doi: 10.7690/bgzdh.2019.07.017

基于 Verlet 积分的高速布料模拟方法

刘　渊
（西南科技大学城市学院工程技术学院，四川 绵阳 621000）

摘要：为提高游戏和互动应用的可信度和运行速度，以 Verlet 积分为核心，提出一种迭代算法。分别阐述基于惩罚和改进的方案，采用张弛方法解决一致性约束问题，给出布料模拟系统软件实现流程和效果图，并进行实验验证。结果表明：该方法可随时停止，是一个非常有用的精度换时间的折衷方案，在游戏布料模拟等对精度要求相对较低的应用环境中，其效率高于现有的其他方法。

关键词：布料模拟；碰撞检测；Verlet 积分
中图分类号：TP391　　**文献标志码：**A

A High Speed Cloth Simulation Method Using Verlet Integration

Liu Yuan
(*School of Engineering Technology*, *City College of Southwest University of Science & Technology*, *Mianyang* 621000, *China*)

Abstract: For improving reliability and operating speed of game and mutual application, taking Verlet integration as core, present an iterated algorithm. Introduce plans for penalty and improvement, solving consistency problems with relaxation methods, cloth simulation software process to realize process and effect picture, pictures showing the procedure and test result. The method is verified by test. Result shows that the algorithm can end in any time, it can effectively exchange precision for time. This method is much faster than other methods in circumstances like simulating cloth in games which does not need high precision.

Keywords: cloth simulation; collision detection; Verlet integration

0　引言

布料模拟是游戏及互动应用中的常见问题，蓝磊等研究的重点是模拟的稳定性与真实性[1-3]。笔者则另辟蹊径，提出了一种用精度换时间的折衷方案。总的来说，该方法的成功是由于改进与整合了以下技术：1) Verlet 积分方法；2) 用投影的方式来处理碰撞和穿透；3) 利用张弛方法来构建简单的约束求解器；4) 采用平方根估计方法提升速度；5) 以带有约束的粒子群方式来为布料建模；6) 采用具有计算穿透深度能力的优化碰撞引擎。

1　Verlet 积分方法

1.1　传统的欧拉积分方法

仿真的核心内容是粒子系统。典型的，在粒子系统运行中，每个粒子含有 2 个主要变量：位置矢量 $\boldsymbol{X}$ 与速度矢量 $\boldsymbol{V}$。在时间步循环中，新的位置矢量 $\boldsymbol{X}$ 和速度矢量 $\boldsymbol{V}$ 一般遵循以下公式计算：

$$\boldsymbol{X}' = \boldsymbol{X} + \boldsymbol{V}\Delta t,$$
$$\boldsymbol{V}' = \boldsymbol{V} + \boldsymbol{a}\Delta t。 \tag{1}$$

式中：Δt 是时间步长；$\boldsymbol{a}$ 是利用牛顿第二定律 $\boldsymbol{f}=m\boldsymbol{a}$ 计算的加速度（$\boldsymbol{f}$ 是施加在粒子上力的总和）。这就是简单的欧拉积分[4]。

1.2　Verlet 积分方法

笔者选择另一种积分方法：不再存储每个粒子的位置和速度，而是存储其当前位置 $\boldsymbol{x}$ 和前一位置 $\boldsymbol{x}^*$。保持一个恒定的时间步长，改进规则（或称积分步骤）如下：

$$X' = 2X - X^* + a\Delta t^2,$$
$$X^* = X。 \tag{2}$$

上式被称为 Verlet 积分方法[5]，并常常在模拟分子动力学时使用。由于速度是隐含给出，使得速度和位置更不容易超出同步范围；因此，该方案相当稳定（例如，在水中产生水波的演示动画也使用了类似的做法）。该方案之所以有效是由于 $2x-x^*=x+(x-x^*)$ 与 $x-x^*$ 是当前速度的一种估计（事实上是上一时间步中移动的距离），这并不总是非常精确（能量可能会在系统中损失，也就是说消散了），但是它快速而稳定。通过将更新规则改变为 $x' = 1.99x - 0.99x^* + a^* \mathrm{D}t^2$，可以将少量的阻力引入

收稿日期：2019-03-14；修回日期：2019-04-25
作者简介：刘　渊（1981—），男，四川人，硕士，讲师，从事计算机图形学研究。E-mail: itit_ly@163.com。

系统。

在每个步骤的最后，每个粒子的当前位置 ***x*** 被保存在相应的变量 ***x**** 中。当处理大量粒子时，简单的交换队列指针就能有效地优化算法。

结果代码如下(Vector3 类应该包含适当的成员函数和用于处理矢量的重载过的操作符)：

```
//用于物理模拟的代码
class ParticleSystem {
  Vector3 m_x[NUM_PARTICLES]; //当前位置
  Vector3 m_oldx[NUM_PARTICLES]; //前一位置
  Vector3 m_a[NUM_PARTICLES]; //受力累加器
  Vector3 m_vGravity; //重力
  float m_fTimeStep;
public:
  void  TimeStep();
private:
  void  Verlet();
  void  SatisfyConstraints();
  void AccumulateForces();
//(构造函数, 初始化)
};
// Verlet 积分步骤
void ParticleSystem::Verlet() {
  for(int i=0; i<NUM_PARTICLES; i++) {
    Vector3& x = m_x[i];
    Vector3 temp = x;
    Vector3& oldx = m_oldx[i];
    Vector3& a = m_a[i];
    x += x-oldx+a*fTimeStep*fTimeStep;
    oldx = temp;
  }
}
//这个函数用于累积每个粒子的受力情况
void ParticleSystem::AccumulateForces()
{
//所有粒子都受到重力影响
  for(int i=0; i<NUM_PARTICLES; i++)
m_a[i] = m_vGravity;
}
//需满足约束条件
void ParticleSystem::SatisfyConstraints() {
//暂时不用管这个函数
}
void ParticleSystem::TimeStep() {
  AccumulateForces();
  Verlet();
  SatisfyConstraints();
}
```

当开始使用约束条件并开始处理布料模型时，该方案的优势会变得明显。相比其他方法，上述积分方法切实提高了稳定性并减少了计算量。

2　碰撞和接触处理

2.1　基于惩罚的方案

基于惩罚的方案通过在穿透点中插入弹力来处理接触问题[6]。虽然这种方案非常容易实现，但也有严重的不足。比如：这种方案很难选择合适的弹簧刚度，使得一方面物体不会穿透过深；另一方面，构成的系统不会变得不稳定。在其他模拟物理方案中，首先把时间倒回到碰撞发生的时间点，从时间点开始处理碰撞，然后重新开始模拟过程。存在大量碰撞时，这样的代码可能运行非常缓慢，从实时模拟的角度看，没有太大的实用价值。

2.2　改进的方案

笔者采用另一种方案。穿透点被简单地投影到碰撞体外部。这种投影意味着要把穿透点从碰撞体中分离出来，并尽可能保证它们不会距离碰撞体太远。一般来说，这意味着将穿透点沿着碰撞面垂直地向外平移。

例如：假设所有的粒子都需局限在一个以矢量(1 000,1 000,1 000)为对角线的立方体内部，同时假设粒子复原系数为零，即粒子不会在碰撞时弹离表面。为保证所有粒子都落在有效的区间内部，相应的程序代码如下：

```
//在立方体空间中实现粒子群模拟
void ParticleSystem::SatisfyConstraints() {
  for(int i=0; i<NUM_PARTICLES; i++) { //对所有粒子有效
    Vector3& x = m_x[i];
    x = vmin(vmax(x, Vector3(0,0,0)),
    Vector3(1000,1000,1000));
  }
}
```

(vmax 操作取分量最大的矢量，而 vmin 操作取分量最小的矢量)

这样保证了所有粒子都位于立方体内部，并且处理了碰撞与静止接触。Verlet 积分方法的优势在于速度的相应改变是自动处理的。接下来，在对 TimeStep()函数的调用中，速度被自动调整为不包

含任何表面法线方向的分量(对应的复原系数为零)，如图 1 所示。

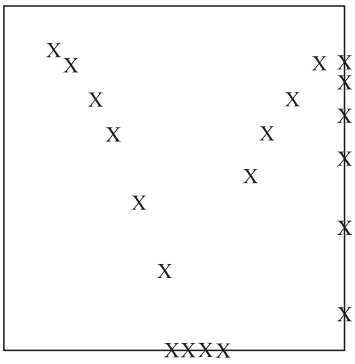

图 1　2 个粒子的 10 个时间步

3　用张弛方法解决一致性约束问题

一般的布料模型是由相互连接的弹簧约束和粒子构成的简单系统[7]。然而，用微分方程来求解这一系统有时困难重重。基于惩罚的系统有时也面临同样的难题：如果使用简单的积分策略，强力的弹簧约束会使一系列的方程式变得趋于刚性，导致不稳定的发生，或者至少会降低系统的运行效能。相反，较弱的弹簧约束会使布料看起来延展性过大[8]。

如果把弹簧约束的刚度设为无穷大，会产生以下结果：系统变得可以用一种稳定、简单而快速的方式求解。如图 1 所示的例子中提到的立方体，可以被看作随时都必须被满足粒子位置的单边约束条件的集合(立方体中每组相对的面对应一个约束条件)：

$$x_i \geqslant 0 \text{and} x_i \leqslant 1\ 000 \text{ for } i = 1,2,3。 \tag{3}$$

该例子中，用将粒子投影到立方体表面的方式调整了穿透位置，从而满足了约束条件(即所有的粒子都不会位于立方体外部)。为了满足式(3)，可以使用以下伪码：

```
//满足式(3)的伪码
for i=1,2,3
set xi=min{max{xi, 0}, 1 000}
```

可以把这一过程看作在粒子与穿透面之间插入刚度无穷大的弹簧。这些弹簧的刚度和阻尼极其高，弹性形变后会立即恢复原状。

以建立长度为 100 的棍状模型为例。具体过程为设置 2 个独立的粒子(位置分别为 x_1 与 x_2)，将他们的距离设置为 100。用数学的方式表示，可以得到以下双边约束条件：

$$\left|x_2 - x_1\right| = 100。 \tag{4}$$

粒子一开始可能被放置在正确的位置，但是经过一次积分后，粒子间的距离可能不会再满足约束条件。为了重新取回正确的距离，可以将粒子映射到式(4)的解集，改变粒子的位置。如图 2，通过直接将粒子间的距离拉远或拉近即可实现。

满足式(4)的伪码如下：

```
//满足式(4)的伪码
delta = x2−x1;
deltalength = sqrt(delta*delta);
diff = (deltalength-restlength)/deltalength;
x1 += delta*0.5*diff;
x2 −= delta*0.5*diff;
```

注意：delta 是个矢量，所以 delta*delta 实际上是个点积。

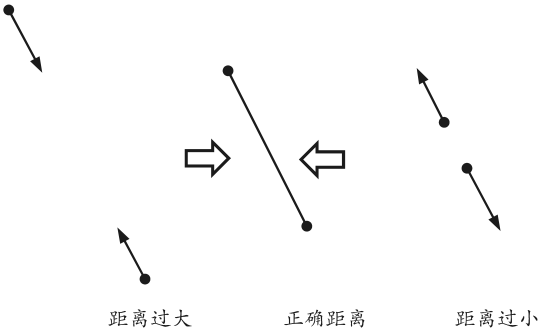

图 2　用移动粒子的方式修正粒子间的距离

通过设置 restlength=100，以上伪码将粒子间的距离拉远或拉近直到它们的距离重新变成 100 这个正确的值。依旧可以将这一方案看成在粒子之间插入一个刚度很大的弹簧，使得它们可以立即被放置在正确的位置。

现在假设粒子仍需满足立方体约束条件。为了满足棍状约束条件，会把一些粒子置于立方体外部。这样就会使一个或多个立方体约束条件变得无效。立即将穿透粒子的位置再次投影回立方体表面的策略可以弥补这一缺憾，但又会破坏棍状约束条件。

为了同时满足式(3)与式(4)的约束条件，笔者选取局部迭代的方式，即将以上 2 段伪码交替地执行若干次，直到结果变得有意义。代码如下(省略了 2 个粒子的初始化部分)：

```
//同时满足立方体与棍状约束条件
void ParticleSystem::SatisfyConstraints() {
for(int j=0; j<NUM_ITERATIONS; j++) {
  //首先满足式(3)
  for(int i=0; i<NUM_PARTICLES; i++) { //对所有粒子有效
    Vector3& x = m_x[i];
    x = vmin(vmax(x, Vector3(0,0,0)),
```

```
        Vector3(1000,1000,1000));
    }
    //然后满足式(4)
    Vector3& x1 = m_x[0];
    Vector3& x2 = m_x[1];
    Vector3 delta = x2-x1;
    float deltalength = sqrt(delta*delta);
    float diff = (deltalength-restlength)/deltaleng-
th;
    x1 += delta*0.5*diff;
    x2 -= delta*0.5*diff;
    }
}
```

虽然单纯循环的方案看起来有些过于简单，却正是文中所寻求的解决方案。将该方法称为张弛方法(依据做法的不同，又可称为 Jacobi 迭代或 Gauss-Seidel 迭代[9])。该方法是通过反复进行依顺序满足多个局部约束条件的过程，最终达至一个同时满足所有约束条件的状态。它也适用于其他需要同时满足多个相互独立的约束条件的场合。

需进行的循环次数取决于所模拟的物理系统以及运动发生的频度。如果过早地结束循环，结果可能不太正确，但由于 Verlet 方法的存在，下一帧的结果会更好，以此类推。这就意味着虽然过早地结束循环不至于得不到结果，却会使结果不太精确。

4　布料模拟系统软件实现

4.1　布料模拟流程

布料模拟的具体流程如下：

1) 建立模型；

2) 运动仿真：

① 初始化各粒子的位置与速度；

② 各粒子受力清零；

③ 利用 Verlet 积分器，计算粒子新的位置；

④ 对各粒子使用张弛方法，使他们满足所有约束条件；

⑤ 将所有粒子的新位置传递到碰撞检测子系统中，判断各基元是否发生碰撞；

⑥ 在碰撞检测中获得接触时间和接触点，回馈到布料模拟主系统中，改变粒子的位置；

⑦ 返回②，继续下一时间步的运动仿真。

4.2　软件设计与实现

在 VC6.0 环境下的 PC 机(CPU I7 3770K，内存 8 g，显卡 Nvidia Geforce GTX 680，操作系统 Windows7 sp2)上初步实现了布料模拟系统(显示部分采用了 OPENGL 函数[10])。为了得到更直观的模拟结果，在程序中设置了一个场景，用大炮轰击窗帘，然后观察窗帘与炮弹发生碰撞后的反应及窗帘如何逐渐恢复平衡状态。该程序主要由以下 3 个模块构成：

1) 布料模拟模块。

具体模拟过程如前文所述，模拟流程见 4.1 节。

2) 投射处理模块。

场景中的投射物(炮弹)都是能以某种特定角度和速度发射的球体，直到发生碰撞前，它们都会在重力的作用下飞行。投射物的受力情况也是利用 Verlet 积分器来计算。利用碰撞检测系统和张弛方法来处理球体与地面及布料(窗帘)的碰撞。在张弛过程中，允许布料对球体发生作用，使球体与布料之间产生具有真实感的碰撞效果。

3) 场景管理器。

笔者编写了一个简单的场景管理器，用于对场景中一系列活动的物体进行模拟、渲染，剔除不活动的物体。另外，场景管理器还允许使用者与场景进行简单的互动。

模拟效果见图 3、图 4。

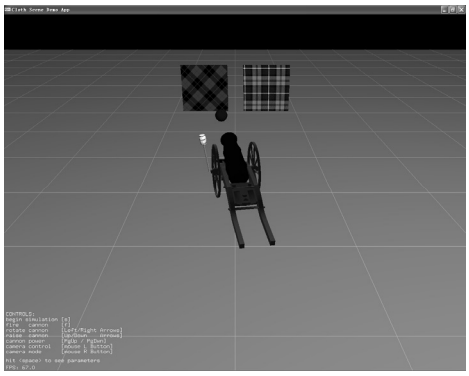

图 3　炮弹在空中飞行

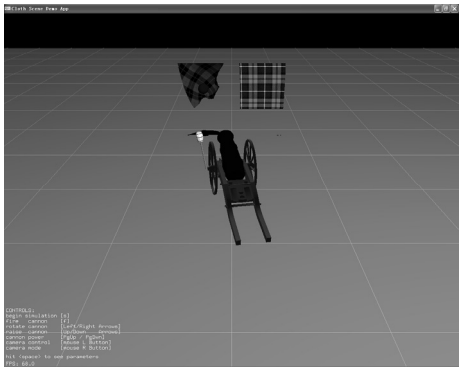

图 4　炮弹与窗帘发生碰撞

(下转第 91 页)

模拟研究[J]. 弹道学报, 2006, 18(1): 90−92.
[3] 顾文彬, 李旭峰, 李裕春, 等. EFP 引爆盖板炸药盒数值模拟[J]. 弹道学报, 2013, 25(3): 59−63.
[4] 章冠人, 陈大年. 凝聚炸药起爆动力学[M]. 北京: 国防工业出版社, 1991.
[5] 崔凯华, 洪滔, 曹结东. 射弹冲击带盖板 CompB 装药起爆过程数值模拟[J]. 含能材料, 2010, 18(3): 286−289.
[6] ALMOND R J, MURRAY S G. Projectile attack of surface scattered munition: Prompt shock finite element models and live trial[J]. Propellants, Explosives, Pyrotechnics, 2006, 31: 83.
[7] URTIEW P A, VANDERSALL K S, TARVER C M, et al. Shock initiation Experiments and modeling of Composition B and C4[C]//Proceeding of the 13th International Detonation Symposium, Norfolk Virginia USA, 2006: 929−939.
[8] TARVER C M, HALLQUIST J Q, ERICKSON L M. Modeling short pulse duration Shock initiation of solid explosives[C]//Proceeding of the 8th International Detonation Symposium, Naval Surface Weapons enter, Albuquerque, NSWCMP86-194, 1985: 951−961.

（上接第 81 页）

4.3　系统效率

为了测试系统的整体性能，笔者在 VC6.0 环境下在 PC 机(CPU I7 3770K，内存 8 g，显卡 Nvidia Geforce GTX 680，操作系统 Windows7 sp2)上进行了如下实验：1) 使用文中系统进行 5 次碰撞(开炮 5 次)，记录每次碰撞检测消耗的时间；2) 将系统中的布料模拟与碰撞检测部分换成 SOLID2.0 包，重复以上过程，记录碰撞检测消耗时间。结果如表 1 所示。

表 1　文中系统与典型系统效率比较　　ms

次数	文中系统	基于特征值的算法	空间分解法	基于物体空间方法	基于 SOLID2.0 的系统
1	152	165	160	172	180
2	153	166	158	170	182
3	155	168	165	171	179
4	159	164	160	172	180
5	155	166	162	169	178

5　结束语

从以上结果可以看出，文中系统在模拟布料时比一些经典碰撞检测算法效率高。系统效率的提高主要是来自 2 个方面：

1) 以 Verlet 积分方法为核心的数种有效方法的改进与整合；

2) 采用随时可以停止的有限迭代方法，用精度换时间的策略。

参考文献：

[1] 蓝磊. 基于位置约束的布料动态模拟研究[D]. 厦门: 厦门大学，2014.
[2] 铁铮. 面向服装变形的三维布料仿真技术研究与实现[D]. 上海: 东华大学, 2017.
[3] 周玲莉. 基于物理模型的布料实时动画关键技术研究[D]. 济南: 山东财经大学，2015.
[4] 卢路加, 张君会, 赵志稳. 欧拉积分性质及应用[J]. 亚太教育, 2015, 11(20): 123−125.
[5] VERLET L. Computer experiments on classical fluids. I. Thermo dynamical properties of Lennard-Jones molecules[J]. Physical Reviews, 1967, 159: 98−103.
[6] 于晓霞. 碰撞检测技术在游戏中的应用与实现[D]. 太原: 太原理工大学, 2007: 25−28.
[7] 刘浩. 基于质点—弹簧模型的实时三维布料模拟系统[D]. 上海: 上海交通大学, 2007: 132−135.
[8] GUIDO B, DENNIS A, FRANZE W. Virtual Reality Systems Modelling Haptic Two-Finger Contact with Deformable Physical Surfaces[C]//2007 International Conference on Cyberworlds. Los Alamitos, CA, USA: IEEE Computer Society Press, 2007: 114−123.
[9] METCALF M, PRESS W H, TEUKOLSKY S A, et al. Numerical recipes in Fortran 90, Vol 2[M]. Cambridge, UK: Cambridge University Press, 1993: 38−44.
[10] 柯莹, 梁惠娥, 王宏付. 基于 OpenGL 的虚拟服装二维样版设计与修改[J]. 东华大学学报(自然科学版), 2018, 44(3): 35−39.