

使用隐式粒子进行流体模拟

高级游戏编程

丹·恩格森
乔金·基尔比
乔尔·艾克

2011 年 12 月 20 日



抽象的

本报告涵盖了使用混合方法实施流体模拟。
所使用的方法很大程度上基于 Robert Bridson [1] 所著的 Fluid Simulation for Computer Graphics 一书的内容,通常称为 PIC/FLIP 方法。在该方法中,使用粒子跟踪流体表面,并通过在变形速度场上强制零散度来保持其质量。我们将速度场存储在 [2] 中描述的交错网格上,这极大地有助于满足质量守恒标准。我们还通过在规则网格上评估 [3] 中引入的改进的 Blobbies 符号距离函数以及行进立方体的标准实现来重建实际的流体表面。

我们已经使用五个不同的模拟案例测试了我们的实现,并获得了出色的结果,所有这些都与基准数据一起显示在报告中。

1 简介

火、水和空气，曾经被认为是四个基本要素中的三个组成所有已知的物质，似乎有比曾经想象的更多的共同点。这三种都是不同类型的流体，或者在通常发生的力（例如重力）的影响下能够流动和变形的化合物。

流体的复杂行为一直吸引着人类，它只是在我们来的最近几十年里充分理解复杂的方程控制他们的动态行为。体液模拟非常有用，因为它可以帮助在建筑物和施工的设计中，而且因为它允许人们模拟

迟到一个永远不会发生的事件。例如，一个城市的洪水，一个大爆炸或其他任何适合的东西一部现代故事片。

2 背景

为了理解该方法的不同步骤是如何工作的，需要一些背景信息。本节包含对基本方程的解释、相关向量微积分的简短重复以及对臭名昭著的稳定流体方法的简短描述。

也说明了两者的区别欧拉和拉格朗日观点以及使用交错的好处网格。

2.1 控制方程

控制流体流动的两个方程是称为 Navier-Stokes 方程。这基本方程在方程中给出图 1 描述了速度场 V 随着时间的推移而发展。自然，时间这个速度场的导数必须是受外力影响。这是建模的通过等式 1 中的 F 项，通常表示重力、风或用户相互作用。 $\nu \nabla^2 V$ 项是 diffu

sion 项，它模拟的粘度流体。

粘度是一个常用的术语流体模拟，这仅仅意味着流体抵抗流动的趋势是观察它的厚度。例如，水很容易流动，因此具有低粘度。蜂蜜很厚，抗流动这就是它的粘度高的原因。这粘度直接控制由标量 ν 。应该注意的是该术语通常在以下情况下被丢弃模拟水。插值引入的数值耗散量错误通常是一个很好的替代品项，否则将由泊松方程。

$$\frac{\partial V}{\partial t} = F + \nu \nabla^2 V - V \cdot \nabla V - \frac{\nabla p}{\rho} \quad (1)$$

$$\nabla \cdot V = 0 \quad (2)$$

$V \cdot \nabla V$ 项是自平流模拟速度场如何的术语在自身内部流动。这看起来有点奇怪，但这很自然，因为速度场代表了运动的运动

流体的质量。最后一项是压力梯度 ∇p 在密度 ρ 上 $\frac{\nabla p}{\rho}$ ，的减法。当将整个方程与方程 2 中的约束相结合时，此项被替换

通过减去伪压力梯度。这是任何梯度的结果

标量是无卷曲的，因此在执行无分歧时删除 Helmholtz-Hodge 的速度场如[4]中所述的分解。

2.2 表示流体

有两种常用的表示流体的方法，它们完全基于不同的观点。一种方法是思考流体作为微小粒子或原子的集合，它们共同构成流体的体积。该表示是拉格朗日表示，并且已经产生了一个纯粹的家庭

基于粒子的流体模拟
平滑粒子流体动力学,或 SPH
是最常见的。这些方法可以
有益,因为它们的定义很直观
并且可以理解,但有一个很大的缺陷。
计算基于粒子密度总是足够的假设。情况并非如此,因为粒
子是

能够分散,允许区域
密度低。在这些地区,
计算变得不那么准确
根本没有足够的信息存储在
附近的粒子。
另一种表示流体的方法是
从欧拉的角度来看。为了这
表示,从观察流体
一个感兴趣的领域,分为
多个细胞。这形成了一个网格,
这就是为什么欧拉表示
也称为基于网格的表示。基于网格的表示不

与基于粒子的表示一样,在低密度区域遇到同样的问题。相
反,出现了其他问题。为了使流体表面

足够详细,网格必须包含
的众多细胞。这并不罕见
对于纯基于网格的流体模拟
使用数百万个细胞。这种效果是由于
根据奈奎斯特准则,该准则指出
信号的采样频率必须是
最高频率分量的两倍。换句话说,模拟
只保证捕获大于两个单元格大小的表面细节

宽度。这就是为什么细胞数
对于纯基于网格的方法,它必须非常大。

流体模拟也可以使用混合方法,它结合了基于粒子的方
法的表面捕获可能性,并将其与

辅助网格来执行精确的质量守恒。解释这种方法
方法部分详细介绍。

2.3 算子和向量场

为了了解速度场如何演化以及质量如何

守恒约束保持不变,则
需要了解以下三个运算符。等式 3 显示了梯度
操作员。它在标量场上运行,并且
产生一个向量场,其中每个偏导数作为其分量。什么时候

运算符应用于 a 的单元格
网格结构,应该解释为
标量场 q 的局部变化。在
换言之,财产的交换
由标量 q 描述。

$$\nabla q = \frac{\partial q}{\partial x}, \frac{\partial q}{\partial y}, \frac{\partial q}{\partial z} \quad (3)$$

等式 4 显示了散度算子。发散与梯度算子有关,涉及相
同的部分

导数,而不是测量每个
单独的导数,它测量总
局部变化。如下图所示,运行
在向量场上并产生一个标量场。
运营商应被解释为
任何财产的总增加或减少
被向量场 $\sim u$ 传输。

$$\nabla \cdot \sim u = \frac{\partial \sim u}{\partial x} + \frac{\partial \sim u}{\partial y} + \frac{\partial \sim u}{\partial z} \quad (4)$$

拉普拉斯算子是讨论过的三个算子中最复杂的一个

这里。它可以在等式 5 中看到,并且在标量场 q 上运行。尽
管
表达式涉及偏导数
看起来很复杂的二阶,实际上是散度算子

应用于标量场的梯度。在
换句话说,一种简洁的方式来说明两个运算符的顺序应用

之前定义的。当应用于
网格结构的单元格,它应该被解释为材料的总交换

所有相邻单元格之间。

$$\nabla^2 q = \frac{\partial^2 q}{\partial x^2} + \frac{\partial^2 q}{\partial y^2} + \frac{\partial^2 q}{\partial z^2} \quad (5)$$

2.4 交错网格

为了应用连续算子
对于离散网格结构,算子

首先需要区别对待。这需要介绍一种特殊类型的网格，即交错标记和单元格 (MAC) 网格，由 Foster 和 Metaxas 在 [2] 中介绍。与普通规则网格相比的主要区别在于速度场的分量是分离的和偏移的。如图 1 所示，速度场的采样点与单元面边界重合。请注意，所有标量

$$\begin{aligned}(\nabla q)i+1/2,j,k &= \frac{q_{i+1,j,k} - q_{i,j,k}}{s} \\(\nabla q)i,j+1/2,k &= \frac{q_{i,j+1,k} - q_{i,j,k}}{s} \\(\nabla q)i,j,k+1/2 &= \frac{q_{i,j,k+1} - q_{i,j,k}}{s}\end{aligned}\quad (6)$$

离散散度算子的定义如方程 7 所示。与离散梯度一样， s 表示像元大小。散度作用于速度场的分量对，因此其评估位置必须在单元的中心。

将由速度场传输，它们的样本点位于单元格的中心。这对于微分运算符的定义很重要。

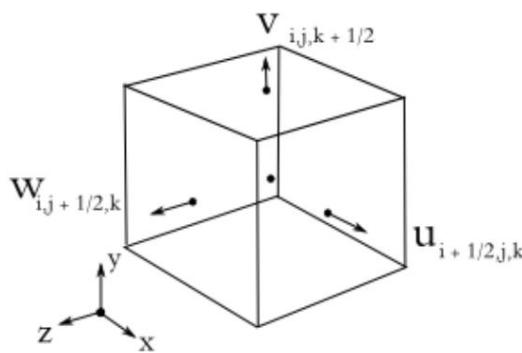


图 1：交错网格

使用这种结构的好处是，先前定义的算子的离散对应部分的评估变得大大简化了。该结构引入了奇怪的半指
数

正半指数是存储在单元格之间共享的面上的样本值

及其随后的单元格。

考虑方程 6 中所示的离散梯度，其中 s 是单元格的大
小。梯度应用于存储在单元中心的标量 q 。因此，评估位
置必须恰好位于两个相邻单元之间。请注意，此评估是
按组件进行的，并导致三个不同的标量，每个标量具有不
同的评估位置。

$$\begin{aligned}(\nabla \cdot \sim u)i,j,k &= \frac{\sim u_{i+1/2,j,k} - \sim u_{i-1/2,j,k}}{s} \\&+ \frac{\sim u_{i,j+1/2,k} - \sim u_{i,j-1/2,k}}{s} \\&+ \frac{\sim u_{i,j,k+1/2} - \sim u_{i,j,k-1/2}}{s}\end{aligned}\quad (7)$$

离散拉普拉斯算子是梯度和散度算子的组合。为了理解它是如何在交错网格上定义的，首先考虑梯度算子的
应用。如前所述，梯度的评估位置在两个相邻单元之间。
如果将散度算子应用于这六个标量，我们将得到等式 8。
由于两个算子有效应用，缩放

因子 $\frac{1}{2}$ 是自然的。

$$\begin{aligned}(\nabla^2 q)i,j,k &= \frac{q_{i+1,j,k} + q_{i-1,j,k}}{s^2} \\&+ \frac{q_{i,j+1,k} + q_{i,j-1,k}}{s^2} \\&+ \frac{q_{i,j,k+1} + q_{i,j,k-1}}{s^2} \\&- \frac{6q_{i,j,k}}{s^2}\end{aligned}\quad (8)$$

请注意，这两个离散
拉普拉斯算子和离散散度算子在单元中心进行评估。由
于泊松方程具有方程 9 中所示的形式，使用交错网格显
然有利于求解这些方程。特别是因为它允许使用仅单个
单元格大小的宽度来评估中心差异。

另请注意，评估位置与速度场分量的样本位置一致。

$$\nabla \cdot \sim u = \nabla^2 q \quad (9)$$

2.5 稳定流体

引入了稳定流体方法

乔斯·斯塔姆于 1999 年 [4]。这是第一个流体的无条件稳定方法

模拟并引入概念

半拉格朗日平流。马厩

流体方法计算近似值

Navier-Stokes 方程的解；

等式 1 和 2。

方程 1 的解由下式获得

依次计算贡献

从方程的每个部分，其中

每一步输入由输出给出

上一步的。

为了强制执行给定的约束

通过等式 2，对求解方法中得到的最终速度场进行投影

到其无散度部分。

整个过程如图 2 所示，其中 $\sim w_0$ 是来自

上一个时间步， $\sim w_4$ 是最后一个，

无散度，速度场。

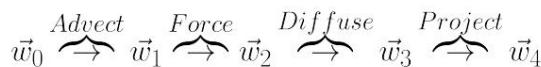


图 2: 稳定流体方法

3 方法

3.1 网格转换

为了将多边形网格对象表示为流体或固体对象以进行模拟，必须进行转换

从多边形到体素表示。

为此，读取了一个多边形对象

来自 .obj 文件并作为

对象被处理，它的顶点坐标

被转换成离散的网格坐标。对应的体素

网格坐标，其中的体素

顶点位于，然后设置为单元类型

对象是表示，例如流体或固体。

为了避免三角形的问题

显示大于网格分辨率，则以随机方式在 trian 上选择一些点

gle 表面并转换为网格坐标。

此方法创建一个体素化的外壳，

或表面，类似于原始多边形对象。然而对于模拟，表面是不够的，体素化的模型

必须填充才能创建卷。

体积是通过填写

表示中体素之间的间隙。

这是通过选择一个体素来实现的

标有正确的细胞类型和

然后依次沿着 x,y 和 z 方向步进，直到找到另一个体素

具有相同的细胞类型。所有体素

然后将两者之间设置为正确的

细胞型。

该方法是在所有多边形对象都是 Mani 折叠曲面的假设下工作的；不能有洞

网格，没有多边形的自相交和
网格内没有内壳。

3.2 混合粒子法

如前所述，在计算机图形学中模拟流体效应存在两种主要方法，即使用

拉格朗日粒子或欧拉网格。

这两种方法各有千秋

和弱点，拉格朗日粒子

平流部分非常好，但

压力和可压缩性约束有问题。幸运的是，

欧拉网格方法在

求解压力和不可压缩性约束，但由于半拉格朗日平流的插值误差，

该方法在平流部分存在问题。可以看出在哪里

拉格朗日方法有其困难，

欧拉方法非常好，并且

通过将这两种方法结合起来，让拉格朗日粒子处理平流部分，欧拉网格处理压力和不可压缩约束，更好地模拟水效应

可以实现。

存在许多不同的混合方法

比如粒子水平集，The Particle in

细胞 (PIC) 和流体隐式部分

cle (FLIP) 方法,其中后两者方法将在本文中进一步讨论
纸。粒子水平集方法是
Foster 和 Fedkiw 在 [5] 中介绍
PIC方法被介绍为
早在 1963 年由 Harlow 在 [6] 中
后来由 Brackbill 和 Ruppel 改进
[7] 1986 年采用 FLIP 方法。
FLIP方法,然后介绍到
朱和布里德森的不可压缩流动
在 2005 [3] 和今天是最先进的
流体模拟技术。

3.2.1 PIC/FLIP 方法

即使 PIC 和 FLIP 方法基本上是相同的方法,但与速度更新略有不同,流体特性差异很大。仅使用 PIC 的流体更粘稠比 FLIP 流体,这是由于速度的双重插值导致的数值耗散,但更多的是

一会儿。然而,FLIP 方法具有粘度小,因此非常适合水效果,但表面上存在不希望的可见噪音。通过早期结合PIC和FLIP方法,可以得到一种粘度很小、没有表面噪声的流体。

严重的数值耗散
PIC方法是由插值引起的。
粒子速度转移到
网格通过插值引入一些平滑,然后

平滑的速度值被用于
使用 Navier 求解新速度
斯托克斯方程,然后被内插回替换粒子的粒子
旧速度。如前所述,由于这个
过度的双重插值,一个PIC流体
会显得比 FLIP 更粘稠
体液。Brackbill 和 Ruppel 解决了这个问题
问题,而不是插值
新计算的粒子速度
从网格和替换速度,
他们插值了速度的变化
并将其添加到已经存在的粒子速度中。通过这
样做,只有平滑
从一个插值执行和

因此平滑不会累积为
在 PIC 方法中并进行 FLIP
方法几乎没有数值耗散。下面是 PIC/FLIP 流体模拟的逐步
解决方法

并更深入地了解不同
脚步。

1. 初始化网格和粒子位置
位置和速度。
2. 将粒子速度传输到交错网格。

3. FLIP:保存网格速度的副本 联系。

4. 计算和施加外力。
5. 执行狄利克雷边界条件。

6. 将所有体素分类为流体、固体或空气。

7. 使用预条件共轭计算拟压力梯度
梯度法。

8. FLIP:通过以下方式更新粒子速度

减去新的网格速度
从保存的网格速度,插值并将差异添加到
粒子速度。

9. PIC:通过用新网格速度插值和替换旧速度来更新粒子速度
粒子。

10. PIC/FLIP:通过插值 FLIP 更新粒子速度
和 PIC 速度来自 8. 和 9. 和
对于每个粒子称重 PIC 和
具有系数 α 的 FLIP 速度在 0 和 1 之间。

11. 更新粒子位置
使用新创建的 ODE 求解器
速度。

3.2.2 初始化粒子

每个被归类为流体的体素都会
里面有种子粒子。粒子被播种在一个抖动的网格中,就像

渲染器中的超级采样,其中八个
每个流体体素中的粒子由
将粒子随机放置在 $2 \times 2 \times 2$
子网格。更多的粒子可以
在每个体素中使用,但不一定给出明显更好的结果,但

相反,可能会减慢计算速度。

Bridson 在 [3] 中指出,八个粒子
每个流体单元足以产生良好的结果。

具有正常 n 的实心细胞,
见方程 11 其中 V 是速度
流体单元和 n 是正常的
相邻固体。

$$V \cdot n = 0 \quad (11)$$

如果一个流体单元格有一个固体相邻单元格,
检查速度分量并
如果任何速度分量指向相邻的实体单元,则速度

预计将与表面一起
固体细胞。由于仿真是在交错的 MAC 网格上进行的,并且
速度被分成单独的分量,速度的投影是

3.2.3 传递粒子到网格

由于粒子是随机放置的
在空间中进行压力和不可压缩性的计算

在网格上,某种插值是
需要将附近的粒子速度转移到网格。网格速度为

通过附近的粒子更新
位于立方体中的加权平均粒子速度的三线性插值

中心的网格单元宽度的两倍
是在网格速度分量。

3.2.4 计算外力

重力等外力
添加来自任何用户交互的力
通过简单的欧拉积分得到速度,如等式 10 所示,其中 V

代表速度, F 是外力
 Δt 是时间步长。

$$V_{\text{new}} = V_{\text{old}} + F \Delta t \quad (10)$$

一个简单的欧拉积分就足够了
因为这个任务是无条件稳定的
相当小的 Δt 。这是因为有
外力与速度场之间不存在反馈回路。

3.2.5 强制狄利克雷边界 (健康)状况

狄利克雷边界条件状态
不应该有流入或流出

3.2.6 体素分类

首先,每个非实体体素的类型
被清除并设置为空气。然后体素 con
包含一个或多个粒子被设置为
液体。实体体素设置为从
模拟的开始和
不要改变他们的类型。

3.2.7 保持质量

质量守恒的速度场是无散度速度场的同义词。
因此,保持流体的质量等于强制零散度
传输质量的速度场。
模拟步骤的这一部分是
计算量最大的一个,因为它涉及求解泊松方程。重新调用前
面提到的拉普拉斯算子以及它是如何定义为求和的

二阶的偏导数。需要强调的是,有
是每对两个之间的依赖关系
相邻的细胞和解决方案
泊松方程实际上意味着求解
大规模线性方程组。前
可以承担这个任务,一个推导
如何获得泊松方程是
必需的。

亥姆霍兹-霍奇分解
指出任何矢量场都可以表示为无散部分 Vdf 和

如等式所示的无卷曲部分 Vcf
12.

$$V = Vdf + Vcf \quad (12)$$

此外,向量微积分指出
根据定义,任何标量场的梯度都是无旋的。因此我们可以
替换
我们速度场的无旋部分
未知标量场的梯度,
 ∇q 。这如公式 13 所示。

$$V = Vdf + \nabla q \quad (13)$$

与任何等式一样,对等号两边都应用运算符确实
不改变平等。因此,允许
将散度算子应用于两者
双方。由于散度是线性算子, $\nabla \cdot (Vdf + \nabla q)$
等于 $\nabla \cdot Vdf + \nabla \cdot \nabla q$
我们得到方程 14。

$$\nabla \cdot V = \nabla \cdot Vdf + \nabla \cdot \nabla q \quad (14)$$

等式 14 可以简化为
分解步骤将 Vdf 定义为无散度。因此应用于 Vdf 的散度算子
必须相等

为零。此外,由于我们定义了拉普拉斯
运算符作为随之而来的应用
一个梯度算子和一个散度算子,我们得到方程 15。

$$\nabla \cdot V = \nabla^2 \quad (15)$$

方程 15 是泊松方程
其中 q 是未知的标量场
解方程。通过重新排列等式 13,我们得到等式 16

清楚地表明,我们可以通过找到未知的标量场 q 并减去其
梯度 ∇q 来强制速度场的质量守恒。

$$Vdf = V - \nabla q \quad (16)$$

等式 15 是需要的等式
为了找到标量场而被解决
 q ,通常称为伪压力。

等式的左边可以是
简单地通过评估速度场的发散度来计算。右手边

边包含未知的标量场和
拉普拉斯算子。的定义
算子如公式 8 所示,并且
指出未知数的相邻值
计算中应使用标量字段。当然,这是不可能的,因为

标量场是未知的。然而,系数是完全已知的,因为它们仅取
决于小区的本地配置。

也就是说,如果有直接相邻的固体细胞或液体可以自由交
换
材料通过每个细胞表面。

拉普拉斯算子的定义
边界变得有些不同
作为狄利克雷边界条件的流体单元表明不应该有流动

通过坚实的边界。结果在
拉普拉斯算子是系数
对于固体单元(当考虑相邻的流体单元时)设置为零,并且

中心系数增加一。
随着系数的计算和
存储,得到一个庞大的线性方程组。该系统在
形式 $Ax = b$ 其中 A 是系数矩阵, b 是每个单元的散度。

该系统中的方程数为
与细胞数量直接相关
模拟网格。如果有 $w \cdot h \cdot d$
细胞,系数矩阵的大小是
 $(w \cdot h \cdot d)^2$.

由于最多可以有六个其他单元格
毗邻每一个细胞,这个系统是非常
稀疏并且可以通过利用这一点的迭代方法有效地解决

财产。内存需求
稀疏系数矩阵为 $w \cdot h \cdot d \cdot 4$ 。这
可以与虚拟大小进行比较
前面提到的矩阵。

为了求解泊松方程,
Bridson [1] 推荐了预条件共轭梯度法

修饰不完整的前置条件
Cholesky 分解类型。这种方法
收敛速度快,可操作
在稀疏系统上。一个实现

PCG 方法的介绍是以 Bridson 书中的伪代码为指导进行的。

尽管应该注意,可以使用任何不需要显式表示 A 矩阵的迭代方法,例如 Jacobi 迭代技术。

求解泊松方程时,从速度场中减去所得标量场的梯度,保证质量守恒。

3.2.8 更新粒子速度

粒子需要使用存储在 MAC 网格上的新计算速度进行更新。这是通过将八个相邻网格速度的速度三线性插值到粒子来完成的,对于 PIC,使用新速度更新速度,或者对于 FLIP,插值速度变化并将其添加到现有粒子速度中。PIC 和 FLIP 的线性组合可以

用于获得低粘度、水状、无表面噪音的流体。这可以是

在等式 17 中可以看到,其中 $\sim u_{\text{new}}$ 是新的粒子速度, α 是 PIC 混合因子。该因子决定了应该有多少 PIC、粘度或数值耗散,其中 1.0 是纯 PIC,0.0 是纯 FLIP。`lerp()` 函数表示三线性插值函数。

$$\begin{aligned} \sim u_p^{\text{新}} &= \alpha \cdot \text{lerp}(\sim u_{\text{new}}^{\text{网格}}, \sim u_p) \\ &+ (1 - \alpha)[\sim u_{\text{old}} + \text{lerp}(\Delta \sim u_{\text{grid}}, \sim u_p)] \end{aligned} \quad (17)$$

3.2.9 节能灯条件

CFL 条件规定粒子在每个子步骤中应始终移动少于一个网格单元。它是通过获取单元格宽度并将其除以网格中的最大速度来获得稳定的 t 来完成的。

然后将 `stabledt` 与实际时间步 `dt` 进行比较,如果大于 `dt`,则将 `stabledt` 设置为 `dt`。然后粒子在六个子步骤中平流,直到它

到达 `DT`,通常有大约五个子步骤。

3.2.10 更新粒子位置

粒子位置使用 Runge Kutta 2 ODE 求解器平流,与简单的 Euler ODE 求解器相比,它是稳定的。但是,由于 RK2 求解器中的错误,粒子偶尔会穿透固体边界。这可能导致

已经穿透固体体素而被卡住的粒子。为了解决这个问题,已经穿透固体体素的粒子沿法线方向向后移动到固体体素外部单元宽度的一半。

3.3 中间存储

流体求解器分为两种不同的耳鼻喉零件;一个模拟程序负责所有关于流体运动的计算,一个可视化程序将模拟数据作为输入并从数据中生成 .obj 文件、网格。可视化程序还具有直接可视化粒子或表面的能力。

随着模拟程序的进行,它将当前时间步长的粒子位置顺序写入二进制文件。只写粒子位置的原因在于粒子的数量非常大,并且这些位置是重建表面所必需的,因为它们决定了流体的位置。

粒子的位置由三个浮点数表示,即笛卡尔坐标 x、y 和 z。由于每个浮点数占用 4 个字节的内存,一个粒子的位置占用 12 个字节。

附录 A 中的表 2 显示了几种不同的模拟数据的大小

粒子数量,每个文件保存 250 帧。

3.4 表面重建

执行模拟并将其存储在磁盘上后,必须重建表面才能在标准渲染管道中进行渲染。由于大多数渲染管道都经过优化以处理三角形,因此这通常简化为对模拟数据进行三角剖分。这不是一件容易的事,因为这个问题没有简单的解决方案。表面重建通常在两个不同的通道中实现。首先,使用函数从常规网格上的基础数据初始化标量字段。此函数的唯一要求是将数据集映射到实值标量。请注意,实值意味着该函数可以采用正值和负值,而标量表示单个值。在第二遍中,选择计算的标量场的某个值 (iso-level) 进行可视化。重建的表面

在函数假设的地方创建
价值。

3.4.1 有符号距离函数

在处理点云时, `map_ping` 函数通常测量到最近点的距离或到一些最近点的平均距离。 Zhu 和 Bridson 介绍了一种方法, 称为

[3] 中改进的 Blobbies。该方法针对每个网格角计算其附近点的平均位置和半径, 判断网格角是否位于该平均位置的平均半径内。这为靠近的角落提供了负距离

对于那些附近点很少的人来说, 到很多点和正距离。实际上, 这形成了一个有符号距离函数, 其定义可以在

等式 18。但是, 当固定搜索半径内不存在点时, 函数未定义, 这可能会导致问题。

该函数在每个网格角定义, 这就是为什么它是有意义的

对每个网格角进行评估。然而, 这是非常低效的, 因为这种实现的时间复杂度是 $O(m \cdot n)$, 其中 n 是点的数量, m 是网格角的数量。为了优化, 我们将搜索半径 R 设置为网格间距的三倍, 将粒子半径 r_i 设置为网格间距的一半并遍历粒子。这使得每个点的影响很容易确定, 因为它只会影响位于其附近的角落。因此, 时间复杂度从 $O(m \cdot n)$ 降低到 $O(n)$ 。

$$\Phi(\sim xg) = \text{len}(\sim xg - X w_i \sim xi) - X w_i r_i \quad (18)$$

使用形状良好的核函数对点进行加权, 如等式 19 所示, 如等式 20 所示。核函数随着点与网格角之间的距离的增加而衰减, 在等于搜索的距离处恰好为零半径, R 。每个权重都针对来自附近点的所有贡献进行归一化。因此, 通过累积贡献及其内核值, 然后是最终的标准化步骤, 最有效地实现此加权功能。

$$w_i = \frac{k1(\text{len}(\sim xg - \sim xi)/R)}{\sum_j k1(\text{len}(\sim xg - \sim xj)/R)} \quad (19)$$

$$k1(s) = \max(0, (1 - s^2)^{\frac{3}{2}}) \quad (20)$$

由于内核对其输入变量进行平方且输入始终是向量的范数, 因此我们用优化替换此内核, 避免平方根。改进的内核如等式 22 所示, 修改后的权重函数如等式 21 所示。请注意, $\sim xg$ 和 $\sim xi$ 之间的距离计算为平方距离或点积, 它不需要平方根运算。

$$w_i = \frac{k2((\sim xg - \sim xi) \cdot (\sim xg - \sim xi)/R^2)}{\sum_j k2((\sim xg - \sim xj) \cdot (\sim xg - \sim xj)/R^2)} \quad (21)$$

$$k2(s) = \max(0, (1 - s)^3) \quad (22)$$

3.4.2 行进方块

介绍了一种简单而优雅的曲面重建算法

[8] 中的 Lorensen 和 Cline。新的方法名为Marching
立方体并从此成为等值面生成的行业标准。

该方法基于将实值函数评估为离散标量场

在规则网格上定义。它应该是
注意到这个网格不同于
在模拟过程中使用的一个。他们
不需要具有相同的分辨率,但暗示该网格应

分辨率由模拟中跟踪粒子的密度决定。如果每个流体播种
八个粒子

细胞,表面的最佳分辨率
重建网格将是根据奈奎斯特准则的模拟网格。

该方法的第一步是评估网格中每个单元角的有符号距离
函数。由于我们的

没有在模拟域的任何地方都定义有符号距离函数,必须特
别考虑

在哪里评估函数。函数未定义的单元格角是

标记为外部以执行以下步骤
的算法。

继第一步之后,给定一个
某些等值,将单元角分类为
在表面内部或外部。
因为我们使用的是有符号距离函数,它代表有符号距离

从表面上看,感兴趣的等值
为零。确定哪些单元格角
因此,在外部对应于测试值是否大于零

反之亦然。

取决于角落如何
分类为内部或外部,为每个单元形成一个配置。这个配置需
要枚举。自从

一个细胞有八个角,有

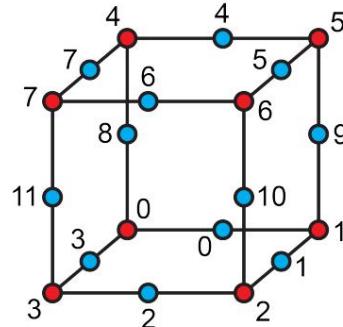


图 3: 角和边缘编号

可以正好是 256 种不同的组合,
每个形成一个独特的案例。该配置非常适合存储在可用作
索引的单个无符号字节中

放入案例表中,详细说明形成表面的三角形。符号

如图3所示用于编码一个
对于标记为内部的每个角落。

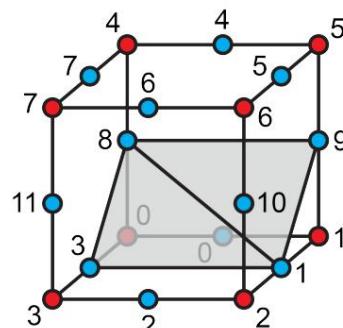


图 4: 一个独特的案例

对于只有角 0 和 1 的单元格
在里面,对应的索引将是
00000011 最低有效位是
位于右侧。这等于 3
因此案例 3 将从
三角案例表。从标准
Marching Cubes 的实现,这个
对应放置两个三角形。一
在顶点 3、1 和 8 之间,一个在顶点 8、1 和 9 之
间。这种情况在图 4 中可见,它清楚地表明

构造封装的表面
角落 0 和 1. 的实际位置

顶点是通过对位于拐角处的标量值进行插值来确定的,如等式 23 所示,其中 α 为

顶点 i 和之间的混合因子
顶点 j 。

$$\alpha = \frac{\phi_j - \phi_i}{\phi_j - \phi_i} \quad (23)$$

原始的 Marching Cubes 算法还详细说明了顶点法线如何从标量中的值计算场地。梯度被简单地评估在每个网格角落并插值到顶点位置。不过,这也是有问题,因为函数可以在表面之外未定义。相反,表面法线在每个顶点处求和并最终归一化。作为顶点

法线是从拓扑中计算出来的表面而不是标量场的梯度,得到的质量

顶点法线的数量较低。他们俩计算方法还不完全相同都产生足够的顶点法线质量为我们的目的。
需要注意的是,原文 Marching Cubes 算法可以创建生成的表面会有洞的模糊情况。中使用的方法

[9] 详细说明了如何找到这些案例并正确纠正。

3.5 导出网格

在表面重建过程中,
顶点列表、顶点法线列表和
创建三角形列表。将待添加到顶点列表的顶点与现有顶点进
行比较。如果他们

位于一个大的阈值内
(10—5),旧顶点被重用,因此三角形可以共享顶点。

这样可以减少顶点的数量
模型,也可以用来禁用
创建不需要的微小拉伸
三角形,只需提高门槛
价值。应该注意的是,我们的实现使用了蛮力方法

时间复杂度为 $O(n)$
每个新顶点都与之进行比较

已经存在的。一个有效的实施应该利用数据

降低时间复杂度的结构
这个操作的。
从这些列表中,表面必须是以高度可访问和标准化的格式存储。我们从 Alias Wavefront 中选择了 mat (.obj) 的目标文件,因为它
是基于文本的,易于实现。这
格式也符合索引
人脸集数据结构,包括
之前描述的列表。

4 个结果

我们的实现使用五个测试不同的测试用例。测试用例一和
其中两个旨在展示如何改变模拟流体的粘度

只需更改 PIC 影响因子即可。所有模拟案例如表 1 所示,其
中附加的模拟数据

如附录 A 中的 2 所示。

案例描述	
1	破坝 (FLIP)
2	溃坝 (PIC)
3	与龙决裂
4	下降的立方体
5	飞龙

表一 : 仿真案例列表

以下附录 B 中显示的所有图像,
使用我们实施的模拟数据在 Autodesk 3DSMAX 中渲染。
对于每种情况,一组 250 帧

被模拟、重建和渲染。

5 讨论

我们发现使用混合
与纯欧拉相反的方法
流体模拟方法允许
捕捉高度详细的行为
流体。我们的方法不受限制
网格分辨率,因为流体的基本表示是集合

的粒子。这使我们能够执行

以相对较低的网格分辨率进行模拟,同时仍然产生详细的表面,节省时间和内存。我们的方法还可以捕捉飞溅中的细节比欧拉方法更准确。

但是我们的有一个缺点方法;从表面重建一组粒子,例如点云,是比较一个级别更难执行集合和生成的复杂度曲面会创建更多的多边形。表面的复杂性可以通过设置限制来稍微简化关于必须的粒子数存在于体素中以生成表面。然而,这将减少细节飞溅和权衡等功能必须介于细节和复杂之间。

我们的实现能够执行即使在相对较短的时间内模拟随着粒子数量的增加。主要是由于巧妙的数据结构和注意缓存内存和缓存工作。如表 2 所示,生产所需的大部分时间一帧用于渲染和表面重建。我们注意到作为我们模拟数据的复杂性增加,例如当发生大的飞溅时,重建阶段需要更长的时间并且产生更复杂的几何图形更多的多边形。这是我们计划解决的问题,下一节将提到可能的解决方案化。

6 改进和未来工作

我们实现的一个主要改进领域是表面生成。一种第一步是改变进入的方式哪个粒子可能会影响网格点。在当前的实现中,粒子影响 a 内的网格点距其位置一定半径的球体。这种影响范围可以改变

成更椭圆的形状,其方向和尺寸基于局部粒子密度。这将产生的最关键的改进是能够在表面上定义更清晰的特征。作为

它站在今天,非常清晰的特征由于球形而被平滑粒子影响的形状。这是在 [10] 中有更详细的描述。我们还想介绍对生成的后处理操作表面,主要是为了减少不必要的多边形,并通过这样做减少渲染时间。其中我们想要实施的操作是;网格平滑以减少可能由低电平引入的类似噪声的行为局部粒子密度。抽取只是为了减少多边形的数量。我们相信网

基于多边形面积的抽取和曲率将非常有效地减少多边形数量,同时保留网格中的精细细节。

至于我们实现的模拟部分,我们想介绍一下在体素处播种泡沫粒子卷曲的幅度高于某个阈值,以强调流体的狂野行为。我们还想在我们的模拟器中引入双向耦合,这意味着流体可以与固体物体或其他流体相互作用,在更复杂的方式,例如浮动固体物体或油水混合物。我们还想改进网格结构,以便我们可以表示固体具有正确斜率的物体大大地。使用现有结构做由于体素表示,这将导致类似楼梯的行为。

Bridson 提出了一种实现方法这在 [1] 中。最后,我们考虑了实现自适应网格结构。在这个结构,网格只会被定义在靠近流体实际位置的位置,它将跟随流体移动。我们认为这将减少

仿真时间显着。

参考

- [1] R. Bridson,流体模拟
电脑图像。 AK Peters/CRC 出版社,2008
年 9 月。
- [2] N. Foster 和 D. Metaxas, “液体的真实动
画” ,在图形模型和图像处理中,

第 23-30 页,1995 年。
- [3] Y. Zhu 和 R. Bridson, “动画
沙子作为流体, ”在 ACM SIGGRAPH 中
2005 年论文,SIGGRAPH '05, (新
美国纽约州约克市) ,第 965-972 页,ACM,
2005 年。
- [4] J. Stam, “稳定流体” ,第 26 届计算机图形
和交互技术年度会议论文集,SIGGRAPH
'99,

(美国纽约州纽约市) ,第 121-128 页,
ACM 出版社/Addison-Wesley 出版公司,
1999 年。
- [5] N. Foster 和 R. Fedkiw, “液体的实用动
画” ,第 28 届计算机图形和交互技术年度会
议记录,SIGGRAPH '01,

(美国纽约州纽约市) ,第 23-30 页,
ACM,2001 年。
- [6] M. Evans 和 F. Harlow,
用于流体动力学计算的单元内粒子方法。
LA-2139,1957 年。
- [7] J. Brackbill 和 H. Ruppel, “翻转：
一种自适应分区的方法，
流体的细胞内粒子计算
二维流动, ”杂志
计算物理学,第一卷。 65,
第 314-343 页,1986 年 8 月。
- [8] 我们洛伦森和他克莱恩，
“行进立方体:高分辨率
3D 表面构造算法， ”
SIGGRAPH 计算。 图形。 ,
卷。 21,第 163-169 页,1987 年 8 月。
- [9] TS Newman 和 H. Yi, “A
行进立方体的调查

算法，”计算机与图形，
卷。 30,第 854-879 页,2006 年 10 月。

[10] J. Yu 和 G. Turk, “重建
基于粒子的流体表面
使用各向异性内核，”在
2010 ACM SIG GRAPH/Eurographics 研
讨会论文集
关于计算机动画,SCA '10,
(Aire-la-Ville,瑞士,瑞士) ,第 217-225
页,Eurographics
协会,2010。

基准

	1	2	3	4	5
模拟网格 128x64x64	128x64x64	128x64x64	128x64x64	128x64x128	128x64x64
粒子密度 2	32	32	32	3	4 ³
颗粒 438 976 5% 8 分钟	917 600	917 600	595 200		402 880
PIC 影响	5% 34	45% 18	5% 25		5%
Simulation	分钟	分钟	分钟		7 分钟
Reconstruction	82 分钟	64 分钟	71 分钟	155 分钟	36 分钟
Rendering 表面数据	483 min 396 min 模拟数据	2.56 GB 2.56GB	310 分钟 926 分钟 366 分钟		
			1.66 GB 1.22 GB 1.12 GB		
	820 MB 709 MB 829 MB	1.34 GB 572 MB			

表 2:所有模拟案例的基准数据

B 图像



图 5:溃坝 (FLIP)



图 6:溃坝 (PIC)



图 7:与龙决裂



图8:下落的立方体

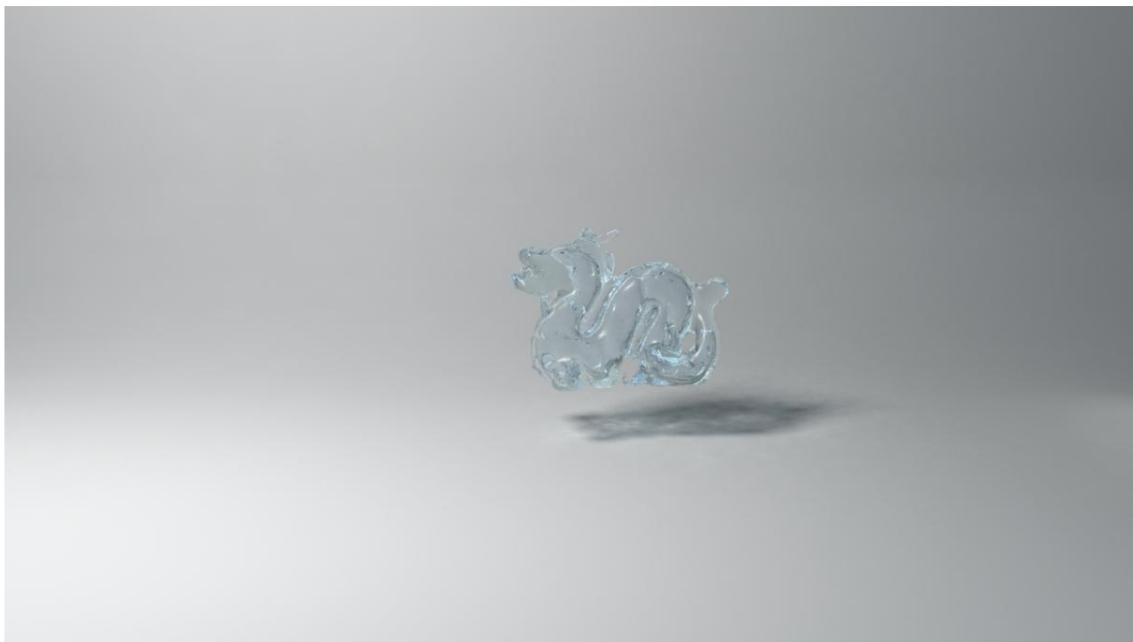


图9:飞龙