微可压缩 SPH 流体的稳定性固体边界处理算法

邵绪强，周　忠，张劲松，吴　威*
（北京航空航天大学虚拟现实技术与系统国家重点实验室　北京　100191）
（wuwei@buaa.edu.cn）

摘　要：固流耦合，即流体的固体边界处理一直是基于物理的流体模拟技术的研究重点.为解决 SPH 流体模拟中固流耦合存在的交界面处流体粒子衰减和穿透问题，提出一种固体采样边界粒子与动量守恒保持的位置-速度修正方案相结合的固流耦合方法.首先在预处理阶段对快速格子形状匹配(fast lattice shape matching，FLSM)模型表示的固体边界进行表面和内部边界粒子采样；然后在运行过程中计算流体粒子密度和受力时考虑边界固体粒子的相对贡献；最后利用动量守恒保持的位置-速度修正方案对流体粒子进行位置和速度的修正.为了提高计算速度以满足交互式应用需求，把每个迭代步长内的计算完全并行化后加载到 GPU 上进行加速处理.实验结果表明，该算法实现了微可压缩 SPH 流体与刚体以及弹性体的双向耦合，并可以高效、稳定地模拟固流耦合中的非穿透、液滴飞溅、溶解等复杂现象.

关键词：固流耦合；流体模拟；并行计算；拉格朗日粒子模型；形状匹配
中图法分类号：TP391.41

Stable Solid Boundary Handling Algorithm of Weakly Compressible SPH Fluids

Shao Xuqiang，Zhou Zhong，Zhang Jinsong，and Wu Wei
(*State Key Laboratory of Virtual Reality Technology and Systems*，*Beihang University*，*Beijing*　100191)

Abstract：Fluid-solid coupling，namely solid boundary handling，has been an important research topic in physically based fluid animation. In this paper，we propose a novel stable and fast solid boundary handling algorithm for weakly compressible smoothed particle hydrodynamics (WCSPH). First，we sample the deformable solids with inner and surface boundary particles in preprocessing stage. Then we compute the relative contributions of boundary particles to their neighboring fluid particles during the density and force computation of SPH fluids. Finally，in order to prevent the penetration artifacts near the fluid-solid interfaces simultaneously，we employ a momentum-conserving velocity-position correction scheme to adjust the velocities and positions of fluid particles whose distances to solid boundaries are smaller than a certain threshold. For improving the efficiency，we entirely implement the unified particle framework on GPUs using CUDA to accelerate the computation of each time step. The results show that the proposed method can handle the stable two-way coupling of WCSPH fluids and deformable solids modeled by fast lattice shape matching (FLSM)，and simulate complex phenomena in the coupling，such as non-penetrations，splashes and melting.

Key words：fluid-solid coupling；fluid simulation；parallel computing；Lagrangian particle modle；shape matching

---

收稿日期：2013-08-27；修回日期：2013-11-26.基金项目：国家“八六三”高技术研究发展计划(2012AA011801).邵绪强(1982—)，男，博士，讲师，主要研究方向为计算机图形学；周　忠(1978—)，男，博士，副教授，博士生导师，CCF 高级会员，主要研究方向为计算机图形学、分布式虚拟现实与计算机视觉；张劲松(1987—)，男，博士研究生，主要研究方向为计算机图形学；吴　威(1961—)，男，博士，教授，博士生导师，CCF 高级会员，论文通讯作者，主要研究方向为计算机图形学、分布式虚拟现实与计算机视觉.

流体固有的流动特性使其时刻都与固体边界存在着交互，并且在交互过程中产生复杂现象，例如液滴飞溅、泡沫、漩涡. 因此，固流耦合也就是流体的固体边界处理一直都是计算机动画中流体模拟技术[1]的重要研究内容.

基于光滑粒子动力学(smoothed particle hydrodynamics，SPH)方法的流体模拟中存在 3 类固流耦合方法：惩罚力方法[2-4]、直接力方法[5]和边界粒子方法；其中边界粒子方法又分为镜像粒子方法[6-8]和冻结粒子方法[9-12]. 惩罚力方法通过对流体粒子(fluid particle，FP)施加一个其与固体边界距离成反比例关系的反向作用力来实现固流耦合，它在固流交界面处会产生 FP 衰减问题，进而导致流体密度和压强的分布噪声. 此外，惩罚力方法必须选择足够小的时间步长来避免流体穿透固体表面. 直接力方法通过利用预测修正方法来计算交互力以及速度，实现了刚体和流体的双向交互，该方法支持使用较大的时间步长，但是会在固流交界面处因 FP 堆积而引起不规则密度分布问题. 边界粒子方法通过考虑边界粒子对流体的贡献可以保证在固流交界处获得连续的密度和压强分布. 在镜像粒子方法中，固体边界附近的 FP 被映射到边界的对面，得到镜像的虚拟 FP，但对于复杂的固体边界尤其是可变形固体，生成这些镜像粒子将变得极其复杂. 冻结粒子方法在预处理阶段对固体边界进行采样生成边界粒子，运行阶段这些边界粒子被当作 FP 进行处理；该方法可以产生光滑的密度和压强分布，但其在大的速度差下很难保证固流交界面的非穿透约束.

针对 SPH 流体在处理固流耦合时交界面处存在的粒子衰减和穿透问题，本文通过对固体边界进行采样生成表面和内部边界粒子，在交互过程中计算边界粒子对 FP 的相对贡献，并结合一个动量守恒保持的位置-速度修正方案，实现了微可压缩 SPH 流体(weakly compressible SPH，WCSPH)模型[13]和快速格子形状匹配(fast lattice shape matching，FLSM)变形模型[14]的耦合，保证了固流交界面处流体密度和压强的光滑分布和非穿透性约束. 我们把整个计算过程并行化后加载到 GPU 上进行加速执行，实现了大粒子规模下交互级固流耦合模拟.

1 相关工作

2003 年，Müller 等[15]利用 SPH 方法在计算机图形学中实现了逼真的交互式流体模拟. 此后，SPH 方法被广泛地用来模拟不可压缩流[13,16]、头发[17]、溶解[18]、多相流[19]和粘弹性固体[20]. 然而，更复杂的流体现象发生在流体和固体边界的耦合过程中.

在基于 SPH 的流体模拟中，大多数模拟器通过计算一个以 FP 到固体边界的距离衡量的惩罚力来处理流体的固体边界问题. 2004 年，Müller 等[2]利用惩罚力方法模拟了微可压 SPH 流体和有限元变形固体的交互，其对固体的三角形表面采样生成边界粒子，并计算 SPH 粒子到这些边界粒子的距离. 然而，该方法必须对计算惩罚力的刚度参数进行精心调整，以防止固流交界面处的压强分布噪声和穿透现象. 因此，惩罚力方法通常必须使用比较小的时间步长来保证固体边界处流体压强的均匀分布. 为了采用较大的时间步长，Becker 等[5]提出了直接力耦合方法，利用一个预测修正方案来计算交互力和速度，实现了流体和刚体的单向和双向交互. 直接力方法避免了固体交界面处的 FP 堆积现象，但是不能处理可变形的固体边界. 2012 年，Yang 等[4]提出了一种基于 GPU 的实时耦合方法来处理微可压 SPH 流体和非线性有限元变形模型的双向交互，该方法结合了直接力方法和预测修正方案，并可以处理不同的固体边界，避免了穿透现象，但仍会产生 FP 堆积现象.

镜像粒子方法通过为固体边界处的 FP 动态生成镜像虚拟粒子，并把这些粒子引入到 FP 的密度计算中去，进而实现稳定的固流耦合，可以保证固体边界处流体压强的均匀分布. 文献[6-7]使用镜像粒子方法，分别实现了 SPH 流体与直线和弯曲固体边界的交互. Schechter 等[8]提出了一种粒子采样方法为流体的自由表面和固体边界生成一个狭窄的镜像粒子层；它可以解决假性数值表面张力效果，并可以保证质量守恒，但是对于可变形固体边界很难生成镜像粒子.

2007 年，Solenthaler 等[9]提出了一个统一粒子框架来模拟流体、刚体和可变形物体以及它们之间的耦合. 该方法通过对固体边界进行粒子采样，并考虑了这些粒子对 FP 密度以及压力的贡献，因此保证了固流交界面处流体密度的均匀分布. 但该方法必须采用小的时间步长来保证固体变形的稳定性，以及固体表面的非穿透性. Ihmsen 等[10]通过把上述方法与直接力方法结合实现了一个新的单向固流耦合方案，不仅可以保证固体边界处流体密度的光滑分布，还可以使用较大的时间步长. Akinci 等[11]通过在固体表面采样一层边界粒子，并考虑它们对

流体的相对贡献，实现了 SPH 流体和刚体的动量守恒保持双向交互. 该方法解决了由于粒子采样导致的边界非均匀问题，但是不能解决可变形固体边界. 在其后续工作中[12]，通过自适应采样边界粒子，把该方法扩展到了流体与可变形固体边界的耦合模拟中. 然而，该方法仅仅依靠一层边界粒子不能保证边界的非穿透，尤其是当固体和流体之间的速度差较大时. 在文献[11]算法的基础上，本文对 FLSM 变形固体进行表面边界粒子(surface border particle, SBP)和内部边界粒子(inner border particle, IBP)采样，并结合一个动量守恒保持的位置-速度修正方案实现了 WCSPH 和可变形固体边界的稳定耦合.

2　稳定性固流耦合算法

2.1　微可压缩 SPH 流体

在拉格朗日粒子方法中，控制流体运动的非线性偏微分方程 Navier-Stokes 方程表示为 $\nabla \cdot \boldsymbol{v}_i = 0$，$\frac{\mathrm{d}\boldsymbol{v}_i}{\mathrm{d}t} = -\frac{1}{\rho_i}\nabla p_i + \mu \nabla^2 \boldsymbol{v}_i + \boldsymbol{F}_i$；其中，$\boldsymbol{v}_i$ 表示 FP i 的速度，ρ_i 为密度，p_i 为压强，μ 表示流体粘度系数，$\boldsymbol{F}_i$ 表示外力和.

根据 SPH 方法[13]，位置为 $\boldsymbol{x}_i$ 的 FP i 的物理量 A_i 可以根据其邻居粒子进行核插值得到，即

$$A_i = \sum_j m_j \frac{A_j}{\rho_j} W(x_{ij}, h) \tag{1}$$

其中，$\boldsymbol{x}_{ij} = \boldsymbol{x}_i - \boldsymbol{x}_j$，$m_j$ 为粒子质量，W 为紧支域半径为 h 的核插值函数. 所以根据式(1)，粒子 i 的密度为 $\rho_i = \sum_j m_j W(x_{ij}, h)$.

FP 之间存在相互作用力分为压力 $\boldsymbol{F}_i^p$ 和粘滞力 $\boldsymbol{F}_i^v$，它们计算公式分别为

$$\boldsymbol{F}_i^p = -\sum_j m_j \frac{p_i + p_j}{2\rho_j} \nabla W(\boldsymbol{x}_{ij}, h)$$

和

$$\boldsymbol{F}_i^v = \mu \sum_j m_j \frac{\boldsymbol{v}_{ji}}{\rho_j} \nabla^2 W(\boldsymbol{x}_{ij}, h);$$

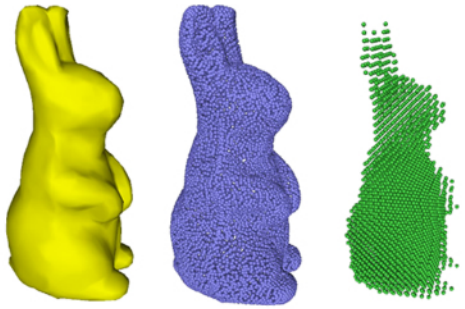
其中，FP 压强我们用文献[13]中的微可压缩 SPH 流体模型 WCSPH 计算，即 $p_i = \frac{\rho_0 c_s^2}{7}\left(\left(\frac{\rho_i}{\rho_0}\right)^7 - 1\right)$；$\rho_0$ 为流体初始密度，c_s 为声速.

2.2　固体边界采样和变形

我们利用改进的 FLSM 算法[14]对流体的可变形固体边界进行了稳定性模拟，提出了基于邻居格子顶点的目标位置进行 SPH 插值的方法，以驱动网格表面的变形.

2.2.1　边界粒子采样

如图 1 a 所示，对于输入的任一个三角形网格模型，我们对其进行粒子采样，生成 SBP 和 IBP，用于和 WCSPH 流体的交互. 首先，对于模型表面的每一个三角形，根据初始 FP 间距 r_0 进行剖分，并利用文献[2]算法中的高斯积分法则采样生成如图 1 b 所示 7 个顶点作为模型的 SBP；然后，对模型建立一个规则、有符号距离场，通过该规则距离场对模型的内部进行均匀粒子采样，生成如图 1 c 所示 IBP，同时 IBP 也作为 FLSM 变形模型的格子顶点控制弹性变形. 在本文的固流耦合算法中，SBP 用于计算固体和流体的相互作用力，在位置-速度修正中用于防止 FP 穿透固体表面；而 IBP 接受来自 SBP 的交互力，并基于 FLSM 算法控制模型的弹性变形.



a 三角网格模型　　b SBP　　C IBP

图 1　边界粒子采样

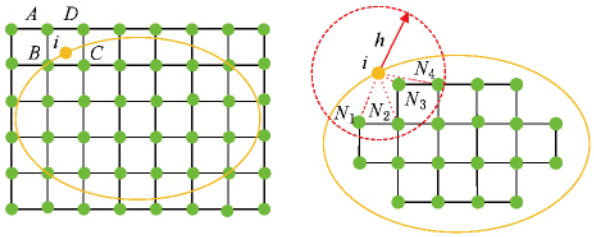
2.2.2　固体表面变形

本文提出了一种基于邻居格子顶点的目标位置进行 SPH 插值的方法，以驱动网格顶点 GP 的变形. 在原始的 FLSM 中，嵌套在格子中的三角形网格表面通过三线性插值更新位置，以实现变形，如图 2 a 所示，格子 $ABCD$ 中的网格顶点 i 在 t 时刻的位置为

$$\boldsymbol{x}_i^t = h_1 \boldsymbol{x}_A^t + h_2 \boldsymbol{x}_B^t + h_3 \boldsymbol{x}_C^t + h_4 \boldsymbol{x}_D^t \tag{2}$$

其中，h 为网格顶点 i 在格子 $ABCD$ 中的体积或面积坐标. 然而，线性插值式(2)在处理固流交互时不够准确，因为只有物体内部格子顶点 B 和 C 受到来自 FP 的交互作用力，而在计算 i 的位置和速度时，物体外面的格子顶点 A 和 D 却参与了插值计算. 针对这个问题，如图 2 b 所示，在预处理阶段搜索任一三角形顶点 i 的初始邻居格子顶点 IBP N_j，然后在每个时刻 t，采用基于邻居格子顶点的目标位置进行 SPH 插值的方法来驱动 i 的变形，即

$$\boldsymbol{x}_i^t = \frac{\sum_j W(\boldsymbol{x}_{ij}^0, h)(\boldsymbol{R}_j \boldsymbol{x}_{ij}^0 + \boldsymbol{g}_j)}{\sum_j W(\boldsymbol{x}_{ij}^0, h)};$$



a 初始FLSM算法[14] b 改进后FLSM算法

图 2 固体表面变形

其中，$x_{ij}^0 = x_i^0 - x_j^0$，x_i^0 和 x_j^0 是粒子的初始位置，g_j 为格子顶点的目标位置，R_j 为粒子的平均刚度旋转矩阵.

2.3 双向耦合

由于 SPH 方法的核函数是球形函数，因此位于自由表面和固流交界面处的 FP 存在不对称的邻居粒子分布，这将使 SPH 核插值公式在计算流体密度时产生误差，进而导致流体压强在边界处分布产生噪声，也就是所谓的粒子衰减问题. 为了进行稳定的固流耦合模拟，本文基于文献[11]算法提出了一种新的 SPH 固体边界处理算法.

如图 3 所示，在计算固流交界面附近 FP i 的密度时，我们同时考虑 IBP 和 SBP 对流体密度的相对贡献，其计算公式为

$$\rho_i = m_i \sum_j W(\boldsymbol{x}_{ij}, h) + m_i \sum_k \varphi(\boldsymbol{x}_k) W(\boldsymbol{x}_{ik}, h) \quad (3)$$

其中，j 表示相邻 FP，k 表示相邻的 IBP 和 SBP，$\phi(x_k) = \dfrac{V_k^t}{V_{\min}^0}$ 为时间相关的相对贡献函数，$V_{\min}^0$ 为初始时刻流体体积的最小值，V_k^t 是第 k 个相邻边界粒子的当前体积. 式(3)通过考虑边界粒子以 $\phi(x)$ 为度量的相对贡献，解决了由于粒子采样以及变形带来的固体边界非均匀特性. 与文献[11]算法相比，本文算法考虑了 IBP 对流体密度的贡献，在固流交界面处获得了更光滑的密度和压强分布.

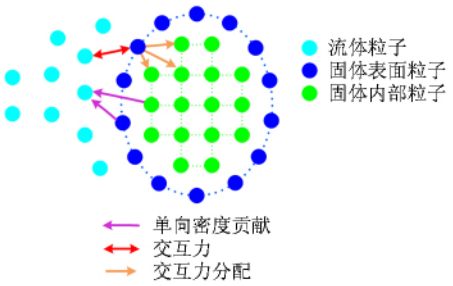


图 3 固流双向交互示意图

而在计算固体和流体之间的相互作用力时，本文算法只考虑 SBP 和 FP 之间的相互作用力；然后作用在 SBP 上的交互力通过 SPH 核插值公式分配到相邻的 IBP 上；最后，IBP 在耦合力的作用下控制物体的变形. 具体方法如下：

对于一对相邻的 FP i 和 SBP j，j 对 i 的交互力它们之间的耦合力 $\boldsymbol{F}_{i\leftarrow j}^c$ 包括压力 $\boldsymbol{F}_{i\leftarrow j}^p$，粘滞力 $\boldsymbol{F}_{i\leftarrow j}^v$ 以及界面张力 $\boldsymbol{F}_{i\leftarrow j}^I$. 考虑 SBP 的分布非均匀特性，其对 FP 的作用力同样采用时间相关的相对贡献函数 $\phi(x)$ 进行度量，因此 3 种耦合力计算公式分别为

$$\left.\begin{aligned} \boldsymbol{F}_{i\leftarrow j}^p &= -m_i \varphi(\boldsymbol{x}_j)\left(\frac{p_i}{\rho_i^2}\right) W(\boldsymbol{x}_{ij}, h) \\ \boldsymbol{F}_{i\leftarrow j}^v &= \mu m_i \varphi(\boldsymbol{x}_j) \frac{\boldsymbol{v}_{ji}}{\rho_j} \nabla^2 W(\boldsymbol{x}_{ij}, h) \\ \boldsymbol{F}_{i\leftarrow j}^I &= -\eta_j m_i \varphi(\boldsymbol{x}_j) W(\boldsymbol{x}_{ij}, h) \end{aligned}\right\} \quad (4)$$

根据牛顿第三定律，FP i 对 SBP j 的作用力为 $\boldsymbol{F}_{j\leftarrow i}^c = -\boldsymbol{F}_{i\leftarrow j}^c$. 最后通过把 SBP j 上的由式(4)计算得到的交互力分配到相邻的 IBP 上，并由 IBP 控制固体的变形. 对于任一 IBP k，它获得的交互力为

$$\boldsymbol{F}_k = \frac{\sum_j W(\boldsymbol{x}_{kj}, h) \boldsymbol{F}_j^c}{\sum_j W(\boldsymbol{x}_{kj}, h)}$$

2.4 位置-速度修正方案

如果固体和流体之间的速度差比较小，上述的耦合算法可以保证流体对固体表面的非穿透约束. 然而，当固体和流体之间的速度差超过一定阈值时，流体就会穿透固体表面，造成模拟的失真. 为了保证大速度差下流体对固体表面的非穿透，本文提出了一种动量守恒保持的位置-速度修正方案，对穿过固体表面的 FP 进行位置和速度的修正.

在模拟过程中，对于一对交互粒子：FP i 和 SBP j，如果满足 $\boldsymbol{v}_{ij} \cdot \boldsymbol{n}_j < 0$ 和 $|\boldsymbol{x}_{ij}| < r_0$，则认为 FP 在固体表面 $\boldsymbol{x}_j$ 处发生了穿透；其中，$\boldsymbol{n}_j$ 为 $\boldsymbol{x}_j$ 的固体表面法线，r_0 为初始 FP 间距. 根据穿透假定，如图 4 所示，FP i 在多个 SBP 的位置 $\boldsymbol{x}_j$ 处发生了穿透. 我们假定此时存在一个虚拟的边界粒子 k 和 FP i 发生交互，并在交互过程中对 i 的速度和位置进行修正

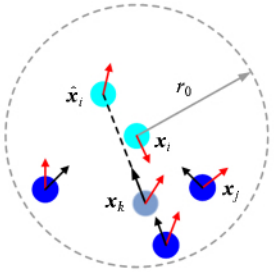


图 4 速度和位置修正

以避免穿透的发生. 虚拟边界粒子的物理属性值为 FP i 的所有相邻的被穿透 SBP 物理属性值的加权平均，即

$$A_k=\frac{\sum_j m_j A_j W(\boldsymbol{x}_{ij},r_0)}{\sum_j m_j W(\boldsymbol{x}_{ij},r_0)} \tag{5}$$

其中，$W(\boldsymbol{x}_{ij},r_0)=\max\left(0,\left(1-\frac{|\boldsymbol{x}_{ij}|^2}{r_0^2}\right)^3\right)$.

我们首先对 FP i 的位置进行修正，其计算公式为 $\hat{\boldsymbol{x}}_i=\boldsymbol{x}_k+r_0\boldsymbol{n}_k$；其中，法向量 $\boldsymbol{n}_k$ 由式(5)计算得到.

然后，根据固体边界的材料属性和动量守恒定律对 FP i 的速度进行修正；把 FP i 和虚拟边界粒子 k 的速度向位置 $\boldsymbol{x}_k$ 处的表面法线和切向方向进行投影，得到 $\boldsymbol{v}_i^n$，$\boldsymbol{v}_i^t$，$\boldsymbol{v}_k^n$ 和 $\boldsymbol{v}_k^t$. 根据法线和切线方向的动量守恒定律得到

$$m_i\boldsymbol{v}_i^n+m_k\boldsymbol{v}_k^n=m_i\hat{\boldsymbol{v}}_i^n+m_k\hat{\boldsymbol{v}}_k^n \tag{6}$$

$$m_i\boldsymbol{v}_i^t+m_k\boldsymbol{v}_k^t=m_i\hat{\boldsymbol{v}}_i^t+m_k\hat{\boldsymbol{v}}_k^t \tag{7}$$

其中，$\hat{\boldsymbol{v}}_i^n$，$\hat{\boldsymbol{v}}_i^t$，$\hat{\boldsymbol{v}}_k^n$ 和 $\hat{\boldsymbol{v}}_k^t$ 是碰撞之后的速度分量.

为了保证固体表面处的非穿透约束，需确保法线方向上碰撞之后的粒子速度相等，即 $\hat{\boldsymbol{v}}_i^n=-\hat{\boldsymbol{v}}_k^n$，将其代入式(6)，得到 $\hat{\boldsymbol{v}}_i^n=\hat{\boldsymbol{v}}_k^n=\frac{m_i\boldsymbol{v}_i^n+m_k\boldsymbol{v}_k^n}{m_i+m_k}$.

至于在切线方向上的速度修正，利用文献[4]方法定义一个变量∂来控制不同的滑移条件，即

$$\partial=\frac{\hat{\boldsymbol{v}}_i^t-\hat{\boldsymbol{v}}_k^t}{\boldsymbol{v}_i^t-\boldsymbol{v}_k^t} \tag{8}$$

其中，$\partial=0$ 表示碰撞中的切向无滑移边界，$\partial=1$ 表示自由滑移边界. 将式(8)代入式(7)中得到

$$\hat{\boldsymbol{v}}_i^t=\frac{(m_i+m_k\partial)\boldsymbol{v}_i^t+m_k(1-\partial)\boldsymbol{v}_k^t}{m_i+m_k}$$

和

$$\hat{\boldsymbol{v}}_k^t=\frac{(m_k+m_i\partial)\boldsymbol{v}_k^t+m_i(1-\partial)\boldsymbol{v}_i^t}{m_i+m_k}.$$

最后，虚拟粒子的速度变化 $\Delta\boldsymbol{v}_k=\hat{\boldsymbol{v}}_k-\boldsymbol{v}_k$ 通过 $\boldsymbol{v}_j+=\frac{\Delta\boldsymbol{v}_k\cdot\boldsymbol{v}_j}{|\boldsymbol{v}_k|}$ 分配到相邻的被穿透的 SBP 上.

2.5　固流耦合中的溶解模拟

本文耦合算法可以模拟固流耦合过程中由于热交换产生的溶解. 在我们的粒子模型中，每一个粒子被分配一个温度值 T_i，在模拟过程中，每个粒子都会与周围的邻居粒子交换热量. 此外，每个固体粒子存储一个和材料相关的溶点 T_{melt}. 根据文献[18]中的 SPH 插值公式，每个粒子由于热传递而引起的温度变化为

$$\frac{\mathrm{d}T_i}{\mathrm{d}t}=\sum_j\frac{m_j}{\rho_j}\frac{K_iK_j}{K_i+K_j}T_{ji}\nabla^2W(\boldsymbol{x}_{ij},h);$$

其中，K 表示固体材料的热传递系数.

随着热传递的进行，固体粒子的温度不断升高. 当到达溶点时，固体粒子将变为 FP. 因此，当固体粒子发生状态转化时，固体变形模型应该能够使其从所属固体脱离. 本文利用扩展的 FLSM 模型来模拟固流耦合过程中固体的溶解：我们只对可溶解固体的内部采样生成 IBP，并且把那些直接邻居粒子少于 26 个的 IBP 当作 SBP 来处理和流体的交互. 对于任一个转化为 FP 的 IBP 粒子，将它的直接邻居粒子链表清空，然后利用 WCSPH 流体模型计算它的运动. 对于剩下的固体粒子，重新计算它们的区域结构. 此外，在溶解过程中，如果一个 IBP 的直接邻居粒子数目少于 26 个，那么它将变为 SBP，用于计算和流体的相互作用力.

3　算法的 GPU 实现

为了实现 SPH 流体的实时模拟，GPU 的并行加速技术[21-22]得到了广泛应用. 本文把计算中需要的数据进行了设备端的优化存储，以充分利用 GPU 强大的并行计算能力：对于在计算中不变的只读数据，例如将粒子的初始位置、初始邻居粒子列表等作为纹理数据存储在设备纹理内存上；而对于计算中每个时间步长里需要更新的可读写数据，例如密度、压强、位置、速度、作用力、目标位置、形状匹配算法的区域结构、固体粒子的直接邻居粒子链表、最优变换等信息存储在显存的全局内存上. 本文将所有粒子的每种物理信息都存储在同一数组中，并用四元组的第 4 位来区分粒子属于哪一个物体的哪种类型粒子(FP：0；SBP：1，…，n；IBP：$n+1$，…，$2n$；其中，n 是物体的数目). 如图 5 所示，如果粒子 i 的标志位

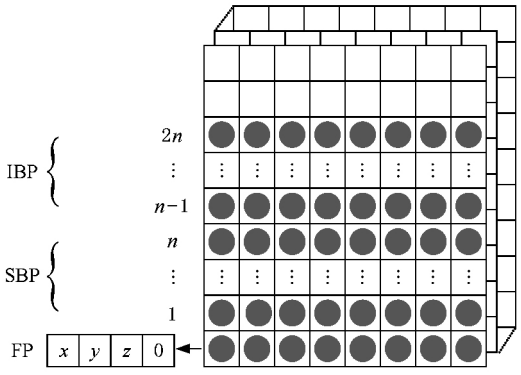


图 5　GPU 端数据存储

$I_i>n$ 并且 $I_i\%n=3$，那么粒子 i 是第 3 个物体的 IBP 粒子. 另外，为了模拟固体的变形和溶解，我们为每个 IBP i 在全局内存中分配一个一致大小的内存，并按距离大小来存储属于同一个区域的所有邻居 IBP 粒子. 对于每个 IBP i，还在全局内存中分配 2 个数组来存储所属的区域的数目和直接邻居粒子列表；这 2 个数组用来计算平均的目标位置和动态确定溶解模拟过程中的 SBP.

本文利用 WCSPH 算法在 GPU 上并行执行来计算流体的运动以及流体和固体的作用力. 在模拟过程中，搜索每个粒子的邻居粒子是必须的也是最耗时的操作. 基于文献[23]算法，本文启动内核函数在 GPU 上建立一个 *KD*-Tree 进行遍历来快速搜索邻居粒子. 对于 WCSPH 的并行执行，首先启动一个内核函数并根据式(3)来计算每个 FP 的密度和压强；然后在另一个内核函数里计算 FP 的压力、粘滞力、界面张力以及和 SBP 的作用力. 接着，对所有的 SBP 启动另一个内核函数，把它们受到的来自流体的作用力分配到相邻的 IBP 上.

对于所有的的粒子启动一个内核函数，并根据 Leap-frog 积分方法来更新它们的位置和速度. 如果是 FP，则需要利用位置-速度修正方案对其位置和速度进行修正，防止流体穿透固体表面.

固体的运动和变形通过在 GPU 上并行执行的 FLSM 算法进行求解. 首先，对所有的 IBP 也就是格子节点启动一个内核函数，更新它们的直接邻居粒子列表、所属的区域数、索引的区域信息. 然后，对每一个区域启动另一个内核函数，根据形状匹配算法计算最优的平移向量和旋转矩阵，并把平均化的粒子目标位置散射到显卡的全局内存数组上. 如果模拟固体的溶解，还要根据直接邻居粒子数据动态更新 SBP 集合和 IBP 集合.

为了跟踪流体表面并绘制流体，本文采用文献[21]算法在 GPU 上动态确定位于自由表面的 FP，并建立一个规则距离场来定义流体的隐式表面，然后利用 NVIDIA 公司 CUDA 开发包“Marching Cubes Isosurfaces”算法提取出流体的三角形网格流体表面. 最后对输出的三角形网格表面利用开源的光线跟踪绘制引擎 POV-Ray 进行真实感绘制.

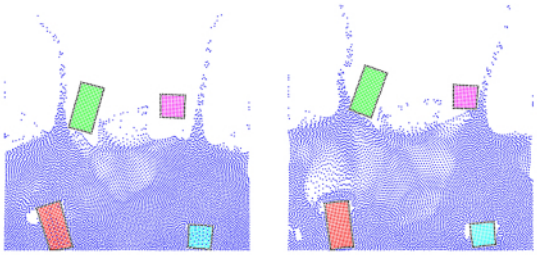
4 实验结果

本文通过一系列实验证明所提出的固流耦合算法是稳定、快速且有效的. 实验硬件平台为 Intel Xeon E5630 CPU 和 NVIDIA Geforce GTX 680 GPU，编程环境为 C++，OpenGL 和 CUDA.

4.1 稳定性分析

本文算法对固体进行表面和 IBP 采样，在计算流体密度和耦合力的过程中考虑这些边界粒子相对贡献，并结合动量守恒保持的位置-速度修正方案，解决了 SPH 固体边界处理中存在的粒子衰减以及穿透问题，实现了稳定的固流耦合模拟.

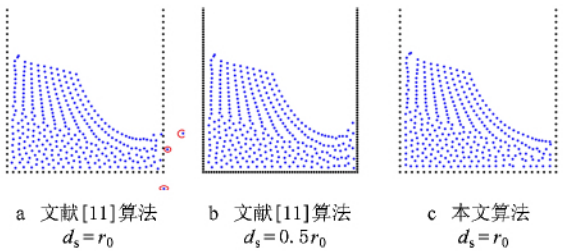
图 6 所示为的二维场景中，4 个不同密度的弹性固体(从左到右密度分别为 5 000，300，200，3 000)以较大的入水速度掉入水中的交互效果；黑色粒子代表 SBP，蓝色代表 FP，其他颜色代表 IBP. 图 6 a 给出了不施加位置-速度修正方案的耦合结果，结果表明，在固流速度差较大的情况下，只考虑边界粒子的相对贡献会发生穿透；图 6 b 是加入了位置-速度修正方案的交互结果. 从对比结果可以看出，在整体耦合结果基本保持一致的情况下，本文的位置-速度修正方案避免了较大固流速度差下的穿透现象.



a 不施加位置-速度修正　　b 施加位置-速度修正

图 6 避免固流交互中的穿透

在图 7 中给出的水坝与刚性立方体盒子交互的二维模拟场景中，将本文算法与文献[11]算法进行了对比；其中 FP 的初始间距为 r_0，边界粒子的采样距离记作 d_s. 文献[11]算法是否发生穿透取决于边界粒子的采样密度. 如图 7 a 所示，当边界粒子的采样距离 $d_s=r_0$ 时，会有较多的 FP 穿透固体表面(红色圆圈标注的粒子). 如图 7 b 所示当采样距离 $d_s=$



a 文献[11]算法 $d_s=r_0$　b 文献[11]算法 $d_s=0.5r_0$　c 本文算法 $d_s=r_0$

图 7 与文献[11]算法的对比

0.5r_0 时，几乎没有 FP 穿过固体表面. 而图 7 c 所示本文算法由于引入了动量守恒保持的位置-速度修正方案，在相对稀疏的采样距离 $d_s=r_0$ 的情况下也不会发生穿透.

4.2　真实感交互效果

本节给出了利用本文算法模拟的真实感固流耦合效果. 如图 8 所示，我们模拟了水流冲击在下端固定的弹性兔子表面上的双向耦合结果. 基于 FLSM 兔子模型在水流的作用下发生弹性变形，同时水流在兔子模型的反向作用产生了大量的飞溅水滴. 由于本文算法在交互力计算中引入了界面张力，因此可以模拟水流在固体表面的滑落现象.

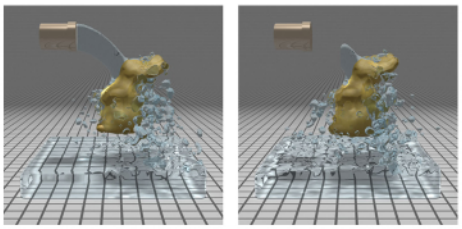


图 8　水流冲击底端固定的弹性兔子模型

图 9 模拟了水流流入空的弹性立方体盒子内部的双向耦合效果. 在水流的冲击下，盒子发生了弹性变形并逐渐被水填满. 从效果图可以看出，在较大固流速度差和固体较大弹性变形的情况下，本文算法可以有效地防止流体对固体表面的穿透.

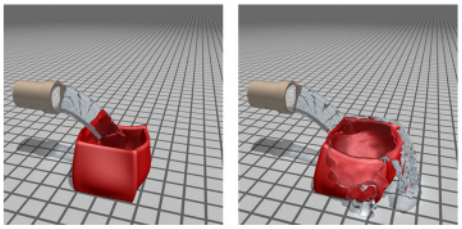


图 9　水倒入空的弹性盒子

利用 FLSM 算法模拟了布料并利用本文算法模拟了布料和水流的交互如图 10 所示. 在模拟过程中，用于表示布料的规则四边形网格顶点被同时选作 IBP 和 SBP. 从交互结果可以得出，本文固流耦合算法可以模拟在只有一层边界粒子下的非穿透双向固流交互.

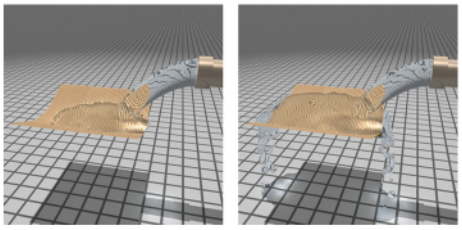


图 10　水流与两端固定的布料的交互

图 11 给出了 4 个不同密度物体（密度从左到右依次是：8 000，300，200，5 000）掉入水中并逐渐溶解的效果模拟. FLSM 可以模拟固流耦合中的溶解：把温度超过一定阈值的固体粒子变为 FP，并重新建立每个固体粒子的区域. 另外，为了模拟可溶解固体和流体的双向交互，固体表面粒子即直接邻居粒子少于 6 个的固体粒子被当作 SBP，并在溶解的模拟过程中每帧都要重新确定 SBP 的集合.

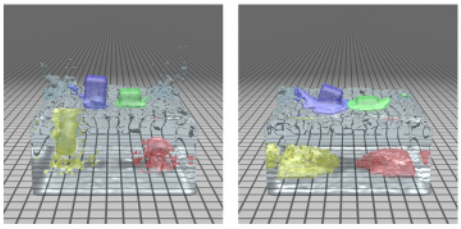


图 11　固流交互中的溶解

在图 12 中，模拟了多个具有复杂几何结构的刚体和弹性体掉入水中的固流交互模拟效果. 从效果图可以看出，本文耦合算法可以稳定、真实地模拟不同密度的复杂固体模型和 WCSPH 流体的双向交互.

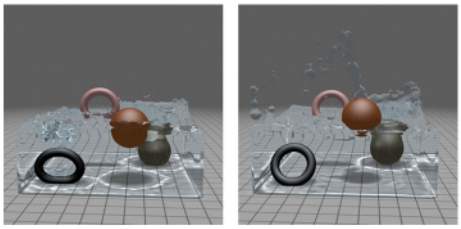


图 12　水坝和多个刚性和柔性固体的交互

4.3　时间效率分析

本文提出的基于 GPU 的固流双向耦合算法具有明显的时间性能优势. 本文把整个模拟流程并行化，并采用 CUDA 编程接口把计算加载到 GPU 上执行来加速每一个时间步长内的物理计算，在大分辨率粒子规模下达到了交互级的模拟速度. 如表 1 所示，我们给出了上述每个三维模拟场景的粒子数以及计算时间统计；其中，# FP 表示流体粒子数，# SBP 表示表面边界粒子数，# IBP 表示内部边界粒子数. 从表中统计数据可以得出，在粒子规模达到如图 12 所示 112k 的模拟场景中，模拟速度达到了 10 帧/s.

表 1 每个步长的计算时间

场景	#FP	#SBP	#IBP	时间/ms
图 8	25k	3.5k	6k	48.7
图 9	15k	5k	3k	39.3
图 10	15k	2.5k		38.6
图 11	40.5k	22k		95.1
图 12	70k	20k	20k	95

5 结 论

本文提出了一种稳定快速的固流耦合算法来处理微可压缩流体 WCSPH 的固体边界问题. 该算法对任一个输入的三角形网格模型进行表面和内部的边界粒子采样,在计算固流耦合时考虑边界粒子对流体的相对贡献,解决了固流交互过程中交界面附近的粒子衰减问题. 同时引入一个动量守恒保持的位置-速度修正方案,保证了较大速度差固流耦合的非穿透约束. 通过把整个算法并行化后加载到 GPU 上进行计算,实现了较大粒子规模下交互级的固流耦合模拟.

参考文献(References):

[1] Liu Youquan, Liu Xuehui, Zhu Hongbin, *et al*. Physically based fluid simulation in computer animation [J]. Journal of Computer-Aided Design & Computer Graphics, 2005, 17(12): 2581-2589 (in Chinese)
(柳有权, 刘学慧, 朱红斌, 等. 基于物理的流体模拟动画综述[J]. 计算机辅助设计与图形学学报, 2005, 17(12): 2581-2589)

[2] Müller M, Schirm S, Teschner M, *et al*. Interaction of fluids with deformable solids [J]. Computer Animation and Virtual Worlds, 2004, 15(3/4): 159-171

[3] Harada T, Koshizuka S, Kawaguchi Y. Smoothed particle hydrodynamics on GPUs [C] //Proceedings of Computer Graphics International. Washington DC: IEEE Computer Society, 2007: 63-70

[4] Yang L P, Li S, Hao A M, *et al*. Realtime two-way coupling of meshless fluids and nonlinear FEM [J]. Computer Graphics Forum, 2012, 31(7): 2037-2046

[5] Becker M, Tessendorf H, Teschner M. Direct forcing for lagrangian rigid-fluid coupling [J]. IEEE Transactions on Visualization and Computer Graphics, 2009, 15(3): 493-503

[6] 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

[7] Morris J P, Monaghan J J. A switch to reduce SPH viscosity [J]. Journal of Computational Physics, 1997, 136(1): 41-50

[8] Schechter H, Bridson R. Ghost SPH for animating water [J]. ACM Transactions on Graphics, 2012, 31(4): Article No. 61

[9] Solenthaler B, Schläfli J, Pajarola R. A unified particle model for fluid-solid interactions [J]. Computer Animation and Virtual Worlds, 2007, 18(1): 69-82

[10] Ihmsen M, Akinci N, Gissler M, *et al*. Boundary handling and adaptive time-stepping for PCISPH [C] //Proceedings of Workshop on Virtual Reality Interaction and Physical Simulation. Aire-la-Ville: Eurographics Association Press, 2010: 79-88

[11] Akinci N, Ihmsen M, Akinci G, *et al*. Versatile rigid-fluid coupling for incompressible SPH [J]. ACM Transactions on Graphics, 2012, 31(4): Article No. 62

[12] Akinci N, Cornelis J, Akinci G, *et al*. Coupling elastic solids with smoothed particle hydrodynamics fluids [J]. Computer Animation and Virtual Worlds, 2013, 24(3/4): 195-203

[13] Becker M, Teschner M. Weakly compressible SPH for free surface flows [C] //Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation. Aire-la-Ville: Eurographics Association Press, 2007: 209-217

[14] Rivers A R, James D L. FastLSM: fast lattice shape matching for robust real-time deformation [J]. ACM Transactions on Graphics, 2007, 26(3): Article No. 82

[15] Müller M, Charypar D, Gross M. Particle-based fluid simulation for interactive applications [C] //Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation. Aire-la-Ville: Eurographics Association Press, 2003: 154-159

[16] Solenthaler B, Pajarola R. Predictive-corrective incompressible SPH [J]. ACM Transactions on Graphics, 2009, 28(3): Article No. 40

[17] Hadap S, Magnenat-Thalmann N. Modeling dynamic hair as a continuum [J]. Computer Graphics Forum, 2001, 20(3): 329-338

[18] Iwasaki K, Uchida H, Dobashi Y, *et al*. Fast Particle-based visual simulation of ice melting [J]. Computer Graphics Forum, 2010, 29(7): 2215-2223

[19] Solenthaler B, Pajarola R. Density contrast SPH interfaces [C] //Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation. Aire-la-Ville: Eurographics Association Press, 2008: 211-218

[20] Clavet S, Beaudoin P, Poulin P. Particle-based viscoelastic fluid simulation [C] //Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation. New York: ACM Press, 2005: 219-228

(下转第 1929 页)

高的计算效率.

本文工作还存在很多的发展空间：如 $f(t)$ 在某点和 t 轴很接近时，由于有限精度等因素的影响，计算出来的包围多项式可能会在此处得到一个根. 如何寻求更加鲁棒的算法来解决这个问题，以及寻求更加高效、具有更高收敛阶的裁剪方法将是本文将来的工作.

参考文献(References)：

[1] Elber G, Kim M S. Geometric constraint solver using multivariate rational spline functions [C] //Proceedings of the 6th ACM Symposium on Solid Modeling and Applications. New York: ACM Press, 2001: 1-10

[2] Jaklič G, Kozak J, Krajnc M, *et al*. On geometric interpolation by planar parametric polynomial curves [J]. Mathematics of Computation, 2007, 76(260): 1981-1993

[3] Liu L G, Zhang L, Lin B B, *et al*. Fast approach for computing roots of polynomials using cubic clipping [J]. Computer Aided Geometric Design, 2009, 26(5): 547-559

[4] Sederberg T W, Nishita T. Curve intersection using Bézier clipping [J]. Computer-Aided Design, 1990, 22(9): 538-549

[5] Patrikalakis N M, Maekawa T. Intersection problems [M] // Shape Interrogation for Computer Aided Design and Manufacturing. Heidelberg: Springer, 2002: 109-160

[6] Choi Y K, Wang W P, Liu Y, *et al*. Continuous collision detection for elliptic disks [J]. IEEE Transactions on Robotics, 2006, 22(2): 213-224

[7] Chen X D, Yong J H, Wang G Z, *et al*. Computing the minimum distance between a point and a NURBS curve [J]. Computer-Aided Design, 2008, 40(10/11): 1051-1054

[8] Collins G E, Akritas A G. Polynomial real root isolation using Descartes's rule of signs [C] //Proceedings of the 3rd ACM Symposium on Symbolic and Algebraic Computation. New York: ACM Press, 1976: 272-275

[9] Rouillier F, Zimmermann P. Efficient isolation of polynomial's real roots [J]. Journal of Computational and Applied Mathematics, 2004, 162(1): 33-50

[10] Chen Jinsong. Stability criterion and algorithm for finding roots of polynomial [J]. Acta Mathematicae Applicatae Sinica, 2003, 26(2): 312-317 (in Chinese)
(程锦松. 稳定性判定与多项式求根算法[J]. 应用数学学报, 2003, 26(2): 312-317)

[11] Liu Dong, Feng Yong, Zhang Chaihuan, *et al*. An improved algorithm for real root isolation of univariate polynomials [J]. Journal of Shanghai Jiaotong University, 2010, 44(11): 1477-1480 (in Chinese)
(刘　栋，冯　勇，张彩环，等. 一种改进的多项式实根隔离算法[J]. 上海交通大学学报, 2010, 44(11): 1477-1480)

[12] Wei Feifei, Zhou Fei, Feng Jieqing. Survey of real root finding of univariate polynomial equation in CAGD/CG [J]. Journal of Computer-Aided Design & Computer Graphics, 2011, 23(2): 193-207 (in Chinese)
(卫飞飞，周　飞，冯结青. CAGD/CG 领域中一元多项式方程求根问题综述[J]. 计算机辅助设计与图形学学报, 2011, 23(2): 193-207)

[13] Kalantari B. Algorithms for quaternion polynomial root-finding [J]. Journal of Complexity, 2013, 29(3/4): 302-322

[14] Farouki R T, Goodman T N T. On the optimal stability of the Bernstein basis [J]. Mathematics of Computation, 1996, 65(216): 1553-1566

[15] Bartoň M, Jüttler B. Computing roots of polynomials by quadratic clipping [J]. Computer Aided Geometric Design, 2007, 24(3): 125-141

[16] Davis P J. Interpolation and Approximation [M]. New York: Dover Publications Press, 1975

(上接第 1922 页)

[21] Goswami P, Schlegel P, Solenthaler B, *et al*. Interactive SPH simulation and rendering on the GPU [C] //Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation. Aire-la-Ville: Eurographics Association Press, 2010: 55-64

[22] Chen Xi, Wang Zhanye, He Jian, *et al*. An integrated algorithm of real-time fluid simulation on GPU [J]. Journal of Computer-Aided Design & Computer Graphics, 2010, 22(3): 396-405 (in Chinese)
(陈　曦，王章野，何　戡，等. GPU 中的流体场景实时模拟算法[J]. 计算机辅助设计与图形学学报, 2010, 22(3): 396-405)

[23] Zhou K, Hou Q M, Wang R, *et al*. Real-time KD-tree construction on graphics hardware [J]. ACM Transactions on Graphics, 2008, 27(5): Article No. 126