

构建虚拟现实的目的是在虚拟的数字空间中模拟真实世界中的物体,这就需要利用建模技术将真实物体转换为逼真的虚拟对象,虚拟对象与真实物体的相似程度也与建模技术紧密相关。与计算机图形学中的建模概念略有不同,虚拟现实建模是指利用数字图像处理、计算机图形学、多媒体技术、传感与测量技术、仿真与人工智能等多个学科的技术,建立逼真的、可交互的、包含多属性的虚拟对象或场景。虚拟现实建模的核心内容主要包括几何、物理和行为三方面的建模。本章首先总体介绍虚拟现实建模的有关概念,然后围绕几何建模与物理建模,给出具体的案例。

## 3.1 虚拟现实建模的有关概念

### 3.1.1 虚拟现实建模的基本内容

#### 1. 几何建模

几何建模主要用于构建虚拟对象的形状、外观与结构等几何属性,得到虚拟对象的几何模型。常见的几何模型数据表现形式有点云(Point Cloud)、网格(Mesh)、体素(Voxel)。其中,点云是三维空间中数据点的集合,适用于描述虚拟对象或虚拟场景的几何外形,其中每个数据点都存储着三维空间的坐标;网格则是根据顶点、边和面的拓扑关系对虚拟对象进行编码获得的一种结构,常见的有完全由三角形面片组成的三角形网格和完全由四边形面片组成的结构网格等;体素是“体积像素”的简称,虚拟对象连续的几何外观被转换为一组最接近虚拟对象的离散晶格,每个体素就是这样的一个晶格,代表三维空间中均匀间隔的样本。

常见的几何建模技术有多边形建模、扫描建模、基于图像的建模等。其中,多边形建模是指利用多边形构造虚拟对象;扫描建模则是指使用三维扫描设备直接获取真实场景的信息,再利用三维重建算法等将构造的虚拟对象或场景存储为点云或体素;基于图像的建模则是指利用摄像机采集的离散图像或连续视频生成全景图像,再通过合适的算法把多幅全景图像组织为虚拟场景。

虚拟现实几何建模的主要任务包含形状建模、外观建模和结构建模,下面分别予以简要说明。

#### 1) 形状建模

要描述虚拟对象的形状,最基本的任务是利用点、线、面等基本几何元素表征虚拟对象

的外边界。目前最常用的形状建模方式可分为显式表示与隐式表示两种。其中,显式表示是指使用点云、网格、体素等结构去表征虚拟对象外边界的位置与拓扑结构信息;隐式表示则是指使用曲线与曲面的参数方程、距离场(Distance Field)、等值集(Level Set)等方法描述虚拟对象的外边界。

### 2) 外观建模

虚拟对象的外观是指虚拟对象独有的质地特征,如表面反射率、纹理等影响虚拟对象真实感的特征。如果不考虑存储和计算的开销,通过增加虚拟对象形状多边形的方法可以刻画出十分逼真的表面,但由于虚拟现实对计算和显示实时性要求高,实际应用中普遍采用纹理映射(Texture Mapping)等技术刻画虚拟对象的外观。采用纹理映射技术,一方面可增加细节层次以及虚拟对象的真实感,另一方面可以减少多边形的数量,从而在不影响实时性的同时,增强虚拟对象和场景的真实效果。

### 3) 结构建模

除了要模拟虚拟对象的形状和外观,很多时候还需描述虚拟对象的空间结构信息,以体现虚拟对象内部结构或虚拟对象之间的空间关系。以虚拟人体为例,可使用骨架结构表示各关节间的关系,并应用于人体动画、人体运动分析、不同人体的匹配等。对复杂的工业装备来说,可建立各组成部分(亦称部件)的结构关系,从而对不同部分进行控制,以满足虚拟装配、虚拟维护等要求。

## 2. 物理建模

除了上述几何属性外,虚拟现实建模还需表征虚拟对象的物理属性或物理过程,如重力、摩擦力、表面硬度、柔软度和变形等。这些物理属性与几何属性的建模结果相互融合,从而可以形成更具真实感的虚拟对象或环境。例如,当用户穿戴具备力反馈的手套抓握虚拟球时,如果虚拟球包含物理属性,那么用户除了能控制和观察自身的手指对球的抓握之外,还能逼真地感受到虚拟球的重量、硬软程度等物理信息。

物理建模(Physically-based Modeling, PBM)是虚拟现实较高层次的建模方式,通过有机地融入物理模型,从而将虚拟对象的物理属性得以体现。近年来,除了静态的物理属性,越来越多物理学中的动态过程,如人体运动解算、流体计算、燃烧计算等,都逐渐与虚拟现实相结合,在虚拟环境中实现了对于面部表情、织物,乃至爆炸等物质与现象的模拟。

物理建模所涉及的范围非常广泛,根据虚拟对象的不同种类,存在不同的建模方法。例如,对于人和动物等有机生物,可通过多关节刚体模型,实现人的行走与面部表情变化、鸟飞鱼游等;对于布料、弹簧等带有弹力阻尼特征的柔性体,可通过弹簧质点模型来重建其运动变化;对于烟雾、火焰等流体,可通过粒子模型来实现其演化过程。一般而言,根据虚拟对象的物理性质,可大致分为刚体建模、柔性体建模与流体建模。

### 1) 刚体建模

刚体建模适用于仅需考虑位置与方向的改变,而不考虑形变的虚拟对象,其涉及的内容主要包括刚体的运动、碰撞检测以及连接和约束等问题。

仅包含一个部件的刚体运动较为简单,可以采用牛顿第二运动定律等力学知识来解决。对于包含多个部件的刚体,如人体、手等,解决方法则相对较为复杂。以人体运动为例,可以采用关键帧方法、运动学方法和动力学方法等多种手段,后两者分别使用运动学和动力学方法建立有关人体运动的物理模型。与运动学方法相比,动力学方法需指定的参数较少,而且

对复杂运动过程的模拟更为逼真,但计算量较大,在运动控制的难度较大。在动力学方法中,解决运动控制问题的策略主要有两类:预处理策略,即首先将所需的约束和控制转换成适当的力和力矩,然后引入动力学方程;基于约束方程的策略,即将约束以方程形式给出,对约束方程个数与未知数相等的情况,采用一般的系数矩阵法快速求解,但对欠约束的情况,约束求解较为复杂。

刚体的碰撞检测主要针对刚体运动中的碰撞进行分析。为了加速计算,一般采用树结构对虚拟环境中的虚拟对象进行组织,通过空间剖分法或层次包围盒法建立树结构。空间剖分法的策略有均匀剖分、BSP树、KD树和八叉树等;层次包围盒法利用形状简单的包围盒将复杂的虚拟对象包裹起来,然后逐步进行碰撞检测的一种方法。包围盒的结构有层次包围球树、AABB(Axis-aligned Bounding Box)层次树、OBB(Oriented Bounding Box)层次树等。空间剖分法适用于分布比较稀疏均匀的几何对象间的碰撞检测,层次包围盒法则适用于复杂环境中的碰撞检测。

对于铰链类型的虚拟对象(如门窗、转动的机械臂)在多个约束情况下的关联运动问题,属于连接和约束建模的范畴。关联运动一般可分为前向关联运动和反向关联运动。对于前向关联运动,一般要在给定关联运动中每个关节的角度和长度的情况下,去求解关节末端所能到达的位置;反向关联运动则是在给定某个位置的条件下确定已知关节模型的可达性。

## 2) 柔性体建模

柔性体不同于刚体,在外力的作用下会产生形变,因此建模难度更大。柔性体建模主要关注的是动力学模型及其迭代求解方法。

柔性体建模中常用的动力学模型有连续体模型、弹簧质点模型等。其中,连续体模型使用本构模型描述不同材料的物理特性,如弹力、阻尼力等,再使用有限元方法模拟不同形变下的力或能量。此外,为得到更加理想的模拟效果,还可使用力传感器与视觉传感器等设备去采集实际材料的力与形变的关系信息,更加准确地描述虚拟对象的静态与动态特性。连续体模型主要用于虚拟现实对于布料、毛发、松软组织、人体器官、肌肉、面部表情的模拟。

弹簧质点模型将柔性体表面视为离散的质点与连接质点的弹簧组成的规则网格结构。质点间通过弹簧相连,受到弹簧弹力和阻尼力,遵循胡克定律,即当实际长度大于松弛长度时,弹簧将对两端的质点产生拉力;反之,在外力作用下变形后,弹簧会产生倾向恢复原样(松弛状态)的张力。根据不同连接方式与功能,又可将该模型中的弹簧分为结构弹簧、剪切弹簧与弯曲弹簧等,分别用以描述柔性体的拉伸、剪切与弯曲功能。弹簧质点模型主要使用显式积分法逐帧地计算下一帧的加速度、速度与位移等,从而实现形变的演进。弹簧质点模型主要用于布料、人体软组织等对象的建模。与连续体模型相比,弹簧质点模型计算复杂度较低,但在稳定性、触觉和视觉表现上稍有不及。

## 3) 流体建模

流体建模主要使用一些计算流体力学模型,如使用纳维-斯托克斯(Navier-Stokes, N-S)方程对流体的运动进行建模。N-S方程常见的迭代求解方法有基于网格的方法和无网格的方法。基于网格的方法主要使用拉格朗日网格或欧拉网格模拟流体在固定网格单元上的运动,是目前流体模拟的主要方法。但是,基于网格的方法容易产生网格畸变导致计算误差过

大。此外,该方法难以模拟大形变现象,如动态裂纹扩展、流固耦合等。无网格的方法通过使用一系列任意分布的节点(或粒子)来求解具有各种边界条件的 N-S 方程,节点或粒子之间不需要网格进行连接,因此不仅可以保证计算的精度,还可以降低计算的难度。光滑粒子流体动力学(Smoothed-Particle Hydrodynamics, SPH)就是一种有代表性的无网格方法。在虚拟现实,计算流体力学模型可用来对烟雾、燃烧、流水,以及云雨雾雪等自然现象进行建模。

### 3. 行为建模

虚拟现实中的行为建模主要研究虚拟环境中自治对象建模方法,如游戏中由计算机控制的角色,元宇宙中由人工智能生成的智能体等。在早期工作中,行为建模的研究集中在军事仿真领域,如 ModSAF、STOW、WARSIM 2000 等分布式虚拟战场环境中由计算机生成的兵力。随着虚拟现实研究与应用的发展,行为建模已拓展到公共安全、教育、文化娱乐等众多领域,如应急仿真规划系统(Emergency Simulation Program, ESP)等。近年来,随着数字孪生和元宇宙等概念的兴起,各类虚拟现实应用对自治对象行为的智能水平提出了越来越高的要求。这一领域属于与人工智能的交叉研究范畴,与生成式人工智能(Artificial Intelligence Generated Content, AIGC)及其应用都有密切的关联,虽然进展缓慢,但却是未来虚拟现实发展的重要关注点。

根据自治对象的类型,可以将行为建模分为个体建模与群体建模两类。其中,群体对象又包括聚合类对象和自治对象组织两类。个体对象是仅包含单一个体的自治对象。这类对象行为建模的内容一般包括建立目标任务的概念模型、体系结构、行为规则、所需条件等。聚合类对象包括多个个体,但可以使用多种解析度来表示。例如,既可以将其当作整体单一的对象,也可将其看作多个个体。根据具体的事件对群体的影响,可以采用不同的解析度。例如,在考查高温对人群行为的影响时,可以将人群看作一个整体来建模;但在考查火灾对人群行为的影响时,需要采用高解析度来建模不同的个体。聚合类对象行为建模主要内容包括计算模型、多解析度表现方法,以及聚合、解聚规则等。

自治对象组织的行为建模更加复杂。自治对象组织是若干独立存在个体的组织,其中每个个体进行自主决策,同时服从组织的控制。因此,自治对象的行为表现为个体行为和整体组织行为两个层面,不同层面对行为建模的要求不同。在个体方面,为支持个体与其他自治个体间的协调与交互控制,需要对社会行为的相关概念进行建模,并对组织内部行为进行分层,以支持个体行为和社会行为;在整体组织方面,需要对交互协议、社会规范等问题进行建模。

目前常用的行为建模方法主要包含基于有限状态自动机的建模方法、面向专家系统的建模方法、基于 Agent 的建模方法等。自治对象的简单反应性行为可以采用有限状态自动机进行建模。其中,自治对象每种可能的反应动作被表示为一个状态,发生的事件控制状态间的转换。采用有限状态自动机是军事仿真中常用的行为建模方法。面向专家系统的建模方法将自治对象看成一个近似的专家系统,将其行为建模看作知识的获取、表示和推理系统建立的过程,比较适合个体和群体的建模。对于确定性知识,采用基于逻辑、规则、框架的表示,以及相应的推理系统;对于不确定性知识,可采用模糊逻辑、神经网络、基于范例的推理和贝叶斯方法等。此外,还可使用强化学习等方法以提高自治对象的求解复杂问题的能力。基于 Agent 的建模方法将人工智能中关于 Agent 的研究引入虚拟现实中,适用于个体和群

体的行为建模。相较上述方法,基于 Agent 的建模方法一方面能够描述个体对象的自主性、自治性和智能性等特征,如建模个体对象的信念、意图、愿望等;另一方面,Agent 之间的通信、协商、协作等可以描述自治对象组织的协作特性。目前,基于 Agent 的行为建模已得到越来越多的应用。

3.1.2 虚拟现实建模的特点

虚拟现实强调沉浸感、逼真性,既要求有较高的真实感,强调自然的交互方式,又要满足建立在实时性基础上的交互性要求。换言之,虚拟现实要求在具有高真实感的环境中,产生沉浸感,并且可以满足实时性和交互性的要求。与传统计算机辅助设计和计算机动画不同,虚拟现实建模具有以下特点。

(1) 由于要实时操控和处理虚拟对象,建模方法与传统计算机辅助设计中以几何造型为主的建模有所不同。例如,在计算机辅助设计中,往往通过增加模型的几何复杂度来提高建模的准确度。但在虚拟现实,更倾向于使用纹理、层次细节等技术来提升虚拟对象的逼真度。

(2) 虚拟现实建模的内容相较于传统计算机图形学中的建模更为丰富,除了对虚拟对象的外形、表观、结构等信息进行表征外,还对虚拟对象的物理属性、行为属性等进行建模,从而为用户提供具有沉浸感的交互体验。

虚拟现实建模与其他建模技术的主要区别如表 3.1 所示。

表 3.1 虚拟现实建模与其他建模技术的主要区别

| 差异 | 虚拟现实建模                                       | 计算机辅助设计建模                                 | 计算机图形学建模                               |
|----|----------------------------------------------|-------------------------------------------|----------------------------------------|
| 特点 | 在综合考虑真实感、实时性和交互性的前提下,侧重交互性和实现意图;模型细节较少,实时性较高 | 侧重准确性,较少考虑实时性和交互性;模型细节较多,通常以牺牲实时性来获得较高的精度 | 侧重真实感的表达,较少考虑实时性和交互性;模型细节较多,渲染效果可以预先计算 |
| 用户 | 能够身临其境地与虚拟环境进行交互,无时间限制,可真实详尽地探索虚拟环境          | 能够交互,但不考虑沉浸感                              | 能够交互,但同样较少考虑沉浸感                        |
| 应用 | 主要用于需要对用户输入做出反应的仿真领域,如飞行训练、游戏和视景仿真等          | 主要用于工业制造与仿真计算领域,如机械零件设计、芯片电路仿真等           | 主要用于影视、流媒体、电子游戏等领域,以预先设计好的演示为主         |

可以看出,虚拟现实建模在确保实时性交互的基础上,极力提高整个虚拟环境的真实感,力图实现真实感与实时性的平衡。

3.1.3 虚拟现实建模评价指标

虚拟对象建模的质量将直接影响整个虚拟现实系统,因此需要了解虚拟现实建模的主要技术指标。常见的虚拟现实建模的评价指标主要包括以下几条。

(1) 真实感。真实感是度量用户对虚拟对象感知精确度的指标,包括但不限于视觉真实感、触觉真实感等。

(2) 实时性。虚拟现实应用要求虚拟对象或场景的显示帧率不低于某一阈值,否则会影响用户的视觉感知质量。

(3) 交互延迟。虚拟现实应用对交互延迟有较高的要求。响应时间太长会大大影响用户的体验。

(4) 易用性。虚拟现实应用强调用户的交互,因此所构建的虚拟对象在几何、物理以及行为等方面要尽可能易于交互,贴近真实场景中的表现。

本章的后续内容将重点围绕虚拟现实建模中的几何建模与物理建模,结合实际应用需求,介绍几个实例。

## 3.2 几何建模实例——点云简化

### 3.2.1 背景知识

如 3.1.1 节所述,目前常用的虚拟对象的几何模型数据表现形式有点云、网格和体素。其中,网格,尤其是三角形网格,具有强大的表面表达能力,而且 GPU 现在已普遍支持对三角形面片的几何处理。然而,三角形网格需要保存拓扑关系,在计算时需要维持拓扑一致性,因此在诸如变形等需要保持拓扑一致性的任务中计算复杂度较高,且具有一定的条件限制。随着三维扫描设备的快速发展,点云逐渐成为计算机图形学的一个研究热点。下面首先介绍点云的特点以及点云简化的概念。

#### 1. 点云的特点

与传统的网格相比,点云无拓扑结构,数据简单灵活。直接对点云进行处理具有如下优点。

(1) 3D 扫描仪通常能够输出大规模点云数据,如果可以直接对点云进行建模与渲染,可以避免网格建模带来的不精确性和人工开销。

(2) 由于不需要考虑拓扑一致性,点的添加和删除非常高效,因此点云适用于处理动态变化的物体。此外,在计算过程中不需要维护拓扑结构,从而简化了程序的数据结构,节约了存储空间。

(3) 当虚拟对象的几何复杂度和精度提升时,基于点的渲染方法效率要高于基于三角形面片的渲染方法。假设某一网格包含的三角形面片的数量已经超过了屏幕上像素的数量。此时,将一个三角形投影到屏幕上的区域已经小于一个像素点,这将导致逐个渲染每个三角形失去意义,从而降低三角形光栅化的效率。基于点的渲染方法可以避免无效的光栅化。因此,点云更适合表征复杂的、高精度的虚拟对象。

(4) 随着虚拟对象几何外形的复杂化,基于层次细节(Level of Detail, LOD)的表示变得十分重要。例如,在靠近视点时需要表征更多几何外形的细节,在远离视点时则可以适当简化一些细节。当构建虚拟对象的 LOD 时,由于点云不需要任何拓扑信息,也不需要存储和保持全局的拓扑一致性,使得层次构建更加简单。

#### 2. 点云简化的概念

点云的质量取决于采样密度,然而较大的采样密度往往导致虚拟对象的几何复杂度剧烈上升,不利于存储与处理。特别对于存储与计算资源有限的设备,如移动终端,大规模点云往往导致较高的交互延迟,降低了用户的视觉体验。因此,需要利用简化技术对点云进行优化。

目前针对点云的简化方法基本是衍生自网格的简化方法,如迭代法、聚类法以及粒子仿

真法等。其中,迭代法是指不断地从点云中去掉按某种标准计算的贡献值或误差值最小的点,直到误差值或贡献值达到阈值为止,从而得到一个简化子集结果。该方法类似于渐近网格的简化方法。这类方法较为简单高效,但是不能保证全局采样点的均匀分布。聚类法是指把输入点集按照一定规则进行聚类,划分成一些小的子集,这些小的子集不能超过给定的上限范围,如直径的大小、法向锥角的变化等。如果采用基于采样点层次结构的聚类简化方法,该方法会通过空间二分法将点集递归地进行划分聚类。也可通过主成分分析找到关键点,然后进行聚类,从而得到保持几何特征的简化结果。这类方法简单且快速,但是误差较大,而且没有优化策略,所以结果通常会包含很多多余的点。粒子仿真法是指通过在点上施加斥力使点的分布均匀化的方法。该方法首先在表面随机分布需要的粒子数,然后通过点的斥力移动粒子的位置直到达到平衡。该类方法能较好地控制采样密度,但是由于收敛速度较慢,处理大规模数据时效率较低。

上述三类简化方法存在的缺点在于,它们将点云中的点看成单纯几何意义上的点,且不能预先指定逼近误差,因此简化效果往往不能准确表示虚拟对象的结构,也难以收敛。基于此,业界提出了基于渲染图元的简化方法。这类方法在简化过程中考虑到点云的渲染图元的几何空间影响,并能用预先指定的全局误差进行控制,从而能够得出较好的简化效果。以常见的点的渲染图元 splat 为例,首先在指定全局误差下,生成每个采样点的最大 splat,然后采用贪心算法选择能覆盖整个表面的最小子集,最后通过全局优化算法使所有的 splat 均匀分布。基于渲染图元的方法相较上述三类方法能够取到较好的简化结果,但是计算复杂,时间开销较大。

下面具体介绍一种基于 splat 的点云简化算法,该方法将移动最小二乘法(Moving Least Squares,MLS)与基于 splat 的点云简化方法相结合,利用 MLS 优化 splat 最小子集的选择,进一步提升了点云简化的效率。

### 3.2.2 基于 MLS 和 splat 的点云简化算法

点云简化的目的是在指定的最大误差范围内得到最小的点集。因此,如何计算误差是非常重要的。如 3.2.1 节所述,基于 splat 的方法以 splat(具有方向的圆形表面或椭圆表面,颜色值从中心向周围逐渐减少)作为渲染图元。此时,在计算简化前后的点云误差时,直接使用原始点云采样后的点的子集,或使用 splat 邻域范围内的中心位置的点组成的子集,与简化后的点云进行比较,难以准确反映点云简化的质量。

基于 MLS 和 splat 的点云简化方法的核心思想是使用 MLS 投影算子计算 splat 的中心点,使得 splat 与其覆盖范围内的原始点云采样的点的子集之间的误差和最小。由此得到一个代表邻域的误差最小的 splat。通过上述方法获得的所有 splat 的中心点的集合,就是一个简化后的点云。

整个算法包括两个步骤。首先,为每个原始点云的采样点创建对应的 splat 集。包括对每个点计算其 MLS 投影点作为 splat 的中心点,根据结果计算每个对应的 splat 在误差范围内的最大覆盖面积。其次,通过贪心算法选择一组能够覆盖整个模型且数量最小的 splat 集合。选择标准是循环选取覆盖点数增值最大的 splat,直到选择的 splat 集合能够覆盖所有输入采样点集。算法流程如图 3.1 所示。

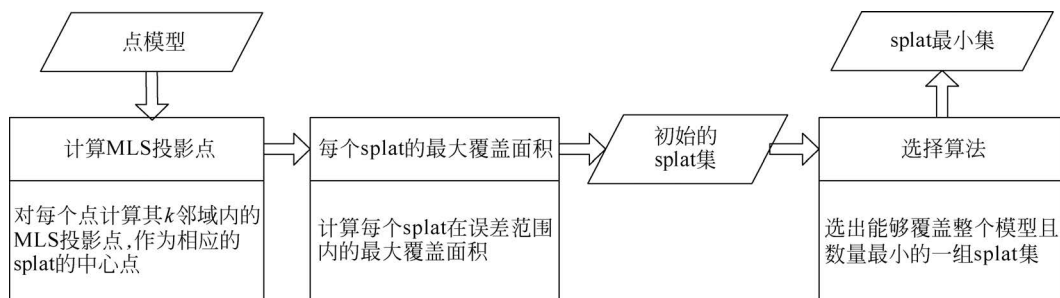


图 3.1 基于 MLS 和 splat 的点云简化算法流程

## 1. 算法实现

### 1) 创建 splat 集合

#### (1) MLS 投影计算 splat 中心点。

MLS 是根据邻域点进行局部多项式逼近的方法,MLS 投影表示将点投影到 MLS 逼近的表面上。对输入点进行 MLS 投影,可以得到代表邻域的误差最小的点,将其作为 splat 中心点,可以得到误差最小的代表邻域的 splat 面,从而得到分布更加合理的 splat 集。

假设待求点  $x$  的邻域点集为  $\{p_i | i=1, 2, \dots, k\}$ ,可使用下列公式来迭代评估投影位置:

$$p(x) = \frac{\sum_{i=1}^k \theta(\|p_i - x\|) p_i}{\sum_{i=1}^k \theta(\|p_i - x\|)} \quad (3.1)$$

其中,权值函数  $\theta$  使用高斯函数,如下所示:

$$\theta(d) = e^{-\frac{d^2}{\text{para}^2}} \quad (3.2)$$

其中,para 表示高斯参数,是一个固定值,在后续实验中,para 参数设为 0.3。

接下来,使用协方差分析来评估法线,其原理主要基于主成分分析。首先根据待求点  $x$  及其邻域点计算其协方差矩阵  $C$ ,公式如下所示:

$$C = \begin{bmatrix} (p_1 - P(x))\theta(\|p_1 - P(x)\|) \\ \vdots \\ (p_k - P(x))\theta(\|p_k - P(x)\|) \end{bmatrix}^T \begin{bmatrix} (p_1 - P(x))\theta(\|p_1 - P(x)\|) \\ \vdots \\ (p_k - P(x))\theta(\|p_k - P(x)\|) \end{bmatrix} \quad (3.3)$$

协方差矩阵表示了点邻域内采样点的分布情况,对其进行特征值分析可以评估其局部表面的属性。最小特征值  $\lambda_0$  对应的特征向量  $v_0$  可被认为逼近曲面的方向。不同于传统的协方差矩阵,这里考虑与距离相关的权值函数  $\theta$  的影响,即在协方差矩阵中考虑了 splat 从中心到边缘随着距离的增加影响逐渐降低的效果。

#### (2) splat 覆盖区域。

获得 splat 的法线和中心点之后,需要知道每个 splat 的覆盖面积,这取决于预先指定的全局最大误差阈值  $\epsilon$ 。如果用 splat 来代表其覆盖的邻域范围,则邻域点中某点  $p_i$  的误差即点  $p_i$  到 splat 在法线方向上的距离可以用下列公式求解:

$$\epsilon_i = |\langle (p_i - c), n \rangle| \quad (3.4)$$

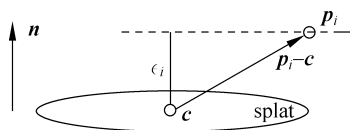
图 3.2 点  $p_i$  与 splat 之间的关系

图 3.2 展示了点  $p_i$  与 splat 之间的关系。  
 $\epsilon_i$  表示  $p_i$  到 splat 在法线方向上的距离,即最终的误差值, $c$  表示 splat 的中心点, $n$  表示 splat 的法线, $p_i$  是邻域点。

从近到远遍历 splat 中心点的邻域点,用上述方法计算每个邻域点的误差,如果小于全局最大误差阈值  $\epsilon$ ,则将其放入该 splat 的覆盖范围并继续遍历,否则停止。最后得到在全局最大误差阈值  $\epsilon$  范围内的每个 splat 的覆盖范围。

在该算法中,需要预先知道邻域关系。这里通过对输入点构建一个 KD 树来获取点模型之间的邻域关系。算法如果遍历到 splat 中心点的最后一个邻域点时,其误差仍然小于阈值,则需要扩大邻域范围来继续遍历计算,直到达到或超过阈值。

## 2) splat 最小集选择算法

选择算法的目的是获取能覆盖整个表面的 splat 最小集。具体地,通过贪心算法不断地选取覆盖点数增值最大的 splat,直到选择的 splat 子集能够覆盖所有的输入采样点。

图 3.3 展示了 splat 最小集选择算法的具体更新过程。

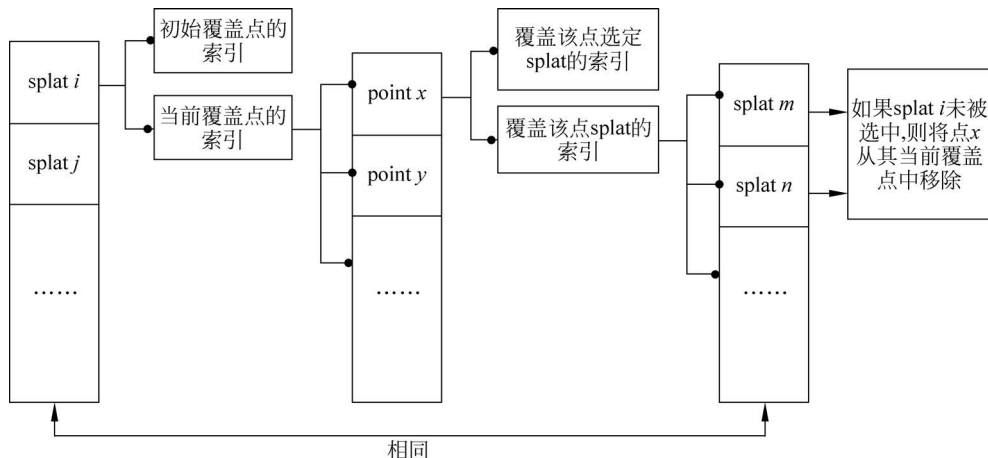


图 3.3 splat 最小集选择算法的具体更新过程

首先,在初始 splat 集中选出一个覆盖点数最多的 splat,加入已选 splat 队列,然后更新未选 splat 的参数,包括当前覆盖的点数,如图 3.3 所示,当 splat  $i$  被选取时,需要找到所有与 splat  $i$  交叉的未选 splat,然后从这些 splat 的当前覆盖点集中删除 splat  $i$  覆盖的点。

接下来,选择覆盖点数增加最多的 splat,即在未选 splat 队列中选择一个当前覆盖点数最多的 splat 加入已选 splat 队列。然后依照同样的方法更新未选 splat 的参数。

重复上述选择过程,直到选择的 splat 集覆盖了所有的采样点。需要注意的是,由于在更新过程中不断地删除未选的 splat 所覆盖的点,因此会出现有些 splat 覆盖的点集为空的情况。这时直接将该 splat 从未选 splat 队列中移除,减少冗余的遍历,进一步提升计算效率。

## 2. 实验结果

下面展示算法的结果。实验平台是配置为 Pentium4 3.0GHz CPU 与 1GB 内存的机器。

首先,使用少量的点验证 MLS 投影的效果,实验结果如图 3.4 所示。其中,绿色点表示点  $r$  (圆点)的最近 20 个邻域点,  $P_r$  (方点)是点  $r$  的 MLS 投影点。

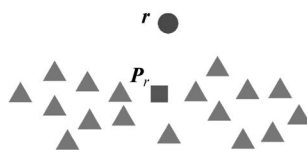


图 3.4 MLS 投影

从图 3.4 可以看出,MLS 投影点  $P_r$  比点  $r$  更适合用来代表  $r$  的邻域。

接着,在不同几何复杂度的点云上进行了简化实验,结果如表 3.2 所示。可以看出,算法可以有效地简化模型,且计算时间较短。

表 3.2 在不同点模型上使用不同的误差阈值  $\epsilon$  进行简化

| 模 型       | 输 入 点 数 | $\epsilon$ | 简化后的点数 | 简化时间(s) |
|-----------|---------|------------|--------|---------|
| bunny     | 35 945  | 0.2        | 477    | 18      |
|           |         | 0.1        | 864    | 13      |
|           |         | 0.05       | 1559   | 11      |
|           |         | 0.02       | 3150   | 10      |
|           |         | 0.01       | 4616   | 6       |
| santa     | 75 781  | 0.1        | 638    | 76      |
|           |         | 0.02       | 2525   | 40      |
| horse     | 100 000 | 0.1        | 890    | 111     |
|           |         | 0.05       | 1695   | 74      |
|           |         | 0.02       | 3459   | 49      |
|           |         | 0.01       | 5406   | 43      |
| igea      | 134 345 | 0.05       | 1082   | 329     |
|           |         | 0.02       | 2950   | 122     |
|           |         | 0.01       | 5494   | 136     |
| armadillo | 172 974 | 0.2        | 9737   | 215     |
|           |         | 0.1        | 16 729 | 223     |
|           |         | 0.05       | 25 764 | 229     |
| dragon    | 437 645 | 0.01       | 22 576 | 831     |

图 3.5 展示了简化后的点云的渲染结果。其中,图 3.5(a)至图 3.5(d)展示的是原始虚拟对象,图 3.5(e)至图 3.5(h)展示的是使用算法简化后的虚拟对象。可以看出,经过算法简化的点云在点数减少的同时能够保持原始虚拟对象的几何特性。

与此同时,使用点数规模更大的点云进行了实验,结果如图 3.6 所示。

其中,图 3.6(a)和图 3.6(b)是 armadillo 模型,图 3.6(c)和图 3.6(d)是 dragon 模型。可以看出,算法对于大规模点云同样具有良好的简化效果。

最后,图 3.7 展示了不同的全局最大误差阈值  $\epsilon$  对点云简化的渲染效果的影响。

其中,图 3.7(a)至图 3.7(c)为基于 splat 的渲染,图 3.7(d)至图 3.7(f)为基于球的渲染,每个球的半径等于 splat 的半径。从图 3.7 可以看出,所选择的 splat 子集在不同全局最大误差下均能有效覆盖虚拟对象的表面。



图 3.5 点云模型及简化结果



图 3.6 大规模点云及简化结果

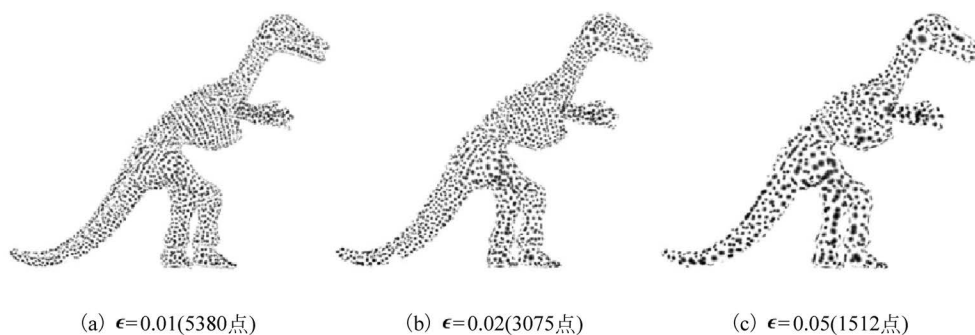


图 3.7 不同误差阈值下简化结果的各种渲染效果

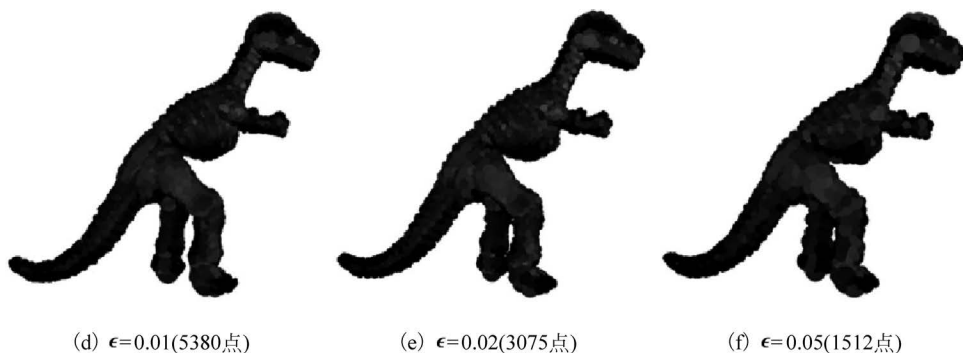


图 3.7 (续)

### 3.3 物理建模实例——虚拟人体的运动合成

#### 3.3.1 背景知识

通过几何建模,虚拟人体具备了空间上的展示与交互能力,再经过物理建模后,进一步具备与虚拟环境实时交互的能力。典型的虚拟人体物理模型主要包含两方面的内容,分别是虚拟人体的静态表示与运动制作,下面分别予以简要说明。

##### 1. 虚拟人体的静态表示

虚拟人体的静态表示又可进一步分为人体骨架系统的构建与关节运动描述两方面。

##### 1) 人体骨架系统构建

在人体骨架系统构建方面,目前业界有两种通用标准,分别是 VRML 使用的 H-Anim 标准与 MPEG 提出的虚拟人体动画标准。

H-Anim 标准是一种用于描绘动画的 3D 虚拟人体模型标准。该标准规定虚拟人体由 H-Anim Humanoid、H-Anim Joint、H-Anim Segment、H-Anim Site、H-Anim Displacer 节点组成,其中 H-Anim Humanoid 节点是其他节点的容器。H-Anim Joint 节点用于描绘虚拟人体模型的关节,所有的关节按层组织成具有父子关系的骨架树,该骨架树就是虚拟人体的骨架系统,虚拟人体的姿态就是该骨架系统决定的。H-Anim Segment 节点用于连接两个 H-Anim Joint 节点,它表示虚拟人体的骨骼。H-Anim Site 节点作为反向运动学(Inverse Kinematics,IK)系统的终止受动器,也被称为末端控制器,作为虚拟人体姿态变换的约束,表示运动传到某个极限位置就停止。H-Anim Displacer 节点的目的则是将一系列人体动作组合起来,形成一个复杂的动作。对应到人体,通常将脊柱末端的骨关节定义为 H-Anim Humanoid 并作为骨架模型的根,将人体关节和骨骼段定义为不同的 H-Anim Joint 和 H-Anim Segment,由此得到一个层次化的模型。

MPEG 的虚拟人体动画标准与之不同,在 MPEG 中,虚拟人体模型由一组节点组成。其中,顶层节点(BodyNode)至少包括两个子节点:人体的运动参数(BAP)和表示人体模型定义的参数(BDP)。人体运动参数包含 296 个描述虚拟人体骨架属性的参数,这些参数可被应用于 H-Anim 兼容的虚拟人体几何模型,并生成相同的虚拟人体运动。

## 2) 关节运动描述

在关节运动描述方面,通常会将虚拟人体看作一种具有多个关节的刚体,因为在角色运动过程中,每个关节并不会发生形变。由此,通过定义每个关节的自由度即可完全确定关节的运动。此外,由于骨架系统本身的层次化特性,除根节点外,其余每个关节可根据各自的自由度,在以父关节为坐标原点的局部坐标系内旋转。例如,股骨关节具有 3 个自由度,表示可在 3 个方向上相对于它的父节点(髋关节)进行旋转。

常见的表示旋转自由度的方法有旋转矩阵、欧拉角、四元数等。其中,旋转矩阵分别建模  $x$ 、 $y$ 、 $z$  方向的旋转矩阵,然后进行复合。这种方法需要使用 9 个数表示旋转的 3 个自由度,用作虚拟人体关节自由度的表示略显冗余,因此一般只用于中间计算过程。对于在三维空间里的一个参考系,任何坐标系的取向,都可以用 3 个欧拉角来表现。参考系又称为实验室参考系,是静止不动的。而坐标系则固定于刚体,随着刚体的旋转而旋转。因为用欧拉角描述的自由度为绕坐标系中 3 个轴旋转的方向值,所以大部分运动捕获设备采集的数据都以欧拉角的形式给出,根据运动编辑目标改变欧拉角的数值能够确定得到一个旋转,这是欧拉角最大的优点。然而,欧拉角本身也有缺点,主要有以下 3 点。其一,一般欧拉角描述的人体的旋转都是按照坐标系的某个特定的次序进行的,或者是  $xyz$ ,或者是  $zyx$ 。其二,用欧拉角定义的旋转有 12 种不同的方式,每种方式得到的复合旋转矩阵都不同,这就要求对人体旋转方向的确切了解,否则有可能得到不同的旋转,导致错误的结果。其三,“万向节死锁”(Gimbal Lock)现象,即如果把  $y$  轴的值指定为  $\frac{\pi}{2}$  时,会发现绕  $z$  轴旋转与绕  $x$  轴旋转得到的结果会较为近似,此时若对欧拉角插值可能会得到一个毫无意义的结果,不符合人体运动范围的规定。四元数可表示矢量和物体的旋转,并且冗余信息少,它提供了一种比旋转矩阵更为有效的方法。四元数的优点是它的很多基本操作如乘法、倒置等运算都直接对应到相应的旋转操作。而且,四元数的插值计算,即球面线性插值(Spherical linear interpolation, Slerp)使用起来极其方便。四元数的缺点在于其 4 个数和人体关节自由度并没有直接联系,用户无法直观理解它。上述自由度的三种表示方法之间可以相互转换,在此不加赘述。

为了方便对虚拟人体骨架模型的运动进行描述,一般规定两种坐标系:一种是用于描述根节点位置和朝向的世界坐标系(也称为绝对坐标系),另一种是关节的局部坐标系(也称相对坐标系)。在这里,以手臂为例说明人体各关节之间的相对运动,如图 3.8 所示。

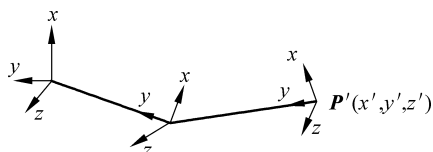


图 3.8 手臂刚体结构

由关节之间相对运动和关节的结构可知,手臂末端关节相对世界坐标系的坐标  $P(x, y, z)$  可表示为:

$$P = MT_n^0 P' \quad (3.5)$$

其中,  $P'(x', y', z')$  为末端关节在自身的局部坐标系中的坐标,  $M$  为根关节到世界坐标系的变换矩阵。  $T_n^0$  为第  $n$  个相对坐标系到根坐标系的变换矩阵,它

可表示为:

$$T_n^0 = T_1^0 T_2^1 \cdots T_n^{n-1} \quad (3.6)$$

通过对任意两个坐标系之间关系的计算,可以方便地得到人体的相对运动。

## 2. 虚拟人体的运动制作

当前,业界对于虚拟人体实时运动的制作方法主要有以下4种。

### 1) 关键帧方法

在三维计算机动画中,所谓的关键帧是指动画师设计的系列关键画面,中间帧则由计算机通过插值以及各种平滑操作来生成。对于虚拟人体而言,关键帧方法是指利用关键帧表示虚拟人体各个关节的位置、姿态以及对应的时间等信息,然后利用算法在关键帧之间进行插值以及平滑操作生成连续的运动序列。

通过关键帧技术生成的虚拟人体运动的质量和关键帧的质量有很大的关系,而关键帧的制作水平则取决于设计者对于人体运动知识的精通程度以及个人的软件掌握程度。对于较为简单的对象来说,可以采用直接的运动学方法定义关键帧。然而,由于人体的关节众多,关节自由度的总数也比较多,同时需要每秒能提供几十帧动画数据才能获得流畅的视觉效果,因此即使采用了反向运动学算法,关键帧制作的工作量还是比较大的,因此该方法更多应用于关节较少的应用情境中,特别是卡通风格的虚拟人体动画。

### 2) 运动学方法

运动学是力学的一门分支学科,专门用来描述物体的运动规律,即物体在空间中的位置随时间的演进而进行的改变,完全不涉及作用力或质量等因素。在运动学方法中,通常利用方程来显式地表示虚拟人体的运动和时间之间的函数关系,将定义的高层运动参数转换成虚拟人体的低层骨架运动参数。计算机通过改变少量的高层运动参数自动生成人体运动。运动学方法一般以生物力学原理为计算基础,采用正向或反向运动学方法,计算过程不必一定满足物理规律,只需要将运动学参数(比如虚拟人体各部分的位置、速度和加速度等)赋予虚拟人体相关的骨骼关节即可。

采用运动学方法生成虚拟人体的运动,由于需要用户手动地给出各个关节的局部坐标系以及关节的自由度值等信息,所以对于复杂的多刚体结构来说非常烦琐,并且生成出的运动数据掺杂了太多的人为因素,显得比较生硬。因此,运动学方法通常只用于已知动作捕捉数据的情况。

### 3) 动力学方法

动力学方法与运动学方法不同,它基于虚拟人体骨骼的质量分布以及施加于每个关节上的力矩,根据物理方法(牛顿运动定律等)计算得到身体各部分的位移数据,最终生成运动。设计驱动虚拟人体各个关节运动的比例微分控制器(Proportional Integral Derivative Controller, PIDC)是动力学方法的关键和难点。由于采用动力学方法计算得到的虚拟人体运动满足自然世界中的物理规律,因此相对于运动学方法来说,能够生成相对更为真实的虚拟人体动画。

动力学方法的主要优点是完全不需要动作捕捉数据就可以生成比较真实的运动,缺点是需要手动地调整很多的参数,并且在自然度方面远不及运动学方法。

### 4) 利用动作捕捉设备直接制作

动作捕捉是一种高度逼真的虚拟人体运动生成方法。该方法通过运动数据的获取系统详细地记录表演者每个关节的位置和旋转数值,这些数值经过处理后再赋值到虚拟人体上,可以逼真地还原出表演者的运动。利用动作捕捉设备虽然可以实时捕捉到高度逼真的人体运动,然而这种方法也有其自身的缺点:首先,人必须要穿上相对厚重和不甚方便的动作捕

捉设备；其次，在对虚拟人体进行控制时，虚拟人体必须要和实际穿着动作捕捉服装的表演者有着相同的骨架结构以及相似的体型；第三，尽管表演者可以依照虚拟环境的约束条件做出相应的交互动作，但是由于现实环境与虚拟环境不可避免地会出现差异，所以动作捕捉设备生成的动作通常并不能够使虚拟人体顺利地虚拟环境进行交互，所以在应用中直接使用运动捕捉设备对虚拟人体进行控制并不现实，业界现在多用动作捕捉设备获取数据，然后将这些数据与其他方法相结合来实现具体的运动控制。

动作捕捉设备是由陀螺仪、加速度传感器以及角速度传感器等多种传感器一起构成的综合传感控制器设备，在捕捉人体运动的时候，这种综合传感器被安装在人体的重要关节（如手腕、脚腕、膝关节）上，以便在运动过程中能够实时计算出人体的运动姿态，进而捕捉到当前人体运动的所有细节并以一定的格式存储。使用 Xsens 进行动作捕捉的实例如图 3.9 所示。

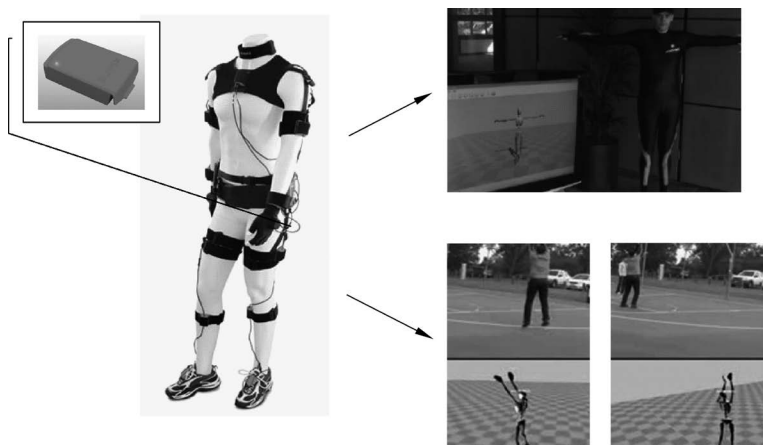


图 3.9 Xsens 公司的产品进行运动控制

上述每种运动制作方法各有其优缺点，在实际应用中，通常结合一种或多种方法来控制虚拟人体的运动，从而提升虚拟人体运动的稳健性和自然度。由于动作捕捉数据捕捉了人体运动的所有细节，并且从某种程度上体现了人类复杂的情感，所以理所当然地成为当前制作虚拟人体运动的主要方法之一。

下面将介绍一种基于动作图的虚拟人体运动合成方法。该方法首先构建虚拟人体的物理模型，然后通过基于动作图的交互式运动合成方法实现对虚拟人体的运动控制。

### 3.3.2 基于动作图的虚拟人体运动合成

所谓运动合成，是指对无序的动作捕捉数据进行组织，根据需求对其中的若干数据序列进行某种方式的连接与融合，从而得到新的运动数据序列。动作图是一种基于图的运动片段组织方式，本质上是由节点和边组成的有向图，其中的节点对应于静态的人体骨架系统（也可称为人体姿态），而边则对应于连接姿态的运动片段。用户通过搜索动作图就能合成出逼真的运动序列。

现有的动作图可分为两类：一类是基于搜索的动作图，即按照用户的要求在动作图中进行搜索，选取与用户要求最相近的运动片段；另一类是基于融合的动作图，即对图中的动作捕捉数据进行融合。前者主要对原始动作捕捉数据进行连接，其合成运动的能力直接依

赖于动作捕捉数据；后者则利用加权平均等方式对原始动作捕捉数据进行加工，从而在一定程度上增强了合成运动的表示能力。然而，基于融合的方法合成运动的表示能力有限，用户的控制精度不高。此外，目前融合方法对根节点位置的计算，只是样本数据中根关节位置的简单加权平均，合成的运动序列的逼真性将会受到很大的影响，容易产生脚步滑动或者根关节朝向抖动的问题。

本节介绍的基于动作图的交互式运动合成方法，首先使用一种基于特征的人体动作捕捉数据自动分割方法将原始数据进行分割，然后基于分割的数据片段构建动作图。最后，在实时运动控制过程中，利用动作图生成符合环境约束的运动，用户可以实时改变运动轨迹，控制虚拟人体沿着指定的轨迹运动。在控制虚拟人体沿着新轨迹运动时，采用对路径曲线弧长参数化的方法，获得原始运动在目标轨迹上的位置和朝向，从而将虚拟人体重定位到目标轨迹曲线上，完成路径合成部分。通过本节介绍的算法，可以实时地控制虚拟人体在指定路径任意长度的运动，算法整体流程如图 3.10 所示。

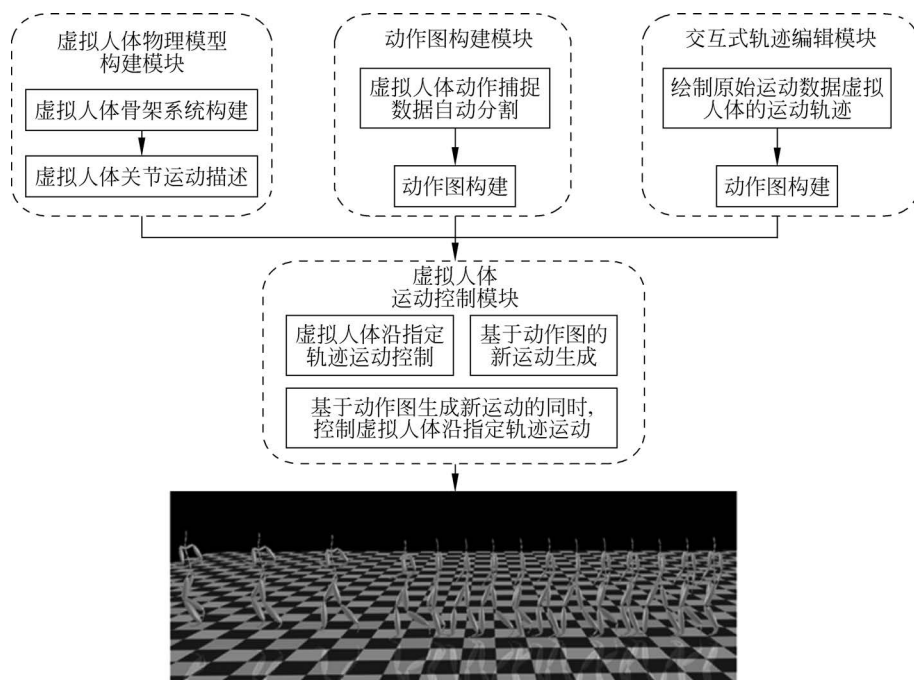


图 3.10 基于动作图的交互式运动合成算法的结构图

下面介绍具体算法实现过程以及相应的实验结果。

## 1. 算法实现

### 1) 虚拟人体物理模型构建

如前所述，虚拟人体的运动数据通常由动作捕捉设备获取。动作捕捉设备通常是由陀螺仪、加速度传感器以及角速度传感器等多种传感器一起构成的综合传感控制器设备，在捕捉人体动作时，这种综合传感器被安装在人体的重要关节（如手腕、脚腕、膝关节）上，以便在人体运动过程中实时计算出人体的运动姿态，进而捕捉到当前人体运动的所有细节。通常动作捕捉数据文件有 HTR、BVH 以及 ASF/AMC 这 3 种格式，每种格式的动作捕捉数据都采用层次化的运动描述方法记录人体的运动。这里重点介绍 ASF/AMC 格式的动作捕

捉数据文件。

ASF/AMC 格式的动作捕捉数据文件主要包含两部分。其中,ASF 格式文件记录人体骨架数据,AMC 格式文件记录运动数据。ASF 格式文件给出了人体骨架模型,并设定了初始姿态。ASF 格式文件包括捕获系统说明、度量单位、文档描述、根节点信息、关节信息、关节之间的层次结构 6 部分。AMC 格式文件与 ASF 格式文件的内容相对应,以帧为单位记录了每一帧中根关节的平移和旋转向量,以及其他各个关节在各个自由度方向上的旋转,在 ASF 文件中,旋转以欧拉角的形式给出。

根据 H-Anim 标准以及 ASF/AMC 格式数据文件的分析,首先根据 ASF 格式数据文件构建出虚拟人体的骨架模型,如图 3.11 所示。

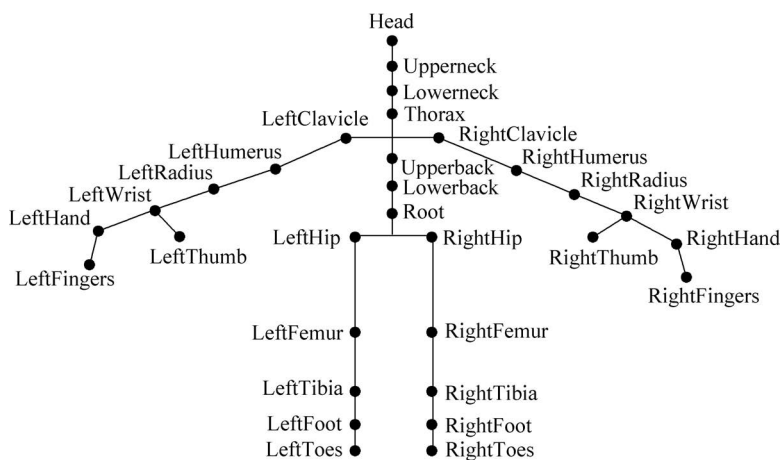


图 3.11 虚拟人体的骨架模型

由图 3.11 可见,所构建的骨架模型包含 31 个关节点,构成了树形的组织形式。整个人体骨架模型有一个根节点 Root,从根节点开始向下延伸到各个关节点,形成各个层次的人体骨骼子树。根节点有 6 个自由度,分别是 3 个方向的平移自由度和 3 个方向的旋转自由度,3 个方向的平移量决定了该骨骼树所表示的人体姿态的位置,而 3 个方向的旋转量决定了该骨骼树所表示的人体姿态的朝向。其余各个子关节均有 1~3 个自由度,表示在其父关节为坐标原点的局部坐标系下可能的旋转状态。此外,ASF 格式文件结构还给出了虚拟人体的初始姿态以及各关节的局部坐标系信息。接下来先从骨架模型构建出虚拟人体的层次树,用于后续算法实现,如图 3.12 所示。

由图 3.12 可见,骨架模型中的每个关节点都成为层次树的一个节点。其中,每个节点存储着关节的自由度以及相应的运动描述信息。

## 2) 动作图构建

对动作捕捉数据进行合理分割是构建动作图的前提和基础。与普通的运动数据分割方法不同,用于构建动作图的分割方法需要考虑两方面的特殊要求:一方面,运动片段的长度越短,对运动合成的控制越灵活;另一方面,为了方便用户控制,运动片段中应当包含明确的语义信息。针对这些特殊要求,这里采用了一种基于特征的动作捕捉数据分割方法,使用该方法得到的运动数据片段既包含明确的语义信息又足够灵活;同时,该方法实现了动作捕捉数据的自动分割,避免了烦琐的手工操作,提高了动作捕捉数据的分割效率、精度和稳

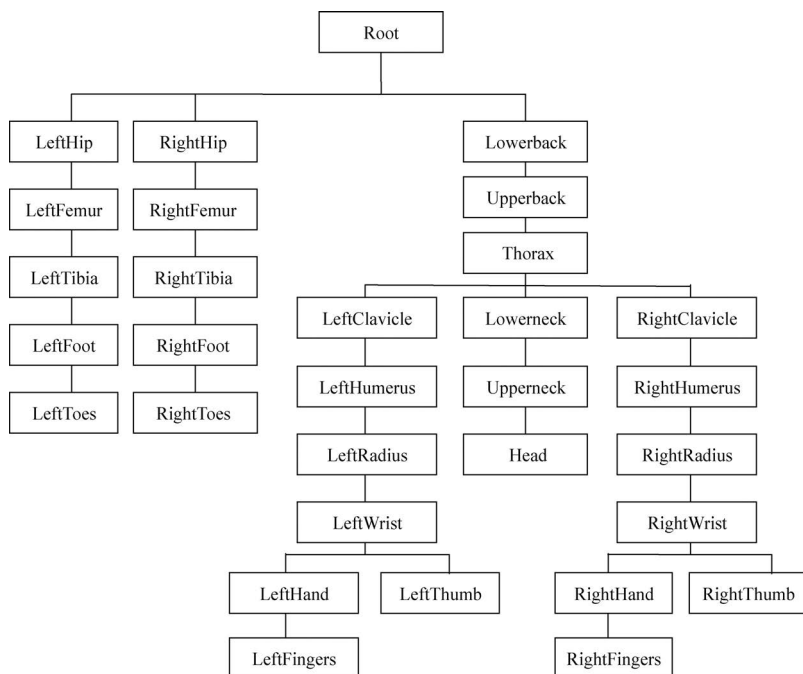


图 3.12 虚拟人体的层次树模型

健性。本节考虑的运动数据仅包括双足运动。

在动作捕捉数据中,人与环境接触的关键姿态,如双足运动中脚接触地面,体现了该运动的运动特征。基于运动特征,可以给出具有明确语义信息的运动类别。本节提出的基于特征的运动捕获数据自动分割法所依据的特征是双足运动的周期性。

在自动分割之前,首先对原始运动数据进行平滑,因为原始的动作捕捉数据因为种种原因可能存在很多不稳定的因素(也被称为噪声),必须对其进行平滑过滤。

本节方法采取的数据平滑算法如下式所示:

$$\begin{aligned} \text{data}_{\text{frame}} = & d_7 \times \frac{1}{28} + d_6 \times \frac{2}{28} + d_5 \times \frac{3}{28} + d_4 \times \frac{4}{28} + \\ & d_3 \times \frac{5}{28} + d_2 \times \frac{6}{28} + d_1 \times \frac{7}{28} \end{aligned} \quad (3.7)$$

其中,  $\text{data}_{\text{frame}}$  为指定帧的运动数据,  $d_{\text{spread}}$  为插值的数据,其表示如下式所示。frame 表示当前修改的数据在数据中处于第几帧。spread 为跨度值,  $\text{spread}=1, 2, \dots, 7$ :

$$d_{\text{spread}} = \frac{1}{2}(\text{data}_{\text{lowIndex}} + \text{data}_{\text{highIndex}}) \quad (3.8)$$

在公式(3.8)中, lowIndex 和 highIndex 分别指定了对当前帧的数据进行平滑操作使用的窗口大小。其中 lowIndex 为计算  $d_{\text{spread}}$  时使用的第一个数据的索引,且满足下式:

$$\text{lowIndex} = \begin{cases} \text{frame} - \text{spread}, & \text{若} (\text{frame} - \text{spread}) \geq 0 \\ 0, & \text{若} (\text{frame} - \text{spread}) < 0 \end{cases} \quad (3.9)$$

highIndex 为计算  $d_{\text{spread}}$  时使用的第二个数据的索引,且满足下式:

$$\text{highIndex} = \begin{cases} \text{frame} + \text{spread}, & \text{若} (\text{frame} + \text{spread}) < \text{totalframe} \\ \text{totalframe}, & \text{若} (\text{frame} + \text{spread}) \geq 0 \end{cases} \quad (3.10)$$

其中, totalframe 为数据的总帧数。

在对运动数据进行平滑处理之后,下面对运动数据进行自动分割。不失一般性地,假定分析的运动是走路,那么对于跑步等其他双足运动的自动分割也可以以此类推。其中,单位步长的运动数据的自动分割步骤如下。

首先,计算一只脚(左脚或者右脚)在所有运动数据帧中的全局坐标系下的绝对位置,取脚在高度方向的坐标值;接着,寻找所选脚的第一个关键帧,即开始迈第一步的帧数,选择依据是梯度由负变为正并且当前全局位置在全局最低点位置的 10% 邻域内;然后,找到所选脚的第二个关键帧,即第一步的最高点的帧数,选择依据是梯度由正变为负并且当前全局位置在全局最高点位置的 10% 邻域;最后,寻找所选脚的第三个关键帧,即第一步的结束帧,选择依据同第一个关键帧的选取方法。

依照上述步骤便可从特定运动数据获得单位步长的数据,具体的步长分割如图 3.13 所示。

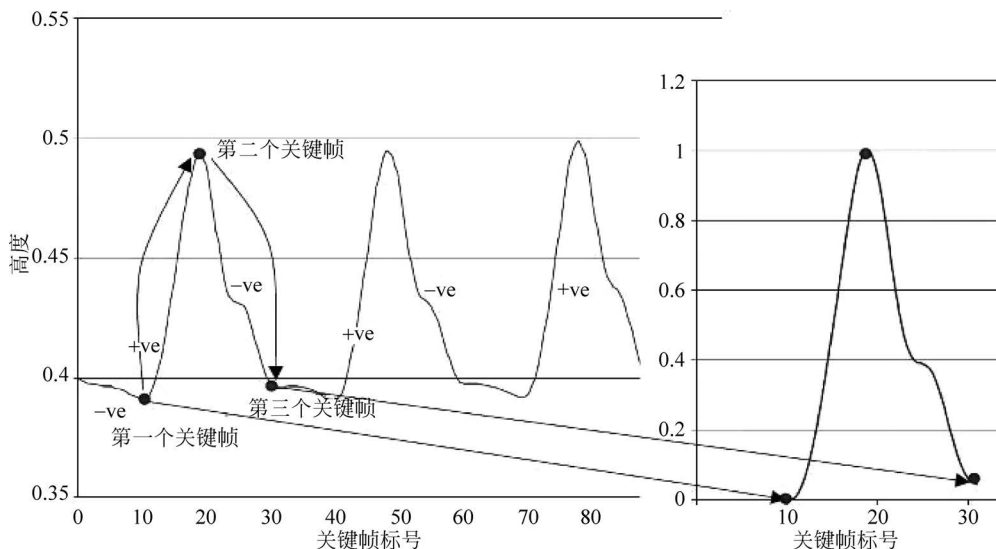


图 3.13 运动数据的步长分割

在此基础上,可实现对连续的双足运动数据进行自动分割,得到若干单位步长的数据,将其作为运动数据片段来构建动作图。

动作图的构建过程主要包含 4 个步骤。

(1) 计算帧间距离。设运动数据片段的个数为  $n$ , 对于任何一个运动数据片段  $i < n$ , 分别计算该运动数据片段的每一帧与其他运动数据片段(记为  $j$ , 满足  $j \neq i$  并且  $j < n$ ) 每一帧之间的距离。若将数据片段  $i$  中的帧表示为  $A_i$ , 将运动数据片段  $j$  中的帧表示为  $B_j$ ,  $D(A_i, B_j)$  表示两帧的距离。在计算距离时, 考虑指定大小为  $k$  的两个窗口内骨架系统的距离, 前者的窗口起始帧为  $A_i$ , 后者窗口的结束帧为  $B_j$ 。  $D(A_i, B_j)$  可以通过计算两个骨架系统对应点  $p_i$  和  $p'_j$  距离的加权平方和来求得。

(2) 选择过渡点。在计算出相应帧之间的距离之后, 给定一个阈值, 然后将所有距离小

于给定阈值的两个帧作为候选的过渡帧。然而,需要注意的是,距离函数的局部最小值并不意味着可以在两个运动片段之间生成高质量的过渡运动数据,而只意味着其相对其邻居来说比较理想。所以阈值的选取是一个比较关键的因素,通常根据经验值选取。

(3) 计算过渡运动数据片段。选定了满足给定阈值的过渡帧号之后,下一个步骤是根据过渡帧号计算两个运动数据片段的过渡数据。假设  $D(A_i, B_j)$  小于给定的阈值,则通过对第  $A_i$  帧到第  $A_{i+k-1}$  帧之间的数据和第  $B_{j-k+1}$  帧到第  $B_j$  帧之间的数据进行融合。这里  $k$  是位于过渡区域的帧数,可以根据实际需要自行设置。融合的第一步是将两个运动数据的根节点的坐标原点和坐标系统一到一个坐标系下,第二步是计算过渡数据的每一帧。具体地,在计算第  $p$  帧 ( $0 \leq p < k$ ) 时,利用线性插值计算根节点的位置,利用球面线性插值计算关节旋转信息。

(4) 动作图剪枝。经过第三个步骤以后,动作图大致上就构建起来了,但有可能在图中存在一种特殊的顶点,它不存在于任何的环路中,不能作为动作图的顶点,所以需要将这种“死点”去掉。

### 3) 交互式轨迹编辑

基于所构建的动作图,可使用交互式的轨迹编辑来控制虚拟人体的运动合成,从而使之按照预设的轨迹进行平滑运动。具体地,使用 Kochanek-Bartels 样条函数对虚拟人体的轨迹进行拟合。Kochanek-Bartels 样条曲线的设计本身就是为了模拟动画轨迹,特别是当对象运动发生改变时,可以通过为参数取非零值而进行模拟。这里的运动发生改变,是指一个卡通角色突然停止运动、改变方向,或者与另一个对象碰撞。下面介绍 Kochanek-Bartels 样条曲线的形式。

Kochanek-Bartels 样条曲线是 Cardinal 样条曲线的扩展。Cardinal 样条曲线是插值分段三次曲线,并且每条曲线段的端点位置均指定切线。一个 Cardinal 样条曲线完全由 4 个连续控制点给出。中间两个控制点是曲线段的端点,其他两个点用于计算曲线斜率。在 Cardinal 样条曲线中,一个控制点的斜率值可以由两个相邻的控制点的坐标进行计算,如图 3.14 所示。

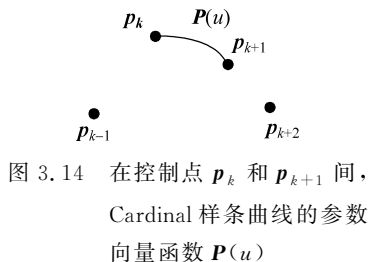


图 3.14 在控制点  $p_k$  和  $p_{k+1}$  间, Cardinal 样条曲线的参数向量函数  $P(u)$

假设  $P(u)$  是两个控制点  $p_k$  和  $p_{k+1}$  间的参数三次函数式,则从  $p_{k-1}$  到  $p_{k+1}$  间的 4 个控制点用于建立 Cardinal 样条段的边界条件如下式所示:

$$P(0) = p_k \quad (3.11)$$

$$P(1) = p_{k+1} \quad (3.12)$$

$$P(0)' = \frac{1}{2}(1-t)(p_{k+1} - p_{k-1}) \quad (3.13)$$

$$P(1)' = \frac{1}{2}(1-t)(p_{k+2} - p_k) \quad (3.14)$$

其中  $P(\cdot)'$  表示该点处的斜率。参数  $t$  称为张量参数,它的作用是控制 Cardinal 样条曲线与输入控制点的松紧程度。图 3.15 展示了  $t$  取很小值和很大值时 Cardinal 样条曲线的形状。

对于 Kochanek-Bartels 样条曲线,它的边界条件被定义如下式所示:

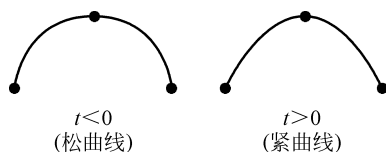


图 3.15 张量参数在 Cardinal 样条曲线形状中起到的作用

$$\mathbf{P}(0) = \mathbf{p}_k \quad (3.15)$$

$$\mathbf{P}(1) = \mathbf{p}_{k+1} \quad (3.16)$$

$$\mathbf{P}(0)' = \frac{1}{2}(1-t)[(1+b)(1-c)(\mathbf{p}_k - \mathbf{p}_{k-1}) + (1-b)(1+c)(\mathbf{p}_{k+1} - \mathbf{p}_k)] \quad (3.17)$$

$$\mathbf{P}(1)' = \frac{1}{2}(1-t)[(1+b)(1-c)(\mathbf{p}_{k+1} - \mathbf{p}_k) + (1-b)(1+c)(\mathbf{p}_{k+2} - \mathbf{p}_{k+1})] \quad (3.18)$$

其中,  $b$  是偏离参数,  $c$  是连续性参数。这里的张量参数  $t$  与 Cardinal 样条公式中的  $t$  作用同样, 都是用于控制曲线段的松紧程度。偏离参数  $b$  来调整曲线段在断点处弯曲的数值, 因此曲线段可以偏向一个端点或者另一个端点。参数  $c$  控制切向量在曲线段边界处的连续性, 若  $c$  取非零值, 则曲线在曲线段边界处的斜率上具有不连续性。

通过将虚拟人体走路的轨迹使用如上所述的 Kochanek-Bartels 样条函数进行拟合, 允许用户通过拖动曲线上的控制点修改虚拟人体运动的轨迹, 从而实现了交互式的轨迹编辑。在修改虚拟人体运动轨迹时, 首先调整视点, 如图 3.16 所示, 通过拖动鼠标对轨迹进行编辑。修改轨迹操作完成后, 将视点恢复, 原始运动轨迹得到修改, 如图 3.17 所示。

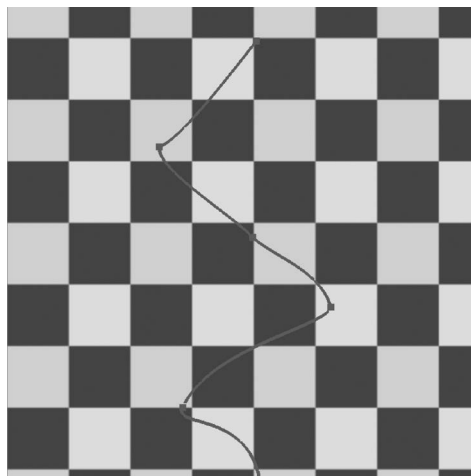


图 3.16 视点调整以后调整运动轨迹

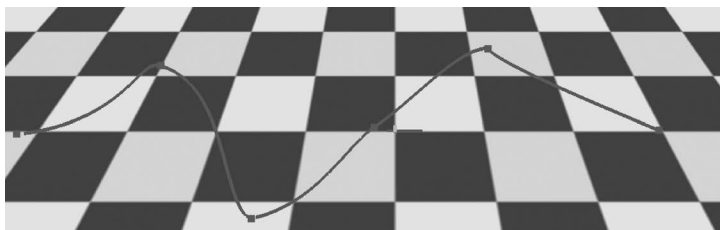


图 3.17 视点还原以后的轨迹

## 2. 实验结果

### 1) 运动合成的效果

图 3.18 展示了运动数据自动分割方法结果。通过对动作捕捉数据进行自动分割,得到了 4 个单位步长的运动数据片段,从左至右依次是左腿跑、右腿跑、左腿走与右腿走。

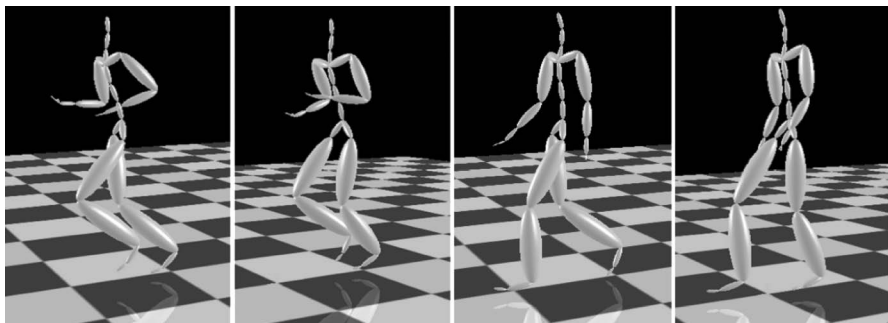


图 3.18 通过自动分割得到的运动数据片段

图 3.19 展示了基于上述动作图构建方法,利用图 3.18 自动分割得到的单位步长运动数据片段合成的运动序列。

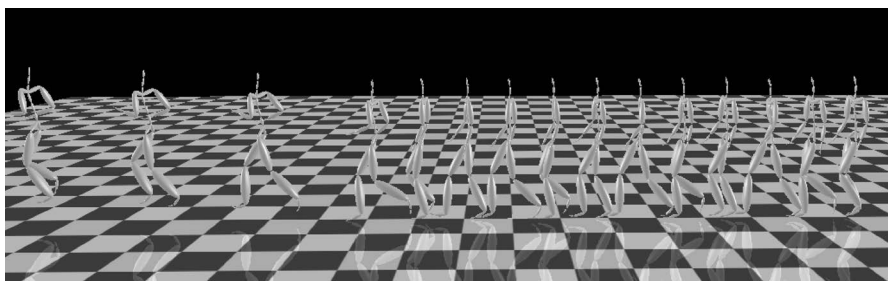


图 3.19 基于动作图生成的由走到跑的运动序列

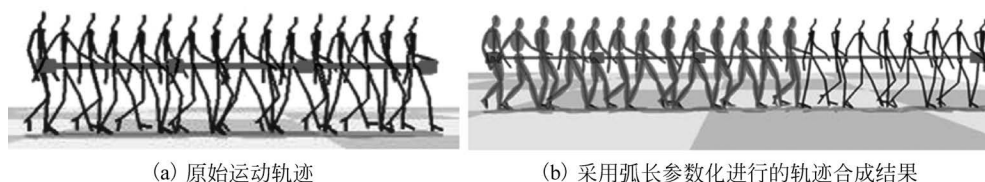
如图 3.19 所示,合成的虚拟人体运动序列较好地实现了由走到跑的变化,不同帧之间变化较为自然。

### 2) 交互式运动编辑的效果

轨迹合成的实质是在目标轨迹上,找到原轨迹上每一帧相对应的坐标位置和方向,并保证虚拟人体的脚步不产生滑动现象。

在求解虚拟人体在目标轨迹上每一帧的根节点的坐标位置和方向时,采用基于轨迹弧长参数化的方法来解决。将虚拟人体的位置以弧长为单位映射到目标轨迹上,使得相同的时间内移动相同的距离,从而保证虚拟人体在目标轨迹上移动的速度与在原始轨迹上的速度一致。采用该方法,既消除了由轨迹变换产生的脚步滑动,又能很好地保留了运动的动力学特性及原始特征,但当目标轨迹长度与原始轨迹长度不一致时,并不能保证运动结束在轨迹末端。因此,需要根据目标轨迹长度合成新的运动,使得运动可以进行到轨迹末端,这就需要利用上面生成得到的运动图进行合成,如图 3.20 所示,图 3.20(a)为原始运动轨迹,图 3.20(b)为采用弧长参数化轨迹的方式进行轨迹合成,该目标轨迹要长于原始轨迹,多出的部分由合成的新运动补偿,使得运动可以进行到轨迹末端。

弧长参数化的方法为给定弧长  $s$ ,得到在新轨迹上距离起始点的距离为  $s$  的点的坐标。



(a) 原始运动轨迹

(b) 采用弧长参数化进行的轨迹合成结果

图 3.20 弧长参数化轨迹合成

弧长公式为  $s = A(t) = \int_0^t \sqrt{x(u)^2 + y(u)^2 + z(u)^2} d(u)$ , 由于积分函数不可逆, 不能直接得到给定弧长对应的参数  $t$ , 实验中以帧为单位, 计算虚拟人体在新轨迹上的位置 (其中, 每一帧经过的位移等于原始轨迹上在当前帧经过的位移), 具体计算的过程是一个离散的积分过程, 而位移的计算是通过上文 Kochanek-Bartels 样条的构造公式得到的。

图 3.21 为已知原始动作捕捉数据, 通过交互指定运动轨迹, 使得虚拟人体沿着新轨迹运动的示意图。图 3.22 为根据动作图生成运动并沿着用户指定的轨迹运动的示意图。可以看到, 合成的虚拟人体能够沿着用户指定轨迹运动, 且没有明显的滑步现象。

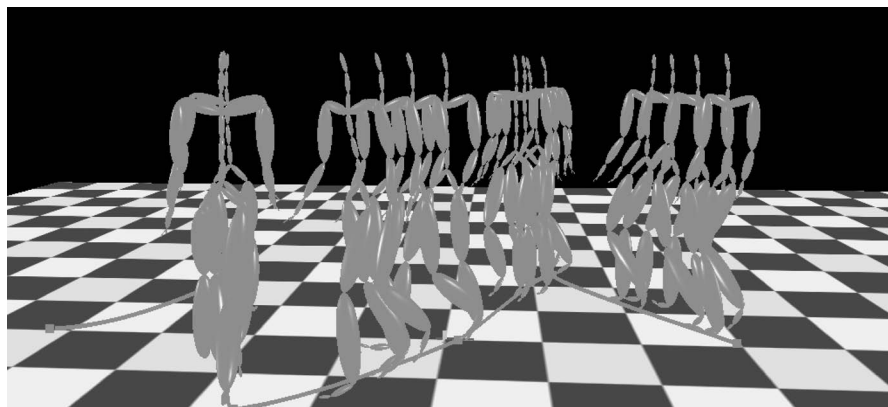


图 3.21 虚拟人体沿用户交互指定的轨迹运动

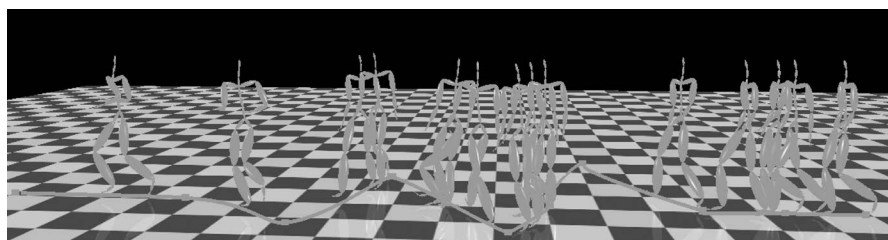


图 3.22 根据动作图生成运动并沿着用户指定轨迹运动

### 3.4 物理建模实例——基于 SPH 的热带气旋建模

#### 3.4.1 背景知识

云景瑰丽多姿的视觉特征和复杂的物理过程一直是气象学和计算机图形学研究人员关注的热点。气象研究人员通常借助地面观测图像或雷达数据分析云的宏观特征, 利用卫星

云图反演云的参数,了解云的构成和发展变化,还会通过求解大气方程组对云景的形成、发展和消散等过程进行仿真。计算机图形学研究人员则主要利用两类经典方法,即基于过程的方法和基于物理的方法,对云景进行仿真建模。前者侧重利用噪声纹理和交互技巧对云景进行建模,依赖频繁的参数调整;后者则借助简化的大气方程组仿真云的动态变化过程,由于初始边界条件和云景的形状呈非线性关系,建模人员仍然需要花费大量的精力才能为初始边界条件设置合适的参数。

受到软硬件的限制,现有方法大多侧重对小规模的云景进行建模,而且多聚焦于云的视觉效果,主要应用于影视动漫和3D游戏中。这些方法构建的云景与现实世界的云景在视觉上比较接近,但其外在形状和内部属性都与现实的云景存在较大的差异,物理真实性有待提高。在虚拟现实应用中,往往需要与云景进行交互,这种情形不单要求在建模视觉上的逼真,还要求与现实云景“物理接近”。这里的“物理接近”不仅体现在视觉效果的相似,而且体现在形状特征、属性构成都尽量符合现实。例如,在军事仿真应用中,虚拟场景中的云景应尽量符合实际天气状况。在天气分析应用中,预报人员则希望真实、实时、逼真地再现大尺度云系的动态效果。在这种要求下,高度的真实性成为云景建模的新要求。

随着探测技术和数值仿真技术的发展进步,人类获取气象数据的能力大大提高。这些数据的出现为构建真实的三维云景提供了必要条件。在此基础上,研究和分析与云景有关的气象数据(如自然图像、卫星云图和数值模拟数据等),并从数据中提取与云景有关的信息,从而构建具有气象学意义的三维云景。在这种建模框架下,最终得到的云景与输入数据在形状特征、属性构成以及场景规模等方面都具有一定相关性。这种方法在一定程度上克服了经典方法的弱点,同时满足了虚拟现实应用的需求,逐渐成为一个新的研究方向和热点。

云景的物理建模侧重于对云景的动态演化过程进行表征,主要方法包括基于过程、基于物理以及数据驱动的方法。从云景的尺度来看,基于过程的方法和基于物理的方法依赖用户的参数设置,比较适合小规模的云景建模;数据驱动的方法则能够满足大规模云景的建模。从云景的动态演化来看,基于物理的方法和数据驱动的方法更容易构造连续时间序列的云景。从建模的输入输出看,基于过程的方法以纹理噪声函数、粒子系统的参数或云景的基本形状为输入,通过计算函数的取值或变换基本形状以产生建模结果,由于输入和建模过程都不具备气象学意义,因此建模结果也不具备气象学意义;基于物理的方法以初始边界条件为输入,通过求解简化的物理方程以生成云景在不同时刻的形状,尽管可以输入真实的气象数据,但由于物理方程过于简化,建模结果还是无法具有气象学意义;数据驱动的方法侧重对气象数据的分析,其核心是利用气象学模型从数据中提取与云景有关的信息,以构建云景的物理属性,最终在虚拟场景中再现云景的演化。三类方法没有明显的界线,具体的工作中往往会用到多种方法。

### 1. 基于过程的方法

基于过程的方法主要涉及三类方法,即噪声纹理函数、粒子系统和可控形状等。对于这些方法,建模人员需要通过设定参数或借助交互手段来构造云景的形状。由于建模结果和输入在视觉上相关性不高,往往需要经过多次的交互才能得到满意的云景形状。

#### 1) 噪声纹理函数

噪声纹理函数是早期云景建模的重要技术。对象的属性(密度和颜色等)被看作空间位

置、时间以及其他参数的函数,因此复杂的自然现象可以通过一些参数进行描述。这类技术主要用于建模具有分形边界的云,如卷云、层云和边界不规则的积云,而浓积云和积雨云则不适合用噪声纹理函数来建模。

图 3.23(a)展示了基于噪声纹理函数的云景建模结果。该方法预先利用隐函数来生成一个规则体数据,然后对体数据进行噪声扰动,从而生成三维云景。

## 2) 粒子系统

粒子系统(Particle System)通常用来建模自然场景,它将自然场景看作大量不规则的、随机分布的运动粒子集合,其中每个粒子具有一定的属性,如位置、颜色、透明度、形状、生命周期等。它们不断运动,改变形状,甚至消亡,从而表现出自然场景的总体外形及其变化规律。粒子系统的特点是实现简单,代码量少,描述相同细节的自然场景时比其他方法需要的内存较少。其不足之处在于,当表现细节程度较高或规模较大的自然场景时,需要大量的粒子,计算开销较大。因此,粒子系统适合建模单朵积云或小规模的积云场景,不适合建模尺度较大的层云和卷云。

元球(Metaball)可以看作粒子系统的推广。单个元球可以通过中心位置、半径和密度三个参数来定义,而所有元球在空间上叠加就形成了物体的密度场。1996年,Nishita等首先通过指定少量的元球构成云景的基本形状,然后在基本形状上随机叠加一些新的元球形成新的形状,经过多次迭代得到云景的最终形状。1998年,Dobashi等采用元球表示台风云系,并借助一个极其简化的单次散射模型从红外云图中搜索每个元球的参数。2010年,Dobashi等将元球的中心位置限定到同一平面上,通过搜索元球的半径和密度从单幅自然图像中重建层云,其建模效果如图 3.23(b)所示。

## 3) 可控形状

这类技术主要用于建模比较稠密的云景,比如积云和层云。由于这些云具有确定的形状,建模人员可以借助图形界面设计云景的形状。

早期,Nishita等通过交互指定云景的基本形状,然后扰动基本形状,以产生具有细节的云景形状。后来,Bouthors等通过在云景的边界迭代地放置粒子来生成积云。他们将小粒子放到大粒子上,当粒子半径接近于0时即指定云景的边界。2003年,Rana通过摆放小立方体来建模云景。类似地,Wang提出了一种大规模三维云景的实时模拟方法,该方法首先通过摆放盒子来生成云景的基本形状,然后用具有位置、大小、角度和颜色的面片填充基本形状形成云景的细节。2008年,Wither等提出利用草图的方法建模云景形状,他们首先勾勒出云的二维廓线,然后将二维廓线经过平移、旋转和缩放等操作形成云的三维形状,如图 3.23(c)所示。

## 2. 基于物理的方法

云景的生成涉及地表气流受热上升、上升气流遇冷凝结形成小水滴、小水滴合并形成云滴等过程。显然,精确描述这些过程是极为困难的。早期的方法对云景的建模多借助一些状态转移规则控制云景的动态变化,后来的方法则利用简化的大气方程组描述云景的生成和发展过程。相比于基于状态转移的模型,大气方程物理含义明确,参数设置比较直观,能够获得更为逼真的动态效果。与基于过程的方法类似,由于基于物理的方法依赖于偏微分方程,建模结果和输入成非线性关系,云景的形状因此很难预料,导致该方法不能达到“所见即所得”的建模效果。近年来,许多运行在CPU的算法都被迁移至GPU上,使得基于物理

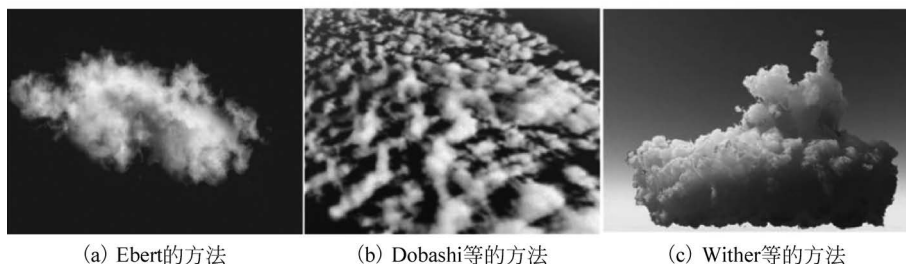


图 3.23 基于过程的云景建模的几种典型方法

的建模方法效率获得了显著提升。然而,由于 GPU 存储能力的局限,这类方法主要用来构建小规模云景,常见的有基于状态转移与基于大气方程组的方法等。

#### 1) 基于状态转移的方法

基于物理的云景建模通常采用三维规则网格表示云景。每一个网格节点都具有一个或多个属性(如颜色、密度、速度等)。1984年,Kajiya 演示了一个简单的方法对云景进行模拟。后来,Dobashi 等利用细胞自动机模型表现云景的变化过程。细胞自动机模型通过简单的状态转移过程控制云景的变化,如图 3.24(a)所示。

#### 2) 基于大气方程组的方法

随着硬件计算能力的提高,研究人员开始利用简化的大气方程组建模云景的运动。2002年,Overby 等使用 Stam 的稳定流体算法在规则网格上通过求解流体力学方程和云相变方程模拟云景的生长消散过程。同年,Miyazaki 等通过求解类似的方程组来模拟积云。在此基础上,Harris 等利用 GPU 并行求解大气方程组,实现了三维云景的交互式物理仿真,如图 3.24(b)所示。此外,Mizuno 等还提出了双流体(2 Fluid Model,2FM)模型,用于模拟火山喷发时的云景。2008年,Dobashi 等提出了可控形状的云景物理建模方法,该方法将用户的控制曲线转化为约束力,构造与控制曲线基本吻合的积云,特别适合艺术设计应用,效果如图 3.24(c)所示。



图 3.24 基于物理的云景建模的几种典型方法

### 3. 数据驱动的方法

观测数据和数值模拟数据是两类典型的气象数据。其中,观测数据根据观测平台可以分为地面观测数据(自然图像、雷达数据等)、机载观测数据、卫星遥感数据(卫星云图)等;数值模拟数据主要是指各种天气预报模式的输出数据,例如气象研究与预报模式(Weather Research and Forecasting Mode, WRFM)数据。目前利用气象数据建模云景的工作相对较少,主要原因是相关数据的获取难度较大。然而,随着深度学习等技术的发展,数据驱动的

方法越来越受到研究人员的重视。

下面介绍一种基于 SPH 的云景建模方法,该方法属于基于过程的方法,以热带气旋的 WRF 数据为输入数据,利用 SPH 框架建模云景的物理属性,从而实现云景的动态演化。

### 3.4.2 基于 SPH 的热带气旋建模方法

#### 1. 算法实现

##### 1) 基于位置的动态算法框架

自 Lucy、Gingold 和 Monaghan 等分别提出了光滑质点流体动力学(Smoothed Particle Hydrodynamics,SPH)方法,并且成功将其应用于天体物理领域之后,SPH 法已经被应用于冲击波模拟、流体动力学、水下爆炸仿真模拟、高速碰撞等材料动态响应的数值模拟领域。然而,SPH 在计算过程中对步长的要求比较苛刻,因此 Miles 和 Matthias 在 2013 年在设计了基于位置的流体(Position Based Fluid,PBF)算法,通过求解一组位置约束,从而强制保证不可压缩性。传统 SPH 方法通过先分析所有的受力情况,然后再计算仿真对象的加速度,进而求解速度和位移的过程,而 PBF 方法则是先应用平流估计,即主要外力作用和惯性产生的位移,然后在此位置上通过迭代求解满足相应的应力作用的位移直到收敛,因此在较大时间步长下,可以实现较为真实的云景运动。

PBF 算法流程如图 3.25 所示。

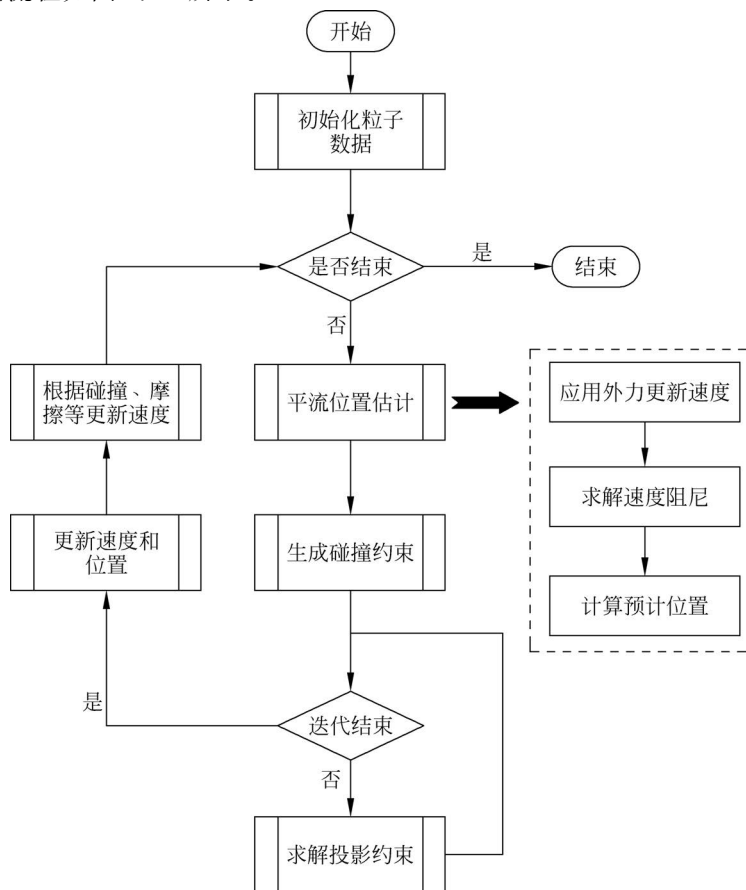


图 3.25 PBF 算法流程图

PBF 提出通过一组包含  $N$  个粒子和  $M$  个约束方程表示动力学对象。其中,每个粒子  $P_i (i \in [1, 2, \dots, N])$  包括质量  $m_i$ 、位置  $x_i$  和速度  $v_i$  等属性。每一个约束方程  $C_j (j \in [1, 2, \dots, M])$  包含基数  $n_j$  (表示粒子的数目), 约束函数  $C_j: R^{3n_j} \rightarrow R$ , 粒子索引  $\{i_1, i_2, \dots, i_{n_j}\}, i_k \in [1, 2, \dots, N]$ , 一组稳健的参数  $k_j \in [0, 1]$  和等式/不等式约束。

若约束方程是一个等式, 需要满足  $C_j(x_i, \dots, x_{i_{n_j}}) = 0$ ; 如果约束函数为不等式, 则需要满足  $C_j(x_i, \dots, x_{i_{n_j}}) \geq 0$ 。参数  $k_i$  用于控制约束的强度, 且其取值范围在  $0 \sim 1$ 。图 3.26 给出了 PBF 算法的伪码。

| 算法 1: PBF 算法 |                                                                         |
|--------------|-------------------------------------------------------------------------|
| 1            | <b>for</b> 所有粒子 $P_i$ <b>do</b>                                         |
| 2            | 初始化粒子数据 $x_i = x_i^0, v_i = v_i^0, w_i = \frac{1}{m_i}$                 |
| 3            | <b>end</b>                                                              |
| 4            | <b>loop</b>                                                             |
| 5            | <b>for</b> 所有粒子 $P_i$ <b>do</b>                                         |
| 6            | 应用外力更新速度 $v_i \leftarrow v_i + \Delta t f_{\text{ext}}(x_i)$            |
| 7            | 求解速度阻尼                                                                  |
| 8            | 计算预计位置 $p_i \leftarrow x_i + \Delta t v_i$                              |
| 9            | <b>end</b>                                                              |
| 10           | <b>for</b> 所有粒子 $P_i$ <b>do</b>                                         |
| 11           | 生成碰撞约束 $x_i \rightarrow p_i$                                            |
| 12           | <b>end</b>                                                              |
| 13           | <b>while</b> 迭代次数 $<$ 最少迭代次数 <b>do</b>                                  |
| 14           | 求解约束方程 $(C_1, C_2, \dots, C_{M+M_{\text{coll}}}, p_1, p_2, \dots, p_N)$ |
| 15           | <b>end</b>                                                              |
| 16           | <b>for</b> 所有粒子 $P_i$ <b>do</b>                                         |
| 17           | 校正速度 $v_i \leftarrow \frac{1}{\Delta t}(p_i - x_i)$                     |
| 18           | 校正位置信息 $x_i \leftarrow p_i$                                             |
| 19           | <b>end</b>                                                              |
| 20           | 更新速度 $(v_1, v_2, \dots, v_N)$                                           |
| 21           | <b>end</b>                                                              |

图 3.26 PBF 算法的伪码

由图 3.26 可见, 算法的第 1~3 行是初始化每一个粒子的属性。PBF 的核心思想是算法的第 6~8 行、第 13~15 行和第 16~19 行。第 6~8 行中使用显式的欧拉积分计算粒子的平流运动, 主要计算对象受到外力后估计的粒子新位置。第 13~15 行是迭代求解过程, 求解约束方程也是算法的难点所在。通过迭代求解约束方程(根据不同仿真对象增加不同约束), 校正平流运动(第 16~19 行)估计的位置  $p_i$ , 直到其满足相关的约束函数为止。

除了估计粒子的新位置之外, 还需要更新粒子的运动速度。速度修改在算法的第 6、7、17 行中。其中, 第 6 行主要是应用那些不能转换为约束函数的系统外力更新速度。常见的系统外力有重力和科里奥利力等。第 7 行表示速度阻尼, 在必要时才增加此项。最后, 在第 17 行中, 根据摩擦和恢复系数对碰撞粒子的速度进行校正。

在求解过程中,  $C_1, C_2, \dots, C_M$  为预设的约束方程。此外第 11 行又在每一个计算步长

中生成了  $M_{\text{coll}}$  个碰撞约束。因此,在第 14 行求解约束函数的过程中,需要对固定的  $M$  个约束函数和  $M_{\text{coll}}$  个碰撞约束进行求解。其具体求解过程为,以  $M+M_{\text{coll}}$  个约束函数和粒子的预计位置  $\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_N$  作为输入,求解过程中修改预计位置直到满足所有的约束函数。这个过程中得到的方程组是非线性的。同时,不等式的约束函数将产生不等式结果。为了求解一系列的等式和不等式方程组,PBF 算法使用 Gauss-Seidel 迭代法进行求解。

## 2) 带旋转的基于 PBF 的云景建模方法

为更符合云景的物理运动过程,可对 PBF 框架进行若干修改,设计带旋转的基于 PBF 的云景建模方法,算法流程如图 3.27 所示。

| 算法 2: 带旋转的基于 PBF 的云景建模算法 |                                                                                         |
|--------------------------|-----------------------------------------------------------------------------------------|
| 1                        | <b>for</b> 所有粒子 $\mathbf{P}_i$ <b>do</b>                                                |
| 2                        | 应用外力更新速度 $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t f_{\text{ext}}(\mathbf{x}_i)$ |
| 3                        | 计算预计位置 $\mathbf{p}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$                   |
| 4                        | <b>end</b>                                                                              |
| 5                        | <b>for</b> 所有粒子 $\mathbf{P}_i$ <b>do</b>                                                |
| 6                        | 查找邻居粒子 $N_i(\mathbf{p}_i)$                                                              |
| 7                        | <b>end</b>                                                                              |
| 8                        | <b>while</b> 迭代次数 $<$ 最少迭代次数 <b>do</b>                                                  |
| 9                        | <b>for</b> 所有粒子 $\mathbf{P}_i$ <b>do</b>                                                |
| 10                       | 计算 $\lambda_i$                                                                          |
| 11                       | <b>end</b>                                                                              |
| 12                       | <b>for</b> 所有粒子 $\mathbf{P}_i$ <b>do</b>                                                |
| 13                       | 计算位置校正 $\Delta \mathbf{p}_i$                                                            |
| 14                       | 执行碰撞检测和响应                                                                               |
| 15                       | <b>end</b>                                                                              |
| 16                       | <b>for</b> 所有粒子 $\mathbf{P}_i$ <b>do</b>                                                |
| 17                       | 校正位置 $\mathbf{p}_i \leftarrow \mathbf{p}_i + \Delta \mathbf{p}_i$                       |
| 18                       | <b>end</b>                                                                              |
| 19                       | <b>end</b>                                                                              |
| 20                       | <b>for</b> 所有粒子 $\mathbf{P}_i$ <b>do</b>                                                |
| 21                       | 校正速度 $\mathbf{v}_i \leftarrow \frac{1}{\Delta t}(\mathbf{p}_i - \mathbf{x}_i)$          |
| 22                       | 应用涡旋约束和 XSPH 黏性                                                                         |
| 23                       | 校正位置信息 $\mathbf{x}_i \leftarrow \mathbf{p}_i$                                           |
| 24                       | <b>end</b>                                                                              |

图 3.27 带旋转的基于 PBF 的云景建模算法

与 PBF 算法相比,带旋转的基于 PBF 的云景建模算法在第 2 行中使用的外力包括重力和科里奥利力;同时将 PBF 算法中碰撞检测和响应过程放到求解不变形约束的约束求解过程中(第 12~14 行)。这是由于流体的粒子碰撞与刚体不同,作用时间短且速度变化大,且在密度约束过程中已经包含了部分流体粒子的碰撞,因此将碰撞检测和响应过程移到约束求解过程中更符合流体运动的特点。同理,将 PBF 中求解投影约束过程分解为第 9~18 行。此外,为解决未预期的阻尼,还增加了涡旋约束和 XSPH 黏性。下面重点介绍修正的部分。

(1) 科里奥利力。热带气旋(台风)呈螺旋状的主要原因是地球是旋转体。由于地球本

身在旋转,在地球上观察的直线运动,其真实速度还需要惯性体系中和地球自转相同的速度。因此,沿着地球坐标系看到的直线方向运动,实际上会朝着某一个方向偏移,这也是台风都呈逆时针运动的原因。在建模热带气旋时,这是不可忽略的现象。

在以旋转体为参考系中的质点沿直线运动时,导致其偏离原有运动方向的力就是科里奥利力。从物理学角度考虑,科里奥利力在惯性系中并不存在,而是惯性作用在非惯性系中的一种表现形式,是为了方便计算而引入的一个假想惯性力。科里奥利力表示如下列公式所示:

$$\mathbf{f}_c = -2m(\boldsymbol{\Omega} \times \mathbf{v}) \quad (3.19)$$

其中, $m$  是质点质量, $\boldsymbol{\Omega}$  是地球的旋转角速度, $\mathbf{v}$  是相对于地球参考系的运动速度。 $\times$  是向量积符号。

(2) 密度约束。为保证粒子的密度,在求解非线性约束系统对每一个粒子生成一个约束函数。每一个约束函数都是该粒子与其邻居粒子位置相关的函数,定义如下列公式所示:

$$C_i(\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n) = \frac{\rho_i}{\rho_0} - 1 = 0 \quad (3.20)$$

其中, $\rho_0$  表示剩余密度, $\rho_i$  为使用标准 SPH 密度估计的估计值,表示为:

$$\rho_i = \sum_j m_j W(\mathbf{p}_i - \mathbf{p}_j, h) \quad (3.21)$$

其中, $m_j$  是第  $j$  个粒子的质量, $W$  是紧致域半径为  $h$  的光滑核函数。常见的核函数有 poly6 核函数、Spiky 核函数等。

由于公式(3.21)是非线性方程,在光滑核函数的边界可能出现梯度消失的情况,因此在约束函数中添加一些约束力来松弛约束,如下列公式所示:

$$C(\mathbf{p} + \Delta\mathbf{p}) \approx C(\mathbf{p}) + \lambda \nabla_p C(\mathbf{p}) \nabla_p C(\mathbf{p}) + \epsilon \lambda = 0 \quad (3.22)$$

其中, $\epsilon$  是预设的松弛参数。在此情形下,所有的位置更新将包含所有邻居粒子的密度约束,如下列公式所示:

$$\Delta\mathbf{p}_i = \frac{1}{\rho_0} \sum_j (\lambda_i + \lambda_j) \nabla W(\mathbf{p}_i - \mathbf{p}_j, h) \quad (3.23)$$

(3) 拉伸不稳定性。在 SPH 仿真中常见的一个问题是由于粒子周边邻居粒子过少导致负压强,导致粒子聚集的现象,如图 3.28(a)所示。通过增加人工压力,可保持压力的非负性,减少粒子的凝聚性,如图 3.28(b)所示。

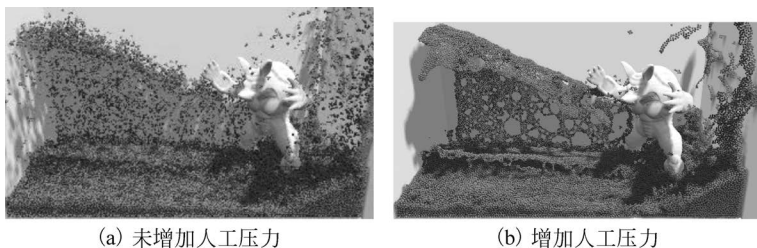


图 3.28 增加人工压力前后的粒子聚集情况

具体地,在光滑核函数中增加一个人工压力,如下列公式所示:

$$s_{\text{coor}} = -k \left( \frac{W(\mathbf{p}_i - \mathbf{p}_j, h)}{W(\Delta \mathbf{q}, h)} \right)^n \quad (3.24)$$

其中,  $\Delta \mathbf{q}$  是一个具有固定距离且位于光滑核紧致域半径内的点,  $k$  是一个小的正常量。粒子矫正位置在添加人工压力后变为下列公式:

$$\Delta \mathbf{p}_i = \frac{1}{\rho_0} \sum_j (\lambda_i + \lambda_j + s_{\text{coor}}) \nabla W(\mathbf{p}_i - \mathbf{p}_j, h) \quad (3.25)$$

(4) 涡旋约束和黏性。这里使用涡旋约束来模拟阻尼带来的影响, 可使用位置矢量  $\mathbf{N} = \frac{\boldsymbol{\eta}}{|\boldsymbol{\eta}|}$  来计算矫正的力, 如下列公式所示:

$$\mathbf{f}_i^{\text{vorticity}} = \varepsilon (\mathbf{N} \times \boldsymbol{\omega}_i) \quad (3.26)$$

其中,  $\varepsilon$  是预设的因子,  $\boldsymbol{\omega}_i$  是在粒子  $\mathbf{P}_i$  位置的涡旋。此外, 添加的 XSPH 黏性如下列公式所示:

$$\mathbf{v}_i^{\text{new}} = \mathbf{v}_i + c \sum_j (\mathbf{v}_j - \mathbf{v}_i) W(\mathbf{p}_i - \mathbf{p}_j, h) \quad (3.27)$$

其中,  $c$  是常数。

## 2. 实验结果

### 1) 初始状态的构建

传统的粒子系统中粒子属性包含粒子的位置、质量及半径等基本属性, 同时为了建模粒子的运动过程, 还需要粒子的速度、加速度及质量等属性。为了准确计算粒子的速度和加速度, 这里还需引入粒子的初始位置、上一帧位置及上两帧位置。此外, 为了加速计算, 提前计算并保存云粒子的质量倒数等。综上可构建云粒子的属性, 如表 3.3 所示。

表 3.3 粒子系统中的粒子属性

| 属 性   | 英 文 名 称       | 属 性 含 义   |
|-------|---------------|-----------|
| 位置    | Position      | 粒子当前位置    |
| 半径    | Radius        | 粒子半径      |
| 速度    | Velocity      | 粒子当前运动速度  |
| 加速度   | Acceleration  | 粒子运动的加速度  |
| 质量    | Mass          | 粒子质量      |
| 质量倒数  | Inverse mass  | 粒子质量的倒数   |
| 初始位置  | Position0     | 粒子初始位置    |
| 上一帧位置 | Old position  | 粒子上一帧所在位置 |
| 上两帧位置 | Last position | 粒子上两帧所在位置 |

这里使用 WRF 数据建模热带气旋。WRF 包含多种物理量数据的输出, 如网格的经纬度、地表温度、云顶温度、地形高度、云水混合比等。其中, WRF 数据中的速度属性用来表示上述热带气旋粒子系统中每个粒子的运动速度。

### 2) 热带气旋建模结果

经过粒子系统初始状态构建后, 从 WRF 数据集中采样粒子的位置、速度、密度、压强等信息, 利用带旋转的基于 PBF 的云景建模方法即可实现热带气旋的建模。图 3.29 展示了热带气旋的建模结果。数据时间范围为 2010 年 9 月 15 日 0 时至 2010 年 9 月 15 日 11 时 30 分(共 12 小时), 地理范围为东经 113.0 度至 120.8 度, 北纬 22.2 度至 25.8 度。

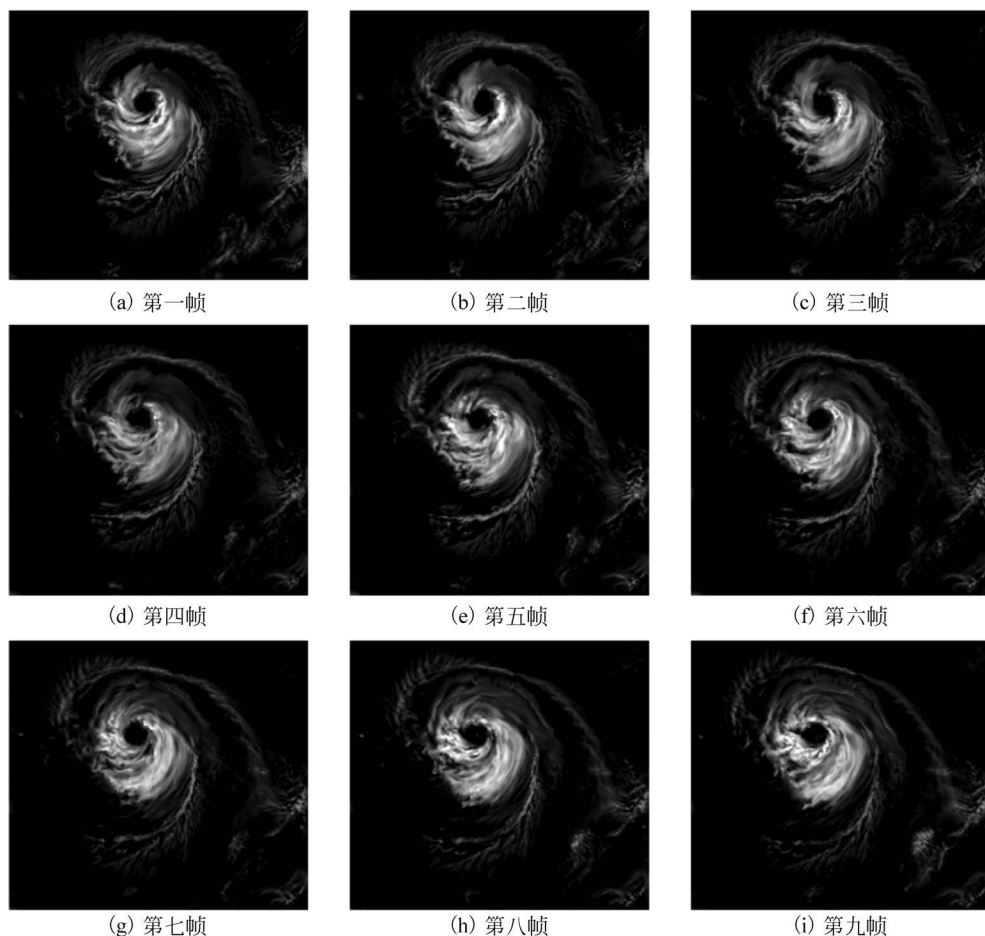


图 3.29 热带气旋建模效果

由图 3.29 可见,从第一帧到第九帧,热带气旋的动态演化过程较为自然,且符合真实场景下的运动规律。由此可见,本节介绍的算法可以有效地实现热带气旋的物理建模。当然,该算法也存在一些不足。例如,建模过程中缺少对热带气旋行进路径机理的分析,不能对热带气旋的行进进行建模。此外,PBF 算法计算量较大,无法支持较大规模的云景建模。

### 3.5 习题

1. 在 Unity 中,构造包括陆地、海洋、植物等元素的虚拟环境。
2. 在 Unity 中添加人物角色,并制作动画。
3. 通过 Kinect 获取室内场景,并构造室内模型。