计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(2):169−174 [doi: 10.15888/j.cnki.csa.006192]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于位置的流体实时交互仿真①

王　坤[1,2], 于　歌[1], 梁　骥[1], 郭丽丽[1]

[1](中国科学院 空间应用工程与技术中心 太空应用重点实验室, 北京 100094)
[2](中国科学院大学, 北京 100049)

摘　要: 针对基于光滑粒子动力学方法的流体交互仿真过程中效率低, 交互细节不够真实等问题, 提出了采用基于位置的流体来模拟刚体工具与流体的交互方法. 该方法在传统的光滑粒子动力学算法的基础上进行改进, 以基于CUDA 并行计算平台实时模拟交互过程, 并结合力觉交互设备实时输出交互力. 实验结果表明仿真过程中的交互力符合预期, 在保证流体模拟的精度的前提下验证了交互力的连续性以及稳定性.

关键词: 光滑粒子动力学; 基于位置的流体; CUDA; 实时; 流体交互

引用格式: 王坤,于歌,梁骥,郭丽丽.基于位置的流体实时交互仿真.计算机系统应用,2018,27(2):169–174. http://www.c-s-a.org.cn/1003-3254/6192.html

Real-Time Interaction Simulation with Position-Based Fluid

WANG Kun[1,2], YU Ge[1], LIANG Ji[1], GUO Li-Li[1]

[1](Key Laboratory of Space Utilization, Technology and Engineering Center for Space Utilization, Chinese Academy of Sciences, Beijing 100094, China)
[2](University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: In view of the low efficiency and lack of real details in the process of the fluid interaction simulation based on SPH (Smoothed Particle Hydrodynamics), an interactive simulation method based on position-based fluid is proposed to simulate rigid body tool and fluid. This method is improved on the basis of the traditional SPH algorithm, and the interactive process is simulated in real time based on CUDA parallel computing platform. Then, the real-time output interaction force is combined with force interaction device. The experimental results show that the interaction force in the simulation process is in accordance with the expectation, and the continuity and stability of the interaction force are verified under the premise of ensuring the accuracy of the fluid simulation.

Key words: SPH; position based fluid; CUDA; real-time; interaction with fluid

引言

刚体与流体的交互是我们生活中常见到的, 比如搅拌一杯水或者手直接与水接触. 在计算机仿真领域, 与流体的力觉交互中, 我们希望得到的是一种实时的逼真的交互体验, 这种交互手段在科学实验仿真、医疗训练等领域是迫切需要的. 这对科学家提供更有效率、更接近真实结果的实验或者训练过程是非常有帮助的.

目前, 模拟刚体与流体的实时交互在视觉与触力觉的真实呈现仍然是一个颇具挑战的问题. 首先对于流体模拟, 在传统的方法中, 通常分为两类: 基于网格模型的欧拉法[1]和基于粒子模型的拉格朗日法, 目前采用最多的是基于粒子模型的拉格朗日法. 由于常用的. 其中, 光滑粒子流体动力学 (Smoothed Particle Hydrodynamic, SPH) 是拉格朗日模型中最流行的粒子方法. Muller[2]表明 SPH 可用于模拟交互式流体的粘滞

① 基金项目: 国家自然科学基金 (Y6030411SY)
收稿时间: 2017-05-04; 修改时间: 2017-05-26; 采用时间: 2017-05-31

力和表面张力, 他于 2004 年[3]将引力、斥力和粘滞力引入流体粒子与变形体网格之间的交互建模中实现了刚体-流体交互的实时仿真.

由于 SPH 算法需要较小的时间步长, 对计算开销较大, 另外算法中采用的核函数的光滑半径较大, 实时性以及计算精度难以达到较高的要求. Muller[4]于 2007 年提出了一种基于位置的动力学方法, 采用了较小的光滑半径以及更大的时间步长, 通过对粒子位置加以约束, 直接更新粒子位置来模拟流体粒子, 避免了需要先计算粒子受力然后才能计算粒子位置改变从而带来的不稳定性, 极大的提升了流体模拟的效率以及真实度.

一般来说, 为了实现刚体与流体在混合环境下的交互, 我们需要为每种类型的对象使用不同的算法, 并在这些异构模型之间建立耦合接口, 这将增加已经非常复杂和耗时的模拟的复杂性和计算成本. Keiser 等[5]将这一算法过程进行改进, 实现了固体、可变形和流体之间的相互作用. 但依然难以达到实时性的要求. 随着硬件性能的不断提升带来的自然加速, 更多的研究者将针对新硬件的体系结构来设计整个算法, 以充分利用硬件新特性带来的优势; 另外结合算法本身的不断创新使整个流体模拟向实时性不断迈进. 而基于位置的流体算法的粒子模型也非常适合将算法实现并行化.

传统的基于 CPU 的流体模拟, 难以达到实时高效的流体仿真. GPU 计算的核心思想是将可以被并行的指令尽可能多的移植到 GPU 上执行, 充分利用 GPU 强大的并行计算能力批量执行指令. Hegeman[6]验证了在 GPU 的并行架构上实现粒子模型的有效性表明 GPU 的运行速度对比纯 CPU 算法的运行速度有超过一个数量级的提升. 2013 年, Domínguez[7]等人优化了现存基于 CPU-GPU 计算平台的 SPH 计算模型, 主要解决了 CPU 与 GPU 仿真数据的传输问题.

本文在各种模拟流体的算法中, 综合考虑了各种算法在交互过程中的利弊, 采用一种 SPH 的改进算法, 结合 Müller 的基于位置动力学 (Position Based Dynamics)[4]得到的基于位置的流体算法 (Position Based Fluid)[8]来实现流体的基本模拟, 其次通过刚性体与流体粒子的碰撞检测与响应计算交互力. 本文着重探讨了刚性体与流体的交互力觉渲染, 基于 CUDA 平台实现算法的并行化, 提供高效稳定的计算结果, 在视觉与力觉上实现真实与实时的流体交互.

1 光滑粒子动力学基础

本文采用基于郎格朗日粒子法来模拟流体的运动, 其中最常用的是光滑粒子流体动力学, 该方法是一种用于流体动力学的插值方法, 可以通过插值计算出流体的物理特性. 用粒子代替流体来模拟流体运动, 它把流体定义为在空间中离散分布并且位置可计算的粒子, 这些粒子具有流体的质量、密度、速度等物理属性. SPH 方法通过积分近似和粒子近似将流体动力学方程进行离散, 每个粒子上的属性函数 $f_i(\boldsymbol{p})$ 通过支持域内相邻粒子的叠加求和计算得到:

$$f_i(\boldsymbol{p}) = \sum_j f_j(\boldsymbol{p}_j)\frac{m_j}{\rho_j}W(\boldsymbol{p}-\boldsymbol{p}_j,h) \tag{1}$$

其中, j 表示粒子 i 支持域内的第 j 个粒子, m_j 表示粒子 j 的质量, $\boldsymbol{p}_j$ 表示粒子 j 的位置, ρ_j 表示粒子 j 的密度, f_j 表示在位置 $\boldsymbol{p}$ 处的某种属性函数, 如密度、压力等, W 表示核函数, h 表示光滑核半径, 如图 1 所示.

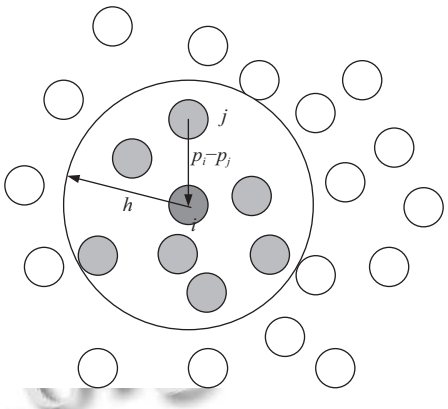

图 1 SPH 粒子算法

2 流体模拟方法

2.1 基于位置的流体

在本文中, 对于流体模拟, 本文采用的是 Macklin[8]提供的基于位置的流体模拟方法, 在 SPH 方法的基础上进行改进, 采用了更大的时间步长以及较小的光滑半径, 可以直接更新粒子位置变化来模拟流体, 避免了 SPH 流体模拟中的不稳定性.

根据 Macklin[8]的方法, 为了实现密度计算的稳定, 在运动中不会发生较大的突变, 对每一个粒子位置施加一个非线性约束系统, 约束的状态方程为:

$$C_i(\boldsymbol{p}_1,\cdots,\boldsymbol{p}_n) = \frac{\rho_i}{\rho_0} - 1 \tag{2}$$

其中, $\boldsymbol{p}_n$ 为 i 粒子的第 n 个邻域粒子的位置, ρ_0 为算法

中给定的初始密度, ρ_i 为标准 SPH 算法的密度计算, 根据公式 (1) 求得密度计算为:

$$\rho_i = \sum_j m_j W(\boldsymbol{p}_i - \boldsymbol{p}_j, h) \tag{3}$$

其中所有粒子质量 m_j 相同, 核函数 W 采用 Müller[3]给出的 Poly6 光滑核函数来求解密度值.

在该 Müller[4]的基于位置动力学算法中, 对粒子位置给定一个修正, 所有粒子的位置满足约束:

$$C(\boldsymbol{p} + \Delta\boldsymbol{p}) = 0 \tag{4}$$

给定位置 $\boldsymbol{p}$ 的变化 $\Delta\boldsymbol{p}$ 在约束函数梯度的大小关系, 其系数为 λ.

$$\Delta\boldsymbol{p} = \lambda \nabla C(p) \tag{5}$$

结合公式 (5) 通过对约束的梯度计算得到:

$$\begin{aligned} C(\boldsymbol{p} + \Delta\boldsymbol{p}) &\approx C(\boldsymbol{p}) + \nabla C(\boldsymbol{p})^T \Delta\boldsymbol{p} \\ &\approx C(\boldsymbol{p}) + \nabla C(\boldsymbol{p})^T \nabla C(\boldsymbol{p}) \lambda = 0 \end{aligned} \tag{6}$$

结合 Monaghan[9]给出的 SPH 方法, 代入公式 (2) 和 (3), 推导出粒子 i 的位置约束梯度函数为:

$$\nabla C_i = \frac{1}{\rho_0} \sum_j \nabla W(\boldsymbol{p}_i - \boldsymbol{p}_j, h) \tag{7}$$

结合公式 (6) 与 (7) 得到 λ 的解为:

$$\lambda_i = \frac{C_i(\boldsymbol{p}_1, \cdots, \boldsymbol{p}_n)}{\sum_k |\nabla C_i(\boldsymbol{p})|^2} \tag{8}$$

因为公式 (2) 中的约束函数与密度计算有关, 而密度计算是非线性的, 当流体粒子太分散时容易导致密度计算结果的不稳定, 根据 Smith[10]提出方法中, 可以通过混合一些约束来提高稳定性, 公式 (6) 修改为:

$$C(\boldsymbol{p} + \Delta\boldsymbol{p}) \approx C(\boldsymbol{p}) + \nabla C(\boldsymbol{p})^T \nabla C(\boldsymbol{p}) \lambda + \varepsilon\lambda = 0 \tag{9}$$

其中 ε 为我们在模拟过程中定义的常量约束参数. λ 的求解修改为:

$$\lambda_i = \frac{C_i(\boldsymbol{p}_1, \cdots, \boldsymbol{p}_n)}{\sum_k \left|\nabla_{p_k} C_i(\boldsymbol{p})\right|^2 + \varepsilon} \tag{10}$$

在每一次循环计算过程中, 都需要计算 λ 的值, 然后结合公式 (5), 按照如下公式计算粒子 i 的位置更新, 其计算为

$$\Delta\boldsymbol{p}_i = \frac{1}{\rho_0} \sum_j (\lambda_i + \lambda_j) \nabla W(\boldsymbol{p}_i - \boldsymbol{p}_j, h) \tag{11}$$

如上所述, 在基于位置的流体算法中, 每次循环执行流体粒子位置的动态更新. 当计算得到粒子的位置更新后, 便可根据这一时间段内进一步计算得出粒子的速度以及加速度. 每次循环中独立解决每个位置约束, 同时, 在每次循环过程中, 对刚性体进行碰撞检测, 每个步骤中我们重新计算每一个粒子的领域粒子, 并且重新计算每个粒子位置的约束值. 这样可以避免对粒子密度的低估, 导致最终结果出现较大偏差.

2.2 刚体与流体的交互力计算

根据 Tse[11]给出的方法中, 刚体与流体的交互过程中, 工具球会受到流体的吸附力, 当工具球与流体粒子的距离在支持域内, 吸附力开始产生作用, 流体粒子 j 与工具球的球心距为$\boldsymbol{r}_j = \boldsymbol{p} - \boldsymbol{p}_j$, 吸附力计算如下:

$$\boldsymbol{F}_{\text{Adh}} = \sum \frac{K_i + K_j}{2} \frac{m_j}{\rho_j} W_{\text{Adh}}(\boldsymbol{r}_j, h) \tag{12}$$

K_i, K_j 为吸附力计算参数, W_{Adh} 为吸附力计算的光滑核函数, 其计算为:

$$W_{\text{Adh}}(\boldsymbol{r}_j, h) = C_d \begin{cases} 0 & \left|\overrightarrow{\boldsymbol{r}_{ij}}\right| < A \\ (\frac{\overrightarrow{r_a}}{Q})^{\varepsilon} - (\frac{\overrightarrow{r_a}}{Q})^{\varsigma} & A \le \left|\overrightarrow{\boldsymbol{r}_{ij}}\right| \le B \\ 0 & \left|\overrightarrow{\boldsymbol{r}_{ij}}\right| > B \end{cases} \tag{13}$$

其中 $\varepsilon > \zeta$, $Q = (B\text{-}A)$. $\overrightarrow{\boldsymbol{r}_a} = (\boldsymbol{r}_j - A)$.

A: 吸附力光滑函数起始点.

B: 吸附力光滑函数结束点.

ε, ζ: 可调节吸附力, 本文中分别为 ε=3, ζ=2.

工具与流体在交互过程中除受到吸附力以外, 还受到因碰撞产生的阻尼力, 其大小计算方式如下:

$$\boldsymbol{F}_{\text{Damp}} = k_{\text{Damp}} * \sum (\boldsymbol{v}_0 - \boldsymbol{v}_j) \tag{14}$$

K_{Damp} 为阻尼系数, $0 > K_{\text{Damp}} > 1$, $\boldsymbol{v}_0$ 为工具球的速度, $\boldsymbol{v}_j$ 为与工具球碰撞的粒子 j 的速度, 可以看出, 碰撞力的大小除了与碰撞系数有关外, 也和工具球与流体粒子的速度差有关, 这也符合我们日常生活中所见到的情景.

3 流体交互算法实现

3.1 流体粒子初始网格构建

在流体模拟阶段, 计算每一个粒子的物理属性需要改粒子邻域内的粒子信息, 因此, 邻域粒子搜索算法是整个算法中比较耗时的步骤, 对流体粒子所在空间进行均匀网格划分是目前被认为最为有效的方法. 基于 CUDA 平台我们可以快速的实现对流体粒子的网格构建, 如图 2.

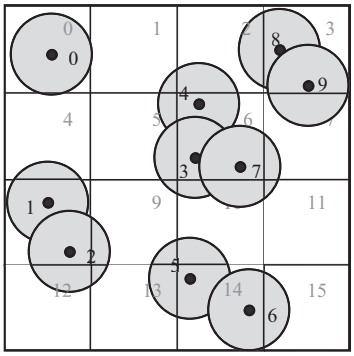

图 2 流体粒子网格构建

采用 Nvidia[12]的粒子模拟方法来实现对流体粒子的初始网格划分并记录每一个流体粒子的领域粒子的索引值, 记录如表 1, 便于执行后续的邻域粒子快速查找, 同时能够快速执行与交互工具的碰撞检测和碰撞响应. 在每一次循环计算中, 我们都需要执行一个完整的 CUDA 程序.

表 1 邻域粒子的索引值记录

网格索引	粒子数目	粒子索引
0	1	0
1	0	
2	0	
3	2	8, 9
4	0	
5	0	
6	3	3, 4, 7
7	0	
8	2	1, 2
9	0	
10	0	
11	0	
12	0	
13	0	
14	2	5, 6
15	0	

3.2 算法流程

整个算法流程图分为如图 3 几个阶段. 由于流体模拟算法的粒子性, 我们可以并行的对每一个粒子的状态并行的求解, 在每一个阶段通过 CUDA 平台并行的对每一个粒子的状态进行求解, 包括对初始粒子网格的建立到邻域粒子的搜索以及对于每一个流体粒子各项物理属性的计算, 在受力计算阶段实时的通过设备输出交互力, 在最后阶段计算粒子的位置、速度等状态并更新绘制显示. 同时将这些状态用于下一个循环中的位置预测计算.

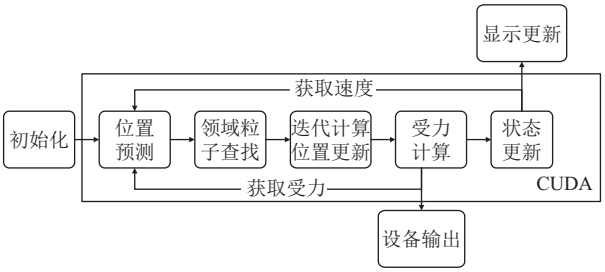

图 3 算法流程

在每一个阶段更为详细的算法过程如下:

算法1. 流体交互算法

#初始化#

1. 设置粒子初始位置、速度等状态量.
2. 根据粒子初始位置划定网格.

#位置预测#

3. 由外力以及设备反馈力更新速度$\boldsymbol{v}_i^* = \boldsymbol{v}_i + \Delta t * f_{\text{Ext}}(\boldsymbol{p}_i)$.
4. 预测粒子位置$\boldsymbol{p}_i^* = \boldsymbol{p}_i + \Delta t * \boldsymbol{v}_i$.

#邻域粒子查找#

5. 根据网格数据查找粒子i的邻域粒子并记录索引.

#迭代过程#

6. 在迭代次数内对粒子i执行.
7. 根据公式(3)计算粒子i的密度.
8. 根据公式(10)计算λ_i.
9. 根据公式(11)计算粒子i的位移$\Delta \boldsymbol{p}_i$.
10. 碰撞检测.
11. 根据$\Delta \boldsymbol{p}_i$更新粒子位置.
12. 重新组织粒子和网格.

#计算受力#

13. 执行碰撞响应.
14. 利用公式(12)计算附着力$\boldsymbol{F}_{\text{Adh}}$.
15. 利用公式(14)计算阻尼力$\boldsymbol{F}_{\text{Damp}}$.
16. 将工具所受合力反馈到力觉设备端.

#粒子状态更新#

17. 更新粒子i位置$\boldsymbol{p}_i^* = \boldsymbol{p}_i^* + \Delta \boldsymbol{p}_i$.
18. 更新粒子i速度$\boldsymbol{v}_i = \frac{1}{\Delta t}(\boldsymbol{p}_i^* - \boldsymbol{p}_i)$.
19. 根据新的粒子位置渲染流体.

4 实验方案

4.1 实验设备及环境

开发环境如表 2 所示, 算法通过 CUDA 并行化, 运行在 GPU 端, 采用的显卡为 NVIDIA Quadro M5000, 包含 2048 个 CUDA 并行处理核心, 所有并行计算的步骤都在 GPU 端执行, 核心数越多, 并行加速效率越高.

交互设备采用的是 Geomagic 公司开发的三自由度力觉交互设备 Geomagic Touch, 如图 4. 该反馈设备具有六自由度操作位置检测, 三自由度力觉交互, 通过 RJ45 以太网接口或 USB 接口连接终端. 可以通过该设

备实时控制工具球在流体中的运动并且反馈出工具球受到的合力, 模拟出真实场景中在流体中交互的受力情况.

表 2 实验设备以及开发环境

软硬件配置	参数
系统	Win7专业版
开发工具	Visual Studio 2015、CUDA 8.0
运行库	GLEW、GLFW、GLM
处理器	Intel Xeon E5-1603V3
内存	32 GB
显卡	NVIDIA Quadro M5000
力觉交互设备	Geomagic Touch

图 4 Geomagic Touch 三自由度力觉交互设备

4.2 实验过程

4.2.1 刚体与流体的交互力模拟验证

为了验证基于位置的流体算法模拟流体与刚体的交互, 我们建立了关于刚性工具在与流体的交互过程中的受力模拟实验, 实验分为两个部分, 第一个部分模拟工具球在交互过程中的受力情况并进行分析, 验证交互过程的真实性; 第二个部分通调整粒子数目来验证力觉交互设备感知到的交互力是否稳定连续来验证交互过程的稳定与实时性.

第一部分的实验我们模拟的是工具球在流体交互过程中的受力情况, 我们设置的初始粒子数目为 20 k, 从视觉角度来说, 已经可以很好的满足这一部分实验的精细度. 粒子的初始位置按照容器内部空间均匀排列, 初始速度为零, 初始受力为容器壁的约束受力以及重力. 一个完整的模拟实验过程包括从工具球向下方向匀速运动, 从刚接触流体表面到开始进入流体, 然后完全没入流体, 在流体中匀速运动直到碰到杯壁返回到中间位置, 最后再匀速向上离开流体, 到工具球最后刚到达流体上表面因吸附力沾染流体, 到最后完全离开流体. 整个实验过程的部分场景如图 5 所示.

4.2.2 交互力模拟实验结果

首先我们验证的是工具球在流体交互中的受力情况, 图 6 模拟的是通过力觉交互设备控制工具球在流体中运动时的受力情况, 如图所示, X 轴、Y 轴和 Z 轴正方向的力分别用实线、点线和虚线来表示, 其方向已经在图 6 中标出. 在整个过程中我们从图中我们可以看到工具球受力是比较平滑的曲线, 没有发生大范围的突变, 说明在我们的算法中实现的力是稳定连续的.

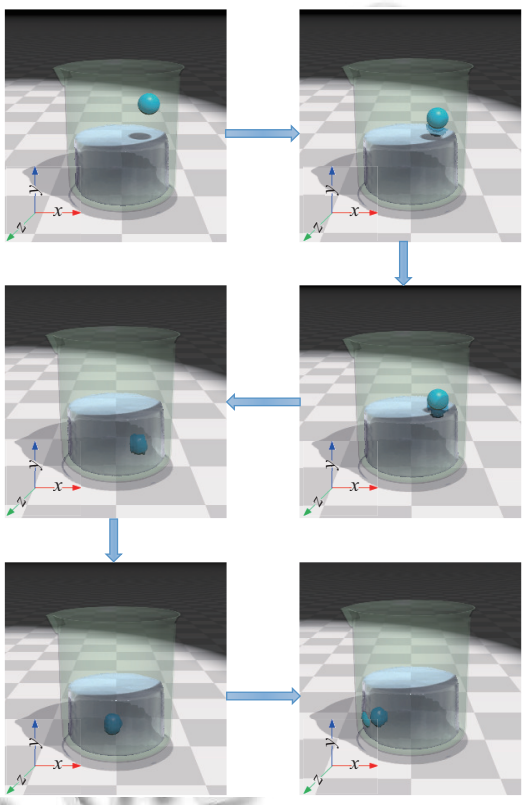

图 5 工具在流体中的交互运动场景

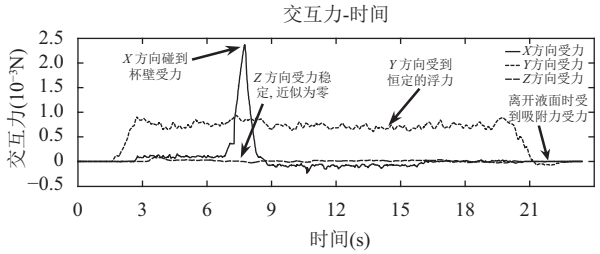

图 6 工具球在流体中运动时的受力情况分析

实验开始时, 当工具球刚开始接触流体表面时, 接触部分受到流体的吸附力, 随着工具球的逐渐进入, 工具球还受到浮力的影响逐渐增大, 到完全没入流体, 工具球受到的浮力趋于稳定. 然后工具球在流体中沿-X 水平方向做匀速运动, 收到一个恒定的阻力. 然后运

动到边界与杯壁发生碰撞, 产生碰撞力, 接着沿 X 方向运动到中间位置, 期间受到-X 方向的恒定阻力, 最后工具球开始向上运动到开始离开流体表面, 竖直方向上受到的流体浮力开始减少, 到即将离开的一瞬间, 工具球在此时没有浸入流体已经不受浮力影响了, 此时工具球还受到一个向下的吸附力, 随着工具球的进一步向上运动, 最终工具球不受到流体交互力的影响. 在整个运动过程中没有 Z 方向的运动, 因此 Z 方向的受力几乎稳定为零, 这也符合实验预期结果.

4.2.3 交互力的连续性验证及结果

对于流体模拟, 需要一定数目的粒子, 才能达到较好的模拟真实性, 但是粒子数增加的同时会带来计算开销的增加, 影响交互的实时性. 在整个交互过程中, 当力觉交互设备的输出频率在 1 kHz 左右时, 我们通过设备能感知到连续的力觉, 而不会有顿挫感. 第二部分的模拟实验中, 在粒子数目能够达到视觉仿真要求的前提下, 通过改变粒子数目来验证算法的效率, 保证在每一次力觉交互循环中的时间步长在 1 ms 左右即可保证力输出的连续性.

如下图 7 所示, 我们测试了不同粒子数目时, 力觉交互设备的输出频率, 粒子数目从 10 k 逐渐增加到 80 k, 每次都测试出整个实验过程中力觉交互设备的刷新率, 粒子数目维持在 10 k 的时候基本上可以满足我们交互过程中的精细度, 继续增加粒子数目可以带来更好的流体模拟效果, 当然粒子数目越多可以模拟更大规模的流体交互, 模拟的流体场景也更为复杂. 当粒子数目达到 60 k 的时候已经远远达到流体模拟的精度要求, 虽然频率稍有下降, 但仍然能够很好的满足交互的实时性, 说明交互过程的实时性以及真实度达到了要求.

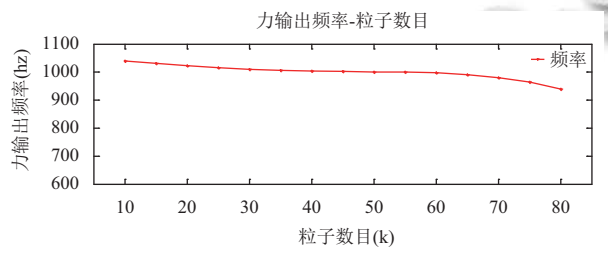

图 7 粒子数目与力觉交互输出频率的影响

5 总结

本文探讨了刚体与流体交互过程中的力觉交互, 实现了基于位置的流体模拟算法并且模拟了刚体与流体的交互过程, 实时计算并渲染出刚体工具所受到的交互力, 有效的避免了传统 SPH 算法的精度较低、计算实时性较差等劣势, 同时刚性工具与流体的交互过程中很好的模拟了流体交互过程中的包括吸附力、浮力等影响比较重要的力觉. 采用 CUDA 并行计算, 极大的提高了流体算法的模拟效率以及精度. 通过模拟实验有效的验证了交互的实时性以及真实度. 对后续流体交互研究具有非常好的指导意义, 我们未来将进一步挖掘流体交互中的诸如湍流等物理现象, 提高交互的真实度.

参考文献

1 Foster N, Metaxas D. Realistic animation of liquids. Graphical Models and Image Processing, 1996, 58(5): 471–483. [doi: 10.1006/gmip.1996.0039]
2 Müller M, Charypar D, Gross M. Particle-based fluid simulation for interactive applications. Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. San Diego, CA, USA. 2003. 154–159.
3 Müller M, Schirm S, Teschner M, *et al*. Interaction of fluids with deformable solids: Research articles. Computer Animation and Virtual Worlds, 2004, 15(3/4): 159–171.
4 Müller M, Heidelberger B, Hennix M, *et al*. Position based dynamics. Journal of Visual Communication and Image Representation, 2007, 18(2): 109–118. [doi: 10.1016/j.jvcir.2007.01.005]
5 Keiser R, Adams B, Gasser D, *et al*. A unified lagrangian approach to solid-fluid animation. Proceedings of the Second Eurographics/IEEE VGTC Conference on Point-Based Graphics. New York, NY, USA. 2005. 125–133.
6 Hegeman K, Carr NA, Miller GSP. Particle-based fluid simulation on the GPU. Proceedings of the 6th International Conference on Computational Science. Reading, UK. 2006. 228–235.
7 Domínguez JM, Crespo AJC, Gómez-Gesteira M. Optimization strategies for CPU and GPU implementations of a smoothed particle hydrodynamics method. Computer Physics Communications, 2013, 184(3): 617–627. [doi: 10.1016/j.cpc.2012.10.015]
8 Macklin M, Müller M. Position based fluids. ACM Transactions on Graphics (TOG), 2013, 32(4): 104.
9 Monaghan JJ. Smoothed particle hydrodynamics. Annual Review of Astronomy and Astrophysics, 1992, 30(1): 543–574. [doi: 10.1146/annurev.aa.30.090192.002551]
10 Smith R. Open dynamics engine v0.5 user guide. Computer Graphics, 2007, 176(2): 121–136.
11 Tse B, Barrow A, Quinn B, *et al*. A smoothed particle hydrodynamics algorithm for haptic rendering of dental filling materials. Proceedings of 2015 IEEE World Haptics Conference (WHC). Evanston, IL, USA. 2015. 321–326.
12 Green S. Particle simulation using CUDA. NVIDIA Whitepaper, 2010.