一种基于二维光滑粒子法的流体仿真方法*

张海超 郑丹晨 边茂松 韩敏†

(大连理工大学电子信息与电气工程学部, 大连 116023)

(2016 年 4 月 27 日收到; 2016 年 7 月 22 日收到修改稿)

针对大场景下流体仿真计算复杂度高的问题, 本文以浅水方程为基础, 提出一种改进的二维光滑粒子方法. 该方法中使用光滑粒子法离散二维浅水方程, 将水深作为粒子的属性, 把计算复杂度降到二维的程度; 同时为了提高邻域粒子的搜索效率, 提出一种基于动态网格的邻近粒子搜索方法; 并使用虚粒子和惩罚力相结合的方法处理边界条件以高效率应对复杂边界; 渲染时, 首先将粒子映射并插值到规则网格内得到流体表面, 避免三维流体表面重构复杂度高的问题, 最后利用 OpenGL 着色语言实现加速渲染, 从而达到大场景下流体实时仿真.

关键词: 光滑粒子法, 浅水方程, 流体仿真, 三维模拟

PACS: 47.11.–j, 47.85.–g **DOI:** 10.7498/aps.65.244701

1 引 言

光滑流体动力学 (smoothed particle hydrodynamics, SPH) 方法 [1] 作为纯粹无网格方法, 避免了网格的复杂操作, 从而应用于溃坝洪水模拟 [2]、河流仿真 [3] 等流体仿真中. 为降低计算复杂度, 许多研究者使用 SPH 方法求解二维浅水方程建立流体模型.

针对 SPH 方法, Ata 和 Soulaïmani [4] 引入 Riemann 解以提高数值模拟的稳定性, 并分析了平坦地形下的溃坝问题; de Leffe 等 [5] 建立了模拟沿海流模型, 考虑了复杂海滩地形和干湿条件. 这些研究多是对 SPH 的求解模型进行理论上的改进和数值上的仿真, 并未对流体的交互、渲染等问题做深入探讨. 为此, Lee 和 Han [6] 给出了二维情况下的用于求解浅水方程的修正 SPH 模型, 并考虑了固体对流体的交互, 但该模型较为简单, 并未研究边界处理以及复杂地形的影响; 在此基础上, Solenthaler 等 [7] 将此模型扩展到任意地形情况, 并考虑了固体和流体的双向耦合交互, 并使用并行架构的技术手段加速计算, 但其边界处理采用简单的排斥力模型. 对于 SPH 边界处理, He 等 [8] 提出一个交错无网格方法, 通过映射质量和动量到这些交错粒子上, 达到将速度场和压力场分离的效果. Cornelis 等 [9] 引进一个候选粒子概念, 实现粒子能够在流体粒子边界中动态交换. He 等 [10] 提出一种带复杂边界流体自适应实时仿真模型, 模型自适应采样规则考虑了几何复杂度、物理复杂度和附加边界条件因素, 并提出一个两步复杂边界碰撞检测方法. 在处理固壁边界问题时, 刘虎等 [11] 从固壁物理原理出发, 应用多层虚粒子, 提出一种新型固壁边界施加模型. 韩亚伟等 [12] 提出一种改进的排斥力模型, 并应用到溃坝算例, 进一步说明该模型能够适应于复杂边界. Hu 等 [13] 提出条形化 (point-in-box, PIB) 搜索算法, 解决对粒子分布形式敏感的缺陷, 同时具有很高的搜索效率. Xia 和 Liang [14] 设计了一个基于 GPU 的 SPH 模型来求解二维浅水方程, 并使用四叉树邻域搜索算法优化模型的性能. Goswami 等 [15] 采用 Z-index 哈希映射和使用统一计算设备架构 (compute unified device architecture, CUDA) 并行排序优化邻域粒子搜索方法, 但这又增加了对硬件的要求, 因此尚未达到

* 国家自然科学基金 (批准号: 61374154) 资助的课题.

† 通信作者. E-mail: minhan@dlut.edu.cn

普遍适用的程度. 对于大场景的流体仿真, 计算复杂度问题一直未能很好地解决.

针对较大场景的流体仿真问题, 尤其是对于大场景下的流体存在较大的变形行为, 传统的基于树形结构的邻近粒子搜索方法并不能很好地解决计算量剧增的问题. 为提高搜索效率, 本文提出动态网格的邻近粒子搜索方法; 对于较大地形, 边界条件较为复杂, 仅仅使用虚粒子法并不能保证计算速度, 本文仅使用类型1的虚粒子以保持边界处的计算精度, 结合使用惩罚力项[16]以防止粒子的边界穿透; 采用相对简单的双线性插值法处理部分缺失值, 同时加密表面网格, 得到流体表面高程图并渲染绘制.

2 求解浅水方程的光滑粒子法

2.1 浅水方程

对于大场景下的流体运动, 垂直方向上的流速水力参数变化小于水平方向, 将纳维-斯托克斯方程沿水深方向积分, 从而得到浅水方程, 其Lagrangian形式为

$$\frac{\mathrm{D}h}{\mathrm{D}t} = -h\nabla\cdot\boldsymbol{u}, \tag{1}$$

$$\frac{\mathrm{D}\boldsymbol{u}}{\mathrm{D}t} = -g\nabla(h+H), \tag{2}$$

其中, $\dfrac{\mathrm{D}}{\mathrm{D}t} = \dfrac{\partial}{\partial t} + u\dfrac{\partial}{\partial x} + v\dfrac{\partial}{\partial y}$ 为全微分算子, h 表示地面上水体高度, $\boldsymbol{u} = (u, v)$ 表示流体水平速度, g 为重力加速度, H 表示地形高程值, ∇ 为梯度算子.

2.2 浅水方程的光滑粒子法求解模型

在光滑粒子流体动力学方法[7]中, 将流体离散为携带物质属性的粒子. 三维情况下, 对每个粒子 i, 其标量函数值 A_i 可由该粒子支持域内所有粒子 j 叠加求和得到:

$$A_i = \sum_j^N \frac{m_j}{\rho_j} A_j W(\boldsymbol{r}_i - \boldsymbol{r}_j, l), \tag{3}$$

式中, N 为粒子总数量, ρ_j 和 m_j 分别为粒子密度和质量, $\boldsymbol{r}_i$ 和 $\boldsymbol{r}_j$ 分别表示对应粒子的三维坐标, $W(\boldsymbol{r}_i - \boldsymbol{r}_j, l)$ 表示支持域长为 l 的光滑核函数.

二维情况下, 光滑粒子表示为具有不同高度的水柱, 采用Rodriguez-Paz和Bonet[17]所提方法, 其高度可由密度 $\rho_{2\mathrm{D}}$ 来表示:

$$h = \rho_{2\mathrm{D}}/\rho_{3\mathrm{D}}. \tag{4}$$

因此在二维下, (3)式修正为

$$A_i = \sum_j^N \frac{V_j}{h_j} A_j W(\boldsymbol{r}_i - \boldsymbol{r}_j, l), \tag{5}$$

其中, V_j 为粒子 j 的体积, h_j 表示粒子 j 的高度. 由Lee和Han[6]总结得出, V_j 相当于三维下粒子的质量, h_j 相当于粒子密度. 而水深 h_j 表示为

$$h_i = \sum_j^N V_j W(\boldsymbol{r}_i - \boldsymbol{r}_j, l). \tag{6}$$

将(6)式应用于浅水动量方程((2)式)中, 其可改写为

$$\frac{\mathrm{D}\boldsymbol{u}_i}{\mathrm{D}t} = -g\sum_j^N V_j \nabla W(\boldsymbol{r}_i - \boldsymbol{r}_j, l) - g\nabla H. \tag{7}$$

本文采用一种新的四次光滑核函数[18], 相较于三次光滑函数, 其二阶导数更为光滑, 因此SPH解能够获得更好的稳定性. 核函数的表达式:

$$W(R,l) = \alpha_{\mathrm{d}} \times \begin{cases} \left(\dfrac{2}{3} - \dfrac{9}{8}R^2 + \dfrac{19}{24}R^3 - \dfrac{5}{32}R^4\right), \\ \qquad 0 \leqslant R \leqslant 2, \\ 0, \quad R > 2, \end{cases} \tag{8}$$

其中 $\alpha_{\mathrm{d}} = \dfrac{15}{7\pi h^2}$, $R = \dfrac{\|\boldsymbol{r}_i - \boldsymbol{r}_j\|}{l}$.

3 二维光滑粒子法的流体仿真改进策略

3.1 动态网格粒子搜索方法

在SPH中, 仿真精度主要受粒子支持域的光滑长度 l 的影响. 对于大场景中的大变形问题, 需对光滑长度进行修正. 本文采用二维情况下的平均密度对光滑长度进行动态调整[17]:

$$l = l_0\left(\frac{\rho_0}{\rho}\right)^{\frac{1}{d}} = l_0\left(\frac{m_0/V_0}{m/V}\right)^{\frac{1}{d}} = l_0\left(\frac{h_0}{h}\right)^{\frac{1}{d}}, \tag{9}$$

式中, l_0 表示初始时刻光滑长度, ρ_0 为初始粒子的密度, h_0 为始时刻粒子水深, d 为维度, 二维情况下值为2.

在对光滑长度进行动态修正时, 通常使用树形搜索方法查找邻域粒子. 然而, 该方法的建树过程较耗时, 并且当粒子处于所分卦限边缘时, 树形法搜索效率较低. 为简化邻域粒子的搜索操作, 本文采用PIB搜索算法思想[13], 提出基于动态网格邻近粒子搜索方法.

该方法主要思路: 设二维空间内有 N 个粒子, 首先根据粒子横坐标 x 值进行升序排序. 对于待搜索邻域内粒子 $P_i(x_i, y_i), i = 1, 2, \cdots, N$, 先搜索横坐标上距 x_i 为 $2l_i$ 的粒子, 再判断这些粒子纵坐标 y 值是否距 y_i 也为 $2l_i$. 如果是, 则计算两个粒子间的距离, 若距离值小于等于 l_i, 则将该粒子索引加入到粒子 P_i 邻近粒子列表中.

为进一步优化搜索方法中的查询操作, 忽略粒子 x, y 值的小数部分, 即每个粒子坐标值为 0 到 k 的一个整数, 如此便可使用计数排序以简化计算复杂度. 在排序算法中, 使用辅助数组存储每个坐标值相等的元素个数, 因此可以快速地得到在排序后的粒子索引, 便于邻域区间的查找. 图 1 为动态网邻域粒子搜索方法. 其具体过程如下.

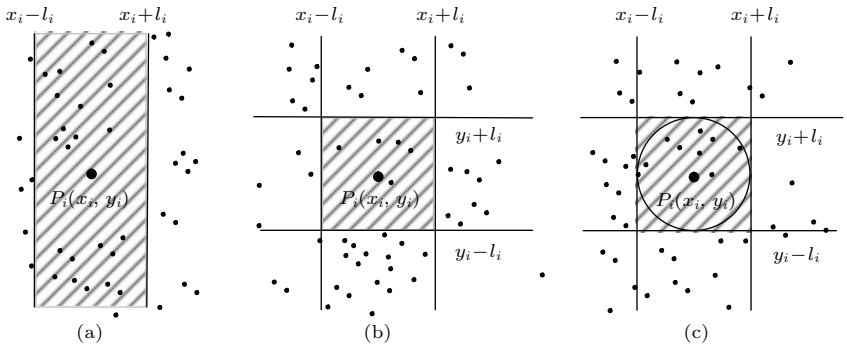

图 1 动态网格的邻近粒子搜索方法

Fig. 1. Neighboring particles search method based on dynamic grid.

步骤 1 拷贝 N 个粒子坐标值和索引号到临时的存储空间 $\boldsymbol{A}[1, 2, \cdots, N]$, $\boldsymbol{A}.length = N$. 根据 N 个粒子的坐标值, 按照 x 坐标进行升序计数排序得到 $\boldsymbol{B}[1, 2, \cdots, N]$, $\boldsymbol{B}.length = N$. 其中, 使用辅助数组 $\boldsymbol{C}[i](i = 0, 1, \cdots, k)$ 存储小于等于 i 的粒子个数.

步骤 2 如图 1 (a) 阴影所示查找邻域粒子, 对待查找的邻域粒子 $P_i(x_i, y_i)$, 首先确定其区间 $[x_l, x_r] = [x_i - l_i, x_i + l_i]$ 内粒子, 即 $\boldsymbol{B}[\boldsymbol{C}[x_l], \cdots, \boldsymbol{C}[x_r]]$ 内的粒子.

步骤 3 然后考察 $\boldsymbol{B}[\boldsymbol{C}[x_l], \cdots, \boldsymbol{C}[x_r]]$ 内的每一个粒子, 如图 1 (b) 阴影部分所示, 判断其 y 值是否也在区间 $[y_i - l, y_i + l]$ 内, 若在, 进入下一步骤, 否则判断下一个粒子. 当遍历所有粒子后, 返回步骤 2, 继续查找其他的邻近粒子.

步骤 4 计算两粒子的距离, 如果小于等于 $2l_i$, 则存储该粒子索引和距离值到索引链表和距离链表中; 否则, 返回步骤 3 继续考察下一个粒子.

由于每次循环都要重新搜索所有粒子的邻域粒子, 因此图 1 中阴影表示的网格大小是变化的, 其内部粒子数量不同, 因此这里只对粒子平均分布情况下进行时间复杂度分析. 首先, 步骤 1 中的复杂度为 $O(N + k)$; 之后在步骤 2 中, 在确定区间操作时采用了辅助数组, 所以时间复杂度为常数 $O(1)$; 步骤 3 中需要对 $2l_i \times N/k$ 个粒子进行相互比较操作, 步骤 4 中需要计算 $4l_i^2 \times N/k^2$ 个粒子的间距和判断操作, 因此, 对每个粒子搜索邻域粒子的时间复杂度为 $O(2l_i \times N/k) + O(4l_i^2 \times N/k^2)$, 总时间复杂度为 $O(N + k) + O(N(2l_i \times N/k)) + O(N(4l_i^2 \times N/k^2))$. 当仿真模拟的范围较大, k 达到 $O(N)$ 时, 上述时间复杂度约为 $O(N) + O(Nl_i)$. 因此, 此方法适合于粒子分布在较大范围场景的情况.

3.2 虚粒子与惩罚力方法的边界处理

在 SPH 中, 边界粒子只受到单边影响使得支持域内粒子数量不足, 从而导致求解错误. 对此问题, 虚粒子法已成为目前常用的解决方案, 示意图见图 2.

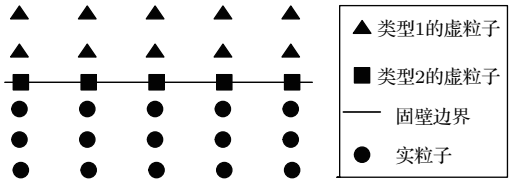

图 2 固壁边界的实粒子与两种虚粒子

Fig. 2. Solid particles and two types of virtual particles at solid boundary.

设置在边界上的为类型2的虚粒子, 对实粒子施加边界排斥力以防止其穿越边界; 类型1的虚粒子为镜像粒子. 但是该方法需布置较密集的类型2虚粒子, 增加了对复杂边界处理的难度, 同时增大了更新实粒子的计算量.

然而在二维SPH中, 自由流体的边界为多边形, 而复杂地形边界是由不规则曲面组成, 如在地形表面设置较多类型2的虚粒子, 则不能够确定流体边界处的情况. 为降低该部分的计算量, 本文提出了一种惩罚力和虚粒子相结合的边界处理方法. 其中, 仅使用类型1的虚粒子以提高边界处的计算精度, 另外, 在边界施加惩罚力项增加排斥力防止实粒子越界. 如图3所示.

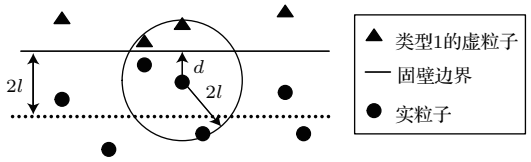

图3 固壁边界的惩罚力和虚粒子结合方法

Fig. 3. The mixed method of virtual particles and punishment at boundary.

图3中, $2l$ 表示粒子支持域长度, d 为粒子到边界的距离值. 如果 d 值小于光滑长度, 则在边界外对称处增加类型1的镜像粒子. 当 d 值小于设定值 $d_{\rm s}$, 对该粒子施加惩罚力, 如(10)式所示:

$$\boldsymbol{f}^{\rm pb}=\begin{cases}[k_{\rm s}d-k_{\rm d}(\boldsymbol{u}\cdot\boldsymbol{n})]\boldsymbol{n}, & d\leqslant d_{\rm s},\\ 0, & d>d_{\rm s},\end{cases}\tag{10}$$

式中, $\boldsymbol{f}^{\rm pb}$ 为所施加的排斥力, $k_{\rm s}$ 是惩罚系数, $k_{\rm d}$ 表示阻尼系数, $\boldsymbol{u}$ 表示粒子的速度, $\boldsymbol{n}$ 表示距离粒子最近边界面的方向向量.

将该项代入到压力项中, (7)式修正为

$$\frac{{\rm D}\boldsymbol{u}_i}{{\rm D}t}=-g\sum_j V_j\nabla W(\boldsymbol{r}_i-\boldsymbol{r}_j,l)-g\nabla H+\boldsymbol{f}_i^{\rm pb}.\tag{11}$$

对边界处的冲击问题, 本文使用Monaghan型人工黏度[19]耗散冲击能量, 提高求解精度, 且能够阻止实粒子靠近时的非物理穿透. 在二维情况下, 其表达式为

$$\Pi_{ij}=\begin{cases}\dfrac{-\alpha_\Pi\bar{c}_{ij}\phi_{ij}+\beta_\Pi\phi_{ij}^2}{h_{ij}}, & \boldsymbol{v}_{ij}\cdot\boldsymbol{r}_{ij}<0,\\ 0, & \boldsymbol{v}_{ij}\cdot\boldsymbol{r}_{ij}\geqslant 0,\end{cases}\tag{12}$$

式中 $\phi_{ij}=\dfrac{l_{ij}\boldsymbol{v}_{ij}\cdot\boldsymbol{r}_{ij}}{\|\boldsymbol{r}_{ij}\|^2+\varphi^2}$, $\bar{c}_{ij}=\dfrac{1}{2}(c_i+c_j)$, $h_{ij}=\dfrac{1}{2}(h_i+h_j)$, $l_{ij}=\dfrac{1}{2}(l_i+l_j)$. 标准常数 $\alpha_\Pi=0.1$, $\beta_\Pi=0.01$, 因子 $\varphi=0.1l_{ij}$, c 和 $\boldsymbol{v}$ 分别表示粒子的声速以及速度矢量.

以人工黏度项修正动量方程, 即将(12)式引入到(11)式中, 得到

$$\frac{{\rm D}\boldsymbol{u}_i}{{\rm D}t}=-g\sum_j V_j\nabla W(\boldsymbol{r}_i-\boldsymbol{r}_j,l)-g\nabla H+\sum_j V_j\Pi_{ij}\nabla W(\boldsymbol{r}_i-\boldsymbol{r}_j,l)+\boldsymbol{f}_i^{\rm pb}.\tag{13}$$

3.3 流体表面重构与渲染

使用粒子法描述流体流动效果时, 流体表面的重构及渲染效果在很大程度上会影响仿真速度. 较大场景中, 流体粒子的疏密程度往往存在不一致现象. 为了使流体表面连续光滑, 需要对流体表面进行重构.

首先, 将每个粒子坐标映射到规则网格中, 并把粒子高度值作为流体表面的高程值. 由于粒子分布的不一致性, 在稀疏的地方对应的网格处存在缺失高程值, 流体表面可能出现不连续现象; 为使得流体表面更加光滑, 需要对网格进行加密处理. 考虑到计算复杂度, 本文采用双线性插值法来处理缺失高程值, 并加密表面网格, 得到流体表面高程数据并渲染绘制. 最后, 将粒子坐标和高程数据以高程图的形式存储, 方便使用OpenGL着色语言(OpenGL shader language, GLSL)渲染编程.

4 仿真实验

为验证本文所提方法的有效性, 实验的硬件平台为3.30 GHz Intel i3-3220 CPU, Intel集成显卡和4G内存, 编程环境为C++, OpenSceneGraph三维渲染引擎, GLSL着色语言进行仿真实验.

为了说明所提动态网格搜索算法的有效性, 增加与基本KD Tree搜索算法的对比. 采用随机点数据集进行对比测试, 结果如图4所示.

根据图4中曲线的上升趋势, 随着粒子数目的增大, 本文所提搜索算法能够快速地搜索邻域粒子, 尤其粒子数目较多时, 更加凸显本文动态网格搜索算法的优越性. 当粒子数目较小时, 本文搜索算法和PIB算法相差不大, 但随着粒子数的增大, 本文搜索算法能够较快速地搜索粒子. 说明本文算法适用于较大范围场景的邻近粒子搜索问题.

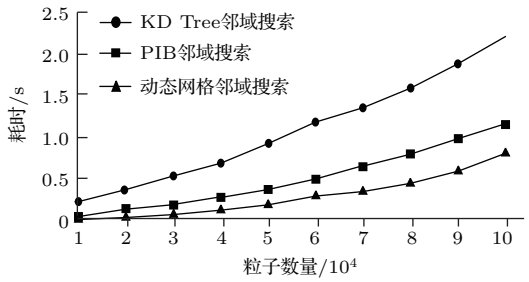

图 4 邻域搜索算法时间对比

Fig. 4. The comparison among neighborhood search algorithms.

为了说明虚粒子和惩罚力相结合的边界处理方法的有效性, 增加水柱坍塌标准算例, 时间步长取0.02, 粒子数为10^4. 不同时刻的结果如图5所示.

从图5(c)结果可以看出并无粒子穿越现象. 而观测仿真实验的整个过程, 发现几乎没有粒子穿越边界, 并经过统计计算过程中穿越底端边界的粒子情况, 在整个迭代过程中本文边界处理方法没有粒子穿透边界, 而虚粒子边界处理方法最大穿越粒子数为21, 图5(d)为某一时刻粒子穿透现象, 说明虚粒子和惩罚力相结合方法的有效性, 保证了算法的无穿透性.

与只使用虚粒子边界方法进行对比, 研究了统计粒子从初始状态运动到长方体底端的耗时情况. 为了统计两种方法的时间差值, 初始粒子设置为10^5. 通过多次统计计算, 只使用虚粒子边界处理方法耗时大约为21 s, 使用本文基于虚粒子和惩罚力相结合的边界处理方法耗时在13 s左右, 从而说明本文边界处理方法的有效性.

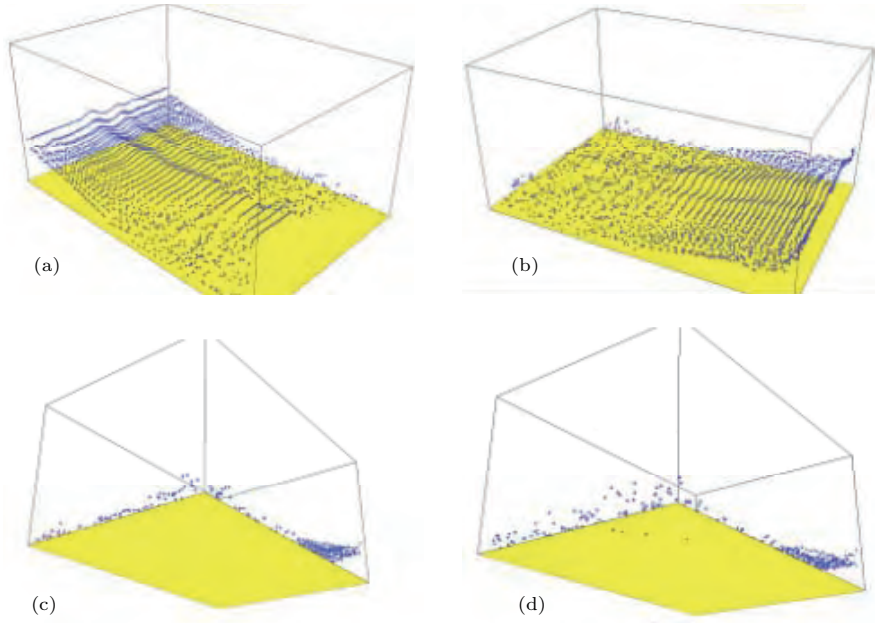

图 5 (网刊彩色) 标准算例不同时刻结果 (a) 0.5 s 俯视图; (b) 1.0 s 俯视图; (c) 1.0 s 仰视图; (d) 虚粒子仰视图

Fig. 5. (color online) The results of standard example at different moments: (a) Plan view at 0.5 s; (b) plan view at 1.0 s; (c) bottom view at 1.0 s; (d) bottom view of virtual particle.

实例一为山丘的斜坡地形(600 m × 200 m, 最高为31 m), 其斜率为0.05, 初始粒子分布在(120 m × 200 m)处, 粒子总数有6×10^3, 其高39 m, 间距2 m. 其中, 初始时刻光滑长度为2.5 m, 时间步长为0.025 s. 取0.5, 1.0, 1.5和2.0 s时刻的渲染结果, 得到渲染图见图6.

图6中, 流体从初始时刻缓慢流下, 经过两个山丘时, 发生碰撞, 粒子从山丘两侧绕过继续向下运动; 当粒子碰撞到大山丘时, 流体粒子仍然绕过, 且此流体粒子并未运动到大山丘后, 见图6(c); 最后流体遇到固壁边界导致流速下降, 如图6(d)所示. 此实例模拟出了流体过山丘的行为, 且实验中帧率能够保持在10 fps左右, 基本能够实现实时仿真.

为了有效地说明本文算法能够适应于复杂场景, 实例二采用某实际水库地形6.2 km × 2.0 km水面进行仿真, 其中, 粒子数为10^4, 取30, 60, 120和180 s四个时刻的渲染结果如图7所示.

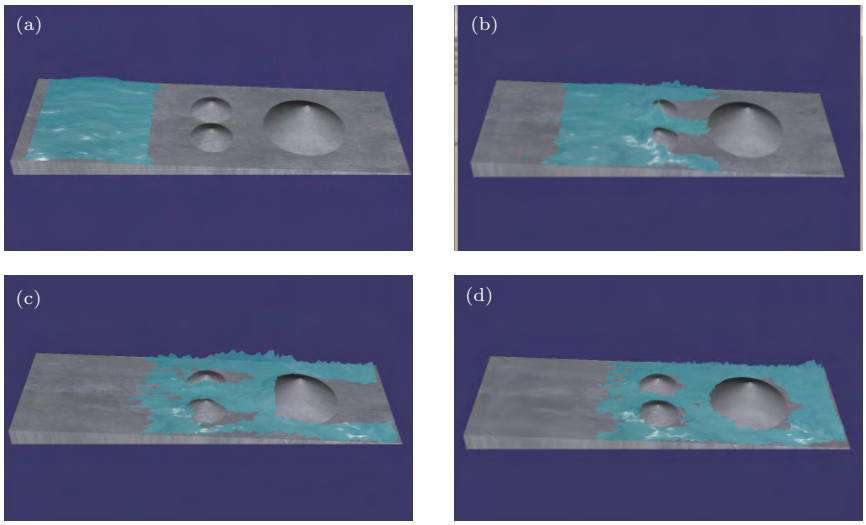

图 6 (网刊彩色) 实例一的四个时刻的流体表面渲染图 (a) 0.5 s 时刻; (b) 1.0 s 时刻; (c) 1.5 s 时刻; (d) 2.0 s 时刻

Fig. 6. (color online) Fluid surface rendering of example 1 at four moments: (a) 0.5 s; (b) 1.0 s; (c) 1.5 s; (d) 2.0 s.

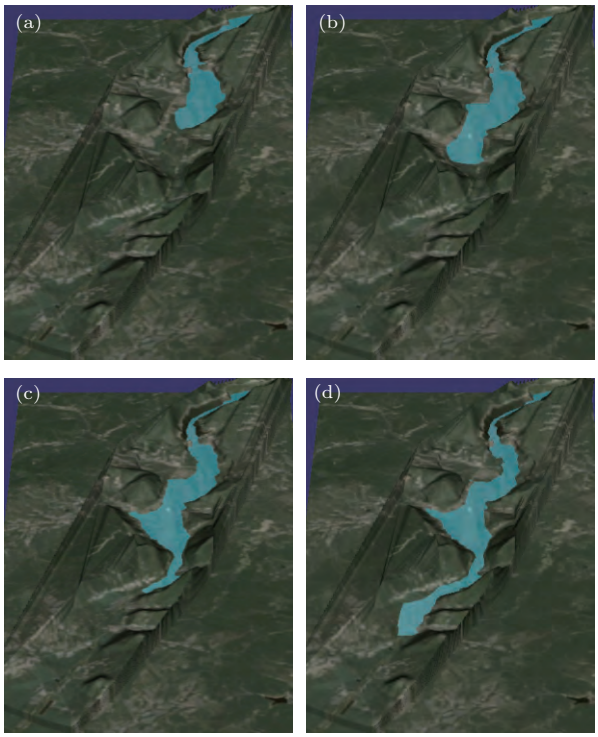

图 7 (网刊彩色) 实例二的四个时刻的流体表面渲染图 (a) 30 s 时刻; (b) 60 s 时刻; (c) 120 s 时刻; (d) 180 s 时刻

Fig. 7. (color online) Fluid surface rendering of example 2 at four moments: (a) 30 s; (b) 60 s; (c) 120 s; (d) 180 s.

从图 7 可以看出, 渲染结果能够实时地可视化流水的高度信息, 自然地展示整个水体的流动. 在实时性方面, 由于加载的地形数据量大, 且粒子数量较多, 使得在模拟环境下的动态效果并不如实例一流畅, 但也达到了边计算边演示的要求, 从而验证了本文算法能够适应于较大复杂场景的流体渲染. 但是, 从仿真实验中可以看出本文算法不能有效地展示出卷浪等细节效果, 这是由于单纯求解二维浅水方程本身不能模拟流体卷浪、破碎等细节行为, 并且采用 GLSL 语言提高渲染速度, 从而能够在普通配置的计算机平台上适应于较大平缓水体的模拟.

5 结 论

本文在传统的光滑粒子法求解浅水方程的基础上, 提出了一种基于二维光滑粒子法的流体仿真改进策略, 以降低大场景下的计算复杂度. 该策略中提出的一种基于动态网格的邻近粒子搜索方法, 可提高在可变光滑长度的情况下邻域粒子的搜索效率, 处理边界条件时使用了虚粒子和惩罚力相结合的方法, 同时添加人工黏滞项以提高仿真精度, 通过采用规则网格映射及插值的方法简化流体表面重构过程, 并利用 GLSL 提高渲染效率, 进一步降低了大场景下流体渲染的计算复杂度. 但是, 本文算法并未解决二维浅水方程不能模拟卷浪的复杂流体行为问题, 降低计算量的同时保持模拟流体细节将是今后研究重点.

参考文献

[1] Lucy L B 1977 *Astron. J.* **82** 1013

[2] Prakash M, Rothauge K, Cleary P W 2014 *Appl. Math. Model.* **38** 1534

[3] Kipfer P, Westermann R 2006 *Proceedings of Graphics Interface 2006* Quebec City, Canada, June 7–9, 2006 p41

[4] Ata R, Soulaïmani A 2005 *Int. J. Numer. Meth. Fl.* **47** 139

[5] de Leffe M, Le Touzé D, Alessandrini B 2010 *J. Hydraul. Res.* **48** 118

[6] Lee H, Han S 2010 *Visual Comput.* **26** 865

[7] Solenthaler B, Bucher P, Chentanez N, Müller M, Gross M 2011 *Proceedings of Workshop in Virtual Reality Interactions and Physical Simulations* Lyon, France, December 5–6, 2011 p39

[8] He X W, Liu N, Wang G P, Zhang F J, Li S, Shao S D, Wang H A 2012 *ACM T. Graphic.* **31** 439

[9] Cornelis J, Ihmsen M, Teschner M 2015 *Comput. Graph-UK* **52** 72

[10] He J, Chen X, Wang Z Y, Cao C, Yan H, Peng Q S 2010 *Visual Comput.* **26** 243

[11] Liu H, Qiang H F, Chen F Z, Han Y W, Fan S J 2015 *Acta Phys. Sin.* **64** 094701 (in Chinese) [刘虎, 强洪夫, 陈福振, 韩亚伟, 范树佳 2015 物理学报 **64** 094701]

[12] Han Y W, Qiang H F, Zhao J L, Gao W R 2013 *Acta Phys. Sin.* **62** 044702 (in Chinese) [韩亚伟, 强洪夫, 赵玖玲, 高巍然 2013 物理学报 **62** 044702]

[13] Hu D A, Long T, Xiao Y H, Han X, Gu Y T 2014 *Comput. Method. Appl. M.* **276** 266

[14] Xia X L, Liang Q H 2015 *Environ. Modell Softw.* **75** 28

[15] Goswami P, Schlegel P, Solenthaler B, Pajarola R 2010 *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* Madrid, Spain, July 2–4, 2010 p55

[16] Yang L P, Li S, Hao A, Qin H 2012 *Comput. Graph. Forum.* **31** 2037

[17] Rodriguez-Paz M, Bonet J 2005 *Comput. Struct.* **83** 1396

[18] Liu M B, Liu G R, Lam K Y 2003 *Comput. Appl. Math.* **155** 263

[19] Ihmsen M, Orthmann J, Solenthaler B, Kolb A, Teschner M 2014 *Proceedings of Eurographics 2014 State of the Art Reports* Strasbourg, April 7–11, 2014 p21

A fluid simulation method based on two-dimensional smoothed particle hydrodynamics*

Zhang Hai-Chao Zheng Dan-Chen Bian Mao-Song Han Min†

(*Faculty of Electronic Information and Electrical Engineering, Dalian University of Technology, Dalian 116023, China*)

(Received 27 April 2016; revised manuscript received 22 July 2016)

Abstract

Smoothed particle hydrodynamics (SPH) method is a kind of meshless method, which is used to solve the problem of fluid simulation without complex operations of the grids. To reduce the computational complexity, SPH method based on the two-dimensional shallow water equations is employed to establish a fluid model. In large scale scenes, taking into account the high computational complexity and the serious distortion problems, in this paper we introduce an improved two-dimensional SPH algorithm according to the shallow water equations. The proposed method with two-dimensional complexity is obtained by discretizing the two-dimensional shallow water equations with SPH, and the depth of water is introduced as the particle's property. The problem of increased amount of calculation cannot be well solved by using traditional neighboring particle search method based on tree structure. To improve the efficiency of search and simplify the search operation of neighborhood particles, in this paper we introduce a point-in-box search algorithm and put forward a neighboring particles searching method on the basis of dynamic grid. Besides, for large scale scenes, by considering that the virtual particle method provides slow computation speed with complex boundary condition, the type-one virtual particles are utilized to ensure that the borders can be calculated precisely by combining the punish force to prevent the phenomenon of particle boundary penetrating. Therefore, a method is further obtained to handle boundary condition efficiently by combining the virtual particles with punish force in this paper. In the process of rendering, the fluid surface is first determined by mapping and interpolating particles into regular grids without the complex reconstruction of surface in three-dimensional. Then, we utilize the bilinear interpolation method to deal with the problem of missing values, and the surface grids are further densified. With OpenSceneGraph three-dimensional render engine, OpenGL Shading Language is adopted to speed up the rendering speed, and in this way, the real-time fluid simulation of large scale scenes can be further achieved. With the basic KD tree searching method employed in the simulations, the comparative experiments are provided to verify effectiveness of the proposed searching method based on dynamic grid. Given the data set obtained from random points, experimental results demonstrate that the method in this paper can be used to solve the problem of neighboring particles searching in large scale scenes. To show the effectiveness of the proposed method on the basis of the virtual particles and the punish force, another experiment based on the collapsing of a water column is further provided. Besides, in this paper we conduct an experiment on a certain actual reservoir terrain to prove that the proposed method can be applied to fluid simulation of large scale scenes.

Keywords: smoothed particle hydrodynamics, shallow water equations, fluid simulation, three-dimensional simulation

PACS: 47.11.–j, 47.85.–g **DOI:** 10.7498/aps.65.244701

* Project supported by the National Natural Science Foundation of China (Grant No. 61374154).

† Corresponding author. E-mail: minhan@dlut.edu.cn