计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(7):101−108 [doi: 10.15888/j.cnki.csa.006975]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于马尔可夫决策过程的群体动画运动轨迹生成①

刘俊君, 杜艮魁

(北京工业大学 信息学部 计算机学院, 北京 100124)
通讯作者: 刘俊君, E-mail: 1808935716@qq.com

摘　要: 近些年来, 群体动画在机器人学、电影、游戏等领域得到了广泛的研究和应用, 但传统的群体动画技术均涉及复杂的运动规划或碰撞避免操作, 计算效率较低. 本文提出了一种基于马尔可夫决策过程 (MDPs) 的群体动画运动轨迹生成算法, 该算法无需碰撞检测即可生成各智能体的无碰撞运动轨迹. 同时本文还提出了一种改进的值迭代算法用于求解马尔可夫决策过程的状态-值, 利用该算法在栅格环境中进行实验, 结果表明该算法的计算效率明显高于使用欧氏距离作为启发式的值迭代算法和 Dijkstra 算法. 利用本文提出的运动轨迹生成算法在三维 (3D) 动画场景中进行群体动画仿真实验, 结果表明该算法可实现群体无碰撞地朝向目标运动, 并具有多样性.

关键词: 群体动画; 马尔可夫决策过程; 运动轨迹; 值迭代

引用格式: 刘俊君,杜艮魁.基于马尔可夫决策过程的群体动画运动轨迹生成.计算机系统应用,2019,28(7):101–108. http://www.c-s-a.org.cn/1003-3254/6975.html

Motion Trajectory Generating Algorithm Based on Markov Decision Processes for Crowd Animation

LIU Jun-Jun, DU Gen-Kui

(College of Computer Science, Faculty of Information Technology, Bejing University of Technology, Bejing 100124, China)

Abstract: Crowd animation has been researched and applied in many domains in recent years, such as robotics, movies, games, and so on. But the traditional technologies for creating crowd animation all need complex calculating for motion planning or collision avoidance, the computing efficience is low. This paper presents a new algorithm for generating motion trajectory based on Markov Decision Processes (MDPs) for crowd animation, it can generate all agents' collision-free motion trajectories without any collision detecting. At the same time, this paper presents a new improved value iteration algorithm for solving the state-values of MDPs. We test the performance of the new improved value iteration algorithm on grid maps, the experimental results show that the new alogithm outperforms the value iteration algorithm using Euclidean distance as heuristics and Dijkstra algorithm. The results of crowd animation simulating experiments using the motion trajectory generating algorithm in three-dimensional (3D) scenes show that the proposed motion generating algorithm can make all agents move to the goal position without any collision, meanwhile, agents' motion trajectories are different when we run the algorithm at different time and this effect makes the crowd animation much more alive.

Key words: crowd animation; Markov Decision Processes (MDPs); motion trajectory; value iteration

① 收稿时间: 2019-01-10; 修改时间: 2019-02-03; 采用时间: 2019-02-18; csa 在线出版时间: 2019-07-01

群体动画是群体行为动画的简称[1], 是指在特定环境下对群体行为的模拟. 群体动画在近些年来得到了广泛的研究和应用, 如机器人学[2]、社会学[3]、交通工程[4]、电影[1]、游戏[5]等. 目前实现群体动画的方法主要有基于力的方法[6]、基于速度的模型[7]、基于场的模型[8]、基于梯度的方法[9]、基于群智能的方法[10]、基于深度强化学习的方法[11]等. 这些方法在对智能体的每一步动作进行规划时, 均需大量的计算对智能体进行碰撞检测或获取当前智能体的环境状态以引导智能体向目标运动, 当智能体的数量很大时, 大量的计算会导致效率较低. 针对这一问题, 本文提出了一种基于马尔可夫决策过程 (Markov Decision Processes, MDPs) 的群体动画运动轨迹生成算法, 该算法在规划智能体的每一步动作时, 无需进行碰撞检测等大量的计算即可实现智能体无碰撞地向目标运动, 因此智能体的轨迹生成效率较高. 同时为了快速求解马尔可夫决策过程的状态-值, 本文提出了一种改进的值迭代算法, 该算法利用贪婪的思想计算出每个状态的初始状态-值, 再将该初始状态-值输入经典的值迭代算法进行迭代优化, 以保证每个状态最终的状态-值能满足预先设定的阈值要求. 在栅格环境中对改进的值迭代算法进行实验, 结果显示此算法的计算效率明显高于使用欧氏距离作为启发式的值迭代算法和 Dijkstra 算法. 在三维 (3D) 动画场景中对本文提出的运动轨迹生成算法进行仿真实验, 结果表明此算法可实现群体无碰撞地朝向目标运动, 并具有多样性.

本文后续结构安排如下: 第 1 节介绍基于马尔可夫决策过程的群体动画建模; 第 2 节介绍本文提出的改进的值迭代算法和群体动画运动轨迹生成算法; 第 3 节介绍实验部分, 并对实验结果进行分析; 第 4 节给出结论与展望.

1 基于马尔可夫决策过程的群体动画建模

马尔可夫决策过程常被用来作为序列决策问题的框架, 在近些年来得到了大量的研究与应用, 如路径规划[12]、深度强化学习[11]等. 本节将对马尔可夫决策过程进行形式化描述, 并利用马尔可夫决策过程对群体动画进行建模, 最后介绍求解马尔可夫决策过程状态-值的值迭代算法.

1.1 马尔可夫决策过程

马尔可夫决策过程常被用来作为序列决策问题的框架, 可以用一个五元组(S,A,P,R,γ)来表示. 其中S代表环境状态的集合, A代表智能体所有动作的集合, $P(s,a,s')$表示智能体在状态s执行动作a到达状态s'的概率, $R(s,a,s')$表示智能体在状态s执行动作a到达状态s'所得到的回报, γ表示折扣因子.

当马尔可夫决策过程用来表示序列决策问题后, 求解马尔可夫决策过程即可表示为求解最优策略 (π) 的问题, 即智能体在所处状态采取最优动作以使智能体所获得的累积回报最大. 累积回报可形式化表示为智能体所处状态的价值 (即状态-值函数$v_\pi(s)$), 如式 (1) 所示.

$$v_\pi(s)=E_\pi\left(\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}|S_t=s\right) \tag{1}$$

为了方便计算状态-值函数$v_\pi(s)$, 可利用贝尔曼方程表示如式 (2) 所示.

$$v_\pi(s)=\sum_a \pi(a|s)\sum_{s',r} p(s',r|s,a)(r+\gamma v_\pi(s')) \tag{2}$$

当$v_\pi(s)\geqslant v_{\pi'}(s)$对所有的$s\in S$和$\pi'$成立时, π为最优策略 (后文用$v_*(s)$表示最优策略对应的状态-值函数), 即智能体在每个状态都采取的最优动作, 如式 (3) 所示.

$$\pi(s)=argmax_a\sum_{s',r} p(s',r|s,a)(r+\gamma V_\pi(s')) \tag{3}$$

求解最优策略常见的方法有策略迭代和值迭代[13]算法. 因值迭代方法在本文所研究的问题中具有更高的计算效率, 本文采用值迭代方法进行求解, 并在 1.3 节给出针对群体动画模型特点的值迭代算法, 在第 2 节中给出本文所提出的改进的值迭代算法.

1.2 群体动画建模

利用马尔可夫决策过程对群体动画进行建模, 即对群体动画的运动环境和在每个环境状态下智能体所能采取的动作进行建模. 本小节首先介绍对群体动画的运动环境进行建模, 然后再利用上一小节介绍的马尔可夫决策过程对群体动画的运动行为进行刻画.

本文采用栅格法[14,15]对环境状态进行建模, 即用矩阵 ($grid$) 的形式对环境状态进行存储表示, 其中将二维平面空间环境用二维矩阵进行存储表示, 将三维立体空间环境用三维矩阵进行存储表示. 首先将环境状态按参与群体运动的智能体的包围盒 (二维平面空间为圆形包围盒, 三维立体空间为球体包围盒) 的直径大小进行栅格化分, 使每个栅格区域的尺寸大小相同且

大于包围盒直径的$\sqrt{2}$倍. 然后在栅格区域中将障碍物所在区域在矩阵中标记为值 1, 自由区域在矩阵中标记为值 0, 如$grid[i][j]=1$表示坐标为(i,j)的栅格区域含有障碍物, 智能体不可进入此区域, $grid[i][j]=0$表示表示坐标为(i,j)的栅格区域为自由区域, 智能体可进入.

对环境状态进行建模后, 即可利用马尔可夫决策过程对智能体的运动行为进行建模, 形式化表示如下:

S: 环境中所有的栅格区域及障碍物信息, 即本文所采用的矩阵表示中矩阵的各元素坐标和值;

A: 智能体的动作空间, 对于二维平面空间, $A=(0,1,2,\cdots,7)$, 即 8 个移动方向对应的动作, 每个动作可使智能体在单位时间移动到对应方向的下一个栅格区域; 对于三维立体空间, $A=(0,1,2,3,\cdots,25)$共 26 个移动方向对应的动作, 其中与智能体在同一高度含 8 个移动方向对应的动作, 上下方各 9 个移动方向对应的动作;

P: 本文假设智能体的运动不受除自身动作以外的因素影响, 即智能体在每个栅格区域s, 采取任何动作a所能达到的栅格区域s'是确定的, 即$P(s,a,s')=1.0$, 同时我们规定当动作a可使智能体进入障碍物所在区域或穿越环境边界时$s'=s$, 即智能体保持在原位置, 当智能体在障碍物所在区域或目标状态区域时, 智能体将永久停留在此区域, 即障碍物所在区域和目标状态所在区域均为吸收状态. 基于此规则和假设, 在使用值迭代算法求解马尔夫决策过程的状态-值时可省略概率P;

R: 当动作a使智能体在自由区域移动时, 我们将栅格之间的欧氏距离的相反数作为智能体移动所得到的回报; 当动作a使智能体进入障碍物所在区域或穿越边界时, 智能体得到负无穷大的回报$(-Inf)$; 当智能体原来所在状态为障碍物区域或目标状态所在区域时, 任何动作a均使智能体收获值为 0 的回报;

γ: 我们取折扣因子为 1.0, 以使每个状态的状态-值的绝对值近似于此状态到目标状态的真实最短路径距离, 同时在值迭代算法求解中可省略此折扣因子, 以简化计算.

此时我们便完成了对群体动画运动环境状态的栅格法建模和智能体运动行为的马尔可夫决策过程建模. 群体相当于多个智能体, 当我们通过上述模型求解出环境中每个栅格区域所对应的状态的最优状态-值, 即$v_*(s)$时, 每个智能体都可根据式 (3) 确定其在当前状态采取的最优策略 (即最优动作), 从而间接完成了对群体动画运动行为的建模.

1.3 值迭代算法

求解马尔可夫决策过程通常采用策略迭代或值迭代[13]算法, 其中策略迭代需在每个状态-值收敛后再进行策略提升, 不断地迭代直到策略收敛, 而值迭代在状态-值每更新一次之后即进行策略提升, 再重复进行状态-值更新和策略提升直至状态-值收敛[16], 此时智能体按式 (3) 即可输出最优策略. 因值迭代算法在本文所研究的问题 (即栅格环境状态所对应的状态-值的求解) 上的效率明显高于策略迭代算法, 本文采用值迭代算法求解环境状态的状态-值. 本小节在算法 1 中给出了结合 1.2 节中群体动画模型特点的值迭代算法, 与经典的值迭代算法相比, 省略了对于概率P和折扣因子γ的考虑. 收敛后每个状态的状态-值的绝对值可近似看作此状态与目标状态的实际最短距离, 因此可以方便的与经典的最短路径算法 Dijkstra 算法的计算效率进行对比.

算法 1. 值迭代算法

1) 对每个栅格区域初始化状态-值为$v(s)=-Inf$, 初始化目标状态区域的状态-值为$v(g)=0$, 初始化阈值$\theta>0$;
2) *while true*:
3) $\Delta=0$;
4) *for s in S*:
5) $v=v(s)$;
6) $v(s)=\max(R(s,a,s')+v(s'))$;//$s'$为$s$可到达的下一状态
7) $\Delta=\max(\Delta,|v-v(s)|)$;
8) *end for*;
9) *if* $\Delta<\theta$:
10) *break*;
11) *end if*;
12) *end while*;
13) 输出$v(s)$;

由于本文的群体动画模型中智能体采取每步动作所能获得的回报R的绝对值除 0 外最小的为 1, 即智能体水平、垂直或竖直方向移动一个栅格的欧氏距离, 因此当算法 1 中的阈值θ取值为 1.0 时, 智能体也可以有效地避开障碍物, 同时基于式 (3) 可实现智能体向目标运动, 本文将在第 3 节实验部分的值迭代算法中均取阈值θ为 1.0, 以加快收敛速度.

2 改进的值迭代算法和运动轨迹生成算法

基于第 1 节中使用马尔可夫决策过程对群体动画的建模, 利用算法 1 求出每个环境状态的状态-值, 利

用式 (3) 已可以实现群体中每个智能体向目标运动, 同时可以有效地避开障碍物 (即矩阵中值为 1 所对应的栅格区域). 在此基础上, 本文提出了一种可实现智能体间无碰撞的运动轨迹生成算法, 同时该算法可保证智能体的运动具有多样性. 为了进一步提升算法 1 的收敛速度, 本文同时提出了一种改进的值迭代算法, 本节将分别予以介绍.

2.1 改进的值迭代算法

对于算法 1 进行改进以进一步提升收敛速度, 一般采用基于启发式信息作为状态-值$v(s)$的初始值[17], 或者采用启发式信息引导智能体进行探索以缩小状态-值进行更新的区域[12].

因群体动画涉及多个智能体的运动, 并且智能体初始位置不确定, 使用启发式信息作为状态-值的初始值, 再利用算法 1 进行迭代更新可求出每个环境状态的状态-值, 进而可方便求出随机初始状态的所有智能体的运动策略.

本文根据贪婪思想设计了一种新的可计算出每个环境状态的估计状态-值$v(s)$的算法 (如算法 2 所示), 将此估计值作为算法 1 中状态-值$v(s)$的初始值, 再利用算法 1 进行迭代计算, 可使计算效率明显高于使用欧氏距离作为启发式信息的值迭代算法和经典的最短路径算法 Dijkstra 算法, 本文将在第 3 部分给出实验结果.

算法 2. 初始状态-值$v(s)$估计算法

1) 初始化所有的状态-值$v(s)=-Inf$, 初始化优先列pq(按从大到小顺序), 初始化集合$block$为空集, 初始化目标状态的状态-值$v(g)=0$, $pq.put((v(g),g))$, $block.add(g)$;
2) *while not pq.empty*():
3) $s=pq.get()[1]$;
4) $ns=get\_neighbors(s)$;//获取状态s可到达的邻近区域
5) *for* s' *in ns*:
6) *if not* s' *in block*:
7) $v(s')=v(s)+R(s,a,s')$;//R为负数
8) $pq.put((v(s'),s'))$;
9) $block.add(s')$;
10) *end if*;
11) *end for*;
12) *end while*;
13) 输出 $v(s)$;

从算法 2 的伪代码中可以看出使用集合$block$存储每次计算的环境状态, 实现了每个环境状态的状态-值仅计算一次, 同时使用优先队列 (基于堆的数据结构实现) 存储每次计算的状态, 可快速地实现状态按状态-值大小排序.

2.2 运动轨迹生成算法

通过算法 1 求出每个环境状态的状态-值$v(s)$后, 即可根据 (3) 式求出每个智能体在当前状态的最优策略, 即最优动作a, 但此时智能体之间可能会发生碰撞. 为了避免智能体之间发生碰撞, 同时为了保证智能体的运动具有多样性, 本文设计了一种运动轨迹生成算法 (如算法 3 所示).

为了保证智能体向目标状态所在区域运动并且保证智能体之间不发生碰撞, 在智能体当前状态s_t采取动作a_t所有可到达状态s_{t+1}(将其他智能体当前所在状态作为该智能体的临时障碍物区域, 即算法 3 第 6 行矩阵$block$中元素为 1 的位置) 中我们优先考虑满足$v(s_{t+1})>v(s_t)$的所有状态s_{t+1}, 当智能体没有满足此条件的状态s_{t+1}时, 我们考虑其他的可到达状态s_{t+1}以及智能体的当前状态s_t, 即当智能体有更接近目标的状态时选取更接近的目标状态, 否则智能体选取其他的可到达状态进行探索, 以便在将来可能进入一个更好的状态.

对于所有的待选取状态s_{t+1}, 我们引入 Boltzmann 分布和轮盘赌方法从中选择一个状态作为智能体运动的下一个状态 (Boltzmann 分布如公式 (4) 所示).

$$P(k)=\frac{e^{\frac{v'(k)}{\tau}}}{\sum_{i=0}^{K}e^{\frac{v'(i)}{\tau}}} \tag{4}$$

取$v'(k)=v(k)-max(v(0),v(1),\cdots,v(K))$, $v'(i)=v(i)-max(v(0),v(1),\cdots,v(K))$以消除状态-值$v(s)$之间的数量级差异, 同时取$\tau$为常数 1, 方便计算同时不影响效果.

为了计算同一时间群体中每个智能体可采取的动作, 即下一个单位时间各智能体所能达到的状态, 我们对每个智能体按当前所在状态的状态-值$v(s)$对智能体进行排序, 其中状态-值越大的智能体优先级越高, 按此优先级顺序依次为智能体按上述方法选取下一个状态并添加到此智能体的运动轨迹队列中, 当所有智能体下一个状态均计算完毕后, 再重复此过程, 直到达到最大的轨迹长度. 按此方法可为每个智能体生成一条运动轨迹, 同时可保证智能体按运动轨迹进行运动时不会发生碰撞.

为了使智能体可以在有运动的障碍物环境中进行无碰撞运动, 我们每隔一段时间检测障碍物位置是否发生变化, 如果发生变化, 我们将障碍物当前位置及一定帧数后的位置所覆盖的区域均作为障碍物处理, 更

新表示栅格环境状态的矩阵, 再重新根据算法 1 计算各状态的状态-值, 然后再按上述步骤继续规划各智能体的运动轨迹.

算法 3. 动轨迹生成算法

1) 随机初始化各智能体的初始状态 (均在自由区域), 为每个智能体 *agent* 设置一个数组用于存储智能体的轨迹*L*[*agent*], 将各智能体初始状态存入对应的*L*[*agent*]中, 初始化矩阵 *block*(大小同环境状态矩阵 *grid*), 在 *block* 中将各智能体初始状态对应位置的元素置为 1, 其余位置的元素置为 0,初始化轨迹长度计数*length*=0;

2) *while length*<*maxLength*:

3) 根据算法 1 中输出的$v(s)$对各智能体排序, 获取排序结果*Agents*;

4) *for agent in Agents*:

5) 在 *block* 中将 *L*[*agent*]的尾部状态对应位置的元素置为 0;

6) 根据*L*[*agent*]的尾部状态及矩阵*block* 获取*agent* 的可到达区域 *neighbors*0(满足$v(s')>v(s)$) 和*neighbors*1(满足$v(s')\leqslant v(s)$);

7) 当*neighbors*0不为空时, 按式 (4) 在*neighbors*0中选取下一状态, 否则在*neighbors*1中按式 (4) 选取下一状态, 将下一状态存入 *L*[*agent*] 尾部;

8) 在 *block* 中将选取的下一状态对应位置的元素置为 1;

9) *end for*;

10) 检测是否有障碍物位置发生变化, 若有则更新环境状态矩阵 *grid*, 重新执行算法 1 获取最新的状态-值$v(s)$;

11) *length*=*length*+1;

12) *end while*;

13) 输出各智能体的运动轨迹*L*[*agent*];

当获取执行算法 3 输出的各智能体的运动轨迹后,在仿真软件中只需将各智能体的运动轨迹映射到栅格环境中各状态区域的真实坐标并让智能体按此真实坐标进行运动即可实现群体运动目的.

3 实验分析

本节将对本文提出的改进的值迭代算法进行实验, 并将其与以欧氏距离作为启发式的值迭代算法和经典的最短路径算法 Dijkstra 算法的计算时间进行比较分析. 同时利用 maya 2009 软件在三维动画场景中对本文提出的运动轨迹生成算法进行群体动画仿真实验.

3.1 改进的值迭代算法实验及分析

我们用矩阵表示二维平面空间和三维立体空间分别对本文提出的改进的值迭代算法 (记作 IVI)、以欧氏距离作为启发式的值迭代算法 (记作 HVI) 和 Dijkstra 算法进行实验, 每组各取 5 个不同大小的矩阵, 在矩阵边界的中心处取一点作为目标点, 在矩阵的中间部分区域设置障碍物区域, 以使测试模型具有一定的代表性. 其中表示二维平面空间的矩阵分别为:

1) 大小: 10×10, 目标: (9, 5), 障碍物区域: (5, 3) 到 (5, 7) 所在直线区域;

2) 大小: 20×20, 目标: (19, 10), 障碍物区域: (10, 5) 到 (10, 14) 所在直线区域;

3) 大小: 30×30, 目标: (29, 15), 障碍物区域: (15, 10) 到 (15, 19) 所在直线区域;

4) 大小: 40×40, 目标: (39, 20), 障碍物区域: (20, 15) 到 (20, 24) 所在直线区域;

5) 大小: 50×50, 目标: (49, 25), 障碍物区域: (25, 20) 到 (25, 29) 所在直线区域.

表示三维立体空间的矩阵为:

1) 大小: 10×10×10, 目标: (9, 5, 5), 障碍物区域: (5, 3, 3) 到 (5, 7, 7) 所在平面区域;

2) 大小: 20×20×20, 目标: (19, 10, 10), 障碍物区域: (10, 5, 5) 到 (10, 14, 14) 所在平面区域;

3) 大小: 30×20×30, 目标: (29, 10, 15), 障碍物区域: (15, 5, 10) 到 (15, 14, 19) 所在平面区域;

4) 大小: 40×20×40, 目标: (39, 10, 20), 障碍物区域: (20, 5, 15) 到 (20, 14, 24) 所在平面区域;

5) 大小: 50×20×50, 目标: (49, 10, 25), 障碍物区域: (25, 5, 20) 到 (25, 14, 29) 所在平面区域.

实验环境为: Intel® Core™ i5-4590 @ 3.30GHz, win7 操作系统, 使用 Python 语言实现.

每次实验均运行 3 次, 取平均时间作为算法求解结束所需时间 (值迭代算法均取阈值$\theta = 1.0$, 在算法 2 具体实现时, 我们采取初始化除目标状态外的所有状态的状态-值为*Inf* 和每步移动的真实距离作为回报, 即回报取正值, 在最终返回全部状态的状态-值时统一取相反数供算法 1 进行迭代优化). 二维平面空间的实验结果见表 1, 三维立体空间的实验结果见表 2.

表 1 二维平面空间计算效率对比 (单位: 秒)

	IVI	HVI	Dijkstra
10×10	**0.0066**	0.0147	0.0072
20×20	0.0258	0.0937	**0.0247**
30×30	**0.0625**	0.2619	0.1171
40×40	**0.1069**	0.5946	0.3280
50×50	**0.1605**	1.0872	0.7309

表 2 三维立面空间计算效率对比 (单位: 秒)

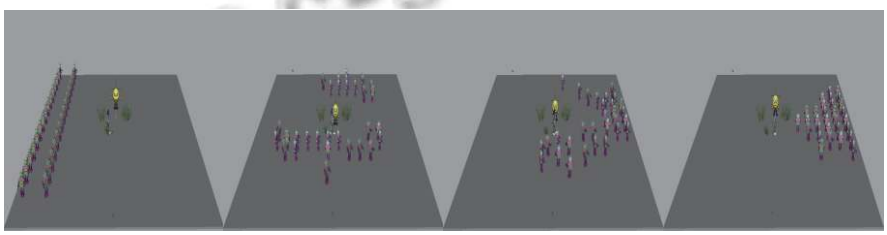
	IVI	HVI	Dijkstra
10×10×10	**0.1791**	0.3925	0.2024
20×20×20	**1.6817**	5.2878	7.7723
30×20×30	**3.9912**	14.4246	37.3508
40×20×40	**7.1431**	30.9954	114.8624
50×20×50	**10.9903**	54.5086	277.3717

通过表 1 和表 2 我们可以看出, 我们提出的改进的值迭代算法仅有一项较 Dijkstra 算法慢 1.1 毫秒, 而其他测试结果与以欧氏距离作为启发式的值迭代算法和经典的最短路径算法 Dijkstra 算法的计算效率相比均具有明显提升. 同时我们通过设置阈值$\theta = 1.0$, 保证了收敛后每次迭代所有状态的状态-值$v(s)$的改变量均在 1.0 以内, 从而保证了各种算法在对各智能体进行运动轨迹规划时具有相近的性能. 因而在满足群体运动规划目标的前提下, 我们设计的算法具有明显优势.

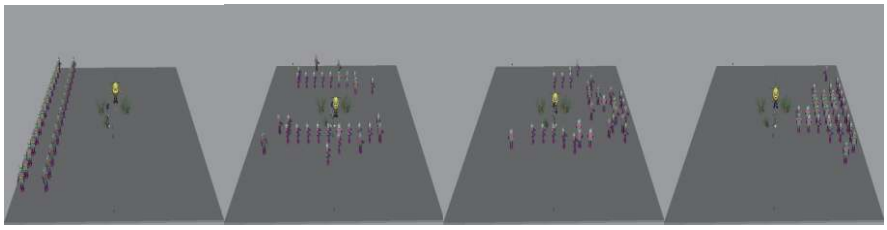
3.2 群体动画仿真实验及分析

我们用 C++语言结合 maya 2009 软件自带的 mel 语言和 api 接口对运动轨迹生成算法在三维动画场景中进行群体动画仿真实验, 将三维动画场景中的地面作为二维平面环境, 立体空间作为三维环境各进行 3 次实验, 二维平面环境中以地面上的机器人作为智能体, 共 30 个, 各智能体的对应的起始位置和目标地点均相同, 三维立体环境中以无人机作为智能体, 共 30 个, 各智能体对应的起始位置和目标点均相同 (机器人和无人机模型下载于 3D 溜溜网: https://www.3d66.com/). 实验效果见图 1 和图 2.

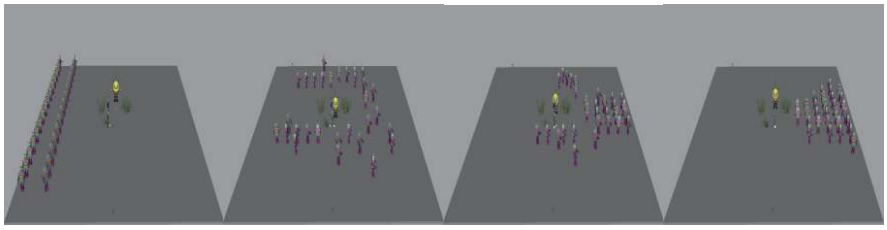
从图 1 和图 2 可以看出, 各智能体均能自动避开障碍物, 同时各智能体之间没有发生碰撞, 最终智能体均聚集在目标点附近. 实验效果体现了本算法实现群体动画的有效性, 不同时间运行的程序, 各智能体在相同时间帧时所处的位置略有不同, 体现了本算法可实现群体运动的多样性.



(a) 第1次第1帧 (b) 第1次第100帧 (c) 第1次第150帧 (d) 第1次第270帧



(e) 第2次第1帧 (f) 第2次第100帧 (g) 第2次第150帧 (h) 第2次第270帧



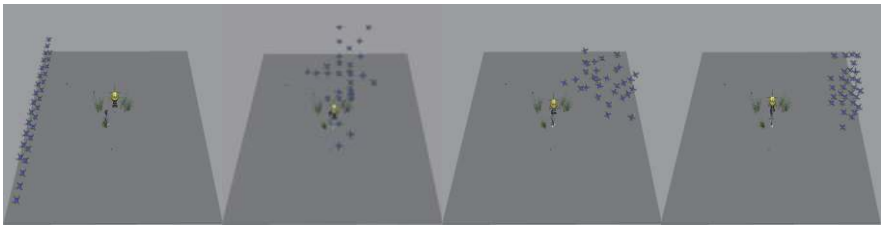
(i) 第3次第1帧 (j) 第3次第100帧 (k) 第3次第150帧 (l) 第3次第270帧

图 1 2D 群体动画仿真结果部分片段截图 (机器人模型下载网址 https://www.3d66.com/reshtml/4768/476880.html)

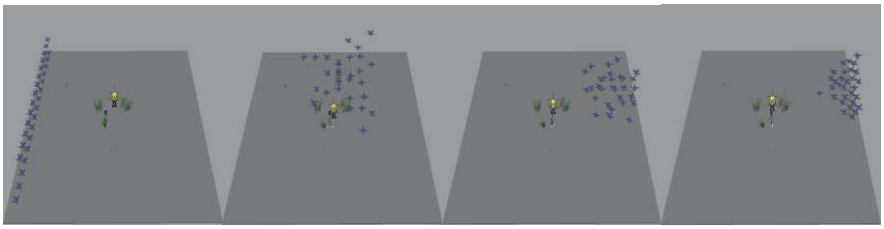
4 结论与展望

本文通过使用马尔可夫决策过程对群体动画进行建模, 同时设计了一种新的改进的值迭代算法和运动轨迹生成算法, 通过实验验证了改进的值迭代算法的计算效率明显高于用欧氏距离作为启发式信息的值迭代算法和经典的最短路径算法 Dijkstra 算法, 并通过三维群体动画仿真实验验证了本文设计的运动轨迹生成算法对于群体运动的有效性和多样性. 我们正在将本文所设计的算法在本实验室的手机短信 3D 动画自动生成系统中进行部署. 今后我们将尝试结合深度强化

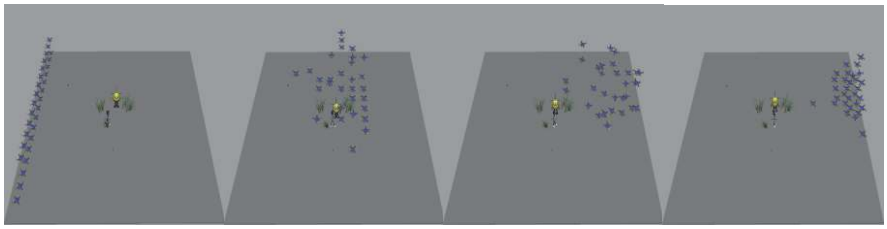
学习算法对智能体的运动进行细化以增强智能体运动轨迹的光滑性.



(a) 第1次第1帧 (b) 第1次第100帧 (c) 第1次第150帧 (d) 第1次第270帧



(e) 第2次第1帧 (f) 第2次第100帧 (g) 第2次第150帧 (h) 第2次第270帧



(i) 第3次第1帧 (j) 第3次第100帧 (k) 第3次第150帧 (l) 第3次第270帧

图 2 3D 群体动画仿真结果部分片段截图 (无人机模型下载网址 https://www.3d66.com/reshtml/5450/545028.html)

参考文献

1 黄东晋, 雷雪, 蒋晨凤, 等. 基于改进 JPS 算法的电影群体动画全局路径规划. 上海大学学报 (自然科学版), 2018, 24(5): 694–702.

2 Molnár P, Starke J. Control of distributed autonomous robotic systems using principles of pattern formation in nature and pedestrian behavior. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2001, 31(3): 433–435. [doi: 10.1109/3477.931538]

3 McPhail C, Powers WT, Tucker CW. Simulating individual and collective action in temporary gatherings. Social Science Computer Review, 1992, 10(1): 1–28. [doi: 10.1177/089443939201000101]

4 Sewall J, Wilkie D, Merrell P, *et al*. Continuum traffic simulation. Computer Graphics Forum, 2010, 29(2): 439–448. [doi: 10.1111/j.1467-8659.2009.01613.x]

5 Henry J, Shum HPH, Komura T. Interactive formation control in complex environments. IEEE Transactions on Visualization and Computer Graphics, 2014, 20(2): 211–222. [doi: 10.1109/TVCG.2013.116]

6 Loscos C, Marchal D, Meyer A. Intuitive crowd behavior in dense urban environments using local laws. Proceedings of 2003 Theory and practice of computer graphics. Birmingham, UK. 2003. 122–129.

7 Ondřej J, Pettré J, Olivier AH, *et al*. A synthetic-vision based steering approach for crowd simulation. ACM Transactions on Graphics, 2010, 29(4): 123.

8 Weiss T, Litteneker A, Jiang CFF, *et al*. Position-based multi-agent dynamics for real-time crowd simulation. Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation. Los Angeles, CA, USA. 2017. Article No. 27.

9 Dutra TB, Marques R, Cavalcante-Neto JB, *et al*. Gradient-based steering for vision-based crowd simulation algorithms. Computer Graphics Forum, 2017, 36(2): 337–348. [doi: 10.1111/cgf.2017.36.issue-2]

10 张超, 魏三强, 陈伟. 一种基于萤火虫算法的群体动画行为控制仿真设计. 重庆理工大学学报 (自然科学), 2017, 31(1): 100–106.

11 Lee J, Won J, Lee J. Crowd simulation by deep reinforcement learning. Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games. Limassol, Cyprus, 2018.
12 Ferguson D, Stentz A. Focussed processing of MDPs for path planning. Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence. Boca Raton, FL, USA, 2004: 310–317.
13 Pashenkova E, Rish I, Dechter R. Value iteration and policy iteration algorithms for Markov decision problem. Proceedings of 1996 Workshop on Structural Issues in Planning and Temporal Reasoning. 1996. 1–15.
14 杨兴, 张亚. 基于改进栅格模型的移动机器人路径规划研究. 农家科技, 2016, (3): 416.
15 王殿君. 基于改进 A*算法的室内移动机器人路径规划. 清华大学学报 (自然科学版), 2012, 52(8): 1085–1089.
16 Sutton RS, Barto AG. Reinforcement learning: An introduction. Cambridge, Massachusetts, London, England: MIT Press, 2018. 82–85.
17 Hansen EA, Zilberstein S. LAO: A heuristic search algorithm that finds solutions with loops. Artificial Intelligence, 2001, 129(1–2): 35–62.