Journal of System Simulation

Volume 28 | Issue 10 Article 26

8-13-2020

GPU-accelerated Cloth Animation Based on Module Division

Kun Zhou
1. Faculty of Electrical Engineering and Computer Science, Ningbo University, Ningbo 315211, China;;

Liu Zhen
1. Faculty of Electrical Engineering and Computer Science, Ningbo University, Ningbo 315211, China;;

Gaoqi He
2. Department of computer science and engineering, East China University of Science and Technology, Shanghai 200237, China;;

Chen Tian
3. Shanghai Dianji University, Shanghai 200000, China;

See next page for additional authors

GPU-accelerated Cloth Animation Based on Module Division

Abstract

Abstract: In order to improve the simulation speed of cloth animation, *a module division parallel computing method was proposed by studying two kinds of cloth simulation models with dynamics and position based dynamics (PBD). A computing task was divided into a plurality of independent blocks by the way of data block. GPU parallel computing was used within the data block to improve the real-time performance and robustness of the cloth simulation. The applicability and effectiveness of the algorithm were verified under different scenarios and stochastic wind field.* Experimental results show that under the same conditions, the proposed method can effectively improve the simulation speed. Moreover, it can meet real-time requirements at higher simulation accuracy.

Keywords

dynamic simulation, cloth animation, GPU parallel computer, wind field, module division

Authors

Kun Zhou, Liu Zhen, Gaoqi He, Chen Tian, Tingting Liu, and Cuijuan Liu

Recommended Citation

Zhou Kun, Liu Zhen, He Gaoqi, Chen Tian, Liu Tingting, Liu Cuijuan. GPU-accelerated Cloth Animation Based on Module Division[J]. Journal of System Simulation, 2016, 28(10): 2476-2484.

第 28 卷第 10 期 系统仿真学报© Vol. 28 No. 10
2016 年 10 月 Journal of System Simulation Oct., 2016

基于模块划分的 GPU 加速布料动画

周昆[1]，刘箴[1]，何高奇[2]，陈田[3]，刘婷婷[1]，刘翠娟[1]
(1. 宁波大学信息科学与工程学院 宁波 315211；2. 华东理工大学计算机科学与工程系 上海 200237；3. 上海电机学院 上海 200000)

摘要：为了提高布料动画的仿真速度，*通过研究动力学和基于位置约束动力学（PBD）的两种布料仿真模型，提出了划分模块的并行计算方法。首先利用数据分块的方式将单个计算任务划分为独立的块；然后对块内采用 GPU 并行计算，进而提高了布料动画仿真的实时性和鲁棒性；最后，观察布料在不同的场景环境和随机风场作用下的仿真动画，验证了该算法的适用性和有效性。*实验结果表明，在相同的条件下，文中计算方法能够有效提高仿真速度，同时在仿真精度较高时，依然可以满足实时性的仿真动画要求。

关键词：动力学仿真；布料动画；GPU 并行计算；风场；模块划分
中图分类号：TP391.9 文献标识码：A 文章编号：1004-731X (2016) 10-2476-09

GPU-accelerated Cloth Animation Based on Module Division

Zhou Kun[1], *Liu Zhen*[1], *He Gaoqi*[2], *Chen Tian*[3], *Liu Tingting*[1], *Liu Cuijuan*[1]
(1. Faculty of Electrical Engineering and Computer Science, Ningbo University, Ningbo 315211, China;
2. Department of computer science and engineering, East China University of Science and Technology, Shanghai 200237, China;
3. Shanghai Dianji University, Shanghai 200000, China)

Abstract: In order to improve the simulation speed of cloth animation, *a module division parallel computing method was proposed by studying two kinds of cloth simulation models with dynamics and position based dynamics (PBD). A computing task was divided into a plurality of independent blocks by the way of data block. GPU parallel computing was used within the data block to improve the real-time performance and robustness of the cloth simulation. The applicability and effectiveness of the algorithm were verified under different scenarios and stochastic wind field.* Experimental results show that under the same conditions, the proposed method can effectively improve the simulation speed. Moreover, it can meet real-time requirements at higher simulation accuracy.

Keywords: dynamic simulation; cloth animation; GPU parallel computer; wind field; module division

引言

随着布料动画仿真技术的不断发展，布料仿真动画被广泛的应用于服装、电影、游戏等多个领域。由于基于物理模型的布料仿真技术能够较为逼真的模拟各类布料的运动效果，从而成为一种常用的布料仿真技术和研究热点。然而，近年来，图形硬件的计算能力日益强大，同时人们对布料仿真动画的速度和质量提出了更高的要求，采用 GPU 并行计算已成为仿真动画中的一种主流计算手段，它被应用于仿真的各大模块，提高布料的仿真效率。

基于粒子的布料仿真模型，即将布料看成一系列点组成的几何体，具有一定的结构。使用粒子进行动画模拟与计算，通过驱动粒子的运动以实现布料运动。其中基于物理动力学和基于位置约束动力

收稿日期：2016-05-31 修回日期：2016-07-09；
基金项目：国家自然科学基金(61373068)，宁波市科技计划项目(2015A610128, 2015C50053, 2015D10011, 2016D10016)；高等学校博士学科点专项科研基金(20133305110004)；
作者简介：周昆(1992-)，男，江西，硕士生，研究方向为布料动画。

学(PBD)等模型是目前仿真中的常用的仿真技术，但随着粒子数量的增加，仿真的效率将大大降低。

GPU 并行计算已成为当前模拟计算的一种重要技术，例如将力学计算，碰撞检测，渲染移植于图形显卡中进行，提高效率。但并行计算需要合理分配 GPU 计算任务，否则容易造成数据操作错误。

1 相关工作

布料模型主要有几何法、物理法、混合法、约束动力学、数据驱动等多种模型[1-4]。其中，由于布料动态效果非常复杂，几何法主要用于静态布料动画。数据驱动依赖于数据源，重用性比较低。物理法和约束动力学有较大的推广。早期，Provot[5]提出的基于离散粒子的弹簧-质点模型成为一种典型的布料力学模型，将布料看成有限的质点和质点间相互连接的无质量弹簧组成的结构体，这种模型后来被不断的进行改进形成了多种布料力学模型。

约束动力学[6]从另一个角度入手，同样是先将布料离散成具有质量、位置、速度等物理属性的粒子，粒子受到内力和外力的作用，外力作用遵守牛顿第二定律，内力和碰撞被看作约束条件；然后将布料的运动看成符合约束条件的动力学系统，相比于力学模型有更好的稳定性，但约束条件和约束模型很难控制，不合理的约束条件容易造成动画失真[7]。Micky[8]提出一种三角形弯曲约束代替Muller[6]提出基于相邻边二面角的弯曲约束条件，以更快的速度达到类似的动画效果。然而，无论采用何种模型，随着问题的规模增大，其计算复杂度也在不断增加，动画难以实时。使用 GPU 并行计算能够有效的缩短计算时间,Tang[9]给出一种图形硬件加速的柔性物体连续碰撞检测算法用于实时的检测柔性物体与环境的碰撞以及形变产生的自碰撞。Sabou[10]提出了显式速度 Verlet 积分和隐式欧拉积分的并行化算法。Lauterbach[11]使用基于层次结构的并行碰撞检测算法处理刚体和形变体的碰撞。纪传舜[12]提出基于包围球的自碰撞检测算法，该算法首先构造粒子路径的 AABB 包围盒，在包围盒内的粒子使用等半径包围球进行运动包围球的连续碰撞检测，该算法相比于传统的方法能实现快速的自碰撞处理，需要执行两次相同的自碰撞计算，算法效率有所降低。

本文通过对动力学和约束动力学这两种常用的布料仿真算法进行分析，将算法实现过程中相互独立的模块提取出来,并利用独立模块并行化的方式提高效率，例如对于超弹性的修正，采用独立划分需要修正的块，提高了模拟的稳定性。最终，实现布料与不同模型的交互效果以及在随机风场下的布料动画效果。对于布料的自碰撞，本文通过改进文献[12]中的方法，仅需要一次碰撞计算即可实现自碰撞处理，提高了计算效率。

2 弹簧-质点模型

弹簧-质点模型由具有物理属性的质点和无质量的弹簧组成，粒子在内力和外力的共同作用下运动。其中粒子之间连接的轻质弹簧主要用于体现粒子之间的相互作用力，其结构如图 1 所示。

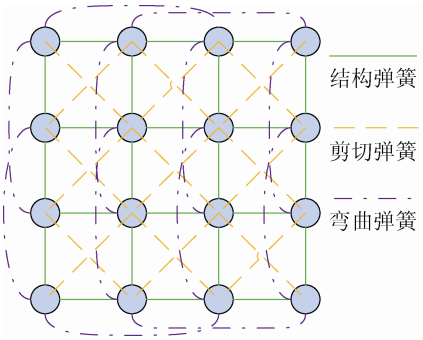


图 1 弹簧质点模型的结构

相邻质点运动产生相互的阻尼力，阻尼力影响领域范围内的质点运动。质点所受的外力主要包括重力、风力、碰撞产生的冲力等等。质点在内力和外力共同作用下运动。首先计算质点受到的加速度，通过当前帧和上一帧质点的位置，使用基于位置的 Verlet 积分方法进行数值计算，受力计算为：

第 28 卷第 10 期 系统仿真学报 Vol. 28 No. 10
2016 年 10 月 Journal of System Simulation Oct., 2016

$$\begin{cases} \boldsymbol{F}^{i}_{\text{ext}} = \boldsymbol{m}_i\boldsymbol{g} + \boldsymbol{F}^{i}_{d} + \boldsymbol{F}^{i}_{c} \\ \boldsymbol{F}^{i}_{\text{int}} = \sum_j \boldsymbol{F}^{i}_{e}(\boldsymbol{x}_j) + \sum_k \boldsymbol{F}^{i}_{vd}(\boldsymbol{v}_k) \end{cases} \tag{1}$$

其中：$\boldsymbol{F}^{i}_{ext}$ 表示第 i 个质点受到的外力；$\boldsymbol{m}_i\boldsymbol{g}$ 为重力；$\boldsymbol{F}^{i}_{d}$ 为全局的阻尼力与质点的速度有关；$\boldsymbol{F}^{i}_{c}$ 为质点运动时碰撞产生的冲力，接触摩擦力等的综合；$\boldsymbol{F}^{i}_{\text{int}}$ 为质点受到的内力，主要包括三种类型的弹簧力 $\boldsymbol{F}^{i}_{e}(\boldsymbol{x}_j)$ 和邻域粒子间的阻尼力 $\boldsymbol{F}^{i}_{vd}(\boldsymbol{v}_k)$。本文采用基于物理位置的 Verlet 积分进行位置更新，速度项由前后帧的最终位置确定：

$$\boldsymbol{v}^{n}_{i} = (\boldsymbol{x}^{n}_{i} - \boldsymbol{x}^{n-1}_{i}) / \Delta t \tag{2}$$

弹簧力 $\boldsymbol{F}^{i}_{e}(\mathrm{x}_j)$ 和阻尼力 $\boldsymbol{F}^{i}_{vd}(\boldsymbol{v}_k)$ 的计算如下所示：

$$\begin{cases} \boldsymbol{F}^{i}_{e}(\boldsymbol{x}_j) = \sum_j \varphi(\boldsymbol{x}_i,\boldsymbol{x}_j)\,|\,l_{ij} - l^{0}_{ij}\,|\,\dfrac{\Delta\boldsymbol{x}_{ij}}{|\,\Delta\boldsymbol{x}_{ij}\,|} \\ \boldsymbol{F}^{i}_{vd}(\boldsymbol{v}_k) = \sum_k \phi(\boldsymbol{x}_i,\boldsymbol{x}_k)\,|\,\dfrac{\Delta\boldsymbol{v}_{ik}\bullet\Delta\boldsymbol{x}_{ik}}{\|\Delta\boldsymbol{x}_{ik}\|^2}\,|\Delta\boldsymbol{x}_{ik} \end{cases} \tag{3}$$

其中：$\varphi(\boldsymbol{x}_i,\boldsymbol{x}_j)$ 表示质点 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$ 的弹性系数：

$$\varphi(\boldsymbol{x}_i,\boldsymbol{x}_j) = k_e(l + l_0)^n / l^n_0 \tag{4}$$

其中：k_e 为线性弹性系数；l 和 l_0 分别表示弹簧当前长度和原长，引入非线性弹性系数可以有效改善超弹性问题。如果两者存在某种弹簧连接，则 $\varphi(\boldsymbol{x}_i,\boldsymbol{x}_j)$ 弹性系数，否则为 0，同理,粒子如果相邻，则 $\phi(\boldsymbol{x}_i,\boldsymbol{x}_k)$ 为阻尼系数 k_d，否则为 0，$\Delta\boldsymbol{x}_{ij},\Delta\boldsymbol{x}_{ik}$ 表示受力的方向。通过计算质点的内力和外力得到粒子的加速度 $\boldsymbol{a}_i$,再更新粒子的位置:

$$\boldsymbol{tmpX}^{n+1}_{i} = 2.0\boldsymbol{x}^{n}_{i} - \boldsymbol{x}^{n-1}_{i} + \boldsymbol{a}^{i}\Delta t^2 \tag{5}$$

式中：$\boldsymbol{x}^{n}_{i},\boldsymbol{x}^{n-1}_{i}$ 分别为第 n 步和第 n–1 步的最终位置，$\boldsymbol{a}^{i}$ 为第 n 步粒子 i 的加速度；上述过程为布料的力学仿真计算模块，计算结果为下一时刻的预测位置 $\boldsymbol{tmpX}^{n+1}_{i}$，之所以称为预测位置，是由于布料由于运动可能发生碰撞，需要进行碰撞的修正。由力学知识可知，弹簧力会同时作用于两质点，采用并行的计算方式必须使用同一帧下粒子的空间位置信息和速度信息计算力的作用，否则会产生误差，在时间步长较大和粒子点较多时容易造成系统失真。

如果采用迭代式的计算方式，即更新完粒子的位置之后直接将粒子的位置保存在位置缓存中，在计算相邻点的粒子时，由于相邻质点粒子的位置可能已经被更新，获取的相对位置是更新之后粒子的位置。例如在第 n 帧时，对于第 j 个质点 $\boldsymbol{p}^{n}_{j}$，计算与 $\boldsymbol{p}_j$ 相连接的质点 $\boldsymbol{p}_i$ 的弹簧力：

$$\boldsymbol{F}^{ij}_{e}(\boldsymbol{x}_i) = \varphi(\boldsymbol{x}_i,\boldsymbol{x}_j)\,|\,\boldsymbol{l}_{ij} - \boldsymbol{l}^{0}_{ij}\,|\,\frac{\Delta\boldsymbol{x}_{ij}}{\|\,\Delta\boldsymbol{x}_{ij}\,\|} \tag{6}$$

其中如果 $\boldsymbol{p}_i$ 的位置已经更新，则造成单步误差为：

$$\Delta\boldsymbol{F}^{ij}_{e} = \boldsymbol{F}^{ij}_{e}(\boldsymbol{x}^{n-1}_{i}) - \boldsymbol{F}^{ij}_{e}(\boldsymbol{x}^{n}_{i}) \tag{7}$$

将数值积分计算模块分为两个模块：第一用于并行更新所有粒子当前的弹簧力和阻尼力即计算粒子的加速度；第二将得到的加速度进行更新质点的预测位置。在模块一中由于质点的位置没有进行更新操作，所以整个系统处于固定的状态下，所有的力的计算具有唯一性，且质点的速度是用前后帧的最终位置与时间差商得到，速度保持不变，因此粒子加速度也是唯一的，从而保证了系统的稳定性。

3 约束动力学模型

3.1 外力和阻尼力的计算

仿真模型被当成由 N 个粒子和 M 个约束条件组成的系统，首先按照类似的方法计算粒子受到的外力，如重力、风力等。而粒子间的相互作用被表达成约束条件，粒子运动为满足约束条件的运动。在外力作用下，更新粒子速度：

$$\boldsymbol{v}_i = \boldsymbol{kd}' \cdot \boldsymbol{v}_i + \boldsymbol{a}^{i}\Delta t \tag{8}$$

为增加系统的稳定性，引入基于质点速度和位置的阻尼作用。这种方式将布料看作具有一定刚性的物体，整体的运动对单个粒子运动产生阻尼作用。

3.2 约束条件

3.2.1 距离约束

约束条件主要分为两类结构约束和碰撞约束，其中结构约束分为距离约束和弯曲约束。距离约束主要用于在拓扑结构上相邻的粒子点，弯曲约束主

要用于表现布料在运动时自然弯曲和褶皱等动画效果，约束条件用于修正粒子的位置(如图 2 所示)。对于距离约束定义如下：

$$C_d(\boldsymbol{p}_1,\boldsymbol{p}_2)=|\boldsymbol{p}_1-\boldsymbol{p}_2|-d \tag{9}$$

式中：d 为初始长度，其对相关的两个粒子位置修正值为：

$$\Delta\boldsymbol{p}=-\frac{C(p)}{|\nabla\boldsymbol{p}C(p)|^2}\frac{\boldsymbol{p}_1-\boldsymbol{p}_2}{|\boldsymbol{p}_1-\boldsymbol{p}_2|} \tag{10}$$

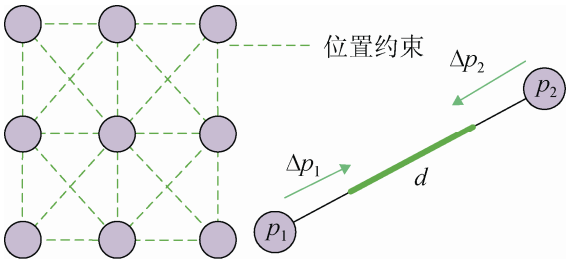


图 2 距离约束

对于 $\boldsymbol{p}_1$ 满足所有距离约束条件的位置修正值为：

$$\Delta\boldsymbol{p}_1=-w_1\frac{C(p_1,\cdots p_N)}{\sum_i w_j|\nabla p_j C(p_1,\cdots p_N)|^2}\nabla\boldsymbol{p}_1 C(p_1,\cdots p_N) \tag{11}$$

式中：w_1 为质量倒数，迭代的刚度系数 κ 与单步迭代次数 u 以及距离刚性系数 K 有关：

$$\kappa=1.0-(1.0-K)^{1/u} \tag{12}$$

3.2.2 弯曲约束

Micky[8]提出一种快速的弯曲约束条件，将相邻非直接相连接的三个质点(如图 3 所示)，本文将弯曲约束看成类似于弹簧质点模型中“弯曲弹簧”的几何结构组以粒子 $\boldsymbol{b}_0$、$\boldsymbol{v}$、$\boldsymbol{b}_1$ 为例，$\boldsymbol{c}$ 为三个顶点组成三角形的质心：

$$\boldsymbol{c}=\frac{1}{3}(b_0+b_1+\boldsymbol{v}) \tag{13}$$

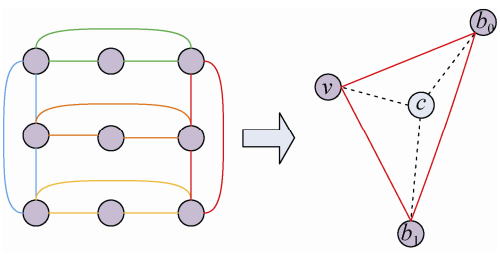


图 3 弯曲约束结构

三者组成的弯曲约束为：

$$C_{tri}(b_0,b_1,v)=\|\boldsymbol{v}-\boldsymbol{c}\|-\boldsymbol{h}_0 \tag{14}$$

式中：h_0 为初始化时刻，顶点 v 到质心的距离；为提高模拟的稳定性采用高斯-塞德尔的迭代方式逐步修正位置以满足约束条件，其单步迭代的修正系数 k'：

$$k'=1.0-(1.0-k_b)^{1/u'} \tag{15}$$

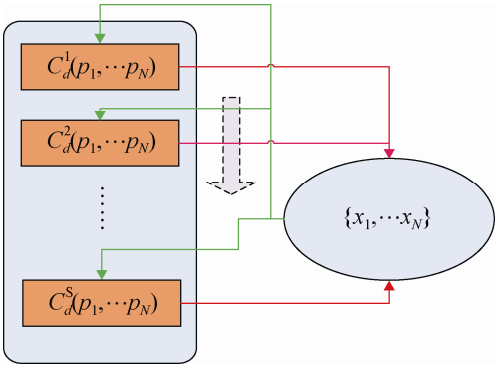
式中：k_b 弯曲系数；u' 为单步迭代次数；w_{tri} 为三个顶点质量倒数之和，三者位置的修正量为：

$$\begin{cases}\Delta\boldsymbol{v}=-\dfrac{4w_v}{w_{tri}}\left(1-\dfrac{\kappa+\boldsymbol{h}_0}{\|\boldsymbol{v}-\boldsymbol{c}\|}\right)(\boldsymbol{v}-\boldsymbol{c})\\ \Delta\boldsymbol{b}_i=\dfrac{2w_{b_i}}{w_{tri}}\left(1-\dfrac{\kappa+\boldsymbol{h}_0}{\|\boldsymbol{v}-\boldsymbol{c}\|}\right)(\boldsymbol{v}-\boldsymbol{c})\end{cases} \tag{16}$$

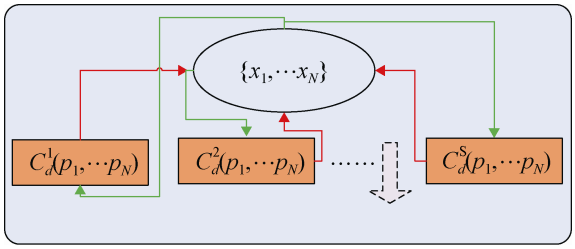
4 模块划分的并行性分析与实现

4.1 弯曲约束和距离约束并行性分析

以距离约束为例，如果利用单核 CPU 进行仿真计算，则所有计算过程都是顺序执行的，其过程如图 4 所示。



(a) 顺序执行



(b) 并行执行

图 4 两种执行方式的对比

第 28 卷第 10 期 系统仿真学报 Vol. 28 No. 10
2016 年 10 月 Journal of System Simulation Oct., 2016

在图 4(a)中对每个距离约束条件，获取与该约束条件相关的粒子位置，计算相应的位置修正值，再将修正后的结果存回到位置集合中，所有约束条件顺序执行。而在图 4(b)中，由于使用并行计算，在同一时刻可能存在两个约束条件同时进行，例如对于约束条件 $C_d^{i_1}(\boldsymbol{p}_j, \boldsymbol{p}_{k_1})$ 和 $C_d^{i_2}(\boldsymbol{p}_j, \boldsymbol{p}_{k_2})$ 同时取得当前粒子 $\boldsymbol{p}_j$ 的位置，会发生写入的错误，即只有一个约束条件对 $\boldsymbol{p}_j$ 的修正量会写回到位置集合中，另一个约束条件将被忽略，从而导致计算错误。

独立划分策略：对于 s 个距离约束组成的集合 $U_d:\{C_d^1, C_d^2, \cdots C_d^s\}$ 每个约束条件与粒子集合中的两个顶点相关，将 U_d 划分成 n_0 个独立的子集合，满足以下三个条件：

1. 每个子集合 $\tilde{U}_d^{n_j}:\{C_d^{i_0^1}, \cdots C_d^{i_{n_j}^1}\}$ 内任意两个约束条件不能存在共同的顶点；

2. 每个子集合的约束条件互不相同，元素数尽可能的均衡，即 $D_n = \min(\sum(n_j - \bar{n})^2)$，保证负载均衡；

3. 所有子集合是全集 U_d 的一种划分，即 $U_d = \tilde{U}_d^{n_1} \cup \tilde{U}_d^{n_2} \cdots \cup \tilde{U}_d^{n_i}$；

在执行时，每一个子集合作为一个模块，模块内并行执行，模块之间顺序执行，这样可以保证每次执行时，对同一个顶点仅执行一次位置读取操作，不会产生错误，过程如图 5 所示。相比于所有约束条件顺序的执行；采用模块划分，模块内并行，模块外顺序执行的方式能够有效的加速计算，粒子点数增加，加速效果将更明显(见第 7 节实验)。

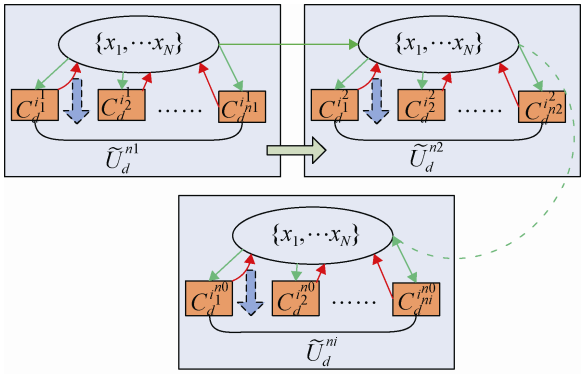


图 5 模块划分并行与顺序执行的协同计算

4.2 模块划分算法实现

针对于 GPU 计算需要将大问题按照数据独立性的原则进行划分计算任务，其中最重要的一个原则就是在保证数据独立性的前提下，尽可能的实现计算任务的均衡化。数据独立性的原则可以保证计算的正确性和迭代计算的稳定性，即快速收敛。在初始化时，把布料看成为二维三角形组成的网格结构体，其中对于顶点的操作，会同时影响到共顶点的若干三角形，同理对于边的操作会同时影响到共边的两个三角形。在布料的超弹性修正或者是 PBD 约束条件都是基于三角形边为基本单位进行操作，但实际是处理与边相关联的两个顶点数据，因此按照 4.1 小节中的条件 1，最少模块数为点集合最大关联的边数，不失一般性，边数最多的顶点为 P_0。因此最少模块数 N_c 为：N_c=max(P_0)为了方便说明我们先讨论最简单的布料拓扑结构，即 $N{\cdot}N$ 的质点阵列。如图 6 所示，N_c 顶点 P_0 关联的边数 N_c=4，因此模块的最少集合数为 4，对于规则的图形我们很容易对其进行 4.1 的划分，通过分析其划分的规则从本质而言是一致的：对于顶点组成的方阵如果我们按照经向经纬方向分为两类，例如以结构弹簧(也是距离约束的一种)为例，其按照经向和纬向分类如图 6 所示。

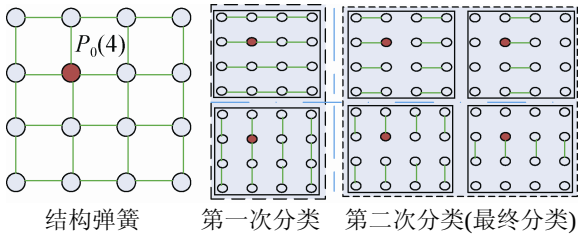


图 6 逐级划分模块

首先这种逐级划分方式很显然满足 4.1 的条件 3；采用数学归纳方法很容易证明这种划分方式满足 4.1 提出的数据独立性条件，在数据量较大时，这种划分也趋于负载均衡即满足 4.1 的条件 2。对于长方形阵列，需要先进行区域划分，转化为若干正方形阵列，则正方形阵列之间数据具有独立性，即转化为上述的情况，对于划分时，阵列之间的连

接边,采用遍历的方式可以很容易加入到集合中。

虽然本文没有详细研究复杂的网格模型，但采用贪心递归的方式进行正方形区域分割，即转化为规则的情况，可以很容易得到其划分结果。按照最大规模正方形为基准，进行上述过程即可。见图 7。

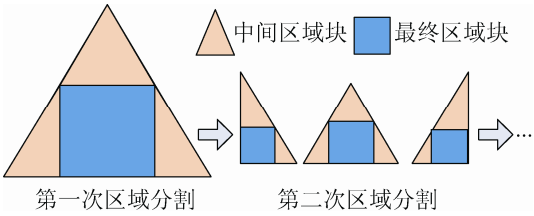
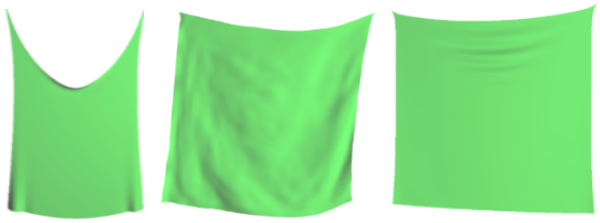


图 7 递归区域分割

4.3 动力学模拟的超弹性问题

把布料当成质点和弹簧组成的系统；在布料悬垂时，固定两个质点，布料在重力的作用下自然下垂最终达到静止的状态，则两个固定的质点与其他部分之间靠两个弹簧相连接，在力学的模拟下容易出现超弹性的问题。按照 Provot 的思想对连接有弹簧的质点进行位置修正，使得形变率缩小在 10%以内，以增加布料动画的真实性。按照约束动力学约束条件并行计算和顺序计算的两种策略，顺序计算仍然会存在对数据操作不同步，例如同时对同一个质点位置进行修正，产生计算错误。按照上述模块划分的策略进行模块内并行，模块之间串行的方法，能够有效的加快计算速度，且保证数据操作的正确性，其结果如图 8 所示。



(a) 非线性弹性系数 (b) 全局并行 (c) 模块划分

图 8 三种计算方式结果对比

图 8 中，粒子规模为 50×50，图 8(b)、8(c)两者迭代的次数均为 4 次。图 8(a)为布料仅使用非线性超弹性方式的布料悬垂动画，在图 8(b)中，利用 GPU 实现完全并行化，抖动明显，图 8(c)中，使用模块划分的方式动画自然稳定，效果更加真实。

5 碰撞处理

5.1 布料与场景碰撞

三维动画里，碰撞检测和处理对动画质量的影响起着重要的作用。常用的碰撞检测算法有基于层次结构的碰撞检测算法和基于空间划分结构的碰撞检测算法。本文采用基于空间划分结构的碰撞检测算法，空间划分即将仿真空间按基本长度均分为单元网格。每个单元网格内或为空，或包含着布料和模型的部分。碰撞检测分为三个阶段，高层碰撞剔除和低层碰撞检测以及三角形对的碰撞检测。

高层碰撞剔除：模型从 3DMax 中导出 OBJ 的文件格式，其中数据包括了顶点的位置、法线、纹理信息。将数据进行规格化之后，得到由三角面片构造成的网格模型，即碰撞的场景。在 GPU 中并行的取出一个三角形，计算三角形的形心。以点 p_i、p_j、p_k、c_{ijk} 为例，c_{ijk} 为三个顶点组成三角形的形心，c_{ijk} 所在空间的单位网格即为三角形 $\{p_i,p_j,p_k,c_{ijk}\}$ 所在的网格，每个网格均为三角形的集合 $G_t:\{T_0,T_1,\cdots T_n\}$。与此类似的，布料在两种模型下均有粒子组成，以任一粒子为三角形的一个顶点，其经纬方向相邻的两个粒子，三者共同组成一个三角形，计算三角形的形心，确定形心所在的网格和邻域内的 26 个网格。在 27 网格外的场景与该三角形空间上不相邻，一个时间步长内不会发生碰撞，即可以剔除这类碰撞。

低层碰撞检测：以布料粒子三角形形心所在的网格为中心，依次取出邻域 27 个网格中的三角形与粒子三角形进行低层碰撞检测。即对于粒子三角形 T_c 和邻域内的场景网格三角形 T_s 两者均处于以各自形心为球心的包围球中，在 Δt 内，如果包围球无相交则不会发生碰撞，否则进入潜在碰撞队列。

三角形碰撞对检测：三角形的碰撞分为两个部分，即边与边的碰撞和点与面的碰撞。本文中的场景均为静态的场景，因此仅需要进行单向的碰撞检

第 28 卷第 10 期 系统仿真学报 Vol. 28 No. 10
2016 年 10 月 Journal of System Simulation Oct., 2016

测。以边与面的碰撞检测为例，设 E_{01} 为粒子三角形的一条边，采用线性插值的方式获取在时间步长内的运动状态，已知 E_{01} 的两个顶点的运动信息 $\boldsymbol{p}_0\{\boldsymbol{x}_0,\boldsymbol{v}_0\}$ 和 $\boldsymbol{p}_1\{\boldsymbol{x}_1,\boldsymbol{v}_1\}$，边上的点可表示为：

$$\boldsymbol{p}_t = \boldsymbol{x}_0 + \alpha(\boldsymbol{x}_1 - \boldsymbol{x}_0) \tag{17}$$

场景三角形 T_s :$\{\mathrm{p}_0',\mathrm{p}_1',\mathrm{p}_2'\}$ 内的点可表示为：

$$p' = \beta p_0' + \lambda p_1' + (1-\beta-\lambda)p_2' \tag{18}$$

联立上述两个等式:

$$(\alpha,\beta,\lambda)\boldsymbol{M} = p_2' - p_0 + \varepsilon \tag{19}$$

其中：ε 为一个极小距离向量，表示当边与 T_s 距离小于 ε，则判断两者相交，α, β, λ 为插值系数。碰撞处理：对于发生碰撞的粒子点，采用位置投影的方式，沿着碰撞面的法向方向，投影至面外，同时修正法向的速度以防止发生进一步的碰撞或穿透，切线方向加以摩擦力效果，减小切向的速度。碰撞效果如图 9 所示。

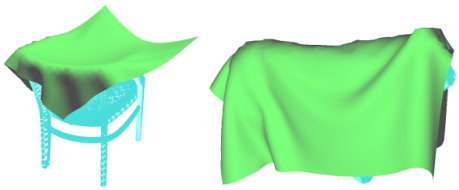


图 9　碰撞效果图

模型为静态的三维模型，只需要在初始化阶段采用空间均分的方式建立空间三角形链表化的信息。对于每一个运动的布料三角形分配一个 GPU 线程，在其邻域内进行碰撞检测，因此除初始化需要较长的时间外，其碰撞检测算法的效率较高。

5.2 布料自碰撞

为了提高动画的实时性，本文没有使用传统的连续碰撞检测方法。根据文献[12]的方法，首先记录粒子当前帧位置和修正后的预测位置，以统一的半径 $R \in (0.3L, 0.4L)$ 、粒子的位置为球心做包围球，根据包围球的运动路径做 AABB 包围盒，采用空间均分的方法，对处于包围盒内的邻域粒子和其包围球的运动路径进行运动球体之间的连续碰撞检测。为了获得碰撞响应的一致性，文献[12]中采用两次碰撞计算的方式进行分别进行碰撞检测和处理，本文采用粒子运动“副本”的方式，进行一次碰撞计算并同时进行碰撞响应。由于副本保存的是无碰撞响应的粒子位置信息，所以并不会由于在一次计算中同时进行碰撞检测和响应而导致的计算错误。在不明显降低动画速度情况下，有效的进行自碰撞的处理。

6　风场模型

本文采用五次多项式插值的柏林噪声产生随机风场[12]，但其计算的时间较长；为了提高动画的实时性，采用预计算的方式生成三维风向噪声纹理缓存和一维风速噪声纹理缓存。即一个长度为 N^4 的缓存数组，以当前时间作为数组索引，从数组中取出一个四维的变量 $\boldsymbol{W}(x,y,z,w)$，文献[13]风速：

$$|\boldsymbol{V}_w| = \boldsymbol{v}_0 + \alpha\sin(\omega t + \varphi_0) + \lambda \boldsymbol{d}(t) \tag{20}$$

自然状态下风表现出随机性，本文使用简化的风速：

$$|\boldsymbol{V}_w| = \lambda \boldsymbol{d}(t)\sin(\omega \mathrm{rand}(t)t + \varphi_0) \tag{21}$$

风速用于表示当前状态风的强度，得到风速后可根据风向得到最终风力为：

$$\boldsymbol{V}_w = \boldsymbol{W}.xyz \cdot |\boldsymbol{V}_w| \tag{22}$$

风场主要分为风速和风向两个因素，风与布料的作用，可看作风以一定的速度和方向与布料发生碰撞的粒子，碰撞产生的冲量使布料随风运动。布料以网格三角形为基本的单位，三角形面片上产生的冲量均分作用于三个粒子顶点。假设在某一时刻时的风速为 V_w，风向 N_w 作用于布料上的第 i 个三角面片 T_i 上，如图 10 所示。

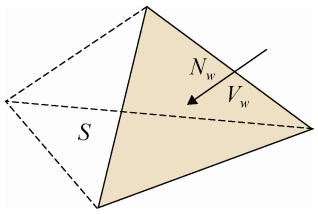


图 10　风作用示意图

由力学公式知，单位时间内，在风向方向正交单位面积上，其产生的冲量与相对速度大小 $\Delta \boldsymbol{V}$ ，

空气密度 ρ 有关：

$$\boldsymbol{I} = C\rho\phi^2(\Delta\boldsymbol{V}) \tag{23}$$

当布料网格三角形在风向上的速度的模长大于或等于风速时，产生的冲量大小为 0。因此有：

$$\phi^2\begin{cases}0 & \text{if } \mathrm{dot}(\boldsymbol{V}_t,\boldsymbol{N}_w)>1.0\\ \boldsymbol{V}_w^2[1.0-\mathrm{dot}(\boldsymbol{V}_t,\boldsymbol{N}_w)]^2 & \text{else}\end{cases} \tag{24}$$

因此在一个时间步长内，T_i 受的风力的冲量为：

$$\boldsymbol{I}_{T_i} = \boldsymbol{I}S\Delta t \tag{25}$$

S 为三角形在风力方向上的投影面积。粒子的位置信息保存于 GPU 的缓存中，使用 OpenGL 可编程管线可直接读取缓存中的粒子位置信息；粒子的法线为共顶点的所有三角形法线的加权平均：

$$\boldsymbol{n} = \mathrm{normlize}(\sum_k \frac{S_i}{\sum S_k}\boldsymbol{N}_i) \tag{26}$$

得到粒子位置和法线信息后对布料进行光照渲染。

7 实验结果

本文实验环境为一台配置为 Inter Core(TM) i7-4790(3.6GHz)CPU、12GB 内存、NV-GTX745 显卡的 PC 机。OpenGL 计算着色器程序实现并行计算，同时使用 OpenGL 进行渲染。对于三种不同精度的布料，对两种布料模型分别使用传统方法和本文算法进行模拟。其对比效果如表 1~2 所示。

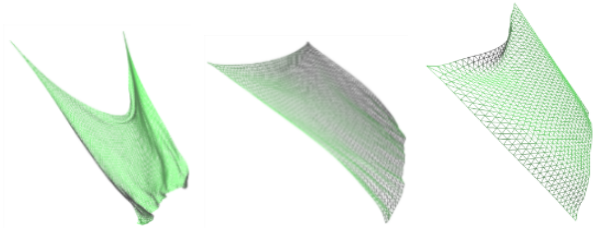
表 1 使用动力学模拟的加速比

精度/迭代	动力学模型		
	CPU/fps	本文/fps	加速比
30×30 /1	62.75	516.32	8.23
50×50 /2	30.33	268.64	8.85
100×100 /5	3.93	150.87	38.39

表 2 使用约束动力学模拟的加速比

精度/迭代	约束动力学模型		
	CPU/fps	本文/fps	加速比
30×30 /2	45.32	315.37	6.96
50×50 /2	15.39	130.12	8.45
100×100 /10	1.27	45.61	35.91

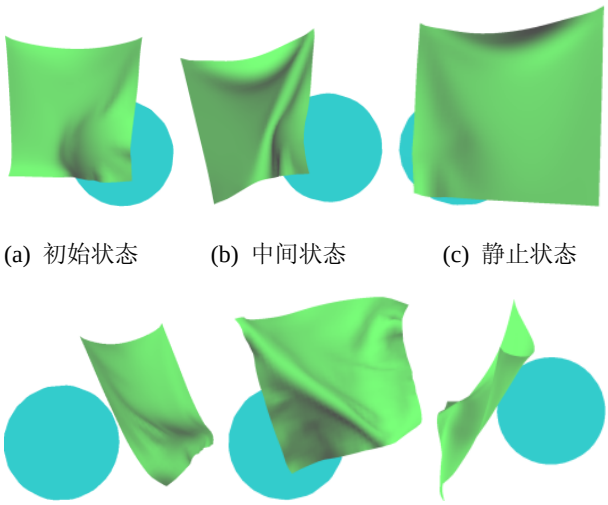
同一精度下，其迭代次数相同。从表 1 和表 2 中可看出本文方法能够有效加速计算。文献[14]中加入随机风场的因素，产生布料随风飘扬的动画，由图 11(a)可以看出，在风力的作用下，超弹性问题变得更为严重(图 11(a))，使用约束动力学和非线性弹性系数与超弹性修正的结合的两种方法，并在此基础上进行并行加速，(图 11(b)，图 11(c))。



(a) 非线性弹性系数 (b) 模块划分-1 (c) 模块划分-2

图 11 布料在风力作用下的动画效果

图 12(a)、12(b)、12(c)分别展示了布料下落与小球发生碰撞，在受力的作用下，最终达到 12(c)中的静止状态。12(d)、12(e)、12(f)展示了布料在静止后加入风力，布料在风力和碰撞的共同作用下发生运动的效果,图中可看出，在风力作用下，动画效果稳定且未出现超弹性。



(a) 初始状态 (b) 中间状态 (c) 静止状态

(d) 风吹效果-1 (e) 风吹效果-2 (f) 风吹效果-3

图 12 碰撞与风力的作用下的布料动画

织物的材料不同，其表现出的物理性质也不尽相同。如图 13 所示，第一行为采用动力学模拟的结果，其参数从左到右分别为(1.0, 1.5, 6)、(1.0, 1.0,

第 28 卷第 10 期 系统仿真学报 Vol. 28 No. 10
2016 年 10 月 Journal of System Simulation Oct., 2016

5)、(1.2, 1.2, 8)、(1.2, 1.5, 2)，三个参数分别表示了布料结构弹簧、剪切弹簧(形变阈值为第一个参数)，弯曲弹簧的形变阈值和计算的迭代次数，第二行采用 PBD 得到的模拟结果，其参数为 (k_1, k_2, κ, u')，具体的参数设置为(1.0, 1.0, 0.0, 3)、(1.0, 0.8, 0.0, 5)、(1.0, 1.0, 0.004, 10)、(0.9, 0.6, 0.004, 20)。

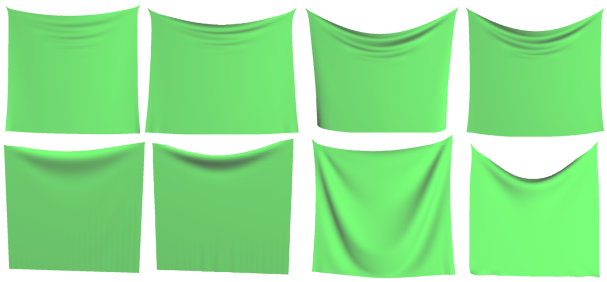


图 13 两种不同的布料仿真模型
采用不同参数得到的布料悬垂效果

8 结论

本文采用模块划分的 GPU 并行计算的方法，实现了动力学和约束动力学的布料动画模拟。首先，通过分析各阶段可并行性，然后采用全局均匀划分子集的方式，实现最大程度的并行化和计算负载均衡的目的。本文将雅可比迭代和高斯塞德尔迭代的优点进行整合，实现模块划分的并行化计算。实验中，在相同的情况下，仅需较少地迭代计算，即可达到稳定地、视觉效果真实的动画。本文主要为布料仿真的 GPU 加速优化提出了一些解决方法，将该方法应用于超弹性修正和约束条件的求解中，证明了本文方法的适用性。在实验中，较好的模拟出布料在风力和碰撞中的动画，通过调整布料参数能够得到物理性质各异的布料动画。但是，自碰撞处理的方式上做了简化，会导致穿透现象发生，需要在后期工作中进行研究以改善动画效果。

参考文献:

[1] Oshita M, Makinouchi A. Real-time cloth simulation with sparse particles and curved faces [C]// Computer Animation, 2001. The Fourteenth Conference on Computer Animation. Proceedings. USA: IEEE, 2001: 220-227.

[2] Wang H, Hecht F, Ramamoorthi R, et al. Example-bas-ed wrinkle synthesis for clothing animation [C]// ACM Transactions on Graphics (TOG). USA: ACM, 2010, 29(4): 107.

[3] Stumpp T, Spillmann J, Becker M, et al. A Geometric Deformation Model for Stable Cloth Simulation [C]// Proceedings of Virtual Reality Interactions and Physical Simulations (VRIPHYS),Grenoble, France. Grenoble, France, 2008: 39-46.

[4] Weil J. The synthesis of cloth objects [J]. ACM Siggraph Computer Graphics (S0097-8930), 1986, 20(4): 49-54.

[5] Provot X. Deformation constraints in a mass-spring model to describe rigid cloth behavior [C]// Graphics interface. Canada: Canadian Information Processing Society, 1995: 147.

[6] Müller M, Heidelberger B, Hennix M, et al. Position based dynamics [J]. Journal of Visual Communication and Image Representation (S1047-3203), 2007, 18(2): 109-118.

[7] 石敏, 毛天露, 夏时洪, 等. 布料动画方法研究进展及问题 [J]. 计算机学报, 2012, 35(12): 2446-2458.

[8] Kelager M, Niebe S, Erleben K. A Triangle Bending Constraint Model for Position-Based Dynamics [C]// Proceedings of Virtual Reality Interactions and Physical Simulations (VRIPHYS), Copenhagen, Danmark, 2010: 31-37.

[9] 唐敏, 林江, 童若锋. 图形硬件加速的柔性物体连续碰撞检测 [J]. 计算机学报, 2010, 33(10): 2022-2030.

[10] Sabou A, Gorgan D. A Parallel, Distributed, High Performance Architecture for Simulating Particle-Based Models [C]// Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2014 16th International Symposium on. USA: IEEE, 2014: 500-507.

[11] C Lauterbach, Q Mo, D Manocha. gProximity: Hierarchical GPU‐based Operations for Collision and Distance Queries [J]. Computer Graphics Forum (S1467-8659), 2010, 29(2): 419-428.

[12] 纪传舜, 刘卉. 基于质点的可变形体自碰撞检测 [J]. 中国图象图形学报, 2011, 16(3): 454-461.

[13] Perlin K. Improving noise [C]// ACM Transactions on Graphics (TOG). USA: ACM, 2002, 21(3): 681-682.

[14] 赖秋凤, 侯进. 基于噪声函数的随机风场作用的动态布料研究 [J]. 计算机仿真, 2015, 32(12): 192-196.

