodynamics[C]//Proc. Int. Workshop on Water Waves and Floating Bodies. 2003.
- [4] Marrone S, Antuono M, Colagrossi A, et al. δ -sph model for simulating violent impact flows [J]. Computer Methods in Applied Mechanics and Engineering, 2011, 200(13-16): 1526-1542.
- [5] Unity Technologies. Water in unity[EB/OL]. 2018. <https://docs.unity3d.com/2018.1/Documentation/Manual/HOWTO-Water.html>, Online; accessed 12-February-2018.
- [6] Epic Games. Unreal engine - water in unity[EB/OL]. 2018. <https://docs.unrealengine.com/latest/INT/Resources/Effects/WaterExamples/>, Online; accessed 12-February-2018.
- [7] Kessler J, Carabaich P, Kinoshita N. Fluid dynamics and lighting implementation in pixeljunk shooter 2[M]//ACM SIGGRAPH 2011 Talks. 2011: 1-1.
- [8] Side Effects Software. Houdini fx[EB/OL]. 2018. <https://www.sidefx.com/products/houdini/fx-features/>, Online; accessed 12-February-2018.
- [9] Side Effects Software. Houdini 16.5 - fluids[EB/OL]. 2018. <https://www.sidefx.com/docs/houdini/fluid/index.html>, Online; accessed 12-February-2018.
- [10] Next Limit Technologies. Realflow[EB/OL]. 2018. <https://nextlimit.com/realflow/>, Online; accessed 12-February-2018.
- [11] Next Limit Technologies. Realflow 10 documentation - particles: liquid - pbd[EB/OL]. 2018. <http://www.support.nextlimit.com/display/rf2016docs/DyDomain+-+Particles3A+Liquid+-+PBD>, Online; accessed 12-February-2018.
- [12] Next Limit Technologies. Realflow 10 documentation - particles: Liquid - sph[EB/OL]. 2018. <http://www.support.nextlimit.com/display/rf2016docs/DyDomain+-+Particles3A+Liquid+-+SPH>, Online; accessed 12-February-2018.

- [13] van der Laan W J, Green S, Sainz M. Screen space fluid rendering with curvature flow[C/OL]// I3D '09: Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games. New York, NY, USA: Association for Computing Machinery, 2009: 91–98. <https://doi.org/10.1145/1507149.1507164>.
- [14] Horvath C J, Solenthaler B. Mass preserving multi-scale sph[J]. Pixar Technical Memo 13–04, Pixar Animation Studios, 2013.
- [15] Yan X, Song Y, Chowdhury N A. Performance evaluation of single svm and lssvm based forecasting models using price zones analysis[C/OL]//2016 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC). 2016: 79-83. DOI: 10.1109/APPEEC.2016.7779474.
- [16] Yu J, Turk G. Reconstructing surfaces of particle-based fluids using anisotropic kernels[J]. ACM Transactions on Graphics (TOG), 2013, 32(1): 1-12.
- [17] Yu J, Wojtan C, Turk G, et al. Explicit mesh surfaces for particle based fluids[C]//Computer graphics forum: Vol. 31. Wiley Online Library, 2012: 815-824.
- [18] Fraedrich R, Auer S, Westermann R. Efficient high-quality volume rendering of sph data[J]. IEEE transactions on visualization and computer graphics, 2010, 16(6): 1533-1540.
- [19] Reichl F, Chajdas M G, Schneider J, et al. Interactive rendering of giga-particle fluid simulations.[C]//High Performance Graphics. 2014: 105-116.
- [20] Xiao X, Zhang S, Yang X. Real-time high-quality surface rendering for large scale particle-based fluids[C]//Proceedings of the 21st ACM siggraph symposium on interactive 3D graphics and games. 2017: 1-8.
- [21] Koshizuka S, Oka Y. Moving-particle semi-implicit method for fragmentation of incompressible fluid[J]. Nuclear science and engineering, 1996, 123(3): 421-434.
- [22] Zhang X, Liu S. Sph haptic interaction with multiple-fluid simulation[J]. Virtual Reality, 2017, 21: 165-175.
- [23] Lucy L B. A numerical approach to the testing of the fission hypothesis[J]. Astronomical Journal, vol. 82, Dec. 1977, p. 1013-1024., 1977, 82: 1013-1024.
- [24] Chen J, Yang K, Yuan Y. Sph-based visual simulation of fluid[C]//2009 4th International Conference on Computer Science & Education. IEEE, 2009: 690-693.
- [25] Andrea C. A meshless lagrangian method for free-surface flows and interface flows with fragmentation[D]. PhD thesis, Universita di Roma La Sapienza, 2005.
- [26] Chen Y, Lee J D, Eskandarian A. Meshless methods in solid mechanics: Vol. 9[M]. Springer, 2006.
- [27] Szewc K, Pozorski J, Minier J P. Analysis of the incompressibility constraint in the smoothed particle hydrodynamics method[J]. International Journal for Numerical Methods in Engineering, 2012, 92(4): 343-369.
- [28] Shadloo M S, Zainali A, Yildiz M, et al. A robust weakly compressible sph method and its comparison with an incompressible sph[J]. International Journal for Numerical Methods in Engineering, 2012, 89(8): 939-956.

- [29] Lee E S, Violeau D, Issa R, et al. Application of weakly compressible and truly incompressible sph to 3-d water collapse in waterworks[J]. *Journal of Hydraulic research*, 2010, 48(sup1): 50-60.
- [30] Lee E S, Moulinec C, Xu R, et al. Comparisons of weakly compressible and truly incompressible algorithms for the sph mesh free particle method[J/OL]. *Journal of Computational Physics*, 2008, 227(18): 8417-8436. <https://www.sciencedirect.com/science/article/pii/S00219991080315X>. DOI: <https://doi.org/10.1016/j.jcp.2008.06.005>.
- [31] Cummins S J, Rudman M. An sph projection method[J]. *Journal of computational physics*, 1999, 152(2): 584-607.
- [32] Violeau D, Rogers B D. Smoothed particle hydrodynamics (sph) for free-surface flows: past, present and future[J]. *Journal of Hydraulic Research*, 2016, 54(1): 1-26.
- [33] Firoozabadi B, Mahdinia M, et al. 2d numerical simulation of density currents using the sph projection method[J]. *European Journal of Mechanics-B/Fluids*, 2013, 38: 38-46.
- [34] Xu R, Stansby P, Laurence D. Accuracy and stability in incompressible sph (isph) based on the projection method and a new approach[J]. *Journal of computational Physics*, 2009, 228(18): 6703-6725.
- [35] Brown D L, Cortez R, Minion M L. Accurate projection methods for the incompressible navier–stokes equations[J]. *Journal of computational physics*, 2001, 168(2): 464-499.
- [36] Asai M, Aly A M, Sonoda Y, et al. A stabilized incompressible sph method by relaxing the density invariance condition[J]. *Journal of Applied Mathematics*, 2012, 2012(1): 139583.
- [37] Harada T, Koshizuka S, Kawaguchi Y. Improvement in the boundary conditions of smoothed particle hydrodynamics[J]. *Computer Graphics & Geometry*, 2007, 9(3): 2-15.
- [38] Tanaka M, Masunaga T. Stabilization and smoothing of pressure in mps method by quasi-compressibility[J]. *Journal of Computational Physics*, 2010, 229(11): 4279-4290.
- [39] Tsuruta N, Khayyer A, Gotoh H. A short note on dynamic stabilization of moving particle semi-implicit method[J]. *Computers & Fluids*, 2013, 82: 158-164.
- [40] Lastiwka M, Basa M, Quinlan N J. Permeable and non-reflecting boundary conditions in sph [J]. *International journal for numerical methods in fluids*, 2009, 61(7): 709-724.
- [41] Monaghan J J, Kajtar J B. Sph particle boundary forces for arbitrary boundaries[J]. *Computer physics communications*, 2009, 180(10): 1811-1820.
- [42] Hu X Y, Adams N A. A multi-phase sph method for macroscopic and mesoscopic flows[J]. *Journal of Computational Physics*, 2006, 213(2): 844-861.
- [43] Zainali A, Tofighi N, Shadloo M S, et al. Numerical investigation of newtonian and non-newtonian multiphase flows using isph method[J]. *Computer Methods in Applied Mechanics and Engineering*, 2013, 254: 99-113.
- [44] Pozorski J, Wawreńczuk A. Sph computation of incompressible viscous flows[J]. *Journal of theoretical and applied mechanics*, 2002, 40(4): 917-937.
- [45] Watkins S, Bhattacharjee A, Francis N, et al. A new prescription for viscosity in smoothed particle hydrodynamics[J]. *Astronomy and Astrophysics Supplement Series*, 1996, 119(1): 177-187.

- [46] Rafiee A, Manzari M, Hosseini M. An incompressible sph method for simulation of unsteady viscoelastic free-surface flows[J]. International Journal of Non-Linear Mechanics, 2007, 42 (10): 1210-1223.
- [47] Sigalotti L D G, Klapp J, Sira E, et al. Sph simulations of time-dependent poiseuille flow at low reynolds numbers[J]. Journal of computational physics, 2003, 191(2): 622-638.
- [48] Yang X, Liu M, Peng S. Smoothed particle hydrodynamics modeling of viscous liquid drop without tensile instability[J]. Computers & Fluids, 2014, 92: 199-208.
- [49] dos Santos Brito C J, Almeida M W S, Vieira-e Silva A L B, et al. Screen space rendering solution for multiphase sph simulation[C]//2017 19th Symposium on Virtual and Augmented Reality (SVR). IEEE, 2017: 309-318.
- [50] Monaghan J J. Smoothed particle hydrodynamics[J]. Reports on progress in physics, 2005, 68 (8): 1703.
- [51] Wilcox D. Turbulence modeling for cfd[J]. DCW industries, La Canada, 1998.
- [52] Pan X j, Zhang H x, Sun X y. Numerical simulation of sloshing with large deforming free surface by mps-les method[J]. China Ocean Engineering, 2012, 26(4): 653-668.
- [53] Morris J P, Fox P J, Zhu Y. Modeling low reynolds number incompressible flows using sph[J]. Journal of computational physics, 1997, 136(1): 214-226.
- [54] Chantasiriwan S. Performance of multiquadric collocation method in solving lid-driven cavity flow problem with low reynolds number[J]. COMPUTER MODELING IN ENGINEERING AND SCIENCES, 2006, 15(3): 137.
- [55] Price D J. Resolving high reynolds numbers in smoothed particle hydrodynamics simulations of subsonic turbulence[J]. Monthly Notices of the Royal Astronomical Society: Letters, 2012, 420(1): L33-L37.
- [56] Meister M, Burger G, Rauch W. On the reynolds number sensitivity of smoothed particle hydrodynamics[J]. Journal of hydraulic research, 2014, 52(6): 824-835.
- [57] Liu M, Liu G. Smoothed particle hydrodynamics (sph): an overview and recent developments [J]. Archives of computational methods in engineering, 2010, 17: 25-76.
- [58] Ataie-Ashtiani B, Farhadi L. A stable moving-particle semi-implicit method for free surface flows[J]. Fluid dynamics research, 2006, 38(4): 241.
- [59] Swegle J W, Hicks D L, Attaway S W. Smoothed particle hydrodynamics stability analysis[J]. Journal of computational physics, 1995, 116(1): 123-134.
- [60] Bonet J, Kulasegaram S. A simplified approach to enhance the performance of smooth particle hydrodynamics methods[J]. Applied Mathematics and Computation, 2002, 126(2-3): 133-155.
- [61] Bonet J, Lok T S. Variational and momentum preservation aspects of smooth particle hydrodynamic formulations[J]. Computer Methods in applied mechanics and engineering, 1999, 180 (1-2): 97-115.
- [62] Monaghan J J. Simulating free surface flows with sph[J]. Journal of computational physics, 1994, 110(2): 399-406.

- [63] Kondo M, Suto K, Sakai M, et al. Incompressible free surface flow analysis using moving particle semi-implicit method[C]//Joint International Workshop: Nuclear Technology and Society-Needs for Next Generation, Berkeley, California, USA. 2008.
- [64] Lee B H, Park J C, Kim M H, et al. Step-by-step improvement of mps method in simulating violent free-surface motions and impact-loads[J]. Computer methods in applied mechanics and engineering, 2011, 200(9-12): 1113-1125.
- [65] Crespo A J C. Application of the smoothed particle hydrodynamics model sphysics to free-surface hydrodynamics[D]. PhD Thesis, Departamento De Fisica Aplicada, Universidade De Vigo, 2008.
- [66] Fang J, Parriaux A, Rentschler M, et al. Improved sph methods for simulating free surface flows of viscous fluids[J]. Applied numerical mathematics, 2009, 59(2): 251-271.
- [67] Kiara A, Hendrickson K, Yue D K. Sph for incompressible free-surface flows. part i: Error analysis of the basic assumptions[J]. Computers & Fluids, 2013, 86: 611-624.
- [68] Bockmann A, Shipilova O, Skeie G. Incompressible sph for free surface flows[J]. Computers & Fluids, 2012, 67: 138-151.
- [69] Hosseini S M, Feng J J. Pressure boundary conditions for computing incompressible flows with sph[J]. Journal of Computational physics, 2011, 230(19): 7473-7487.
- [70] Dehnen W, Aly H. Improving convergence in smoothed particle hydrodynamics simulations without pairing instability[J]. Monthly Notices of the Royal Astronomical Society, 2012, 425 (2): 1068-1082.
- [71] Monaghan J J. Sph without a tensile instability[J]. Journal of computational physics, 2000, 159 (2): 290-311.
- [72] Hori C, Gotoh H, Ikari H, et al. Gpu-acceleration for moving particle semi-implicit method[J]. Computers & Fluids, 2011, 51(1): 174-183.
- [73] Zhu X, Cheng L, Lu L, et al. Implementation of the moving particle semi-implicit method on gpu[J]. Science China Physics, Mechanics and Astronomy, 2011, 54: 523-532.
- [74] Crespo A C, Dominguez J M, Barreiro A, et al. Gpus, a new tool of acceleration in cfd: efficiency and reliability on smoothed particle hydrodynamics methods[J]. PloS one, 2011, 6(6): e20685.
- [75] Harada T, Koshizuka S, Kawaguchi Y. Smoothed particle hydrodynamics on gpus[C]//Computer Graphics International: Vol. 40. SBC Petropolis, 2007: 63-70.
- [76] Krog Ø E, Elster A C. Fast gpu-based fluid simulations using sph[C]//Applied Parallel and Scientific Computing: 10th International Conference, PARA 2010, Reykjavík, Iceland, June 6-9, 2010, Revised Selected Papers, Part II 10. Springer, 2012: 98-109.
- [77] Cirio G, Marchal M, Hillaire S, et al. Six degrees-of-freedom haptic interaction with fluids[J]. IEEE transactions on visualization and computer graphics, 2010, 17(11): 1714-1727.
- [78] Cirio G, Marchal M, Otaduy M A, et al. Six-oof haptic interaction with fluids, solids, and their transitions[C]//2013 World Haptics Conference (WHC). IEEE, 2013: 157-162.

- [79] Cirio G, Marchal M, Le Gentil A, et al. “tap, squeeze and stir” the virtual world: Touching the different states of matter through 6dof haptic interaction[C]//2011 IEEE Virtual Reality Conference. IEEE, 2011: 123-126.
- [80] Pang W M, Qin J, Chui Y P, et al. Fast prototyping of virtual reality based surgical simulators with physx-enabled gpu[J]. Transactions on edutainment IV, 2010: 176-188.
- [81] Wang Z, Wang Y. Haptic interaction with fluid based on smooth particles and finite elements [C]//Computational Science and Its Applications–ICCSA 2014: 14th International Conference, Guimarães, Portugal, June 30–July 3, 2014, Proceedings, Part I 14. Springer, 2014: 808-823.
- [82] Müller M, Charypar D, Gross M. Particle-based fluid simulation for interactive applications[C]// Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation. Citeseer, 2003: 154-159.
- [83] Junior J R d S, Clua E W, Montenegro A, et al. Fluid simulation with two-way interaction rigid body using a heterogeneous gpu and cpu environment[C/OL]//2010 Brazilian Symposium on Games and Digital Entertainment. 2010: 156-164. DOI: 10.1109/SBGAMES.2010.25.
- [84] Junior J R d S, Joselli M, Zamith M, et al. An architecture for real time fluid simulation using multiple gpus[J]. XI SBGames, Brasília, 2012: 8.
- [85] Lorensen W E, Cline H E. Marching cubes: a high resolution 3d surface construction algorithm. siggraph comput graph 21 (4): 163–169[Z]. 1987.
- [86] Blinn J F. A generalization of algebraic surface drawing[J]. ACM transactions on graphics (TOG), 1982, 1(3): 235-256.
- [87] Akinci G, Akinci N, Ihmsen M, et al. An efficient surface reconstruction pipeline for particle-based fluids[M]. The Eurographics Association, 2012.
- [88] Akinci G, Ihmsen M, Akinci N, et al. Parallel surface reconstruction for particle-based fluids [C]//Computer graphics forum: Vol. 31. Wiley Online Library, 2012: 1797-1809.
- [89] Orthmann J, Hochstetter H, Bader J, et al. Consistent surface model for sph-based fluid transport [C/OL]//SCA ’13: Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation. New York, NY, USA: Association for Computing Machinery, 2013: 95-103. <https://doi.org/10.1145/2485895.2485902>.
- [90] Enright D, Fedkiw R, Ferziger J, et al. A hybrid particle level set method for improved interface capturing[J]. Journal of Computational physics, 2002, 183(1): 83-116.
- [91] Premžoe S, Tasdizen T, Bigler J, et al. Particle-based simulation of fluids[C]//Computer Graphics Forum: Vol. 22. Wiley Online Library, 2003: 401-410.
- [92] Monaghan J J, Rafiee A. A simple sph algorithm for multi-fluid flow with high density ratios [J]. International Journal for Numerical Methods in Fluids, 2013, 71(5): 537-561.
- [93] Zhang Y, Solenthaler B, Pajarola R. Adaptive sampling and rendering of fluids on the gpu[M]. University of Zurich, 2008.
- [94] Müller M, Schirm S, Duthaler S. Screen space meshes[C]//Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation. 2007: 9-15.
- [95] Akinci N, Dippel A, Akinci G, et al. Screen space foam rendering[M]. Václav Skala-Union Agency, 2013.

- [96] Ihmsen M, Akinci N, Akinci G, et al. Unified spray, foam and air bubbles for particle-based fluids[J]. *The Visual Computer*, 2012, 28: 669-677.
- [97] NVIDIA. Nvidia advanced rendering[EB/OL]. 2018. <https://www.nvidia.com/en-us/design-visualization/solutions/rendering/>, Online; accessed 12-February-2018.
- [98] Zirr T, Dachsbacher C. Memory-efficient on-the-fly voxelization of particle data.[C]// EGPGV@ EuroVis. 2015: 11-18.
- [99] team D. Xml guide for dualsphysics[Z]. 2016.
- [100] Ihmsen M, Orthmann J, Solenthaler B, et al. Sph fluids in computer graphics[M]. The Eurographics Association, 2014.
- [101] Schechter H, Bridson R. Ghost sph for animating water[J]. *ACM Transactions on Graphics (TOG)*, 2012, 31(4): 1-8.
- [102] Malladi R, Sethian J A. Level set methods for curvature flow, image enhancement, and shape recovery in medical images[M]//Visualization and mathematics: Experiments, simulations and environments. Springer, 1997: 329-345.
- [103] Crespo A J, Domínguez J M, Rogers B D, et al. Dualsphysics: Open-source parallel cfd solver based on smoothed particle hydrodynamics (sph)[J]. *Computer Physics Communications*, 2015, 187: 204-216.
- [104] Parker S G, Bigler J, Dietrich A, et al. Optix: a general purpose ray tracing engine[J]. *Acm transactions on graphics (tog)*, 2010, 29(4): 1-13.
- [105] NVIDIA. Cuda zone | nvidia developer[EB/OL]. 2018. <https://developer.nvidia.com/cuda-zone>, Online; accessed 12-February-2018.
- [106] Whitted T. An improved illumination model for shaded display[C/OL]//SIGGRAPH '05: ACM SIGGRAPH 2005 Courses. New York, NY, USA: Association for Computing Machinery, 2005: 4-es. <https://doi.org/10.1145/1198555.1198743>.
- [107] Mokos A, Rogers B D, Stansby P K. A multi-phase particle shifting algorithm for sph simulations of violent hydrodynamics with a large number of particles[J]. *Journal of Hydraulic Research*, 2017, 55(2): 143-162.
- [108] Vacondio R, Rogers B D, Stansby P K, et al. Variable resolution for sph in three dimensions: Towards optimal splitting and coalescing for dynamic adaptivity[J]. *Computer Methods in Applied Mechanics and Engineering*, 2016, 300: 442-460.
- [109] Mokos A, Rogers B D, Stansby P K, et al. Multi-phase sph modelling of violent hydrodynamics on gpus[J]. *Computer Physics Communications*, 2015, 196: 304-316.
- [110] Board O A R. Openmp application programming interface[EB/OL]. 2017. <https://www.openmp.org/wp-content/uploads/openmp-4.5.pdf>, Online; accessed 12-February-2018.
- [111] Domínguez J, Crespo A, Gómez-Gesteira M, et al. Neighbour lists in smoothed particle hydrodynamics[J]. *International Journal for Numerical Methods in Fluids*, 2011, 67(12): 2026-2042.
- [112] Lauterbach C, Garland M, Sengupta S, et al. Fast bvh construction on gpus[C]//Computer Graphics Forum: Vol. 28. Wiley Online Library, 2009: 375-384.
- [113] Tafuni A, Dominguez J, Vacondio R, et al. Open boundary conditions for large-scale sph simulations[C]//Proceedings of the 11th SPHERIC International Workshop. 2016.

- [114] Ihmsen M, Cornelis J, Solenthaler B, et al. Implicit incompressible sph[J]. IEEE transactions on visualization and computer graphics, 2013, 20(3): 426-435.
- [115] Overby D, Melek Z, Keyser J. Interactive physically-based cloud simulation[C]//10th Pacific Conference on Computer Graphics and Applications, 2002. Proceedings. IEEE, 2002: 469-470.
- [116] Wald I, Dietrich A, Slusallek P. An interactive out-of-core rendering framework for visualizing massively complex models[C/OL]//SIGGRAPH '05: ACM SIGGRAPH 2005 Courses. New York, NY, USA: Association for Computing Machinery, 2005: 17-es. <https://doi.org/10.1145/1198555.1198756>.
- [117] Kniss J, McCormick P, McPherson A, et al. Interactive texture-based volume rendering for large data sets[J]. IEEE Computer Graphics and Applications, 2001, 21(4): 52-61.
- [118] Jiang C, Schroeder C, Selle A, et al. The affine particle-in-cell method[J]. ACM Transactions on Graphics (TOG), 2015, 34(4): 1-10.
- [119] Huang C, Zhu J, Sun H, et al. Parallel-optimizing sph fluid simulation for realistic vr environments[J]. Computer Animation and Virtual Worlds, 2015, 26(1): 43-54.
- [120] Tse B, Barrow A, Quinn B, et al. A smoothed particle hydrodynamics algorithm for haptic rendering of dental filling materials[C]//2015 IEEE World Haptics Conference (WHC). IEEE, 2015: 321-326.
- [121] Cheng H, Zhang A, Ming F. Study on coupled dynamics of ship and flooding water based on experimental and sph methods[J]. Physics of Fluids, 2017, 29(10).
- [122] Kang C m, Wang L, Xu Y n, et al. A survey of photon mapping state-of-the-art research and future challenges[J]. Frontiers of Information Technology & Electronic Engineering, 2016, 17 (3): 185-199.
- [123] Shah M A, Kontinen J, Pattanaik S. Caustics mapping: An image-space technique for real-time caustics[J]. IEEE Transactions on Visualization and Computer Graphics, 2007, 13(2): 272-280.
- [124] Sandim M, Cedrim D, Nonato L G, et al. Boundary detection in particle-based fluids[C]// Computer Graphics Forum: Vol. 35. Wiley Online Library, 2016: 215-224.
- [125] Rustico E, Bilotta G, Herault A, et al. Advances in multi-gpu smoothed particle hydrodynamics simulations[J]. IEEE Transactions on Parallel and Distributed Systems, 2012, 25(1): 43-52.
- [126] Verma K, Szewc K, Wille R. Advanced load balancing for sph simulations on multi-gpu architectures[C]//2017 IEEE High Performance Extreme Computing Conference (HPEC). IEEE, 2017: 1-7.
- [127] Zhang G, Lu D, Liu H. Visualization of fluid simulation: An sph-based multi-resolution method [J]. Concurrency and Computation: Practice and Experience, 2019, 31(23): e4509.

致 谢

到了一生一次的论文致谢环节。好像有很多话要说，但行文至此，也没什么话值得说的。

那就普普通通地致谢吧。

感谢我自己，能够在清华大学的培养体系下完成四年的本科学业。

感谢父母，用精神和金钱支持我读完了四年，没有对我苛求太多，一直默默地支持我。

感谢导师和课题组的支持，让我能够平稳走完最后一段路。

感谢一路上遇到的朋友，安抚我开导我，给予我力量。

还要特别感谢 ThuThesis 节省了论文排版时间。

就这样吧，感谢命运的安排与馈赠，我所经历的一切都将成就我自己。

声 明

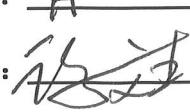
本人郑重声明：所呈交的综合论文训练论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签 名: 尹子琪 日期: 2025.6.13

综合论文训练记录表

学生姓名	卢子期	学号	2021012556	班级	未央-水木 11
论文题目	基于 GPU 的大规模流体粒子 3D 实景可视化方法研究				
主要内容以及进度安排	<p>3D 流体仿真在岩土、水利、汽车、工业等领域的研究和工程分析中发挥着越来越重要的作用。同时，随着计算规模的提升及虚拟现实（VR）和增强现实（AR）等技术的发展，对大规模复杂流体的可视化分析也提出了更高的要求。本毕业设计，基于计算机图形学和流体力学探索高效的 3D 流体渲染技术，研究基于 GPU 和 RTX 技术的高效大规模流体场景的渲染算法，实现超千万级粒子的高效渲染。论文进度安排如下：</p> <p>开题前：确定研究方向，进行文献综述，收集相关资料和研究进展。</p> <p>第 1-2 周：研究和比较不同的流体可视化渲染技术，确定需要实现的技术细节与要求。</p> <p>第 3-6 周：初步实现基于粒子流体（SPH）数据的渲染及可视化分析算法。</p> <p>第 7-11 周：算法优化和发展，实现复杂三维流体的高效、实景渲染，以及流体动力学特性的可视化分析表征。</p> <p>第 12-13 周：对算法进行大规模算例的测试和优化。</p> <p>第 14-15 周：撰写毕设论文，并准备答辩。</p>				
	<p style="text-align: right;">指导教师签字:</p> 				
	<p style="text-align: right;">考核组组长签字:</p>  <p style="text-align: right;">2025 年 1 月 9 日</p>				
中期考核意见	<p>论文工作进展正常，形成了对 3D 大规模流场可视化技术研究现状的基本认识，取得了初步研究成果。</p> <p style="text-align: right;">考核组组长签字:</p>  <p style="text-align: right;">2025 年 3 月 26 日</p>				

指导教师评语	<p>SPH 是目前常用的基于粒子的流体动力学分析方法，在大规模粒子的流场可视成为现在数值计算结果可视化分析的重要需求。卢子期同学根据目前 3D 大规模流场可视化面临的挑战，开展 3D 大规模流场可视化技术研究，基于 GPU 技术将 cLOD 技术应用于流场可视化中，并发展了未来帧提前加载技术；在此基础上设计了适应于优化技术及并行计算的 3D 大规模流场可视化程序，并将其应用于超千万级粒子的大规模流场实景可视化分析中。</p> <p>该生在整个毕设过程中，工作认真踏实，吃苦耐劳，刻苦钻研。论文研究难度较大，内容丰富，体现了作者具有较强的学习能力、综合分析和一定独立工作的能力。达到了清华大学综合论文训练的要求。</p> <p>指导教师签字:  2025 年 6 月 6 日</p>
评阅教师评语	<p>论文针对大规模流体粒子可视化难题，提出了基于 cLoD 的流体粒子动态渲染算法，通过智能分配渲染资源实现效率与质量的平衡。结合多线程预加载技术，显著减少数据加载时间对渲染效率的影响，提高了整个渲染流程的流畅度；开发的交互式原型系统支持实时调节与动画导出，为大规模粒子流体仿真提供了高效可视化方案。论文成果为大规模流体数据可视化提供了新思路，具有重要推广意义。</p> <p>论文成果丰富，内容比较全面，达到了综合论文训练的目的。</p> <p>评阅教师签字:  2025 年 6 月 7 日</p>
答辩小组评语	<p>论文答辩思路清晰，回答问题正确。 达到毕设要求。</p> <p>答辩小组组长签字:  2025 年 6 月 9 日</p>

总成绩: A-
教学负责人签字: 
2025 年 6 月 9 日