doi:10.3772/j.issn.1002-0470.2021.05.006

实时虚拟场景漫游路径的优化算法①

赵晓东② 石程豪③

(河北科技大学信息化建设与管理中心 石家庄 050018)

(河北科技大学信息科学与工程学院 石家庄 050018)

摘 要 传统的虚拟路径漫游是在固定方向路径上的漫游,而实时任意方向上的虚拟路径漫游在路径曲线处有抖动问题且计算量大,使得虚拟现实(VR)场景运行缓慢、晃动和卡顿,给人带来视觉上的晕眩感。针对以上问题提出了基于 Dynapath 算法、微分路径及改进的 Cardinal 曲线插值算法相结合的优化算法。使用 Dynapath 算法规划实时路径和等时间微分路径,并对微分线段等截距处进行初步平滑处理。最后对该路径使用改进的 Cardinal 曲线插值算法拟合出该曲线。测试结果表明,本文算法对实时任意方向虚拟漫游路径的拟合程度优于其他方法,曲线平滑去抖动效果好、计算量小、FPS 值稳定。

关键词 Dynapath 算法; Cardinal 曲线插值算法; 实时虚拟路径漫游; 去抖动; 平滑处理

0 引 言

虚拟现实(virtual reality,VR)场景中摄像机漫游路径的设计直接影响虚拟漫游的真实感和沉浸感,决定摄像机路径的主要因素是漫游路径规划的合理性和曲线路径的平滑性,不合理的路径规划导致计算量大,造成一定的不稳定性。曲线不平滑导致路径在曲线处存在抖动现象,产生严重的晕眩感。

近年来国内外研究虚拟场景漫游多是在固定路径上,少有在实时任意方向上,实时任意方向上摄像机角度变化多,产生大量弯曲路径,导致延迟与抖动更剧烈。实时虚拟漫游多采用改进遗传算法、改进蚁群算法等路径预测的规划方法,但预测后的路径在短距离路径上效果明显,长距离路径上仍存在抖动现象,普适性低。现阶段普遍的优化方法是基于路径预测算法并结合 Bézier 曲线[1]、Hermite 曲线、B 样条曲线等插值算法优化路径,可以在一定程度上处理长距离路径中曲线部分的平滑度,这些方法需要在每一段路径中的曲线部分添加额外控制点。在实时路径中产生的曲线数目多,需要的控制点将会更多,算法复杂、计算量大,从而导致 VR 场景延迟卡顿不稳定。

虚拟现实技术越来越受到研究者的关注。文献[2]利用 probabilistic road map 算法,通过交互或自动路径规划,使虚拟人物在大型场景中漫游。文献[3]对地形模型进行信息配置,提前设定场景中的路径信息,确定摄像机的运动路径。文献[4]采用路径拼接的方法利用路径中路径点的索引对多条路径的路径点进行组合,并对交叉点或者连接点的旋转和俯仰角度进行重新计算。使用二次 B 样条曲线插值处理拟合路径。文献[5]使用八叉树将场景分层构建连接图,使用 A* 算法找到最短路径。文献[6]基于二次有理 Bézier 曲线插值算法对虚拟漫游路径进行优化,并通过提供首尾控制点的位置以及添加 3 个控制点上的形状参数实现对路径的优化。

其中文献[5,6]通过圆弧对路径顿挫处进行平滑处理,其效果明显。但圆弧的弧度大小只能通过

① 河北省重点研发计划(18210803D),河北省科技厅科技支撑计划(17210803D)和河北省教育厅青年基金(QN2018095)资助项目。

② 男,1973 年生,博士,教授; 研究方向:电机与电器,VR 技术,人脸识别;E-mail: zhaoxiaodong@hebust.edu.cn

③ 通信作者,E-mail: 369959287@qq.com

(收稿日期:2020-04-27)

半径调节，相切的两条边必须符合截距相等这一条件，这无法满足多数路径，并且对于处理长距离路径效果不理想。文献[6]适用所有曲线，但插值拟合个别采样点前后平滑度不够。

在文献[5,6]的基础上，本文提出使用 Dynapath 算法、微分路径及改进的 Cardinal 曲线算法相结合的优化算法，来解决实时漫游路径的规划和曲线处抖动的问题。使用 Dynapath 算法对路径进行提前规划，降低 VR 场景运行时计算机的计算量；使用微分路径快速提取采样点，降低对长路径处理的不稳定性，对微分后的路径进行初步平滑，优先处理一部分等截距路径点；最后对不等截距的路径点通过改进的 Cardinal 曲线插值算法对曲线进行平滑处理，Cardinal 曲线插值算法对曲线进行平滑处理时不需要增加额外的控制点，计算量小、处理实时路径效果好[7]。该算法不局限于路径线截距问题，且对曲线平滑处理效果好，对所有路径具有普适性。

1　实时虚拟场景漫游路径的优化算法

该算法通过使用 Dynapath 算法实时预先规划路径[8]，对该路径进行微分处理，并对微分后的路径进行初步平滑处理，再对平滑处理后的路径使用 Cardinal 曲线等距离插值算法拟合出最终漫游路径。

1.1　预先规划路径

在起始点处使用 Dynapath 算法进行路径计算，通过有限范围内的穷举搜索，得到一棵路径树。路径树的范围为 ±90 °（符合正常漫游角度变化），如图1所示。通过VR场景中物体的运动初速度和角度信息与路径树匹配，确定最符合的路径轨迹。图中给出的是 3 个采样周期内得到的轨迹树。当每个周期 T 结束时会记录下此时坐标作为采样点。算法提前规划路径，错开与 VR 场景动画同时运行，减少计算机计算量，降低延时。

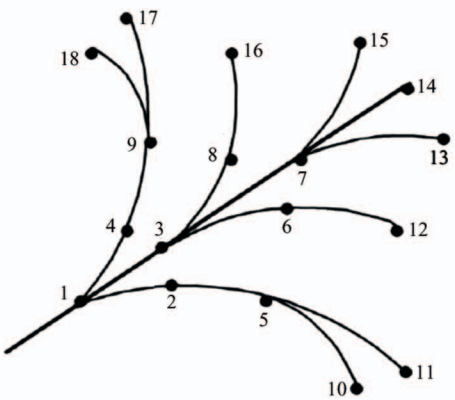


图 1　轨迹树

路径树产生的路径周期相比较 VR 场景的帧数刷新率时间较长，因此还是存在部分延迟现象，所以还需对路径进一步处理。

1.2　微分路径

首先定义每一个周期为 Δt，令 $\Delta t = 0.05$ s。同时把本周期结束时的路径点作为下一个周期的起始路径点，即 (x_i, y_i, z_i)，$i = 1,2,3,\cdots,N_0$。R_k 为第 k 条路径的方向，如图 2 所示，其中 $k = 1, 2, 3, \cdots, n$。虚线段为微分后的路径，实线段为原路径。

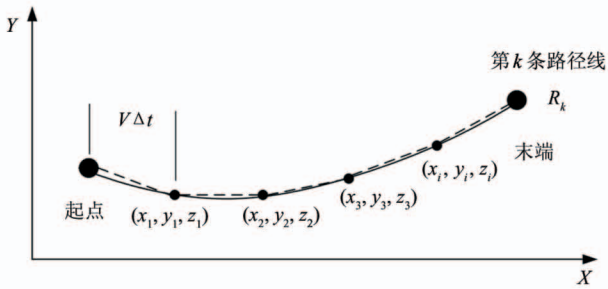


图 2　第 k 条路径线

以第 R_k 条路径为例，把整个路径大周期 T 微分为一个个小周期 Δt。因为在整个路径中，单个周期时间比较短，则整个曲线路径就可以看作是由多个小的直线路径组成。假设在时间 Δt 期间内 $(x_{i+1}, y_{i+1}, z_{i+1})$ 为下一时刻的位置坐标，(x_i, y_i, z_i) 为初始点坐标，虚拟场景中运动模型的位移为

$$\begin{cases} \Delta t_x = \Delta t \times v_x \\ \Delta t_y = \Delta t \times v_y \\ \Delta t_z = \Delta t \times v_z \end{cases} \tag{1}$$

其中，v_x、v_y、v_z 为小周期的初速度分量，Δt_x、Δt_y、Δt_z 为该周期结束时的位移距离，则该周期结束时的坐标为

$$\begin{cases} x_{i+1} = x_i + \Delta t_x \\ y_{i+1} = y_i + \Delta t_y \\ z_{i+1} = z_i + \Delta t_z \end{cases} \tag{2}$$

由式(1)和式(2)可得：

$$\begin{cases} x_{i+1} = x_i + \Delta t \times v_x \\ y_{i+1} = y_i + \Delta t \times v_y \\ z_{i+1} = z_i + \Delta t \times v_z \end{cases} \tag{3}$$

由式(1)~式(3)得到矩阵形式为

$$(x_{i+1} \quad y_{i+1} \quad z_{i+1} \quad 1) = (x_i \quad y_i \quad z_i \quad 1) \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ v_x\Delta t & v_y\Delta t & v_z\Delta t & 1 \end{bmatrix} \tag{4}$$

可以计算得出下一时刻路径点的坐标位置,由此得到运动轨迹。这样解决了因路径过长导致的延迟、运行不稳定等问题。但路径不够平滑会出现强抖动感,下面将去除抖动问题。

1.3 路径初步平滑处理

由此规划出来的路径是由若干直线段构成,在线段与线段的连接处会出现一些不平滑的钝角,需要对路径进行初步平滑处理[9],如图 3 所示,这样可以处理掉部分截距相等的路径,减少下文中插值处理的曲线个数,降低计算量。

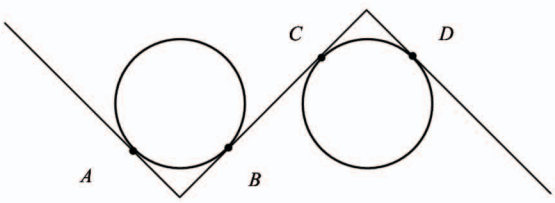


图 3 初步平滑处理

做一个虚拟圆,通过虚拟圆与相邻两个线段相切对路径进行平滑处理,圆半径的计算公式为

$$R = v^2/g \tag{5}$$

其中,R 为转弯半径,v 为场景中模型的行驶速度,g 为重力加速度常数(约为 9.8 m/s^2)。

1.4 Cardinal 曲线等距离插值

1.4.1 插值方式分析

分析造成曲线处延迟的原因。通过设置一个固定的漫游路径,如图 4 所示,并利用传统的 Cardinal 曲线等时间插值方式来测试直线与曲线上的漫游时间是否相同。图 4 中,$AB = BC = CD = 180$ m,设定速度为 5 m/s(任意设定),插值数 n 分别为 30 个、50 个、100 个的情况下测试行驶在曲线与直线上的漫游时间,其中时间单位为 s。

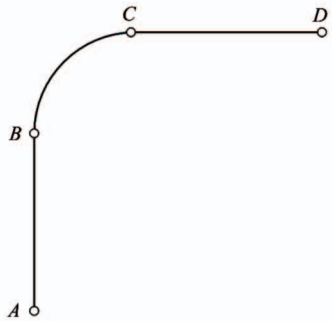


图 4 固定路径

通过表 1 可以得出,直线部分与曲线部分漫游时间差距大,将出现延迟情况。

表 1 等时间插值所需漫游时间

n/个	AB/s	BC/s	CD/s
30	36.88	38.50	36.96
50	42.22	45.84	42.28
100	46.42	48.12	46.30

为了解决这一问题,本文使用 Cardinal 曲线等距离插值。同样使用图 1 中的路径在直线与曲线上进行插值,插值数 n 分别为 30 个、50 个、100 个。速度设定为 5 m/s(与等时间插值实验速度一致),测试情况如表 2 所示。

表 2 等距插值所需漫游时间

n/个	AB/s	BC/s	CD/s
30	36.12	36.69	36.20
50	40.82	41.06	40.84
100	44.42	44.50	44.41

通过表 1 和表 2 的测试结果发现,等距离插值中直线与曲线之间漫游时间起伏小且较稳定,很好地消除了因曲线处漫游时间长导致的延迟。同一线段漫游时间低于等时间插值方式,有效地提高了计算机的计算效率,因此本文采用等距离插值方式。

1.4.2 Cardinal 曲线插值算法的优化

为了进一步消除延迟抖动,对初步平滑处理后的路径进行插值处理。由第 1.4.1 节中的实验分析

可知，同等条件下 Cardinal 曲线等距插值效果比等时间插值效果好。因此本文采用 Cardinal 曲线等距插值方法对曲线进行插值拟合，Cardinal 曲线与控制点的关系如图 5 所示。

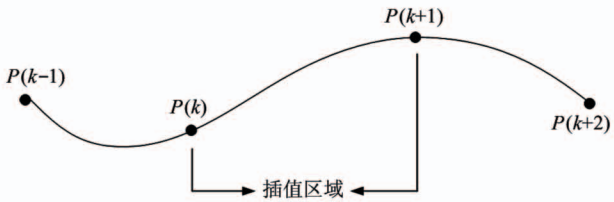


图 5　插值范围

传统 Cardinal 曲线插值方式是将 t 的值从 $1/m$、$2/m$、$3/m$ 依次递增至 1 即可在每两个路径点之间生成 m 个插值点[10]。区别于传统方法，本文将优化为使用等距离插值，因为在 Dynapath 算法中已经把路径微分为多个等周期的小路径，产生的是类等周期路径，周期为 Δt。Δt =0.05 s 是瞬时运动，速度变化量非常小，因此可视为匀速运动。

$$L = \Delta t \times \Phi_i \tag{6}$$

其中 Δt 固定，Φ_i 为初速度，且运动为匀速运动，所以 L 为固定值，如图 6 所示。因此按照进行等距离插值，可以避免重新把该路径的漫游时间计算出来再对时间进行插值，节省了计算机的运算量。

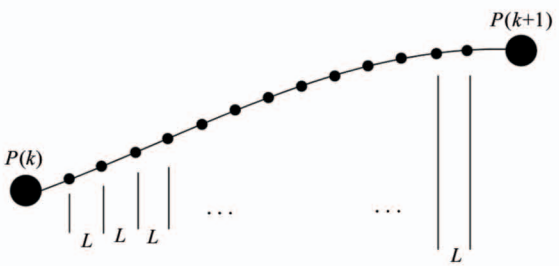


图 6　等距离插值

Cardinal 样条曲线是插值分段三次曲线，一个样条需要 4 个连续控制点给出，可以指定如下的边界条件：

$$P_{(0)} = P_{(k)} \tag{7}$$

$$P_{(1)} = P_{(k+1)} \tag{8}$$

$$P'_{(0)} = \frac{1}{2}(1-t)(P_{(k+1)} - P_{(k-1)}) \tag{9}$$

$$P'_{(1)} = \frac{1}{2}(1-t)(P_{(k+2)} - P_{(k)}) \tag{10}$$

其中，t 是 Cardinal 曲线的的张量参数，控制着该曲线与控制点之间的松紧程度，在程序中设置 t = 0[11]。通过式(9)和式(10)可知，曲线的切线值由控制点本身决定，不需要额外输入。根据边界条件可得 Cardinal 样条插值公式为

$$P(v) = (v^3 \quad v^2 \quad v \quad 1)\begin{bmatrix} -x & 2-x & x-2 & 5 \\ 2x & x-3 & 3-2x & -x \\ -x & 0 & x & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}\begin{bmatrix} P_{(k-1)} \\ P_k \\ P_{(k+1)} \\ P_{(k+2)} \end{bmatrix} \tag{11}$$

其中 $P_{(k-1)}$、P_k、$P_{(k+1)}$、$P_{(k+2)}$ 为曲线上的点。已知曲线上 4 个点 $P_{(k-1)}$、P_k、$P_{(k+1)}$、$P_{(k+2)}$ 的坐标，代入式(11)，读取参数 v 的值，即可获得 $P_kP_{(k+1)}$ 曲线段上不同的插值点。其中，当 v =0 时，插值点位于点 $P_{(k-1)}$ 处；当 v =1 时，插值点位于点 $P_{(k+2)}$ 处。要获得 n 个插值点，只需让 v 的值从 $L_1, L_2, L_3 \cdots L_n$ 一直递增到 1，通过 Cardinal 曲线插值公式对场景摄像机[12]位置进行插值和控制，展开方程：

$$\begin{aligned} P(v) = {} & P_{k-1}(-xv^3 + 2xv^2 - xv) \\ & + P_k[(2-x)v^3 + (x-3)v^2 + 1] \\ & + P_{k+1}[(x-2)v^3 + (3-2x)v^2 + xv] \\ & + P_{x+2}(xv^3 - xv^3) \end{aligned} \tag{12}$$

其中 $x = (1-t)/2$，这样将拟合得到最终的平滑路径。

1.5　算法全过程优化流程图

算法流程图如图 7 所示，其具体步骤如下。

步骤 1　使用 Dynapath 算法搜索出路径树。

步骤 2　从初始节点出发，根据 VR 场景中移动物体的角度大小，匹配路径树中与此角度相同的路径。

步骤 3　对匹配后的路径进行微分处理并计算出微分节点的坐标位置。依次计算出一个周期内的所有节点连接各节点形成路径。微分长路径，提高计算的稳定性。

步骤 4　对微分后的路径进行初步平滑处理。处理截距相等的路径，减少下文中插值处理的曲线次数，降低计算量。

步骤 5　对平滑处理后的路径使用改进的 Car-

dinal 曲线插值算法通过等距离插值拟合出最终路径。

步骤 6 每当角度发生改变,重复步骤 1 ~ 步骤 5。

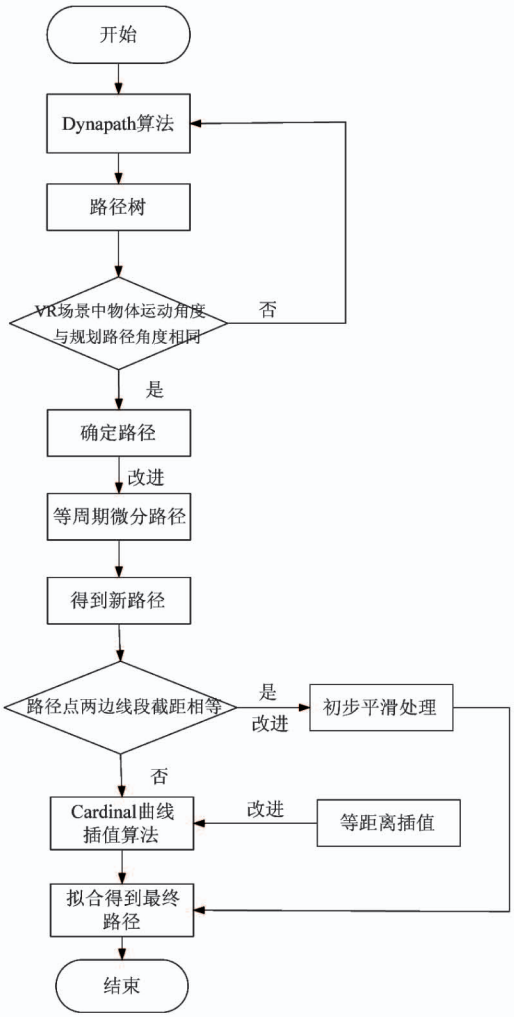
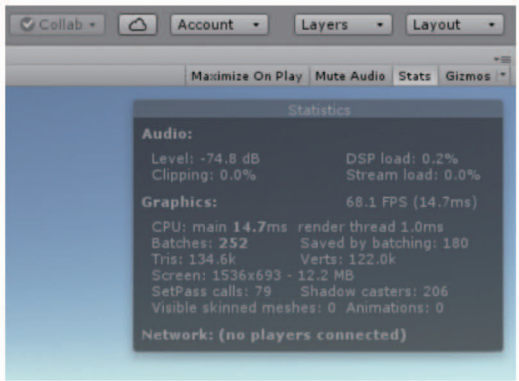


图 7 算法流程图

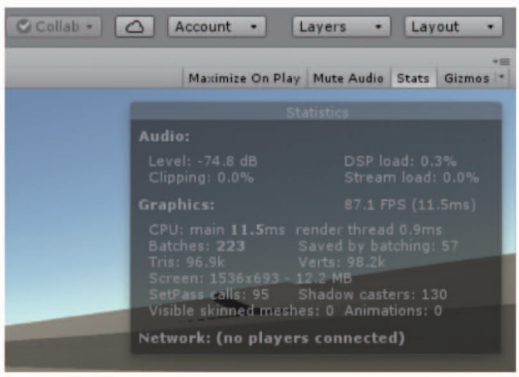
2 测试结果分析

2.1 运行帧数对比

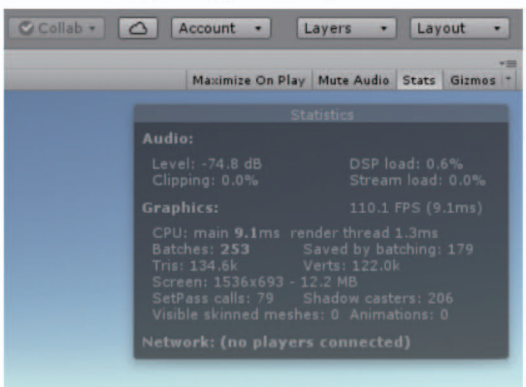
在同一台电脑上使用 Unity3d 软件对同一条曲线路径进行测试。图 8 为 3 种算法的最高 fps 截图。表3为连续7 s内测试结果,由图8可知,文献[5]算法最高为 68.1 fps,平均值为 62.8 fps,方差为 50.92。文献[6]算法最高为 87.1 fps,平均值为 82.9 fps,方差 57.26。本文算法最高为 110.1 fps,平均值为 107.1 fps,方差为 29.96。本文算法 fps 高且方差小,使场景运行更加流畅稳定、无延时卡顿感。



(a) 文献 [5] 方法最高 fps 值



(b) 文献 [6] 方法最高 fps 值



(c) 本文方法最高 fps 值

图 8 运行帧数比较

表 3 不同算法 FPS 数值比较

算法/s	1	2	3	4	5	6	7
文献[5]算法 fps	68.1	64.4	61.5	58.8	55.2	53.1	50.3
文献[6]算法 fps	87.1	85.4	93.1	81.2	77.4	73.9	70.3
本文算法 fps	110.1	107.7	105.2	104.3	102.2	98.8	97.3

2.2　仿真结果对比

由图 9 ~ 图 11 可知，当规划出路径后，采用 3 种不同插值算法对比可知，图 9（对照实验组）经 Bézier 曲线插值处理后的曲线在第 6、7、8 点时平滑，其余各点效果都不好。图 10 采用文献[6]二次有理 Bézier 方法，路径大部分光滑，在第 7、10、14 点时仍存在不平滑现象。图 11 经本文算法处理过的曲线各点顺畅平滑，符合真实的运行轨迹，去抖动效果突出。

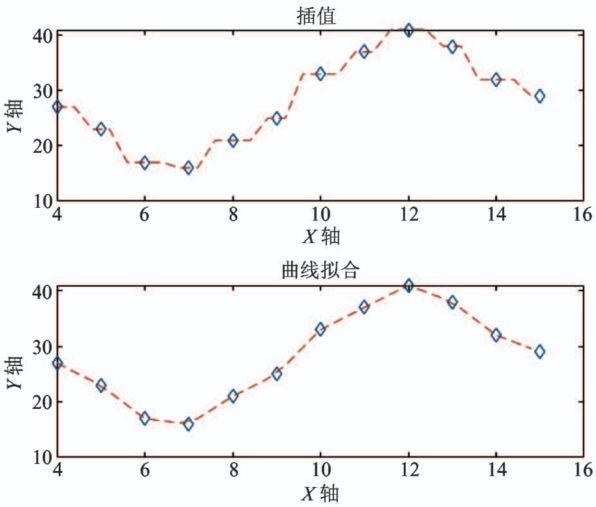


图 9　Bézier 曲线插值

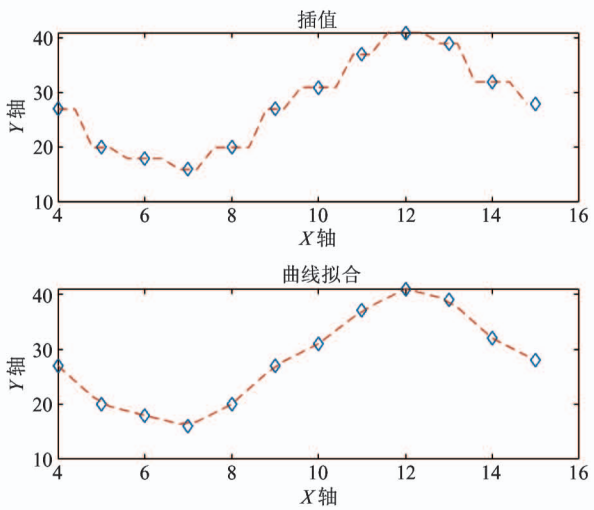


图 10　文献[6]插值方式拟合

3　结　论

传统虚拟路径的漫游，通过设置固定方向路径，使用曲线对固定路径进行等时间插值，计算量大且只适合在固定路径中使用，实时任意方向的虚拟路径因产生路径曲线较多，导致延迟与抖动现象明显。本文提出了一种适合实时路径漫游的方法。该方法通过 Dynapath 算法预测出下一周期的路径，通过反向匹配锁定最符合的路径，并把该路径微分为等距离的直线段，提高处理长距离曲线的稳定性。对微分后的路径进行初步平滑处理，对处理后的曲线通过 Cardinal 曲线进行等距离插值处理，最终拟合出该路径。实验结果表明，该优化算法可有效地解决实时虚拟路径漫游中的延迟与抖动问题。

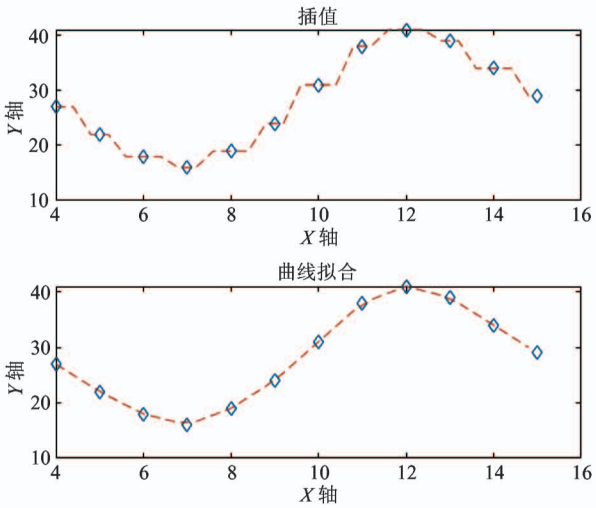


图 11　改进的 Cardinal 曲线插值

参考文献

[1] 陈嘉林，魏国亮，田昕. 改进粒子群算法的移动机器人平滑路径规划[J]. 小型微型计算机系统，2019，40(12)：2550-2555

[2] Salomon B，Garber M，Lin M C，et al. Interactive navigation in complex environments using path planning[C]//Proceedings of Symposium on Interactive 3D Graphics，Monterey，USA，2003：41-50

[3] Christie M，Languenou E，Granvilliers L. Modeling camera control with constrained Hypertubes[C]//Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming，Ithaca，USA，2002：618-632

[4] 修春华，孙秀娟，车德福，等. 三维场景中虚拟漫游路径的优化设计方法[J]. 金属矿山，2015(4)：242-245

[5] 史红兵，毅彬，童若峰，等. 虚拟场景自动漫游的路径规划算法[J]. 计算机辅助设计与图形学学报，2006，

18(4):592-597

[6] 孙红岩, 李翠芳, 孙晓鹏. 基于二次有理 Bezier 方法的虚拟漫游路径优化[J]. 计算机工程与设计, 2013, 34(11):3912-3915, 3928

[7] 罗立宏, 张群英, 冯开平,等. 基于 Cardinal 的虚拟场景自动漫游算法[J]. 桂林工学院学报, 2007, 27(2):278-281

[8] 马向玲, 王欣欣, 王永生,等. 基于滑动时间窗的飞行器航路动态优化方法[J]. 飞行力学, 2010,28(5):92-96

[9] 吴琼. 智能机器人路径规划中的蚁群算法改进[J]. 计算机与网络, 2017,43(9):48-49

[10] 江伟, 章仁江. 保广义凸的曲线插值方法[J]. 计算机辅助设计与图形学学报, 2018,30(9):1686-1691

[11] Grigorieff R D. On cardinal spline interpolation[J]. *Computational Methods in Applied Mathematics*, 2013, 13(1):39-54

[12] Nieuwenhuisen D, Overmars M. Motion planning for camera movements in virtual environments, UUCS-2003-004[R]. Utrecht: Utrecht University, 2003

The optimization algorithm of real-time virtual scene roaming path

Zhao Xiaodong, Shi Chenghao

(Information Construction and Management Center of Hebei University of Science and Technology, Shijiazhuang 050018)

(School of Information Science and Engineering, Hebei University of Science and Technology, Shijiazhuang 050018)

Abstract

The traditional virtual path roaming is to roam on the fixed direction path, while the real-time virtual path roaming in any direction has jitter problem at the path curve, and the amount of calculation is large, which makes the virtual reality (VR) scene running slowly, shaking and stuck, and brings visual dizziness. Aiming at the above problems, a new interpolation optimization algorithm is proposed based on Dynapath algorithm, differential path and improved Cardinal curve interpolation algorithm. The Dynapath algorithm is used to plan the real-time path and the equal time differential path, and the differential line segment with equal intercept is smoothed preliminarily. Finally, the improved Cardinal curve interpolation algorithm is used to fit the path. The test results show that the fitting degree of the algorithm is better than other methods, the curve is smooth and the jitter is good, the calculation is small, and the FPS value is stable.

Key words: Dynapath algorithm, Cardinal curve interpolation algorithm, real-time virtual path, debounce, Smoothing processing