计算机科学
COMPUTER SCIENCE

基于碰撞危急程度和深度强化学习的实时轨迹规划算法

徐林玲，周远，黄鸿云，刘杨

引用本文

徐林玲，周远，黄鸿云，刘杨.基于碰撞危急程度和深度强化学习的实时轨迹规划算法[J]．计算机科学，2023，50(3)：323-332.

XU Linling，ZHOU Yuan，HUANG Hongyun，LIU Yang. Real-time Trajectory Planning Algorithm Based on Collision Criticality and Deep Reinforcement Learning [J]. Computer Science，2023，50(3)：323-332.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220100007

基于碰撞危急程度和深度强化学习的实时轨迹规划算法

徐林玲[1]　周　远[2]　黄鸿云[3]　刘　杨[1,2]
1 浙江理工大学信息学院　杭州 310018
2 新加坡南洋理工大学计算机科学与工程学院　新加坡 639798
3 浙江理工大学图书馆大数据处理与分析中心　杭州 310018
(201930605055@mails.zstu.edu.cn)

摘　要　动态环境的实时碰撞规避是移动机器人轨迹规划中的一个巨大挑战。针对可变障碍物数量的环境,提出了基于LSTM(Long Short Term Memory)和DRL(Deep Reinforcement Learning)的实时轨迹规划算法Crit-LSTM-DRL。首先,根据机器人和障碍物的状态,预测碰撞可能发生的时间,计算各个障碍物相对于机器人的碰撞危急程度(Collision Criticality);其次,将障碍物根据碰撞危急程度由低到高排序,然后由LSTM模型提取固定维度的环境表征向量;最后,将机器人状态和该环境表征向量作为DRL的输入,计算对应状态的价值。在任何一个时刻,针对每一个动作,通过LSTM和DRL计算下一时刻对应的状态的价值,从而计算当前状态的最大价值以及对应的动作。针对不同环境,训练获得3个模型,即在5个障碍物的环境里训练的模型、在10个障碍物的环境里训练的模型和在可变障碍物数量(1～10)的环境里训练的模型,分析了它们在不同测试环境中的性能。为进一步分析单个障碍物和机器人之间的交互影响,将障碍物表示为障碍物和机器人的联合状态(Joint State),分析了在上述3个训练环境下获得的模型的性能。实验结果验证了Crit-LSTM-DRL的有效性。

关键词:轨迹规划;碰撞规避;障碍物危急度;深度强化学习
中图法分类号　TP242

Real-time Trajectory Planning Algorithm Based on Collision Criticality and Deep Reinforcement Learning

XU Linling[1],ZHOU Yuan[2],HUANG Hongyun[3] and LIU Yang[1,2]
1 School of Information Science and Technology,Zhejiang Sci-Tech University, Hangzhou 310018, China
2 School of Computer Science and Engineering,Nanyang Technological University,Singapore 639798,Singapore
3 Center of Library Big Data Processing and Analysis,Zhejiang Sci-Tech University,Hangzhou 310018,China

Abstract　Real-time collision avoidance in dynamic environments is a challenge in trajectory planning of mobile robots. Focusing on environments with variable number of obstacles,this paper proposes a real-time trajectory planning algorithm,Crit-LSTM-DRL,based on long short-term memory(LSTM) and deep reinforcement learning(DRL). First,it predicts the time to the occurrence of a collision between an obstacle and the robot based on their states,and then computes the collision criticality of each obstacle with respect to the robot. Second,it generates the obstacle sequence based on the collision criticality and abstracts a fixed-dimension vector by LSTM to represent the environment. Finally,the robot state and the extracted vector are concatenated as the input of the DRL's value network to compute the value with respect to the system state. At any instant,for each action,it predicts the value of the next state based on the LSTM and DRL models and then the value of the current state; hence,the action generating the maximal value of the current state is selected to control the robot. To evaluate the performance of Crit-LSTM-DRL,it is first trained in three different environments and obtain three models: the model trained in the environment with 5 obstacles,the model trained in the environment with 10 obstacles,and the model trained in the environment with variable number of obstacles(1～10). The models then are tested in various environments containing different number of obstacles. To further investigate the effects of the interaction between an obstacle and the robot,this paper also takes the joint state of an obstacle and

到稿日期:2022-01-04　返修日期:2022-08-14
基金项目:国家自然科学基金(62132014);浙江省科技计划项目(2022C01045);上海工业控制系统安全创新功能型平台开放课题(21170022-N);上海工业控制安全创新科技有限公司资助课题(21170424-J)
This work was supported by the National Natural Science Foundation of China(62132014),Science and Technology Plan Project of Zhejiang Province,China(2022C01045),Opening Project of Shanghai Trusted Industrial Control Platform(21170022-N) and Project Funded by Shanghai Industrial Control Security Innovation Technology Co.,Ltd.(21170424-J).
通信作者:黄鸿云(huanghongyun07@hotmail.com)

the robot as the state of the obstacle and trains another three models in the above training environments. Experimental results show the effectiveness and efficiency of Crit-LSTM-DRL.

Keywords Trajectory planning, Collision avoidance, Obstacle criticality, Deep reinforcement learning

1 引言

移动机器人可以帮助人们完成很多需耗费大量人力以及危险的任务，如环境监测、灾难救援和雷区测绘工作[1]等。为完成这些任务，首要条件是移动机器人具有在复杂环境中自主规划无碰撞轨迹的功能。如何生成从初始位置到目的地的无碰撞轨迹是机器人运动规划的重点和难点，也是限制移动机器人广泛应用的瓶颈。在过去几十年里，研究人员提出了许多碰撞规避的方法，如离散事件系统[2-4]、基于路线图的方法[5-6]、单元分解方法[7]、状态格法[8]、基于采样的方法[9]、人工势场法[10]、交互避免碰撞[11-12]、数学规划的方法[13-14]、优化运动方程方法[15]、图搜索算法 A* 算法[16-17]、遗传算法[18]、蚁群算法[19]。这些算法能够很好地规避碰撞，然而，在复杂拥挤的环境中，它们需要更高的计算成本，这就导致了实时效率的降低，从而影响了运动性能。为了提高计算效率，研究人员将深度强化学习(DRL)应用到机器人运动规划中[20-22]。然而这些方法通常仅考虑具有固定障碍物数量的环境，而无法适用于可变障碍物数量的环境。针对可变障碍物数量的环境，目前的主要方法是利用 LSTM 对障碍物进行编码，利用 LSTM 的隐藏层来表征环境中障碍物的特性[23]：首先根据障碍物到机器人的距离，将障碍物从远到近进行排序，获得障碍物状态序列，将该序列作为 LSTM 的输入，从而计算出 LTSM 的隐藏状态；该隐藏状态将和机器人状态一起作为 DRL 的输入，从而计算机器人的运动行为。因此，距离机器人最近的障碍物对机器人的动作选择的影响最大。然而，虽然某些障碍物与机器人的距离较远，但因其相对速度较大，它与机器人会更早地发生碰撞。因此，在碰撞规避中，它更应该被机器人重视并且记住。

本文同时从时间和空间两个维度定义障碍物的危急程度。首先根据机器人和障碍物的相对速度，预测它们未来是否会发生碰撞，如果会，则以碰撞时间衡量障碍物的危急程度，并按照碰撞时间递减获得可能会碰撞的障碍物危急序列；否则，以相对距离衡量障碍物的危急程度，并按照相对距离的递减加入不会碰撞的障碍物序列。然后合并两个障碍物序列，不会碰撞的障碍物序列在前。此外，本文方法也考虑了动态变化的环境对训练的影响，采用了不定障碍物个数的训练方式。区别于文献[23]通过距离对障碍物进行排序，本文主要的创新点在于：1)根据障碍物和机器人的可能碰撞时间，定义了障碍物相对于机器人运动的危急程度；2)提出了基于危急程度的 DRL 运动规划算法；3)通过大量实验验证了所提方法的有效性。

2 相关工作及预备知识

2.1 轨迹规划的相关工作

传统的轨迹规划算法可以分为离散方法和连续方法。离散方法通过将环境或者机器人运动离散化为一些离散状态，并通过图搜索算法获得机器人运动的状态序列；连续方法通常根据机器人的运动学方程，通过优化算法来产生机器人的运动轨迹[15]。Zhang 等通过加入位置转移传递参数，将 A* 算法和动态规划结合起来，从而解决了三维空间中的路径规划问题[17]；Lin 等利用遗传算法，将 AGV 任务分配和协同路径规划作为一个整体，设计适应性函数来计算 AGV 的货位路径[18]。然而，传统的路径规划算法往往存在计算复杂度高、无法应对复杂多变的环境等问题。因此研究人员尝试将机器学习应用于机器人的路径规划。Bency 等[24]使用循环神经网络以迭代的方式确定端到端轨迹，紧凑且隐式地生成具有最小性能损失的最佳运动计划，但这种方式需要根据环境的改变重新训练，并且暂时只能应用于静态环境中，针对动态环境，它还是无法高效地进行轨迹规划。

随着深度学习软硬件技术的发展，研究人员开始将 DRL 应用到机器人的运动规划中，通过离线训练来降低实时计算复杂度。自从 Google Deep Mind 提出深度 Q 网络(Deep Q-Network, DQN)[25]，深度强化学习开始进入人们的视野。DQN 改进了传统 RL 方法中的价值函数逼近和策略搜索的学习机制[26]，将强化学习的决策能力与深度学习的感知能力相结合，可以根据输入的信息直接进行控制，在实现原理上，更加贴近人类思维。因此在路径规划中，得到了十分广泛的应用。Tai 等[27]提出了一个基于学习的无地图的运动规划器，通过边训练边探索，该规划器在极端复杂的环境中也具有很高的稳定性。Feng 等[28]提出了一种改进的深度强化学习算法(NDQN)，该算法利用差值增长的改正函数对 DQN 算法进行改进，并将其运用到机器人在三维环境中的路径规划中。Wang 等[29]基于深度强化学习，使用运动学方程约束来优化状态空间的搜索与采集，从而提高了模型的训练速率。Li 等[30]将强化学习和深度卷积神经网络相结合，用值函数近似法代替 Q-Learning 中的动作值函数，从而缓解了“维数灾难”问题。上述方法通常需要将状态空间离散化。针对连续状态空间和动作空间，Chen 等[22]针对两个机器人之间无通信的场景，提出通过学习一个隐式编码智能体之间的协同行为及预估到达目标时间的值函数，通过强化学习进行离线学习，最终得到了一个计算高效的两个机器人协同控制策略。随后，Everett 等[23]将该方法扩展到多机器人系统，使用 LSTM 模型对观测到的障碍物进行特征编码，获得固定维度的隐藏状态。但该方法只考虑障碍物和机器人之间的空间关系，即根据障碍物到机器人的距离来构建 LSTM 的输入状态序列。Sawada 等[31]扩展了目标障碍物区(Obstacle Zone by Target, OZT)，让 LSTM 在连续动作空间中进行训练，得到了具有更长安全距离的模型，但该方法只针对船只应用场景。本文同时从时间和空间两个维度对障碍物进行排序，它更能反应障碍物在机器人碰撞规避轨迹决策中的影响。

2.2 深度强化学习(DRL)

本文提出了一种基于深度强化学习的轨迹规划算法。

深度强化学习是将强化学习与深度学习结合起来的新型学习框架。强化学习(Reinforcement Learning，RL)[32]是通过不断试错来和环境进行交互，使用“奖励”来对动作的价值进行量化表示，通过最大化累积奖励(Reward)的方式来学习得到最优策略[33]。强化学习本质上是求解马尔可夫决策过程(Markov Decision Process，MDP)的最优策略，通过参数化的函数来逼近“状态-动作”的映射关系。深度学习(Deep Learning，DL)通过神经网络，模仿人脑的神经元的互连互通机制处理数据，因此拥有强大的感知能力，某些应用场景下，它的感知能力甚至可以赶超人类[34]。目前深度学习被广泛应用在图像特征提取、图像分类以及一些自然语言处理领域。

由于自身结构与学习能力的约束，强化学习只能被用来解决低维问题。为了处理高维的复杂问题，人们尝试加入神经网络进行训练[35]，深度强化学习(DRL)初具模型。经过一段时间的发展，DRL 趋于成熟。近年来，不少研究人员利用 DRL 中深度学习的感知能力来预处理复杂、高维的环境信息，再结合其强化学习的决策能力与环境信息进行交互，最终完成决策过程[36-37]。DRL 算法主要分为两类：值函数算法和策略梯度算法[38-39]。值函数算法一般是指通过迭代更新值函数来得到智能体的最优策略，它一般应用于离散的问题中。策略梯度算法则是直接用近似函数来建立网络，通过网络进行预测，选取动作后得到奖励值，然后沿梯度方向对网络的参数进行优化，从而使奖励值最大化[40]。

3 问题描述

本文关注具有完整约束的机器人在具有可变数量障碍物的未知环境中的运动规划问题。给定目标速率 $v_f>0$，机器人需从初始位置 $\boldsymbol{p}_0=(x_0,y_0)$无碰撞地移动到目标位置 $\boldsymbol{p}_g=(x_g,y_g)$，其中每个机器人表示为一个圆形，其安全半径为 r。将时间以等步长 Δt 离散为 $0=t_0,t_1,t_2,\cdots$；第 $k(t_k=k\Delta t,k=0,1,2,\cdots)$时刻机器人的状态表示为 $\boldsymbol{s}_k=(\boldsymbol{p}_k,v_{k-1},\boldsymbol{p}_g,v_f,r)$，其中 $\boldsymbol{p}_k=(x_k,y_k)$表示 k 时刻机器人的位置，$\boldsymbol{v}_{k-1}=(vx_{k-1},vy_{k-1})$表示机器人在$[(k-1)\Delta t,k\Delta t)$时间内的速度。这里第 k 时刻的速度 $\boldsymbol{v}_k$是控制命令，它需要根据机器人的当前速度(即 $\boldsymbol{v}_{k-1}$)和环境信息计算获得。第 k 时刻的障碍物集合为$\boldsymbol{O}_k=\{o_1,o_2,\cdots,o_{n_k}\}$，其中障碍物 o_i的状态表示为 $\boldsymbol{s}_k^i=(\boldsymbol{p}_k^i,\boldsymbol{v}_k^i,r_i)$，$\boldsymbol{p}_k^i=(x_k^i,y_k^i)$表示障碍物的位置，$\boldsymbol{v}_k^i=(vx_k^i,vy_k^i)$表示障碍物的速度，$r_i$表示障碍物的半径。令 $\boldsymbol{s}(\boldsymbol{O}_k)=(\boldsymbol{s}_k^1,\boldsymbol{s}_k^2,\cdots,\boldsymbol{s}_k^n)$，其维度随着障碍物数量的变化而变化。当 k 表示未来时刻时，$\boldsymbol{v}_i^k$ 既可以是障碍物o_i的当前速度，也可以由 ORCA[11]预测得到。本文中的动态障碍物主要考虑行人和其他机器人，同时假设每个动态障碍物有独立的碰撞规避算法(如 ORCA)，以避免和其他障碍物发生碰撞。

上述机器人的轨迹规划问题可以描述为：

$$\arg\min_{v_0,v_1,\cdots,v_{T-1}} T \tag{1}$$

$$\text{s.t. } \boldsymbol{p}_{k+1}=\boldsymbol{p}_k+\boldsymbol{v}_k\Delta t,\ \forall k\in\{0,1,\cdots,T-1\} \tag{2}$$

$$\|\boldsymbol{p}_k-\boldsymbol{p}_k^i\|\geqslant r+r_i,\ \forall o_i\in\boldsymbol{O}_k,k\in\{1,2,\cdots,T\} \tag{3}$$

$$\boldsymbol{p}_T=\boldsymbol{p}_g \tag{4}$$

根据文献[22]和文献[23]，此轨迹规划问题可以通过 DRL 最大化如下的价值函数来求解：

$$\boldsymbol{v}_k^*=\arg\max_{v\in A}R(\boldsymbol{s}_k,\boldsymbol{s}(\boldsymbol{O}_k),\boldsymbol{v})+\gamma^{v_f\Delta t}V^*(\boldsymbol{s}_{k+1,v},\boldsymbol{s}(\boldsymbol{O}_{k+1})) \tag{5}$$

$$V^*(\boldsymbol{s}_{k+1,v},\boldsymbol{s}(\boldsymbol{O}_{k+1}))=\sum_{k'=k+1}^{T-1}\gamma^{(k'-k-1)v_f\Delta t}R(\boldsymbol{s}_{k'},\boldsymbol{s}(\boldsymbol{O}_{k'}),\boldsymbol{v}_{k'}^*) \tag{6}$$

其中，A 是一组预定义的命令集合，5.1 节中详细介绍了动作空间 A 的组成；$\gamma\in[0,1]$是衰减系数；$R(\boldsymbol{s}_k,\boldsymbol{s}(\boldsymbol{O}_k),\boldsymbol{v})$表示机器人在 k 时刻采取速度 $\boldsymbol{v}$ 的一步激励值，它依赖于 k 时刻机器人和障碍物的状态；$\boldsymbol{s}_{k+1,v}$表示机器人在 $\boldsymbol{s}_k$状态下以速度 $\boldsymbol{v}$ 运动后的下一时刻状态。根据文献[23]，激励函数设为：

$$R(\boldsymbol{s}_k,\boldsymbol{s}(\boldsymbol{O}_k),\boldsymbol{v})=\begin{cases}1, & \boldsymbol{p}_{k+1}=\boldsymbol{p}_g\\ -0.25, & d_{\min}\leqslant 0\\ -0.1+0.5d_{\min}, & 0<d_{\min}\leqslant 0.2\\ 0, & \text{otherwise}\end{cases} \tag{7}$$

其中，$d_{\min}$表示在时间区间$[k\Delta t,(k+1)\Delta t]$内，机器人距离所有障碍物的最小距离。

因此，本文所要解决的问题可以表述为：在未知、障碍物数量可变的环境中，给定机器人的初始位置 $\boldsymbol{p}_0$、目标位置 $\boldsymbol{p}_g$和参照速率 v_f，设计一个基于 DRL 的控制器，使得机器人在尽可能短的时间内从 $\boldsymbol{p}_0$无碰撞地运动到 $\boldsymbol{p}_g$。

4 以障碍物危急程度为导向的深度强化学习框架

针对上述问题，本节首先从时间和空间两方面来定义障碍物的危急程度，然后阐述以障碍物危急程度为导向的基于 DRL 的轨迹规划方法 Crit-LSTM-DRL。其主要思想为：首先根据障碍物可能发生碰撞的时间以及距机器人的距离，定义障碍物的危急程度并由小到大排序，将排序后的障碍物作为 LSTM 的输入，以 LSTM 的固定维数的隐藏状态作为环境的表征向量，最后将该向量和机器人结合作为价值网络的输入，进而计算当前状态的最大价值以及对应的速度命令。

4.1 障碍物危急程度的确定

首先根据文献[21-23]，我们通过坐标变换来减少冗余变量。具体地，通过坐标平移和旋转，将全局坐标系转换为以机器人为中心、以指向目标位置的方向为 x 轴的局部坐标系 $\bar{\boldsymbol{x}}\boldsymbol{p}_k\bar{\boldsymbol{y}}$，如图 1 所示，其中 $\boldsymbol{p}_k$和 $\boldsymbol{p}_g$为机器人的当前位置和目标位置，φ 为向量 $\boldsymbol{p}_k\boldsymbol{p}_g$ 的方向角。因此，在局部坐标系内，机器人的速度可表示为：

$$\bar{v}_{k-1}=(\overline{vx}_{k-1},\overline{vy}_{k-1})^{\mathrm{T}}=\boldsymbol{M}v_{k-1} \tag{8}$$

障碍物 o_i的位置和速度可表示为：

$$\bar{\boldsymbol{p}}_k^i=(\bar{x}_k^i,\bar{y}_k^i)^{\mathrm{T}}=\boldsymbol{M}(\boldsymbol{p}_k^i-\boldsymbol{p}_k) \tag{9}$$

$$\bar{\boldsymbol{v}}_k^i=(\overline{vx}_k^i,\overline{vy}_k^i)^{\mathrm{T}}=\boldsymbol{M}v_k^i \tag{10}$$

其中，$\boldsymbol{M}=\begin{bmatrix}\cos\varphi & \sin\varphi\\ -\sin\varphi & \cos\varphi\end{bmatrix}$，$\varphi=\arctan\dfrac{y_g-y_k}{x_g-x_k}$。因此，机器人的状态可表示为$\bar{\boldsymbol{s}}_k=(d_k^g,v_f,\overline{vx}_{k-1},\overline{vy}_{k-1},r)$；障碍物 o_i的状态可表示为$\bar{\boldsymbol{s}}_k^i=(\bar{x}_k^i,\bar{y}_k^i,\overline{vx}_k^i,\overline{vy}_k^i,r_i,d_k^i,r+r_i)$，所有障碍物的局部状态序列为 $\bar{\boldsymbol{s}}(\boldsymbol{O}_k)=(\bar{\boldsymbol{s}}_k^1,\cdots,\bar{\boldsymbol{s}}_k^{n_k})$，其中，

$$d_k^g=\|\boldsymbol{p}_k-\boldsymbol{p}_g\|_2 \tag{11}$$

$$d_k^i=\|\boldsymbol{p}_k-\boldsymbol{p}_k^i\|_2 \tag{12}$$

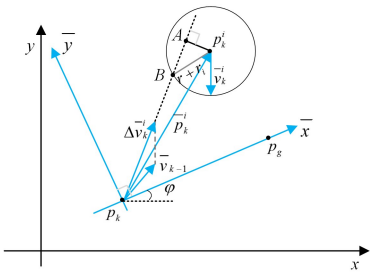


图 1 坐标系变换

Fig. 1 Coordinate system transformation

根据局部坐标系内机器人和障碍物的状态，我们从时间和空间两个维度定义障碍物的危急程度。首先，对任意障碍物 o_i，它相对于机器人的位置为 $\bar{p}_k^i$，它们的相对速度为：

$$\Delta\bar{\boldsymbol{v}}_k^i=\bar{\boldsymbol{v}}_{k-1}-\bar{\boldsymbol{v}}_k^i \tag{13}$$

在运动过程中，机器人和障碍物 o_i 的最短距离满足：

$$d_{\min}^{i\,2}=\|\bar{\boldsymbol{p}}_k^i\|_2^2-d^2(\boldsymbol{p}_k,A) \tag{14}$$

$$d(\boldsymbol{p}_k,A)=\bar{\boldsymbol{p}}_k^i\cdot\Delta\bar{\boldsymbol{v}}_k^i/\|\Delta\bar{\boldsymbol{v}}_k^i\|_2 \tag{15}$$

根据最短距离，我们可以将障碍物分为两类

(1) $\boldsymbol{O}_k^\alpha=\{o_i:d_{\min}^i<r+r_i\}$：对任意属于 $\boldsymbol{O}_k^\alpha$ 的障碍物，如果机器人和障碍物以当前速度继续运行，则它们将在 t_k^i 时刻后发生碰撞，其中 t_k^i 表示为：

$$t_k^i=(d(\boldsymbol{p}_k,A)-d(A,B))/\|\Delta\bar{\boldsymbol{v}}_k^i\| \tag{16}$$

$$d(A,B)=\sqrt{(r+r_i)^2-d_{\min}^{i\,2}} \tag{17}$$

t_k^i 越小，障碍物 o_i 就越危险，机器人碰撞规避时更需保留该障碍物信息。

(2) $\boldsymbol{O}_k^\beta=\{o_i:d_{\min}^i<r+r_i\}$：如果机器人和障碍物以当前速度继续运行，它们将不会发生碰撞，此时，距离机器人更近的障碍物在机器人的碰撞规避决策中起着更重要的作用，但是它们没有第一类障碍物重要。因此，我们有如下定义：

定义 1(碰撞危急程度) 给定任意两个障碍物 o_i 和 o_j，o_i 具有更高的碰撞危急程度，记作 $c(o_i)\geqslant c(o_j)$，如果 o_i 和 o_j 满足以下条件之一：

1) $o_i\in\boldsymbol{O}_k^\alpha$，$o_j\in\boldsymbol{O}_k^\beta$；

2) $o_i,o_j\in\boldsymbol{O}_k^\alpha$，且 $t_i\leqslant t_j$；

3) $o_i,o_j\in\boldsymbol{O}_k^\beta$，且 $d_i\leqslant d_j$，其中 d_i 为障碍物 o_i 到机器人的距离。

定义 2(有序危急障碍物序列) 给定障碍物集合 $\boldsymbol{O}_k=\{o_1,o_2,\cdots,o_{n_k}\}$，序列 $\langle o_1,o_2,\cdots,o_{n_k}\rangle$ 被称为有序危急障碍物序列，记作 $\boldsymbol{O}_k^c$，如果该序列满足 $\forall i<j$，则 $c(o_i)\leqslant c(o_j)$。

根据上述定义，在任意时刻，机器人都可以对感知到的障碍物进行排序，从而得到有序危急障碍物序列。首先，将所有探测到的障碍物分为 $\boldsymbol{O}_k^\alpha$ 和 $\boldsymbol{O}_k^\beta$；其次，将 $\boldsymbol{O}_k^\beta$ 中的障碍物根据它们到机器人的距离进行降序排列；第三，将 $\boldsymbol{O}_k^\alpha$ 中的障碍物根据预测的碰撞时间进行降序排列；最后，将两个子序列依次连接起来，得到有序危急障碍物序列，对应的状态序列记为 $\bar{\boldsymbol{s}}(O_k^c)=(\bar{\boldsymbol{s}}_k^1,\cdots,\bar{\boldsymbol{s}}_k^{n_k})$，其中 $\boldsymbol{O}_k^c=\langle o_1,o_2,\cdots,o_{n_k}\rangle$。

以图 2 所示状态为例，对算法的每步决策进行具体介绍。如图 2 所示，机器人在 $k=19(t=4.75\text{ s})$ 检测到 7 个障碍物 $\{o_1,o_2,\cdots,o_7\}$。此时，机器人的状态为 $\boldsymbol{s}=(-1.58,-0.71,0.48,0,0,4,1,0.3)$，$o_1$ 的状态为 $\boldsymbol{s}(o_1)=(-0.32,-0.99,-0.94,-0.35,0.3)$(从左到右依次表示 x 坐标、y 坐标、x 轴方向上的速度、y 轴方向上的速度、半径)。因此，在机器人的局部坐标系内，机器人的状态为(4.97,1,0.15,−0.45,0.3)，障碍物 o_1 的状态为(0.14,−1.28,−0.63,0.78,0.3,1.29,0.6)。若以当前速度运动，o_1 将在 0.68 s 后和机器人发生碰撞。因此，$o_1\in\boldsymbol{O}_k^\alpha$。同理，我们可以分别对其他障碍物进行相同的分析。最后，我们得到 $\boldsymbol{O}_k^\alpha=\{o_1,o_3,o_5\}$，对应的碰撞时间分别为 0.68,2.19,0.71；$\boldsymbol{O}_k^\beta=\{o_2,o_4,o_6,o_7\}$，它们到机器人的距离分别为 1.54,3.01,2.38,2.09。因此，该时刻的有序危机障碍物序列为 $\boldsymbol{O}_k^c=\{o_4,o_6,o_7,o_2,o_3,o_5,o_1\}$。

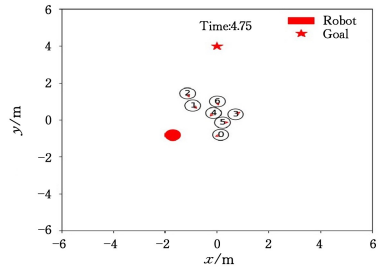


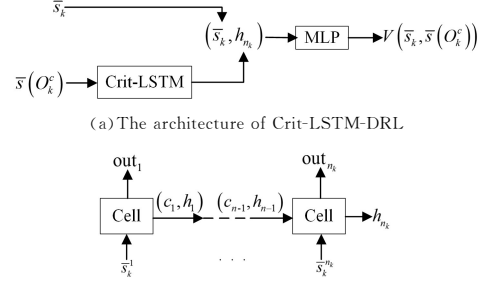
图 2 危急障碍物序列的计算

Fig. 2 Calculation of critical obstacle sequence

本文提出的方法针对每一时刻探测到的障碍物信息进行实时处理并且得出最优动作。每次决策采用的数据都是排序过的障碍物的状态信息，每步决策都是相对独立的，仅与当前时刻探测范围内的障碍物有关。执行决策后，障碍物碰撞的危急程度可能发生改变，因此每一步轨迹规划都需要重新计算碰撞危急程度。因此，如果机器人本身的计算能力不足，可以通过分布式计算解决，即由其他机器人计算各自的危急程度，然后转发。

4.2 Crit-LSTM-DRL 的结构和训练

如图 3(a)所示，Crit-LSTM-DRL 主要包含两个网络结构：LSTM 和多层感知机(Multilayer Perception，MLP)。其中，LSTM 用来处理可变数量障碍物，MLP 用于计算该状态的价值。首先，以有序危急障碍物序列的状态序列作为输入，利用 LSTM 计算其隐藏状态 h_{n_k}：如图 3(b)所示，根据有序危急障碍物序列 $\langle o_1,o_2,\cdots,o_{n_k}\rangle$，LSTM 首先计算 o_1 的隐藏状态 h_1，然后计算 o_2 的隐藏状态 h_2，依次进行，最后得到隐藏状态 h_{n_k}。其次，MLP 以 $\boldsymbol{h}_{n_k}$ 和机器人的状态作为输入向量，计算该状态对应的价值。



(a) The architecture of Crit-LSTM-DRL

(b) The unrolled Crit-LSTM

注：O_k^c 是危急障碍物序列

图 3 Crit-LSTM-DRL 的结构

Fig. 3 Structure of Crit-LSTM-DRL

根据上述网络结构，算法 1 给出了 Crit-LSTM-DRL 的训练过程，其中 $\boldsymbol{s}$ 和 $\bar{\boldsymbol{s}}$ 分别表示在全局坐标系和局部坐标系下的状态。该训练过程主要包括网络初始化（Lines 1－3）和网络训练（Lines 5－26）。在网络初始化阶段，我们首先随机产生一组运动任务，包括机器人的起点位置和目标位置，以及随机障碍物个数和运动任务，然后用传统的轨迹规划算法 ORCA 产生一组可行轨迹（Line 1）；针对每条轨迹的每个状态，计算其价值，从而得到一组（状态，价值）的数据集 E（Line 2）；最后，根据监督学习，训练并初始化网络模型（Line 3）。训练阶段主要包括 4 个步骤：1）根据当前网络 V，产生一组训练轨迹（Line 7－19）；2）根据目标网络，计算这组轨迹中每个状态的价值并更新数据集 E（Lines 20－23）；3）根据更新后的数据集，采用监督学习更新网络（Line 24）；4）以固定间隔更新目标网络（Lines 25－26）。具体地，在根据当前网络 V 产生一个训练轨迹时，我们首先产生一个初始任务 $(\boldsymbol{s}_0, \boldsymbol{s}(\boldsymbol{O}_0))$。其次，在任意时刻 k，机器人根据 $\boldsymbol{v}_k = \arg\max_{\boldsymbol{v}\in A} R(\boldsymbol{s}_k, \boldsymbol{s}(\boldsymbol{O}_k), \boldsymbol{v}) + \gamma^{v_j \Delta t} V(\bar{\boldsymbol{s}}_{k+1,v}, \bar{\boldsymbol{s}}(O_{k+1}))$ 确定当前速度 $\boldsymbol{s}_k$，其中 $\boldsymbol{s}(O_{k+1})$ 通过假设每个障碍物在 $[k\Delta t, (k+1)\Delta t)$ 时间间隔内以恒定的速度 v_i^k 移动进行预测获得。为了提高探测效率，本文采用 ε 贪心策略确定每一时刻的速度。如果机器人到达目的地，或检测到碰撞，或已经达到最大运动时间，则结束此次任务。当机器人以到达目的地或者发生碰撞来结束运动，则生成的轨迹将用于更新训练数据集 E。为了增加模型的稳定性[40]，我们用一个单独的目标网络来计算这些轨迹中各个状态的价值，该目标网络将以频率 C 更新。

算法 1 Crit-LSTM-DRL 的训练算法

输入：最大迭代次数 N_{max}，重放次数 N_{replay}，动作空间 A，机器人运动的最大时间步长 T_{max}，贪心选择概率 ε 和目标价值网络的更新频率 C

输出：价值网络 V

1. 通过 ORCA 算法生成一组轨迹；
2. 根据轨迹初始化重放记忆 E={(state,value)}；
3. 用基于记忆 E 的监督学习初始化一个价值网络 V；
4. 初始化目标值网络 $\widetilde{V}$=V，episode=0；
5. for episode=1：N_{max} do
6. for ite=1：N_{replay} do
7. Sample $\mathbf{s}_0$ and $\mathbf{s}(\mathbf{O}_0)$；
8. traj={($\mathbf{s}_0$,$\mathbf{s}(\mathbf{O}_0)$)}，done=False，k=0；
9. while not done do
10. 在[0,1]之间随机生成一个值 p；
11. if p <ε then
12. 从动作空间 A 中随机选择一个 v_k；
13. else
14. 通过 $\mathbf{v}_k^* = \arg\max_{v\in A} R(\mathbf{s}_k, \mathbf{s}(\mathbf{O}_k), \mathbf{v}) + \gamma^{v_i \Delta_t} V^*(\mathbf{s}_{k+1}, \mathbf{v}, \mathbf{s}(O_{k+1}))$，计算 $\mathbf{v}_k$
15. 移动到下一个状态 $\mathbf{s}_{k+1}$；
16. 检测并更新障碍物状态 $\mathbf{s}(\mathbf{O}_{k+1})$；
17. traj=traj∪{($\mathbf{s}_{k+1}$,$\mathbf{s}(\mathbf{O}_{k+1})$)}，k=k+1；
18. if k ≥T_{max} or collided or reached then
19. done=True；
20. if collided or reached then
21. for ∀($\mathbf{s}_k$,s($\mathbf{O}_k$)∈ traj do
22. $\mathbf{v}_k = \widetilde{V}(\bar{s}_k, \bar{s}(\mathbf{O}_k^c))$
23. 用($(\bar{s}_k, \bar{s}(\mathbf{O}_k^c))$,$\mathbf{v}_k$)更新轨迹记忆 E
24. 使用 E 通过梯度下降更新网络 V；
25. if(episode mod C) ==0 then
26. $\widetilde{V}$=V
27. Return **V**

5 实验分析

5.1 实验设置

本节通过仿真实验评估 Crit-LTSM-DRL 的性能。实验设置如下。

（1）模型参数设置

LSTM 隐藏状态的维度为 50；MLP 的层数为 4，各层的神经元数量分别为 150，100，100 和 1，激活函数采用 ReLU 函数。

（2）机器人运动设置（动作空间）

机器人的参考速率为 1，时间步长为 0.25，共 81 个可选离散速度，即：

$$A=\left\{(v_i\cos\theta_j, v_i\sin\theta_j): v_i=\frac{e^{\frac{i}{5}}-1}{e-1}, \theta_j=\frac{j}{8}\pi, i=0,1,\cdots,5, j=0,1,\cdots,15\right\}$$

（3）训练设置

Crit-LTSM-DRL 在 3 类环境中进行了训练：具有 5 个障碍物的环境、具有 10 个障碍物的环境和具有可变障碍物数量（从 1 到 10 不等）的环境。在每个训练环境中，模型由 3 000 条轨迹进行初始化，DRL 共进行了 10 000 次迭代；在每次迭代过程中，随机产生一条轨迹更新训练数据集，然后训练 100 次。针对 3 类环境，我们训练得到 3 个模型：Crit-LSTM-DRL-5，Crit-LSTM-DRL-10 和 Crit-LSTM-DRL-D。为进一步研究单个障碍物和机器人的交互影响，每个障碍物用自身状态和机器人状态的联合状态 $(\boldsymbol{s}_k, \boldsymbol{s}_i^k)$ 表示其状态并输入给 LSTM。因此，我们另外得到 3 个模型：Crit-LSTM-DRL-5-J，Crit-LSTM-DRL-10-J 和 Crit-LSTM-DRL-D-J。

（4）测试设置

为了验证模型的性能，我们生成了 5 类测试环境：分布于 1～4 的随机障碍物个数的环境（set1）、5 个障碍物的环境（set2）、分布于 6～9 的随机障碍物个数的环境（set3）、10 个障碍物的环境（set4），以及分布于 11～14 的随机障碍物个数的环境（set5）。在一组测试中，每类环境随机产生 300 个测试用例，共 1500 个测试用例，共进行 10 组测试。

（5）比较基准

本文将所提方法与文献[23]的方法（LSTM-DRL）进行了比较。LSTM-DRL 仅从空间维度对障碍物进行分类，将障碍物根据其到机器人的距离进行排序，然后由 LSTM 处理。采用相同设置，我们得到 6 个模型，即 LSTM-DRL-5，LSTM-

DRL-10，LSTM-DRL-D，LSTM-DRL-5-J，LSTM-DRL-10-J 和 LSTM-DRL-D-J。

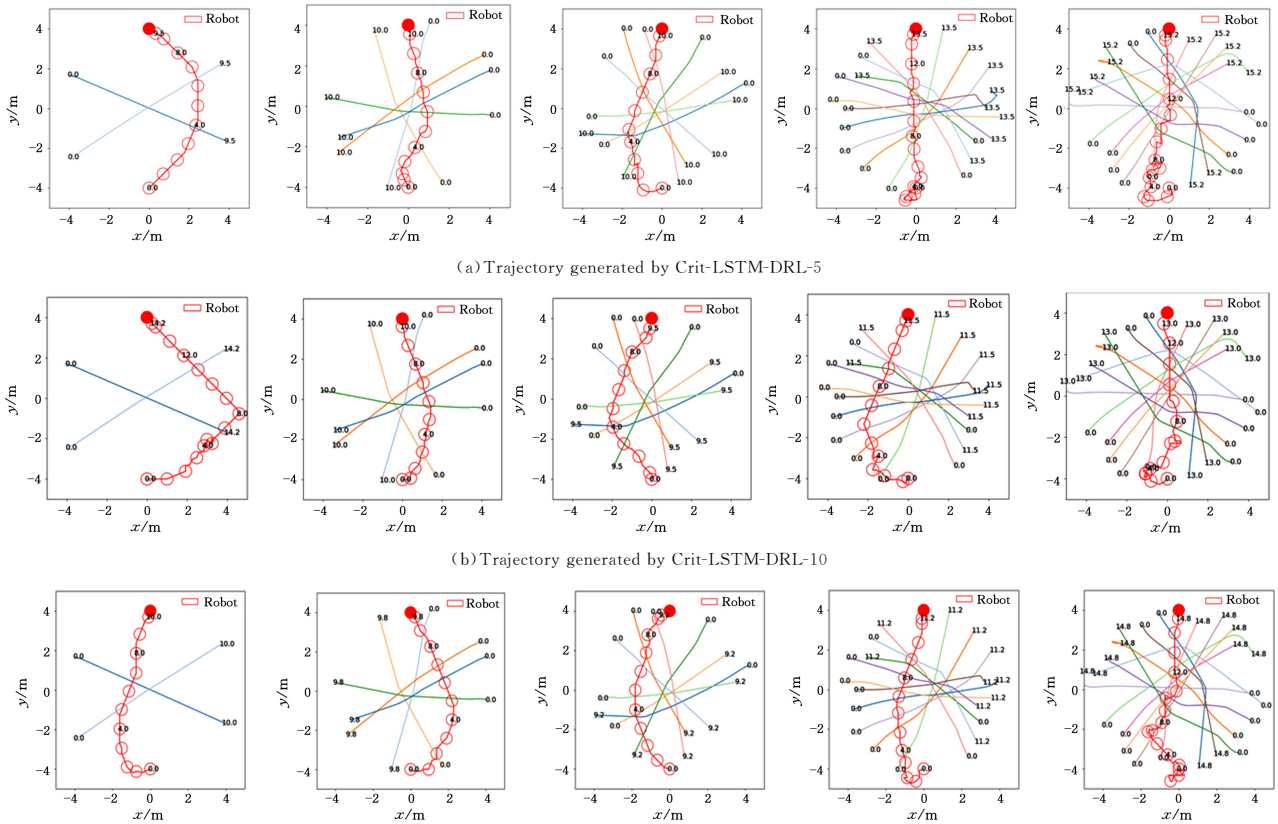
5.2 Crit-LSTM-DRL 有效性评估

(1)计算性能

模型在 CPU 为 Intel © Xeon(R) CPU E5-2697 v3 的电脑上进行训练和测试。Crit-LSTM-DRL 在 3 种不同训练环境下完成 10 000 次迭代所需的时间分别为 26.38 h，55.9 h 和 31.43 h。Crit-LSTM-DRL 在 3 个模型的平均一步决策时间分别为 174.31 ms，177.06 ms，180.50 ms。训练和预测时间主要依赖于动作空间。操作空间越大，时间就越长，因为 Crit-LSTM-DRL 需要查询动作空间中的每个动作并选择最佳动作。本文实验共有 81 个动作，因此一个动作的平均查询时间约为 2.2 ms。

(1)仿真结果

图 4 给出了 Crit-LSTM-DRL-5，Crit-LSTM-DRL-10 和 Crit-LSTM-DRL-D 这 3 个模型生成的来自 5 类不同测试环境的测试用例的轨迹。由图 4(a)可以发现，在包含 2 个、5 个、7 个障碍物的测试用例中，Crit-LSTM-DRL-5 能产生较为光滑的轨迹；但在包含 10 个和 12 个障碍物的测试用例中，Crit-LSTM-DRL-5 生成的轨迹存在严重的徘徊，导致运动时间增加。如图 4(b)所示，Crit-LSTM-DRL-10 在包含 5 个、7 个和 10 个障碍的测试用例中表现良好，在包含 12 个障碍物的测试用例中产生少量徘徊，而在包含 2 个障碍物的测试用例中产生了一个很长的弯路。同理，如图 4(c)所示，Crit-LSTM-DRL-D 在包含 2 个、5 个、7 个和 10 个障碍的测试用例中生成了合适的轨迹，而在包含 12 个障碍物的测试用例中产生了有一定徘徊的轨迹。



(a)Trajectory generated by Crit-LSTM-DRL-5

(b)Trajectory generated by Crit-LSTM-DRL-10

(c)Trajectory generated by Crit-LSTM-DRL-D

注：从左往右依次为在包含 2，5，7，10，12 个障碍物的测试用例中产生的结果

图 4 Crit-LSTM-DRL 在不同测试用例中产生的轨迹

Fig. 4 Trajectories generated by Crit-LSTM-DRL in different test cases

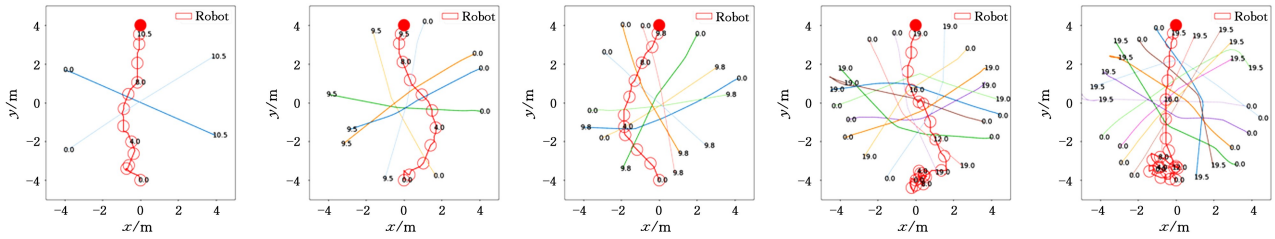
5.3 Crit-LSTM-DRL-J 有效性评估

(1)计算性能

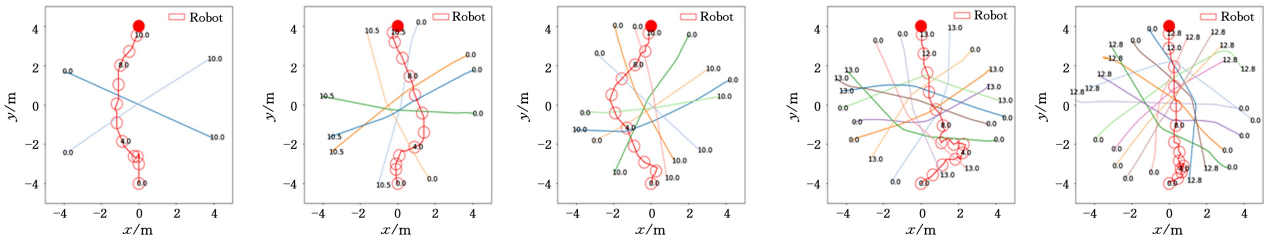
Crit-LSTM-DRL-J 在 3 类训练环境中完成 10000 次迭代所需的时间分别为 25.06 h，48.45 h 和 24.24 h，所得 3 个模型的平均一步决策时间分别是 184.21 ms，176.81 ms，182.52 ms。与 Crit-LSTM-DRL 相同，Crit-LSTM-DRL-J 也需查询动作空间中的每个动作，从而选择最佳动作，其中一个动作的平均查询时间约为 2.26 ms。

(2)仿真结果

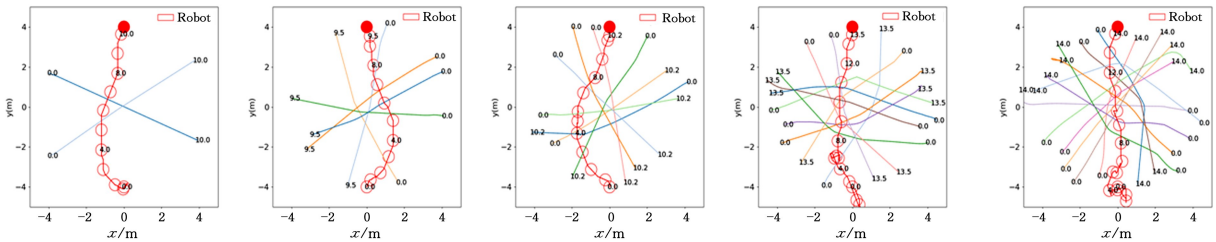
图 5 给出了 Crit-LSTM-DRL-J 在 5 个测试用例中生成的轨迹。如图 5(a)所示，Crit-LSTM-DRL-5-J 在包含 2 个、5 个、7 个障碍物的测试用例中生成了较为光滑的轨迹，而在包含 10 个和 12 个障碍物的测试用例中产生了具有较多徘徊的轨迹。如图 5(b)所示，Crit-LSTM-DRL-10-J 在包含 2 个、5 个、7 个和 10 个障碍物的测试用例中表现良好，而在包含 12 个障碍物的测试用例中产生了一定的徘徊。由图 5(c)可以发现，Crit-LSTM-DRL-D-J 在包含 2 个、5 个、7 个、10 个障碍的测试用例中生成了光滑的轨迹，而在包含 12 个障碍物的测试用例中生成了具有一定徘徊的轨迹。



(a) Trajectory generated by Crit-LSTM-DRL-5-J



(b) Trajectory generated by Crit-LSTM-DRL-10-J



(c) Trajectory generated by Crit-LSTM-DRL-D-J

注：从左往右依次为在包含 2,5,7,10,12 个障碍物的测试用例中产生的结果

图 5 Crit-LSTM-DRL-J 在不同测试用例中产生的轨迹

Fig. 5 Trajectories generated by Crit-LSTM-DRL-J in different test cases

5.4 Crit-LSTM-DRL 和 Crit-LSTM-DRL-J 的性能对比

本节对 Crit-LSTM-DRL 和 Crit-LSTM-DRL-J 的性能进行比较。Crit-LSTM-DRL 和 Crit-LSTM-DRL-J 在训练时间和决策时间上没有显著差异。表 1 列出了它们在不同测试集上的平均成功率、碰撞率和超时率。由表 1 可以发现：1)Crit-LSTM-DRL-5 和 Crit-LSTM-DRL-5-J 在前两类测试集中能够保持较高的成功率；当障碍物个数大于 5 个时，随着障碍物的增加，成功率逐渐下降，碰撞率和超时率升高。由于考虑了障碍物和机器人的交互影响，Crit-LSTM-DRL-5-J 的平均碰撞率降低了 70%。但是，在密集障碍物的碰撞规避过程中，机器人的运动出现了局部徘徊的情况，导致运动时间增加，超时率上升。2) Crit-LSTM-DRL-10-J 的性能优于 Crit-LSTM-DRL-10。Crit-LSTM-DRL-10-J 和 Crit-LSTM-DRL-10 具有相似的平均成功率(0.963 vs 0.966)，但是 Crit-LSTM-DRL-10-J 具有更低的平均碰撞率。3)在动态训练环境下，Crit-LSTM-DRL-D-J 和 Crit-LSTM-DRL-D 的性能没有显著差别，说明 Crit-LSTM-DRL 和 Crit-LSTM-DRL-J 能够在不同环境中有效地进行轨迹规划。Crit-LSTM-DRL-J 通过障碍物和机器人的联合状态来表示它们之间的相互影响，从而在保证成功率的前提下，有效地降低碰撞率。但在碰撞规避过程中，它会导致机器人产生徘徊运动，从而导致超时率增加。

表 1 不同 Crit-LSTM-DRL 模型在不同测试环境中的性能(成功率/碰撞率/超时率)

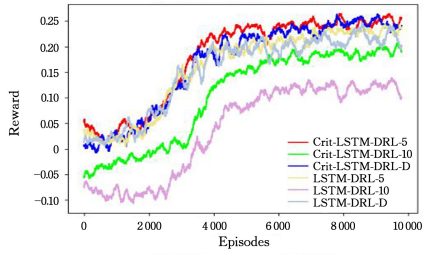
Table 1 Performance of different Crit-LSTM-DRL models in different test environments(success rate/collision rate/timeout rate)

Models	Total average rates	Average rates in different test sets				
		1—4(set1)	5(set2)	6—9(set3)	10(set4)	11—14(set5)
Crit-LSTM-DRL-5	0.894/0.080/0.026	0.97/0.00/0.03	0.99/0.00/0.01	0.96/0.03/0.01	0.88/0.10/0.02	0.67/0.27/0.06
Crit-LSTM-DRL-10	0.966/0.008/0.026	0.90/0.00/0.10	1.00/0.00/0.00	0.99/0.01/0.00	0.98/0.01/0.01	0.96/0.02/0.02
Crit-LSTM-DRL-D	0.978/0.014/0.008	1.00/0.00/0.00	1.00/0.00/0.00	0.99/0.01/0.00	0.97/0.02/0.01	0.93/0.04/0.03
Crit-LSTM-DRL-5-J	0.791/0.024/0.185	0.99/0.00/0.01	0.99/0.01/0.00	0.92/0.01/0.07	0.66/0.03/0.31	0.40/0.07/0.53
Crit-LSTM-DRL-10-J	0.963/0.004/0.033	0.87/0.00/0.13	0.99/0.00/0.01	0.99/0.00/0.01	0.99/0.01/0.00	0.97/0.01/0.02
Crit-LSTM-DRL-D-J	0.970/0.024/0.006	1.00/0.00/0.00	0.99/0.01/0.00	0.98/0.02/0.00	0.96/0.03/0.01	0.93/0.05/0.02

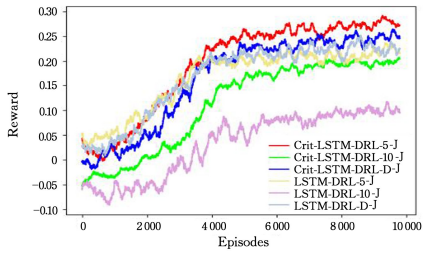
5.5 与 LSTM-DRL 的性能比较

本节比较了 Crit-LSTM-DRL 和 LSTM-DRL 两类模型的性能差异。首先，我们比较了两者在训练阶段的累计激励变化，结果如图 6 所示，其中图 6(a)是障碍物只考虑其自身状态时的训练结果，图 6(b)是障碍物由联合状态表征时的训练结果。从图中可以发现，在同类训练环境中，Crit-LSTM-DRL 比 LSTM-DRL 具有更高的奖励值，说明 Crit-LSTM-DRL 在训练过程中更能收敛于完成给定的运动任务。其次，在相同训练环境中，Crit-LSTM-DRL-J 产生的累计激励值略高于 Crit-LSTM-DRL，说明 Crit-LSTM-DRL-J 在训练后期

产生的碰撞率更低；而 LSTM-DRL-J 和 LSTM-DRL 所产生的奖励值接近，说明 LSTM-DRL-J 的性能没有显著提升，这是因为单纯考虑空间距离，LSTM 无法很好地表征障碍物对机器人碰撞规避决策的影响。最后，对比任意一类模型（即 Crit-LSTM-DRL 或者 LSTM-DRL）在不同环境下的训练结构，可以发现：在包含 10 个障碍物的训练环境中得到的模型的奖励值最低，这是因为随着环境中障碍物的增加，机器人靠近障碍物的概率就越大，从而产生负的一步奖励值。



(a)Cumulative reward curve without the joint state of obstacles

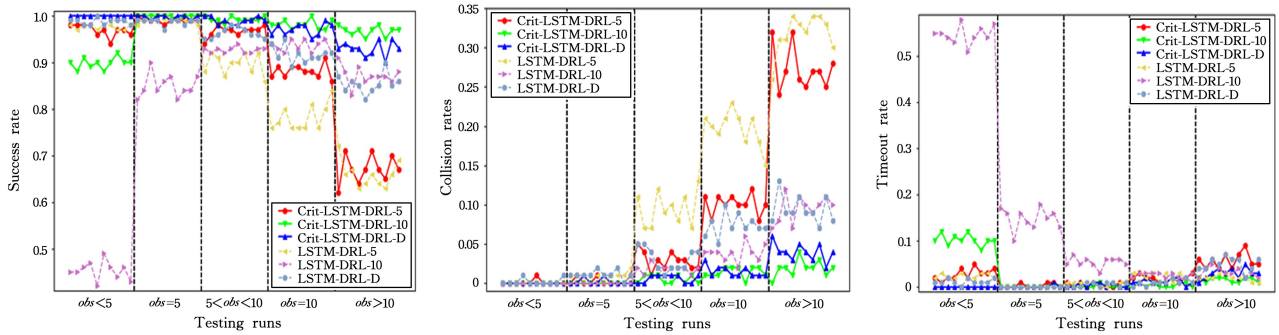


(b)Cumulative reward curve with the joint state of obstacles

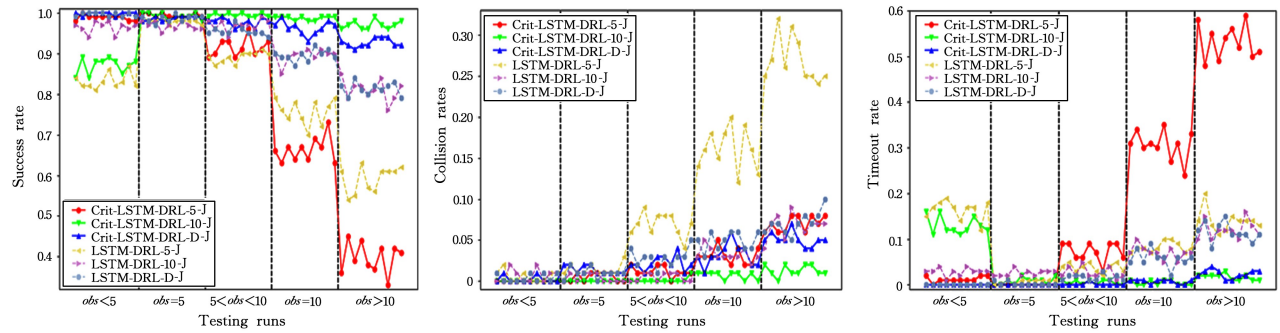
图 6 累积奖励值

Fig. 6 Cumulative reward value

图 7 给出了不同测试环境中的 10 组测试用例集的平均成功率、碰撞率和超时率，表 1 和表 2 分别列出了 Crit-LSTM-DRL 系列和 LSTM-DRL 系列模型在 5 类测试环境中的平均成功率、碰撞率和超时率以及所有测试用例的平均成功率、碰撞率和超时率。图 7(a)给出了障碍物仅由其自身状态表征时所得模型的性能。可以发现：1)与 LSTM-DRL-5 相比，Crit-LSTM-DRL-5 的成功率更高，碰撞率更低。其中，Crit-LSTM-DRL-5 和 LSTM-DRL-5 在前两类测试集中表现出相似的性能，但在第三和第四类测试集中，Crit-LSTM-DRL-5 的成功率明显高于 LSTM-DRL-5；在包含 12 个障碍的测试用例中，两者的成功率都很低。2)Crit-LSTM-DRL-10 的总体性能优于 LSTM-DRL-10。Crit-LSTM-DRL-10 在后 4 类测试环境中都表现出较高的成功率（分别为 100%，99%，98%和 96%），而 LSTM-DRL-10 的成功率相对较低（在第三和第四类测试集中最高仅达到 93%）；在第一类测试集中，Crit-LSTM-DRL-10 的成功率达到 90%，而 LSTM-DRL-10 的成功率仅为 45%。在所有测试集中，Crit-LSTM-DRL-10 的平均成功率为 96.6%，碰撞率仅为 0.8%，而 LSTM-DRL-10 的平均成功率和碰撞率分别为 80.6%和 3.2%。3)Crit-LSTM-DRL-D 在前两类测试集中的性能与 LSTM-DRL-D 相当，但是在后三类测试集中具有显著优势，即可以获得更高的成功率、更低的碰撞率和更低的超时率。综上所述，在所有测试用例中，Crit-LSTM-DRL 的 3 个模型的总平均成功率分别提高了 3.7%，19.9%和 3.8%，总平均碰撞率分别降低了 35.5%，75%和 66.7%。因此，Crit-LSTM-DRL 能够更好地规避碰撞。



(a)Performance comparison between Crit-LSTM-DRL and LSTM-DRL



(b)Performance comparison between Crit-LSTM-DRL-J and LSTM-DRL-J

图 7 不同模型的性能比较

Fig. 7 Performance comparison of different models

图 7(b)给出了障碍物由障碍物自身状态和机器人状态联合表征时所得模型的性能。同理有如下发现：1)Crit-LSTM-DRL-5-J 在障碍物较少的环境中（前三类测试集）的成功率比 LSTM-DRL-5-J 更高。然而，当障碍物个数增加后，

两者都没能很好地完成任务，其中 Crit-LSTM-DRL-5-J 的成功率更低。这是因为在障碍物较多时，为防止发生碰撞，Crit-LSTM-DRL-5-J 会让机器人在某些地方徘徊，从而导致运动时间增加，无法在规定的时间内完成任务。因此，Crit-LSTM-DRL-5-J 具有较低的碰撞率和较高的超时率。2)类似于 Crit-LSTM-DRL，Crit-LSTM-DRL-10-J 比 LSTM-DRL-10-J 具有更高的成功率、更低的碰撞率和超时率，Crit-LSTM-DRL-D-J 比 LSTM-DRL-D-J 具有更高的成功率和更低的碰撞率、超时率。综上所述，Crit-LSTM-DRL-5-J 虽然在总平均成功率上降低了 2.59%，但碰撞率降低了 76.9%；Crit-LSTM-DRL-10-J 和 Crit-LSTM-DRL-D-J 在总平均成功率上提升了 4.4% 和 4.5%，碰撞率显著降低了 81.8%和 25%。

表 2 不同 LSTM-DRL 模型在不同测试环境中的性能(成功率/碰撞率/超时率)

Table 2 Performance of different LSTM-DRL models in different test environments(success rate/collision rate/timeout rate)

Models	Total average rates	Average rates in different test sets				
		1－4(set1)	5(set2)	6－9(set3)	10(set4)	11－14(set5)
LSTM-DRL-5	0.862/0.124/0.014	0.98/0.00/0.02	0.99/0.01/0.00	0.90/0.09/0.01	0.78/0.20/0.02	0.66/0.32/0.02
LSTM-DRL-10	0.806/0.032/0.162	0.45/0.00/0.55	0.85/0.00/0.15	0.93/0.02/0.05	0.93/0.04/0.03	0.87/0.10/0.03
LSTM-DRL-D	0.942/0.042/0.016	0.99/0.00/0.01	0.99/0.01/0.00	0.96/0.03/0.01	0.92/0.07/0.01	0.85/0.10/0.05
LSTM-DRL-5-J	0.812/0.104/0.084	0.83/0.01/0.16	0.98/0.01/0.01	0.90/0.07/0.03	0.76/0.16/0.08	0.59/0.27/0.14
LSTM-DRL-10-J	0.922/0.022/0.056	0.96/0.01/0.03	0.97/0.00/0.03	0.97/0.00/0.03	0.90/0.03/0.07	0.81/0.07/0.12
LSTM-DRL-D-J	0.928/0.032/0.040	1.00/0.00/0.00	0.99/0.01/0.00	0.95/0.03/0.02	0.89/0.05/0.06	0.81/0.07/0.12

根据上述实验结果，我们可以发现基于时间和空间的障碍物危急定义能够有效地反应障碍物在机器人碰撞规避中的影响，从而能够很好地降低碰撞率，保证机器人的安全。

结束语 本文提出了一种新的以障碍物危急程度为导向的、基于深度强化学习的实时轨迹规划算法。该方法首先通过时间和空间两个维度定义障碍物的碰撞危急程度；然后根据障碍物的碰撞危急程度，用 LSTM 来提取固定维度的环境表征向量，该向量和机器人状态一起作为 DRL 的输入；最后，由 DRL 选择适合的速度来最大化价值。本文评估了该方法在不同训练环境下得到的模型在不同测试环境中的性能，同时和现有方法进行了比较。仿真实验结果验证了所提方法的有效性。未来我们将在真实的机器人平台上部署并评估我们的方法；当然，针对多障碍物徘徊问题，我们也将设计新的激励函数以减少超时率；最后，我们也将研究基于学习的方法与传统轨迹规划方法的结合，同时保证安全性和实时性。

参 考 文 献

[1] KHAMIS A，HUSSEIN A，ELMOGY A. Multi-robot task allocation：A review of the state-of-the-art[M]// Cooperative Robots and Sensor Networks 2015. Springer International Publishing，2015：31-51.

[2] LUO J L，NI H J，ZHOU M C. Control program design for automated guided vehicle systems via Petri nets[J]. IEEE Transactions on Systems，Man，and Cybernetics：Systems，2014，45(1)：44-55.

[3] ZHOU Y，HU H，LIU Y，et al. Collision and deadlock avoidance in multirobot systems：A distributed approach[J]. IEEE Transactions on Systems，Man，and Cybernetics：Systems，2017，47(7)：1712-1726.

[4] ZHOU Y，HU H，LIU Y，et al. A distributed method to avoid higher-order deadlocks in multi-robot systems[J]. Automatica，2020，112：108706.

[5] VAN DEN BERG J P，OVERMARS M H. Roadmap-based motion planning in dynamic environments[J]. IEEE Transactions on Robotics，2005，21(5)：885-897.

[6] MARBLE J D，BEKRIS K E. Asymptotically near-optimal planning with probabilistic roadmap spanners[J]. IEEE Transactions on Robotics，2013，29(2)：432-444.

[7] KLOETZER M，MAHULEA C，GONZALEZ R. Optimizing cell decomposition path planning for mobile robots using different metrics[C]// International Conference on System Theory，Control and Computing(ICSTCC). IEEE，2015：565-570.

[8] PIVTORAIKO M，KNEPPER R A，KELLY A. Differentially constrained mobile robot motion planning in state lattices[J]. Journal of Field Robotics，2009，26(3)：308-333.

[9] BIRCHER A，ALEXIS K，SCHWESINGER U，et al. An incremental sampling-based approach to inspection planning：the rapidly exploring random tree of trees[J]. Robotica，2017，35(6)：1327-1340.

[10] TANNER H G，BODDU A. Multiagent navigation functions revisited[J]. IEEE Transactions on Robotics，2012，28(6)：1346-1359.

[11] VAN DEN BERG J，GUY S J，LIN M，et al. Reciprocal n-body collision avoidance [M]// Robotics research. Berlin：Springer，2011：3-19.

[12] ALONSO-MORA J，BEARDSLEY P，SIEGWART R. Cooperative collision avoidance for nonholonomic robots [J]. IEEE Transactions on Robotics，2018，34(2)：404-420.

[13] ABICHANDANI P，FORD G，BENSON H Y，et al. Mathematical programming for multi-vehicle motion planning problems [C]// IEEE International Conference on Robotics and Automation. IEEE，2012：3315-3322.

[14] ZHOU Y，HU H，LIU Y，et al. A real-time and fully distributed approach to motion planning for multirobot systems[J]. IEEE Transactions on Systems，Man，and Cybernetics：Systems，2017，49(12)：2636-2650.

[15] ZHOU Y，HU H，LIU Y，et al. Distributed approaches to motion control of multiple robots via discrete event systems[J]. Control Theory & Applications，2018，35(1)：110-120.

[16] WANG H，WANG X F，ZHANG B，et al. A method for planning the path of mobile robot moving on general terrain[J]. Ruan Jian Xue Bao/Journal of Software，1995(3)：173-178.

[17] ZHANG X D，ZHAN D C，WANG C Y，et al. Key technologies

of indoor navigation based on heuristic path planning and IMU [J]. Ruan Jian Xue Bao/Journal of Software, 2015, 26(S1): 78-89.

[18] LIN Y S, LI Q S, LU P H, et al. Shelf and AGV path cooperative optimization algorithm used in intelligent warehousing[J]. Ruan Jian Xue Bao/Journal of Software, 2020, 31(9): 2770-2784.

[19] ZHU Q B. Ant algorithm for navigation of multi-robot movement in unknown environment[J]. Journal of Software, 2006, 17(9): 1890-1898.

[20] CHEN Y F, EVERETT M, LIU M, et al. Socially aware motion planning with deep reinforcement learning[C]// IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017: 1343-1350.

[21] CHEN C, LIU Y, KREISS S, et al. Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning[C]// International Conference on Robotics and Automation(ICRA). IEEE, 2019: 6015-6022.

[22] CHEN Y F, LIU M, EVERETT M, et al. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning[C]// IEEE International Conference on Robotics and Automation(ICRA). IEEE, 2017: 285-292.

[23] EVERETT M, CHEN Y F, HOW J P. Motion planning among dynamic, decision-making agents with deep reinforcement learning[C]// IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS). IEEE, 2018: 3052-3059.

[24] BENCY M J, QURESHI A H, YIP M C. Neural path planning: Fixed time, near-optimal path generation via oracle imitation [C]// IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS). IEEE, 2019: 3965-3972.

[25] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Playing atari with deep reinforcement learning[J]. arXiv: 1312.5602, 2013.

[26] ZHU Y, ZHAO D, LI X. Iterative adaptive dynamic programming for solving unknown nonlinear zero-sum game based on online data[J]. IEEE Transactions on Neural Networks and Learning Systems, 2016, 28(3): 714-725.

[27] TAI L, PAOLO G, LIU M. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation[C]// IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS). IEEE, 2017: 31-36.

[28] FENG S, SHU H, XIE B Q. Path planning for 3D environment based on improved deep reinforcement learning[J]. Computer Applications and Software, 2021, 38(1): 250-255.

[29] WANG K, BU X J, LI R F, et al. Path Planning for Robots Based on Deep Reinforcement Learning by Depth Constraint [J]. Journal of Huazhong University of Science and Technology (Natural Science Edition), 2018, 46(12): 77-82.

[30] LI H, QI Y M. A Path Planning Method for Robot Based on Deep Reinforcement Learning in Complex Environment[J]. Application Research of Computers, 2020, 37(S1): 129-131.

[31] SAWADA R, SATO K, MAJIMA T. Automatic ship collision avoidance using deep reinforcement learning with LSTM in continuous action spaces[J]. Journal of Marine Science and Technology, 2021, 26(2): 509-524.

[32] NAEEM M, RIZVI S T H, CORONATO A. A gentle introduction to reinforcement learning and its application in different fields[J]. IEEE Access, 2020, 8: 209320-209344.

[33] CHEN Z H, YANG Z H, WANG H B, et al. Recent Researches on Robot Autonomous Grasp Technology[J]. Control and Decision, 2008(9): 961-968, 975.

[34] LE CUN Y, BENGIO Y, HINTON G. Deep learning[J]. Nature, 2015, 521(7553): 436-444.

[35] MOUSSA M A. Combining expert neural networks using reinforcement feedback for learning primitive grasping behavior[J]. IEEE Transactions on Neural Networks, 2004, 15(3): 629-638.

[36] ZHAO D B, SHAO K, ZHU Y H, et al. Review of deep reinforcement learning and discussions on the development of computer Go[J]. Control Theory & Applications, 2016, 33(6): 701-717.

[37] WAN L P, LAN X G, ZHANG H B, et al. A review of deep reinforcement learning theory and application[J]. Pattern Recognition and Artificial Intelligence, 2019, 32(1): 67-81.

[38] LIU Q, ZHAI J W, ZHANG Z Z, et al. A survey on deep reinforcement learning[J]. Chinese Journal of Computers, 2018, 41(1): 1-27.

[39] LIU J W, GAO F, LUO X L. Survey of deep reinforcement learning based on value function and policy gradient[J]. Chinese Journal of Computers, 2019, 42(6): 1406-1438.

[40] LIU Z Y, MU Z X, SUN C Y. An overview on algorithms and applications of deep reinforcement learning[J]. Chinese Journal of Intelligent Science and Technology, 2020, 2(4): 314-326.

[41] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529-533.

XU Linling, born in 1997, postgraduate. Her main research interests include robot intelligent control and deep reinforcement learning.

HUANG Hongyun, born in 1977, master, lecturer. Her main research interests include intelligent system modeling and analysis and information management.

(责任编辑:何杨)