推算错误预测的自适应位置外推算法

周 忠 魏 晟 张国峰 吴 威

(北京航空航天大学 虚拟现实技术与系统国家重点实验室, 北京 100191)

摘 要: 给出了适用于第一人称射击游戏中角色运动特征的位置外推算法. 根据此类游戏角色运动方向多变、轨迹呈连续折线的特点, 选取一阶外推方程和直线收敛算法用于位置推算与收敛, 根据角色运动行为和距离来预测推算错误、调整收敛过程, 减少游戏的网络通信量并提高推算精度. 实验结果表明, 该位置外推算法应用于第一人称射击游戏可以减小发送的状态更新数量和推算轨迹偏差.

关 键 词: 第一人称射击; 位置预测; 预测错误; 动态收敛

中图分类号: TP 391

文献标识码: A 文章编号: 1001-5965(2008)09-1070-06

Error-predictable adaptive dead reckoning algorithm

Zhou Zhong Wei Sheng Zhang Guofeng Wu Wei

(State Key Laboratory of Virtual Reality Technology and Systems, Beijing University of Aeronautics and Astronautics, Beijing 100191, China)

Abstract A dead reckoning algorithm was presented which is suitable for first-person shooting games. Avatars' moving directions change frequently and their traces consist of continuous broken lines. According to the features, first order derivative polynomial and linear convergence algorithms were selected for position extrapolation and convergence respectively. Avatar's moving state was used to predict extrapolation error and adjust convergence procedure dynamically. Extrapolation policy was changed with the distance between avatar and its observer in order to reduce state update packets. Experiment results show that the position extrapolation algorithm can reduce state update packets and deviation of reckoning trace in first-person shooting game.

Key words first-person shooting; position extrapolation; error prediction; dynamic convergence

第一人称射击类多人在线游戏中存在着频繁的角色移动, 产生大量的状态更新消息, 加上网络延时的影响使客户端对于远程物体的仿真总是落后于服务器端执行的真实仿真状态, 当游戏运行于 Internet上时这种影响尤为明显. 外推 (DR, Dead Reckoning) 算法通过牺牲仿真精度来降低仿真应用的网络通信量和减轻网络延时影响, 是网络游戏和仿真的重要算法. 在仿真过程中, DR 算法在观察器上保存物体的运动状态信息, 如当前位置、速度、加速度等, 根据外推算法在本地预测仿真物体的状态. DR 算法主要分为通用目的的 DR 和特殊化 DR 两类, 推算目标包括位置推算和方向推算.

通用目的 DR 不对仿真环境与物体做假定, 适用于各种仿真环境和物体类型. 早期的 DR 算法中, 仿真器以 1Hz的固定频率向观察器发送物体的位置、速度更新, 观察器使用匀速直线运动的外推方程推算物体位置, 不考虑网络延时对推算的影响, 也没有收敛机制[1]. IEEE 1278 的 DIS (Distributed Interactive Simulation) 标准[2]在 1995 年对 DR 算法进行了定义, 给出了适用于不同情况的 11 种推算方程, 包括位置推算和方向推算, 还采用固定平滑点数目的直线收敛算法修正推算偏差.

文献[3]提出基于物体的历史位置点来进行位置推算, 根据物体在最近的运动中转过的角度

收稿日期: 2007-09-10

基金项目: 国家 863计划资助项目 (2006AA01Z331); 国家自然科学基金资助项目 (60603084)

作者简介: 周 忠 (1978-), 男, 江苏海安人, 副教授, zz@vrlab.buaa.edu.cn

动态地选择外推和收敛算法. 该算法传输的状态更新中不包含速度与加速度信息, 进一步减少了通信量. 文献[4]提出动态阈值的 DR 算法, 根据推算目标的兴趣区域与观察者的兴趣区域和敏感区域的相交关系来确定 DR 算法的阈值. 文献[5]在文献[3]的基础上, 提出对角度进行预推算, 当物体转角超过预定义的角度阈值时即发送状态更新以避免推算错误. 文献[6]使用全局同步时钟进行带时间戳的 DR 推算, 观察器根据时间戳进行远程物体的绘制.

特殊化的 DR 算法应用于特定的仿真环境以及推算目标, 利用环境及物体类型相关的信息对 DR 算法进行优化与改进, 使其在特定的应用范围内能够获得更好的推算效果, 如文献[7]提出了对飞行仿真进行优化的 DR 算法, 文献[8]提出对具有关节的物体的推算算法.

对于第一人称射击游戏, 现有的 DR 算法均没有研究其角色运动特征, 并不太适合在此类游戏中使用[9]. 前述一些算法设定可变的偏差阈值但没有考虑收敛过程中角色运动状态改变可能引起的新的偏差. 本文结合第一人称射击游戏的输入控制与角色移动的特点提出了可预测错误的自适应位置外推算法 (EPADR, Error-Predictable Adaptive Dead Reckoning), 用于第一人称射击游戏的角色位置预测, 利用角色运动行为来预测推算错误、调整收敛状态, 根据角色与观察者间的距离设定不同的推算策略.

1　基本思想

在第一人称射击类游戏中, 输入控制与角色移动方式相对于其它类型的游戏具有一定的特点. 绝大多数的射击游戏都通过鼠标控制调整角色朝向, 按键控制平移. 当鼠标移动一定距离, 角色就转动一定角度; 平移控制通常使用 4个按键, 分别代表前后左右, 并允许组合使用, 即前和左同时按下时会产生左前方向的斜向平移. 角色的移动为匀速直线运动, 虽然速度可能变化, 如由走变为跑、角色移动能力提升、在不同重量的武器间切换等, 但是不会在一定时间内连续地变化, 速度的改变均由离散的事件触发, Counter-Strike和 BZ-Flag等游戏均是如此. 这些特点使得角色的运动轨迹总体上表现为连续的折线, 转向轨迹由若干较短的折线组成, 不存在平滑的曲线, 如图 1所示.

图 1中的轨迹是从真实游戏中采样得到的角色移动轨迹, 游戏世界的尺寸为 2 000格 ×2 000格, 角色速度为 50格 /s. 游戏的大多数时间里角色处在非战斗移动状态, 转向具有目的性, 其运动特征为: 直线移动→改变移动方向→直线移动; 在战斗情况下, 用户控制角色躲避攻击, 移动行为较为复杂, 但主要由左右和斜向平移组成. 由于这些特点, 角色的运动轨迹主要呈折线, 没有平滑的曲线运动. 同时, 由于射击游戏通常对于用户没有很强的规则约束, 角色行为具有多样性, 很难采用预定义的宏观行为模式来描述角色的位置移动.

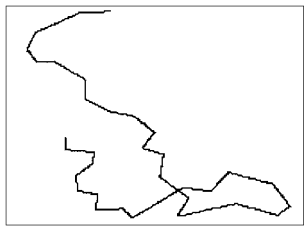


图 1　第一人称射击游戏的角色移动轨迹采样

因此可采用匀速直线运动外推方程预测位置. 其 DR 模型为一阶多项式: $P = P_0 + v_0 \Delta t$

在第一人称射击游戏中, 用户控制的角色的运动行为通常具有一定模式, 而其中部分运动行为模式导致推算错误的可能性较高. 因此, 算法可以不是被动地在发生推算错误时发送状态更新, 而是在检测到会导致推算错误的运动行为时及时通知客户端提早修正. 与偏差超过阈值时才修正相比, 在保持发送的状态更新数量不变的情况下可减小推算轨迹偏差. 而且, 在一些情况下当偏差超过阈值时通过收敛算法纠正错误较为困难[5]. 本文从该想法出发, 对可预测错误的自适应位置外推算法进行研究.

2　EPADR 算法流程

算法根据角色的位置与速度矢量在客户端推算远程角色位置, 检测导致推算错误可能性较大的角色运动行为, 发送状态更新, 并在收敛过程中根据角色运动状态的改变对收敛过程动态调整, 根据推算角色与观察者的距离改变推算策略, 以减少需要发送的状态更新数量和推算轨迹的平均偏差. EPADR 算法在游戏一帧中的执行流程如图 2所示.

游戏的每一帧执行此流程一次, 对异地角色的位置进行预测. 算法开始时根据角色的当前推算状态使用相应的算法模块计算角色位置. 如果角色没有处于收敛状态, 则使用外推方程计算位置; 否则使用分级收敛模块在收敛起始位置和终

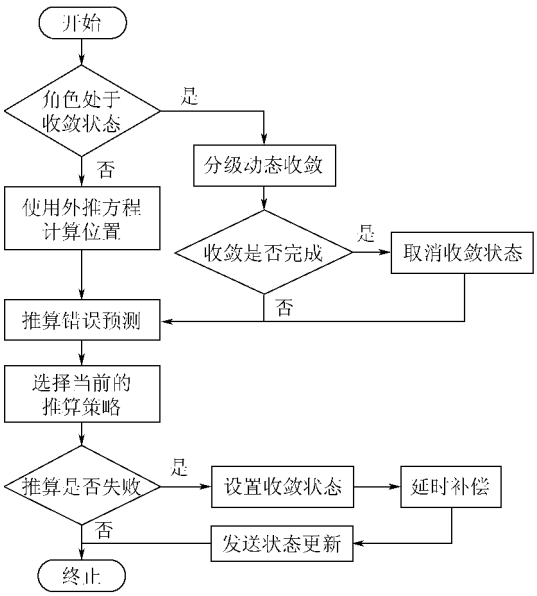


图 2 EPADR 算法执行流程

止位置间插值计算位置. 在完成位置计算后, 通过推算错误预测模块判定角色的当前运动行为. 在根据角色距观察者的距离选择当前的推算策略后, 通过推算偏差和角色运动行为判断当前帧的推算是否失败. 如果没有失败, 则结束当前帧的预测; 否则, 设置角色进入收敛状态, 并通过延时补偿计算出状态更新到达客户端时角色所处的实际位置, 然后将此位置与角色当前速度作为状态更新发送给客户端. 推算错误预测、分级收敛、推算策略是其中的 3个主要部分, 下面将分别介绍.

2 1 推算错误预测

EPADR 算法通过实时检测角色的运动行为, 对于可能使预测偏差超出阈值的行为, 根据其行为效果, 结合当前角色所处的状态与环境, 确定是否需要发送状态更新进行提前纠正.

预先定义第一人称射击类游戏中存在的可能引起预测错误的运动行为, 并根据行为效果为每个行为定义影响因子 EPFactor 根据游戏具体环境和规则为游戏世界中的区域定义运动行为 EPFactor阈值. 当运动行为的 EPFactor大于等于当前游戏区域的 EPFactor阈值时, 即可判断此运动行为在当前区域起作用, 在未来的某个时间很可能会引起推算偏差超过阈值, 需要立即发送状态更新对推算数据进行修正. 运动行为的定义如表 1所示.

由于不存在加速度, 游戏中的预测错误主要是由运动方向的改变引起的, 而动静切换可看作是一种特殊的运动方向改变. EPADR 抽象并定义了 4种改变角色移动方向的主要运动行为, 包括动静切换和 3个角度范围的转向. EPFactor越大表示此运动行为引起预测错误的可能性越高. 3种典型的转向行为如图 3所示.

表 1 用于推算错误预测的角色运动行为的定义

运动行为	EPFactor	判断条件
动静切换	4	当前速度与前一帧速度两者中有且仅有一个为 0
大角度运动转向	3	转向角度范围: (120°, 180°] 稳定状态角度偏差阈值: 5° 判断转向结束的稳定时间: 100ms
中角度运动转向	2	转向角度范围: (80°, 120°] 稳定状态角度偏差阈值: 5° 判断转向结束的稳定时间: 100ms
小角度运动转向	1	转向角度范围: [40°, 80°] 稳定状态角度偏差阈值: 5° 判断转向结束的稳定时间: 100ms
平缓运动	0	不符合其它所有行为定义

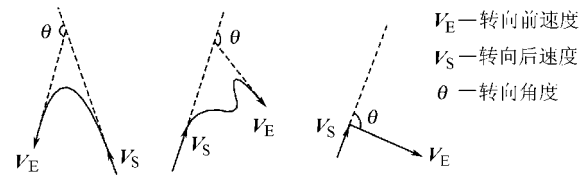


图 3 运动转向行为示例

运动转向行为没有完成时间的限制, 只要推算位置始终在偏差阈值范围之内, 满足表 1中规定条件的任意持续时间的转向行为都属于可能引起预测错误的转向. 运动转向行为中的稳定状态角度偏差阈值和判断转向结束的稳定时间 2个参数用于判断转向行为的结束. 通过前者判断角色是否由速度变化状态进入速度稳定状态, 即当连续 2帧的速度夹角不大于此值时, 则认为角色速度进入稳定状态; 后者用来判断转向行为是否结束, 当角色速度连续处于稳定状态的时间达到此参数值时, 则认为转向行为完成, 可根据此行为的 EPFactor判断是否需要发送状态更新. 运动转向行为判定的伪代码如下:

```
cosθ = V_Real • V_DR; //实际速度方向和推算速度方向的夹角
if( cosθ <= cos(TurnActionAngleMin) && cosθ >= cos(TurnActionAngleMax)) { //偏差速度夹角在转向行为指定的角度范围内
    cosθ = V_Real • V_Previous; //实际速度和前一帧真实速度的夹角
    V_Previous = V_Real; //记录当前真实速度
    if( cosθ >= cos( stableAngleThreshold)) { //角度小于稳定状态角度偏差阈值
        StableTime += Δt; //速度稳定时间递增
        if( StableTime > StableTimeThreshold) { //速
```

```
        度稳定时间超过上限
            IsTurning= false; //清空转向状态标志
            StableTime= 0; //清空稳定状态计时器
            return true; //符合此转向行为, 返回结果
            }
        }
        else
            StableTime= 0; //清空稳定状态计时器
    }
  else{
     IsTurning= true; //设置推算目标的转向状态标志
     V_Previous = V_Real; //记录当前实际速度
        }
}
 else if( IsTurning) { //处于转向状态, 且偏差夹角大于转向行为指定的角度范围
   IsTuring= false; //清空推算目标的转向状态标志
   StableTime= 0; //清空稳定状态计时器
}
 return false; //不符合此转向行为, 返回结果
```

相对于文献[10]中定义的宏观运动模式, 如前进到指定地点、返回基地等, EPADR 算法中定义的是一种微观运动模式, 体现的是角色的轨迹特征而不是行为特征. 定义的运动行为可以根据具体游戏的特点增加或减少, 进行更加特殊化的定义, 调整行为参数等. 错误预测存在失败的可能, 如角色改变移动方向后立即又转回原先的方向, 发生这种情况就会导致多发送一个状态更新, 但不会显著减少推算偏差. 因此通常情况下, 推算错误预测机制需要付出增加少量状态更新的代价, 但是当预测失败的情况大量存在时, 推算错误预测机制就无法达到预期的效果.

2.2　分级收敛

由于第一人称射击类游戏中角色存在很多急转变向运动, 其运动轨迹具有折线特征, 曲线收敛并不能够取得更好的视觉效果, 甚至有可能使角色移动表现得不自然, 而直线收敛更符合角色的运动特性. 此外, 采用不同的收敛方程对平滑效果的影响差别不大[11], 直线收敛算法的计算量小, 产生的误差较小, 更适于采用. EPADR 算法中采用固定收敛时间的直线收敛, 与固定插值点的方法[2]相比, 收敛时间可控, 可避免收敛过程持续时间过长.

在收敛过程中, 实时地检测角色运动状态的改变, 根据其运动行为判断是否需要动态调整收敛状态. 如果发现在收敛过程中角色的运动行为的 EPFactor大于当前区域的 EPFactor阈值, 即表明角色当前的运动行为很可能会引起其实际位置逐渐偏离客户端收敛算法计算的收敛点, 则发送状态更新对客户端推算的收敛方向与速率进行及时校正, 减小推算轨迹的偏差.

为了避免在收敛过程中因角色频繁地改变运动方向而导致发送过多的用于调整收敛的状态更新, 预先将动态收敛划分为 K 级 ($K \geqslant 0$), 开始时处于 0级收敛状态, 每发送一次用于调整收敛的状态更新收敛级别就加 1. 设预定义的收敛时间为 T, 则第 i级收敛状态的收敛时间为 $t_i = T/2^i$, 即收敛级别每加 1, 收敛时间减半. 若当前收敛级别 $i=K$, 则结束收敛过程, 直接将推算目标放置到其真实位置.

EPADR 算法的收敛方程如下.

$$x_t = \begin{cases} x_s + (x_e - x_s) \times t/(T/2^i) \\ \qquad 0 \leqslant i \leqslant K-1,\ 0 \leqslant t \leqslant T/2^i \\ x_e \qquad i = K \end{cases}$$

$$y_t = \begin{cases} y_s + (y_e - y_s) \times t/(T/2^i) \\ \qquad 0 \leqslant i \leqslant K-1,\ 0 \leqslant t \leqslant T/2^i \\ y_e \qquad i = K \end{cases}$$

其中, (x_t, y_t)为当前级别收敛中 t时刻的插值点坐标; T为预定义的收敛时间; K 为收敛级别上限; i为当前收敛级别; (x_s, y_s)和 (x_e, y_e)分别为当前级别的收敛起始点和结束点. EPADR 算法的收敛过程如图 4所示.

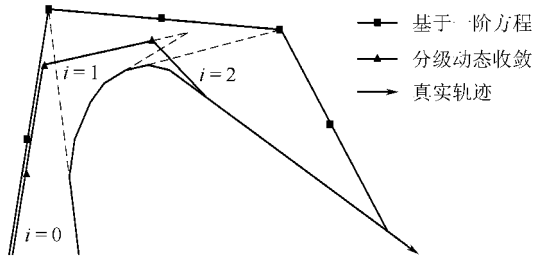


图 4　分级动态收敛示意图

运动行为的 EPFactor大于等于当前区域的 EPFactor阈值时, 角色的运动行为与在当前区域起作用的转向行为匹配, 算法就发送状态更新对收敛方向和速率进行修正. 图 4中 $i=0, 1, 2$的 3段轨迹分别表示了收敛的 3个级别, 级别每增加 1, 收敛趋势就更加明显, 收敛进程也更快. 相比收敛过程中没有调整的收敛算法, 同样发送了 2次状态更新, 分级动态收敛在收敛过程中角色不断

改变运动方向时可以及时修正收敛过程, 减少推算轨迹偏差. 将收敛过程分级可以防止收敛进程持续时间过长, 避免因调整收敛而发送过多的状态更新.

2.3 推算策略

通常情况下 DR 算法采用固定的阈值控制推算偏差, 不考虑物体间的相对位置关系. 在要求保证较高的仿真精度的情况下, 需要使用较小的阈值. 但在游戏中, 当角色间的距离较远时, 精确的位置信息对于游戏用户来说并不重要. EPADR 算法根据推算目标与观察者的相对距离选择推算策略, 提供不同的仿真精度.

在观察者的敏感区域 (SR, Sensitive Region) 中, 远程角色的位置偏差对用户的游戏体验有着较大影响, 而观察者对 SR 之外的区域关心程度较低, 位置偏差对用户的影响相对减弱. 因此以观察者的 SR 为参照划分动态策略区间, 可提供更为细致的推算策略. 策略区间的划分如图 5所示, 分为 4个策略区间: $[0, 0.25R]$, $(0.25R, 0.5R]$, $(0.5R, R]$, $(R, +\infty)$. 其中 R 为观察者的 SR 半径. 策略区间的宽度由远及近递减, 以随着距离的减小向用户提供精确度逐渐增加的异地角色表现.

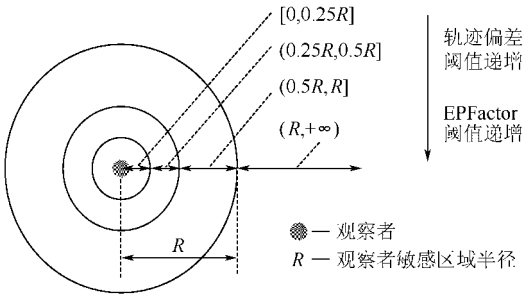


图 5 策略区间的划分

由于角色间的射击等交互主要发生在近距离的情况下, 而且远距离时的三维透视投影效果又使得偏差在视觉效果上得以变小, 根据距离划分策略区间可以在保证用户的游戏体验不受影响的同时减少发送的状态更新数量. 射击游戏中推算偏差主要影响的是用户的射击交互, 射击结果又受到被攻击角色的有效攻击面影响, 因此采用角色宽度的倍数来定义推算策略中的轨迹偏差阈值. 通常情况下, 策略区间距离观察者越远, 所设定的推算轨迹偏差和 EPFactor的阈值均越大, 但也可以根据具体游戏的特点进行特殊化的设定.

EPADR算法的可变推算策略在引入自适应性的同时也增加了一定量的存储开销. 在推算策略可变的情况下, 角色与各个观察者间距离的不同会引起此角色对不同观察者的推算数据各不相同, 需要维护角色的多个推算数据. 当推算数据量较大且游戏中存在大量相关角色的时候, 可变推算策略会急剧加重存储开销.

3 实验与结果分析

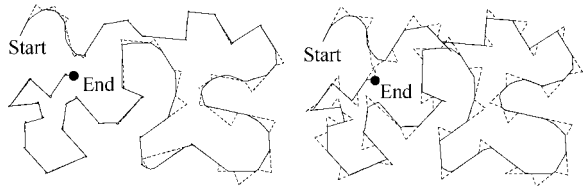
对 EPADR 算法的推算错误预测、收敛和推算策略进行测试, 并与 DIS 标准中适用于第一人称射击类游戏 DR 算法的 DRM 形式[2] (SDR, Standard DR)进行比较.

测试中算法和程序的配置参数如下: 游戏世界尺寸为 1 000格 × 750格, 角色直径为 30格, 角色速度为 30格 /s, 观察者敏感区域半径为 325, 推算目标起始位置为 (100, 600), 运动轨迹采样间隔为 200ms, 其中的坐标和长度均采用游戏世界坐标单位. 为了对比算法的推算效果, 测试没有计入网络延时.

3.1 推算错误预测

对 EPADR 和 SDR 算法进行比较, 测试它们各自发送的状态更新数量和推算轨迹平均偏差. 关闭两者的收敛算法, 并关闭 EPADR 算法的推算策略机制. 测试中操作角色在游戏世界中移动, 同步地对角色真实运动轨迹和 DR 算法实例的推算轨迹进行采样, 并分别记录它们发送的状态更新数量. 算法实例 EPADR3和 SDR3的阈值为 45.

测试结果包括每个算法实例与真实轨迹的对比以及状态更新数量和推算轨迹平均偏差的统计结果, 如图 6和表 2. 表 2中阈值为 45的 EPADR 算法比同阈值的 SDR 算法发送的状态更新数量高 14.9%, 平均偏差减小了 61.1%. 可以看出对于第一人称射击游戏中的角色运动, EPADR算法以增加少量状态更新数据为代价显著减小了推算轨迹的平均偏差.



a EPADR3 b SDR3

圆点—推算的目标角色; 实线—真实轨迹; 虚线—推算轨迹.

图 6 推算错误预测的轨迹对比

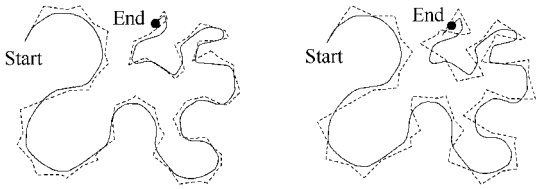
表 2 推算错误预测的状态更新和平均偏差统计结果

算法实例	状态更新	平均偏差
EPADR3	54	4.391
SDR3	47	11.277

3.2　分级收敛

测试中控制角色在游戏世界中连续地转动, 使角色在收敛过程中速度发生改变. 对真实轨迹和它们的推算轨迹进行定期采样, 并记录各算法实例发送的状态更新数量. 算法实例的收敛机制开启, 收敛时间为 1000ms, 直线收敛, 阈值配置同 3.1节. 将 EPADR 算法的推算策略关闭, 收敛级数设为 3.

测试结果包括推算轨迹与真实轨迹的对比图、发送的状态更新数量与推算轨迹平均偏差的统计结果, 如图 7和表 3.



a　EPADR3　　b　SDR3

图 7　分级动态收敛测试的轨迹对比

表 3　分级动态收敛测试的状态更新和平均偏差统计结果

算法实例	状态更新	平均偏差
EPADR3	45	14.632
SDR3	33	25.151

根据表 3, 阈值为 45的 EPADR 算法比同阈值的 SDR 算法发送的状态更新数量高 36.4%, 推算轨迹的平均偏差减小了 41.8%.

结果表明 EPADR 算法发现收敛过程中可能产生的新偏差并及时调整收敛方向与速率, 减少了收敛过程中推算的平均偏差, 但增加了需要发送的状态更新数量.

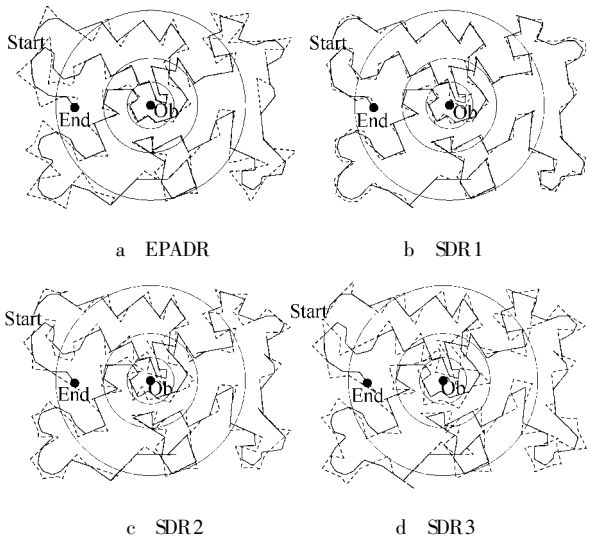
3.3　推算策略

测试中控制角色在观察者的周围随意移动. 使用 EPADR 算法实例和 3个采用不同阈值的 SDR 算法进行轨迹推算, 记录每个算法发送的状态更新以及更新发送时运动角色与观察者间的距离, 定期对轨迹进行采样, 对于每个采样记录采样时运动角色与观察者间的距离. 3个 SDR 算法实例开启收敛机制, 收敛时间均为 1000ms, 阈值分别为 15, 30, 45(对应 SDR1, SDR2, SDR3). EPADR 算法的收敛级数设为 3, 推算策略配置如表 4.

表 4　EPADR 算法的推算策略表

距离范围	阈值	EPFactor
(325.0, +∞)	60	4
(162.5, 325.0]	30	3
(81.25, 162.5]	21	2
[0, 81.25]	15	1

如图 8所示, Ob为观察者, 以其为中心的圆圈对应于 EPADR 算法的各推算策略区间的边界.



a　EPADR　　b　SDR1

c　SDR2　　d　SDR3

图 8　可变推算策略测试的轨迹对比

统计结果如表 5所示.

总体上看, EPADR 算法比 SDR1状态更新数量降低了 15.5%, 轨迹偏差高出了 117.3%. 相对于 SDR2, 状态更新数量和轨迹偏差分别降低了 3.9%和 0.5%. 与 SDR3相比, 状态更新数量高出 12.6%, 轨迹偏差降低了 36.6%. 而通过分段统计的结果看出, EPADR 算法在 [0, 162.5]这个区间内发送的状态更新占总更新数的比例在 4个算法实例中最高, 平均偏差最小. 在这个区间以外, EPADR 算法的状态更新比例相对下降, 平均偏差升高, 但是由于距离观察者较远, 对于用户的游戏体验不会产生较大影响.

表 5　可变推算策略测试的状态更新和平均偏差的分段统计结果

策略区间	EPADR		SDR1		SDR2		SDR3	
	平均偏差	状态更新	平均偏差	状态更新	平均偏差	状态更新	平均偏差	状态更新
[0, 81.25]	11	3.668	10	9.240	9	15.648	6	23.621
(81.25, 162.5]	18	4.536	19	5.447	16	13.694	15	20.151
(162.5, 325.0]	39	8.319	40	4.624	35	11.151	31	18.323
(325.0, +∞)	30	22.496	47	6.962	42	14.187	35	22.089
总　体	98	12.948	116	5.959	102	13.011	87	20.410

(下转第 1100页)

转速且角动量大小相等的不限定框架构型的陀螺群的内奇点必是隐奇点的严格证明,并通过三正交框架构型陀螺群的仿真实例对有限点进行了验证.本文中提出的各种奇点的性质,将作为以后操纵率设计的依据.

参考文献 (References)

[1] 朱敏.空间站的力矩陀螺控制[D].北京:北京航空航天大学宇航学院, 1997
Zhu Min. CMG control for orbital space stations[D]. Beijing: School of Astronautics, Beijing University of Aeronautics and Astronautics, 1997(in Chinese)

[2] 汤亮.使用控制力矩陀螺的航天器姿态动力学与控制问题研究[D].北京:北京航空航天大学宇航学院, 2005
Tang Liang. A research on attitude dynamics and control of spacecraft with control moment gyros[D]. Beijing: School of Astronautics, Beijing University of Aeronautics and Astronautics, 2005(in Chinese)

[3] Meffe M. Control moment gyroscope configurations for the space station[R]. AAS-88-040, 1988

[4] Wie B. Computation and visualization of control moment gyroscope singularities[R]. AIAA-2002-4570, 2002

[5] Yoshikawa T. A steering law for three double-gimbal control moment gyro systems[R]. NASA TM-X-64926, 1975

[6] Wie B. Singularity analysis and visualization of single-gimbal control moment gyro systems[R]. AIAA-2003-5678, 2003

(责任编辑:张 嵘)

(上接第 1075页)

4 结束语

本文给出了适用于第一人称射击游戏角色运动特征的可预测错误的自适应位置外推算法.根据此类游戏角色运动方向多变、轨迹呈连续折线的特点,选取一阶外推方程和直线收敛算法用于位置预测与收敛,通过检测可能导致推算错误的角色运动行为,及时发送状态更新,并在收敛过程中根据角色运动状态的改变对收敛进行分级调整.根据推算角色与观察者的距离改变推算策略,以减少需要发送的状态更新数量和推算轨迹的平均偏差.与DIS标准中DR算法的对比测试表明,EPADR算法在第一人称射击游戏中可以减小游戏的网络通信量并提高位置预测精度.

参考文献 (References)

[1] Berglund E J, Cheriton D R. Amaze: A multiplayer computer game[J]. IEEE Software, 1985, 2(3): 30-39

[2] IEEE 1278-1996, Standard for distributed interactive simulation application protocols[S]

[3] Singhal S K, Cheriton D R. Using a position history-based protocol for distributed object visualization[R]. CS-TR-94-1505, 1994

[4] Cai Wentong, Lee Francis B S, Chen L. Auto-adaptive dead reckoning algorithm for distributed interactive simulation[C]// Proceedings of the Workshop on Parallel and Distributed Simulation. Los Alamitos, CA: IEEE, 1999: 82-89

[5] Duncan T P, Gracanin D. Pre-reckoning algorithm for distributed virtual environments[C]// Winter Simulation Conference Proceedings. Los Alamitos, CA: IEEE, 2003, 2: 1086-1093

[6] Aggarwal S, Banavar H, Khandelwal A, et al. Accuracy in dead-reckoning based distributed multi-player games[C]//Proceedings of the ACM SIGCOMM 2004 Workshops. New York: Association for Computing Machinery, 2004: 161-165

[7] Katz A, Kenneth G. Dead reckoning for airplanes in coordinated flight[C]//Proc of the Tenth Workshop on Standards for Interoperability of Defense Simulations. Orlando, Florida: SISO, 1994: 5-13

[8] Capin T, Esmerado J, Thalmann D. Dead-reckoning technique for streaming virtual human animation[J]. IEEE Transactions on Circuits and Systems for Video Technology, 1999, 9(3): 411-414

[9] Pantel L, Wolf L C. On the suitability of dead reckoning schemes for games[C]//Proc of 1st Workshop on Network and System Support for Games. New York: Association for Computing Machinery, 2002: 79-84

[10] Marshall D, Delaney D, McLoone S, et al. Exploring the spatial density of strategy models in a realistic distributed interactive application[C]//Proc of the 8th IEEE International Symposium on Distributed Simulation and Real-Time Applications. Los Alamitos: IEEE, 2004: 210-213

[11] Lin KuoChi, Wang Morgan, Wang Jie, et al. Smoothing of dead reckoning image in distributed interactive simulation[J]. Journal of Aircraft, 1996, 33(2): 450-452

[12] Singhal S K. Effective remote modeling in large-scale distributed simulation and visualization environments[D]. Stanford, CA: Stanford University, 1996

(责任编辑:文丽芳)