基于路径规划和深度强化学习的机器人避障导航研究

康 振 兴
(合肥工业大学仪器科学与光电工程学院 安徽 合肥 230009)

摘 要 针对移动机器人的长距离避障导航问题,提出结合深度强化学习(Deep Reinforcement Learning,DRL)和路径规划(Path Planning,PL)的避障导航算法。该方法通过快速扩展随机树(Rapidly Exploring Random Tree,RRT)算法在长距离的路径上进行规划,根据生成的路径节点,将长距离路径划分为若干短距离,而在短距离的导航问题上利用深度强化学习的算法,训练一个具有环境感知和智能决策能力的端到端避障导航模型。仿真实验表明,相较于仅用 DRL 的避障导航,该方法使移动机器人的长距离避障导航性能有了大幅度提升,解决了 DRL 在长距离避障导航任务上的局限性问题。

关键词 深度强化学习 路径规划 移动机器人 长距离避障导航

中图分类号 TP18 TP3 **文献标志码** A DOI: 10.3969/j.issn.1000-386x.2024.01.043

ROBOT OBSTACLE AVOIDANCE NAVIGATION BASED ON PATH PLANNING AND DEEP REINFORCEMENT LEARNING

Kang Zhenxing
(*School of Civil and Hydraulic Engineering, Hefei University of Technology, Hefei 230009, Anhui, China*)

Abstract Aimed at the long-distance obstacle avoidance navigation problem of mobile robots, an obstacle avoidance navigation algorithm combining deep reinforcement learning (DRL) and path planning (PL) is proposed. This method used the rapidly exploring random tree (RRT) algorithm to plan the long-distance path. According to the generated path nodes, the long-distance path was divided into several short distances. And for the navigation problem in the short distance, DRL algorithms were used to train an end-to-end obstacle avoidance navigation model with environmental perception and intelligent decision-making capabilities. Simulation experiments show that compared with obstacle avoidance navigation using only DRL, this method greatly improves the long-distance obstacle avoidance navigation performance of mobile robots, and solves the limitations of DRL in long-distance obstacle avoidance navigation tasks.

Keywords Deep reinforcement learning Path planning Mobile robot Long-distance obstacle avoidance navigation

0 引 言

对于自主移动机器人,自主导航是其最基本的功能,即在不发生碰撞的情况下,可以从起始位置运动到目标点。在解决自主导航问题中,传统大多是采用全局路径规划和局部路径规划组合的方法[1-2]。全局路径规划是在全局地图已知的情况下,在静态环境中搜索出一条可行的路径,机器人沿着该路径运动到目标点[3];局部路径规划,需要机器人在运动的过程中,结合自身对环境的感知,避开出现在已规划的路径之中的移动障碍物[4-6]。

近年来,深度强化学习快速发展,由于其端到端的优势,使得在自动控制[7]、通信网络[8]、自然语言处理[9]等很多领域都具有广泛的应用[6]。在机器人避障导航的问题中,基于深度强化学习的方法也有大量的研究,文献[10]采用的 DQN 神经网络的深度强化学习算法[7],为了使得算法的训练容易收敛,将导航目标点的坐标按照其处于机器人的视野角度范围进行离散化,并且机器人的运动速度也按照前进、后退、左转、右转等固定的指令进行离散化处理,但是这对机器人的控制不够精细和灵活。文献[11]引入了动作连续

收稿日期:2020-11-14。康振兴,硕士生,主研领域:深度强化学习。

控制的 DRL 算法，但实验场景和训练任务过于简化，难以做到模型的稳定性。文献[13]和文献[19]在利用 DRL 在训练的 DRL 模型在短距离任务上有较优的性能，但在实际导航任务上缺乏长远规划，仅仅根据机器人当前的环境观测将无法执行长距离的任务，或得到最优的运动路径。本文基于最大熵框架的软-演员评论家(Soft-Actor-Critic，SAC)算法，提出了改进的并行式 SAC(Parallel SAC，PSAC)训练算法，提升了在实际复杂场景下深度强化学习模型的稳定性；并结合全局路径规划的方法，解决了长距离避障导航任务下深度强化学习方法的难以泛化的问题。

本文先利用深度强化学习算法解决复杂短距离场景的避障导航问题，接着针对深度强化学习在处理长距离避障导航的不足之处，提出了结合路径规划和深度强化学习的长距离避障导航方法，并在 Player/Stage 仿真器中进行训练以及测试的仿真实验，实验表明了上述的方法的有效性和稳定性。

1 基于强化学习的避障算法

1.1 强化学习基本原理介绍

强化学习(Reinforcement Learning，RL)是机器学习的一个领域，相比其他机器学习方法，其解决的问题主要面向序列决策，由智能体在环境中不断交互，根据环境给出的回报反馈，迭代更新决策。强化学习基本框架如图 1 所示，智能体在与环境的每一个状态 $s(t)$ 中，会选择行动 $a(t)$，执行完 $a(t)$ 后，进入状态 $s(t+1)$，并且从环境得到一个回报信号 $r(t)$，经过不断的“试错”，智能体的决策会朝着得到更高的回报 $r(t)$ 的方向进行改善更新[12]。

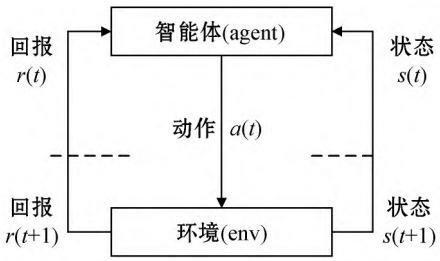


图 1 强化学习基本框架

1) 基于值函数的方法。设智能体在环境中从 t_1 时刻开始运动到 t_2 时刻结束，则定义这段时间里获得累积回报为：

$$R_t = \sum_{t_1}^{t_2} \gamma^{t-t_1} r_t \tag{1}$$

式中：$\gamma(0 \leqslant \gamma \leqslant 1)$ 是时间折扣因子，表示未来的决策回报对现在决策的影响。

动作价值函数 Q 定义为，在状态 s 下采用策略 π 选择某个动作，得到的累积回报的期望，即：

$$Q^{\pi}(s,a) = E_{\pi}[R_t \mid s_t = s, a_t = a] \tag{2}$$

而最优策略表示在状态空间里，根据该策略选择的动作，得到的 Q 值都是最大的，即：

$$Q^{*}(s,a) = \max_{\pi} E_{\pi}[R_t \mid s_t = s, a_t = a] \tag{3}$$

基于值函数的学习方法，就是通过 Q 值来引导策略的更新方向，更高的 Q 值代表更优的策略。Q 值更新的过程如下表示：

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha[r + \gamma \max_{a'}(Q_t(s',a')) - Q_t(s,a)] \tag{4}$$

式中：$\alpha(0 \leqslant \alpha \leqslant 1)$ 是学习率；r 是在状态 s 下执行动作 a 后的回报；s' 表示在状态 s 下执行动作 a 之后的状态。

智能体每一个行程结束时，会更新一次 Q 值表，下一次在环境中运动时会利用已经更新的 Q 值进行动作选择，Q 值越大的动作被选中的概率就越。通过不断交互策略评估和策略改善的步骤，最终迭代至最优策略。

2) 深度强化学习算法。对于连续状态空间和连续的动作空间的情况，传统的基于表格的查询和存储的强化学习算法将会导致维数灾难的问题。而深度神经网络由于其强大的逼近能力，可以用来对值函数加以拟合；在策略梯度的方法中，深度神经网络也同样可以对策略做近似逼近。对于解决观测状态连续、动作离散的问题，有经典的 DQN 算法，其利用卷积神经网络进行环境信息的特征提取，用有限的输出节点表示每个动作对应的 Q 值大小。最终，在 Atari 游戏中表现出很好的效果。对于动作连续的情况，则有 DDPG 算法[14]，PPO 算法[15-16]等结合了策略梯度和值函数学习的算法，在解决高维度的环境观测和动作问题上，大多都行之有效。

1.2 神经网络结构设计

对机器人状态的描述决定了网络结构的输入，本文定义的机器人状态，不仅包括在环境中观测到的障碍物信息，也包括相对于目标点的位置信息，以及机器人自身的速度信息。本文设计的深度神经网络结构如图 2 所示。其中，观测的障碍物信息经过两层一维卷积结构和一层全连接结构，得到深层次的观测特征；之后再与机器人的速度信息，以及目标点的位置信息相融合，得到描述机器人状态的特征向量；神经网络最后一层为全连接层，是机器人状态到动作的映射，最终输出两个二维向量，分别代表机器人速度的期望向量以及速度的对数方差向量。机器人的速度包括直线速度和转角速度两个维度，基于在实际应用中安全方面的

考虑,对机器人最终的执行速度大小做了以下限幅的操作: 使用 sigmoid 激活函数将直线速度其限制在 0 ~ 1 m/s; 对转角速度,则使用了 tanh 激活函数将其限制在 -1 ~ 1 rad/s。

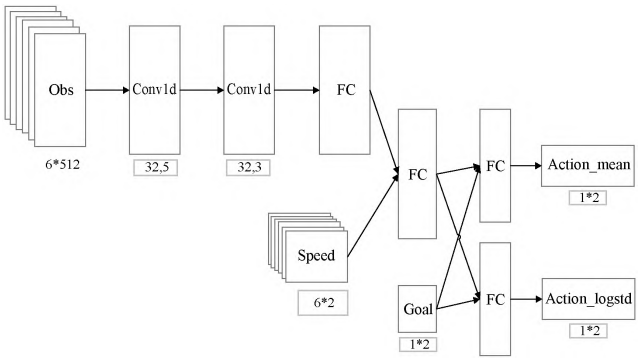


图 2　深度神经网络结构

1.3　回报函数设计

回报函数是机器人在某状态下采取某个动作的奖惩值,本文设计了连续的回报函数,相比于稀疏回报,更利于模型的收敛[10],使机器人更快速地获取避障导航能力。回报函数由两项组成:

$$reward = reward\_step + reward\_final \tag{5}$$

式中: $reward\_step$ 是机器人每一步获得的回报,由机器人前一步离目标点的距离与当前步离目标点的距离差值决定。$reward\_final$ 是机器人运动结束状态的 $reward$,如果机器人发生了碰撞,则 $reward\_final$ 为 -15; 如果顺利到达目标点,取值 +15; 否则在一定的运动时间内既没有碰撞也没有到达目标点,则结束运动状态,取值为 0。

$$reward\_step = ratio * (dist\_pre - dist\_cur) \tag{6}$$

式中: $ratio$ 为比例常数取值为 1.5。

$$reward\_final = \begin{cases} -15 & \text{crash} \\ 15 & \text{reach goal} \\ 0 & \text{其他} \end{cases} \tag{7}$$

1.4　训练算法

为了加快训练速度,提高样本利用率,并平衡机器人探索和利用,本文针对机器人避障导航的具体任务,在 SAC 算法基础上进行,将其改进为并行式算法 PSAC,使得训练收敛更快,模型也更加稳定。PSAC 是将常规的单场景下,单机器人的 SAC 训练模式优化为多个机器人同时在多个场景下训练,其优势在于可大大提高机器人在环境中的采样效率,单位时间里采集的样本数与机器人数成正比,因此可以加快收敛速度; 其次,训练场景下的每个机器人既是避障主体,也互为动态障碍物,使场景更加的动态化; 最后,多机器人并行训练,在环境中采集的训练样本也更多样化,模型的泛化性能更好。SAC 算法是目前在多项任务中表现最优的算法,是基于最大熵框架的方法,相比于其他的算法,其更具备探索性,是目前最高效的深度强化学习算法[18]。SAC 总的策略优化目标为:

$$\pi^* = \arg\max_{\pi} \sum_t E_{(s_t,a_t)\sim\rho_\pi} [r(s_t,a_t) + \alpha H(\pi(\cdot|s_t))] \tag{8}$$

式中: $r(s_t,a_t)$ 是回报函数,$H(\pi(\cdot|s_t))$ 是状态下的熵,α 为回报函数和熵一同作为优化目标的权重。

SAC 也是 AC 框架的一种,其 critic 网络的更新目标为:

$$J_Q(\theta) = E_{(s_t,a_t)\sim D}\left[\frac{1}{2}\left(Q_\theta(s_t,a_t) - \left(r(s_t,a_t) + \gamma E_{s_{t+1}\sim p}\left[V_{\bar{\theta}}(s_{t+1})\right]\right)\right)^2\right] \tag{9}$$

使用梯度下降法优化上述目标时,梯度的计算如下:

$$\hat{\nabla}_\theta J(\theta) = \nabla_\theta Q_\theta(a_t,s_t)\left(Q_\theta(s_t,a_t) - \left(r(s_t,a_t) + \gamma\left(Q_{\bar{\theta}}(s_{t+1},a_{t+1}) - \alpha\log(\pi_\phi(a_{t+1}|s_{t+1}))\right)\right)\right) \tag{10}$$

actor 的优化目标为:

$$\pi_{new} = \arg\min_{\pi'\in\Pi} D_{KL}\left(\pi'(\cdot|s_t) \,\|\, \frac{\exp\left(\frac{1}{\alpha}Q^{\pi_{old}}(s_t,\cdot)\right)}{Z^{\pi_{old}}(s_t)}\right) \tag{11}$$

通过让新的策略与 critic 的 KL 之间的散度尽可能小,即越接近 critic 的分布,则 actor 会获得更大的熵。训练过程中,为了使模型在新环境中的泛化性能更强,机器人在环境中采样训练时采用了域随机化(Domain Randomization)的方法[20]做了数据增强的处理。基于 PSAC 的机器人避障导航任务的训练伪代码如算法 1 所示。

算法 1　机器人避障训练

PSAC 算法

if 进程号 = 0

　初始化 Q_1 网络(θ_1),Q_2 网络(θ_2),policy 网络(φ)

　初始化目标网络 $Q_{1\_t\ arg\ et} = Q_1$,$Q_{2\_t\ arg\ et} = Q_2$

　初始化一个空的样本池 R

for 回合数 = 1,2,…,M do

　while(没有碰到障碍物) do

　　if 进程 = 0

　　进程 0 收集其他进程机器人的状态,组合成状态列表 s_{t_list};

　　$a_{t\_lsit} \sim \pi_\varphi(a_{t\_list}|s_{t\_list})$ 按照策略网络进行动作采样;

　　将计算得到的速度列表中的值对应地分发给其他进程,以控制机器人运动;

else

等待进程 0 发送速度指令;

$s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)$ 从环境中采样得到下一个状态;

if 进程 = 0

$R \leftarrow R \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1})\}$ 新的样本加进样本

$\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$;

for $i \in \{1,2\}$

$\varphi \leftarrow \varphi - \lambda_\pi \hat{\nabla}_\varphi J_\pi(\varphi)$

$\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$

$\bar{\theta}_i \leftarrow \tau\theta_i + (1-\tau)\bar{\theta}_i$ for $i \in \{1,2\}$

end for

得到训练好的策略模型 $\theta_1, \theta_2, \varphi$

2 结合路径规划的长距离导航

2.1 路径规划算法介绍

基于深度强化学习的避障算法,可以在不需要地图的环境中,在复杂的障碍物场景下具备出色的动态避障能力[13,22]。基于 DRL 算法训练避障导航机器人时,如果基于长距离,或是路径迂回地图来训练机器人长距离的点到点避障导航功能,会因为机器人获得的回报函数过于稀疏,回报函数稀疏通常是强化学习的一个难点[23],这会导致训练无法收敛。因而在机器人避障导航任务中,通常选择较短距离,较易探索的障碍物场景进行 DRL 的训练。

另外,由于强化学习的算法自身的局限性,训练好的模型一旦收敛,对环境的探索能力也就随之下降,陷入局部最优化点后便无法自主摆脱陷阱,即强化学习固有的探索(Exploration)与利用(Exploitation)的矛盾[12]。机器人仅仅依赖于当前可见范围内的观测状态做决策,而缺乏全局环境的整体认知,因而容易做出次优的决策,泛化性能随着路径的迂回而逐渐下降。如图 7 所示的情形,基于传统的导航算法就可以完成的避障导航任务,对于 RL 机器人则很容易陷入局部最优解而使最终任务失败。

所以,本文提出一种结合了路径规划和强化学习的联合避障导航算法。

2.2 基于 RRT 的路径规划算法

路径规划,顾名思义就是根据全局的环境信息,在给定的环境中,在不接触环境中障碍物的条件下,规划出一条可以连接起始点和目标点的路线。常用的路径规划算法包括 A^* 算法、人工势场算法、遗传算法等。

RRT 算法[21]属于随机采样的算法,通过在空间中进行路径点的采样,并进行碰撞检测,不需要将环境栅格化建模,可以高效快速地完成对可通行区域的搜索。由于路径点的采样是一个随机事件,概率上是完备的,原则上只要经过足够的探索次数,在任何存在可通行路径的空间下,都可以找到可行的路径,该算法适用于解决未知环境下的完整性规划,被广泛应用于高维空间、约束复杂的路径规划中。

RRT 的过程图 3 所示,该算法以起始点作为树的根节点,以一定的概率 p 向目标点靠近,或是以概率 $1-p$ 向其他偏离目标点的方向做进一步探索。

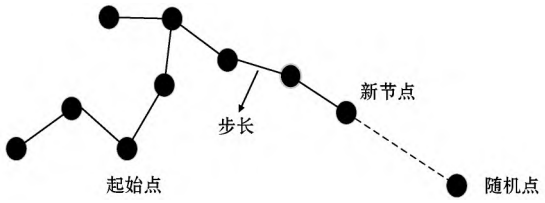


图 3 快速扩张随机树的扩展过程

为了方便描述 RRT 算法的流程,先指定如下的符号,如表 1 所示,RRT 算法流程如算法 2 所示。路径规划的任务,就是在状态空间 Q 中找到一条连通 q_start 到 q_goal 的无障碍物路径。即规划出的路径中的每一个点和连线,都必须规避 Q_obstacle 在 Q-free 空间中。初始化起始节点 q_start,并将其放入 T_k 中作为扩展树的根节点,以初始化 T_k。从起始点开始,开始随机在空间中进行节点的采样得到 q_rand,如步骤 3;遍历扩展树 T_k 中的节点,找到与 q_rand 距离最近的节点 q_near,如步骤 4;再根据节点 q_rand 和 q_near,找到新的扩展节点 q_new,且使得 D(q_near,q_new) = epsilon,判断若 q_new 在 C_free 中,则将 q_new 加入扩展树 T_k 中,树的构建包括节点和边,如步骤 5 - 步骤 8。最终若在 K 次遍历中到达了最终节点 q_goal,则构建结束,根据最终的扩展树查找到由起始点 q_start 到 q_goal 的路径。由于 RRT 算法不需要对空间进行如栅格化等建模的操作,所以适用于在复杂场景等约束条件下的路径规划任务。

表 1 RRT 路径规划问题符号说明

符号	符号意义
q_start	起始点
q_goal	目标点
C	状态空间
C_goal 子集符号 Q	目标区域
C_obstacle 子集符号 Q	障碍物区域
C_free 子集符号 Q	自由状态区域
epsilon	步长
D(x1,x2)	空间 C 中点 x1 与 x2 的距离
T_k	随机树 T 有 k 个节点

算法 2　RRT 伪代码流程

GENERATE_RRT(q_{start}, K, Δt)

1　T. init(q_{start}, K, Δt);
2　for k = 1 K do
3　q_{rand} ← Random_Configuration( );
4　q_{near} ← Nearest_Neighbor(q_{rand}, T);
5　u ← Select_Input(q_{near}, q_{rand});
6　q_{new} ← New_Configuration(q_{near}, u, Δt);
7　T. add_vector(q_{new});
8　T. add_edge(q_{near}, q_{new}, u);
9　return T

3　实验设计和结果

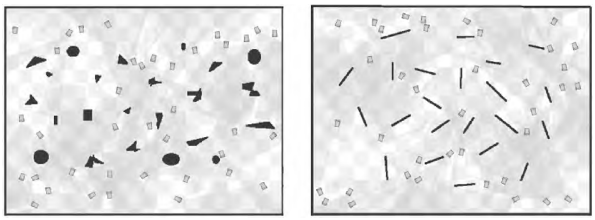
3.1　实验说明

为了验证本文提出的方法的有效性，本文在 Player/Stage 仿真实验平台下实现 DRL 算法的训练和测试。Player 定义了机器人和传感器与 Stage 的通信接口; Stage 是 2D 环境，提供声呐、激光传感器、碰撞检测和执行器等虚拟机器人设备，支持多机器人仿真，在机器人操作系统(Robot Operating System，ROS) 下可以方便地调用，仿真环境是基于 C ++ 搭建和编译的。

SAC 和 PSAC 算法中的神经网络反向求导训练和前向推理测试，均是基于 PyTorch 框架用 Python 语言实现的，其中 PSAC 的多进程算法是基于 MPI4 工具实现的。实验的硬件配置上，显卡型号为 Nvidia 的 1080Ti，显存 12 GB，CPU 的型号为 Intel Core i5，笔记本内存 16 GB。

3.2　模型训练

模型训练时所用的传感器是单线激光雷达，用于感受视野范围内障碍物的距离信息，训练场景设计如图 4 所示。本文设计了两种不同的障碍物组成的训练场景，每个场景分别投放了 38 个机器人，两个场景同时训练，一共开启 76 个进程，每个进程控制一个 RL 机器人进行独立的采样。每当机器人发生碰撞，或到达目标，或在指定时间内没有到达目标都为一个 episode 的结束。训练过程中，随机生成每一个 RL 机器人的起点和目标点，一个 episode 结束时，重新生成起点和目标点进行下一 episode 的采样。每个机器人采集的样本统一保存至样本池，由主进程进行 actor 网络和 critic 网络的参数更新。每个 RL 机器人的速度指令也由主进程根据每个机器人的状态，统一计算并分配到每个进程。网络参数每更新 100 次保存一次。



(a)　　(b)

图 4　PSAC 训练的场景

实验中，机器人的尺寸设置为长 0.44 m，宽 0.33 m; 训练场景(a) (b) 大小均设置为 30 m × 20 m，训练的超参设置如表 2 所示。

表 2　训练参数设置

参数设置	值
优化器	Adam
学习率(lr)	0.000 5
折扣因子(γ)	0.99
样本池大小	1 000 000
目标熵	-2
网络激活函数	ReLU
目标网络平滑系数(τ)	0.003
目标网络更新间隔	1
梯度更新的步数	1

SAC 和 PSAC 的训练过程中每个 episode 的累计回报的变化如图 5 所示。可见由于 PSAC 在单位时间内采集样本更多，所以 episode reward 收敛更快; 另外由于训练 PSAC 的场景下障碍物更多更复杂，所以在训练收敛后每个机器人平均获得的回报函数要低于 SAC。

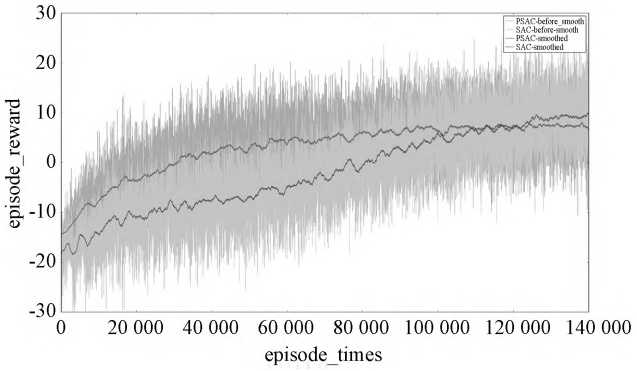
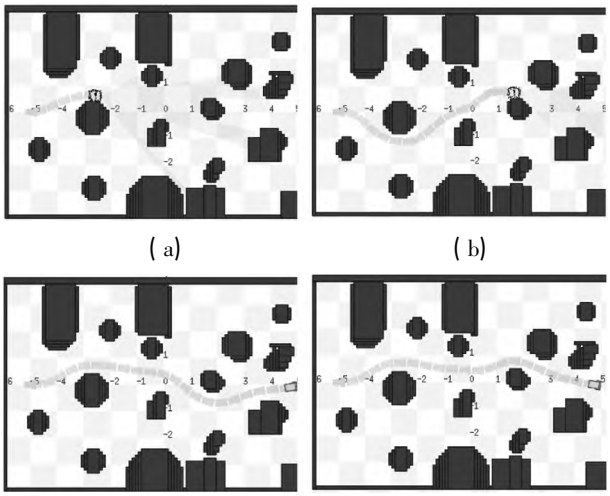


图 5　SAC/PSAC 训练中的 episode reward 变化

3.3　模型测试

为了保证 DRL 模型的泛化性能，DRL 模型的测试场景需要不同于训练场景。图 6 所示是本文提出的 PSAC 算法在测试场景下的表现，指定机器人的初始点为(-5，0) ，目标点是(5，0) 。图 6(a) 、(b) 、(c) 和(d) 的模型编号分别为 100、500、1 000、5 000 步之后的

效果。可见，基于 PSAC 算法训练的模型在 1 000 步，即模型更新次数为 100 000 次时，就具备了较强的稳定性和泛化能力。



(a) (b)

(c) (d)

图 6 PSAC 模型的阶段测试

为了量化避障导航模型的性能，本文选择了避障的运动时间、运动路程、避障成功率等指标进行评估[22]。评估指标说明如表 3 所示。

表 3 避障导航算法的评估指标

参数	描述
Map	是否需要地图(yes/no)
T_path/s	点到点避障导航的时间
L_path/m	点到点避障导航的运动距离
R_success	点到点避障导航的成功率

1) 短距离场景测试。短距离测试场景如图 6 所示，为了验证本文提出的算法的可靠性，实验将 SAC、PSAC 与传统的人工势能场(Artificial Potential Field，APF)避障导航算法在表 5 的指标上进行对比。

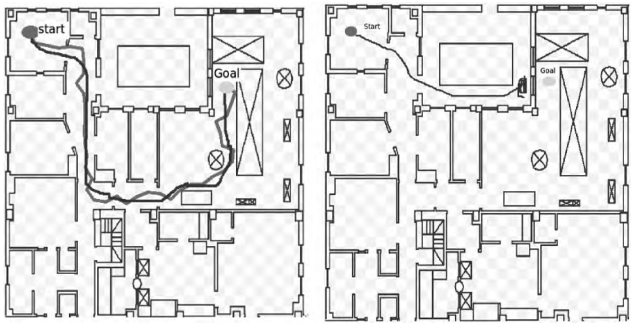
首先，随机地在测试场景下生成机器人的起点，并控制在起点的 4 ~ 8 米距离内再随机生成目标点。最终得到起点-目标点一共 100 对，作为 APF、SAC 和 PSAC 的测试任务。算法的测试效果为 100 个测试的任务的统计平均，如表 4 所示。

表 4 短距离避障导航的评估结果

参数	APF	SAC	PSAC
Map	yes	no	no
T_path/s	13.54	11.21	10.97
L_path/m	11.33	10.56	10.08
R_success/%	84	79	90

结果表明，基于 PSAC 算法训练的模型，比基于 SAC 的更加容易泛化，且比传统的 APF 算法，避障导航时间更短，灵活性更高。

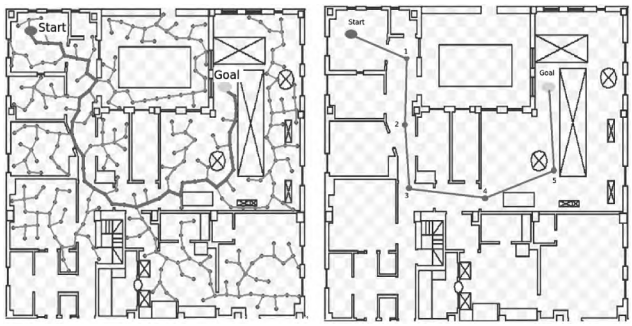
2) 长距离场景测试。如图 7 所示，对比了传统的 RRT + APF 的算法和 PSAC 算法在长距离避障导航任务上的表现。在没有全局路径规划来提供中间目标点情况下，基于 PSAC 的算法容易失败，而传统的导航算法却可以顺利完成任务，且图 7 所示的情况绝非个例。所以基于 PSAC 算法的避障导航比起传统的 APF 算法具备更好的实时动态性能，而不适用于长远距离的导航任务。



(a) RRT + APF (b) PSAC

图 7 长距离避障导航任务对比

RRT + PSAC 算法如图 8 所示，其实现过程描述如下，首先基于 RRT 路径规划算法得到一系列连接起始点与目标点之间的路径节点集合，在节点集合中按照一定的距离选择若干节点作为中间的目标，让机器人依次达到中间的目标点，最终完成长距离的避障导航任务。



(a) RRT (b) RRT + PSAC

图 8 RRT 和 RRT + PSAC 的算法任务对比

设置图 8 的场景为 30 m × 30 m 的大小，随机生成 100 组起点-目标点对，保持起点和目标点间的距离在 20 m ~ 30 m，且保证起点和目标点的 RRT 路径规划有解。使用生成的点集分别进行 PSAC、RRT + APF、RRT + PSAC 的实验。测试效果为 100 个任务的统计平均，实验结果如表 5 所示。结果说明，结合了路径规划和强化学习的避障导航算法可以发挥更大优势。

表 5　长距离避障导航的评估结果

参数	RRT + APF	PSAC	RRT + PSAC
Map	yes	no	yes
T_path/s	56.8	34.3	52.4
L_path/m	46.9	30.5	47.7
R_success/%	94	34	96

4　结　语

本文针对移动机器人的长距离避障导航问题，提出结合 DRL 和 PL 的避障导航的算法，并以代表性的 SAC 算法和 RRT 算法加以验证。该方法充分发挥了深度强化学习的环境感知和决策能力，以及路径规划的全局最优性，通过 RRT 算法在长距离的路径上进行路径点的取样，根据路径的节点将长距离路径划分为若干短距离，在短距离的问题上利用本文改进的 PSAC 算法，训练出一个端到端避障导航机器人。仿真实验表明，相较于仅用深度强化学习进行避障导航，该方法可以通过全局的导航路径优化，大幅度提升 RL 机器人的长距离避障导航性能。

参 考 文 献

[1] 朱大奇，颜明重. 移动机器人路径规划技术综述[J]. 控制与决策，2010，25(7)：961－967.

[2] 刘佳，王杰. 无人水面艇避障路径规划算法综述[J]. 计算机应用与软件，2020，37(8)：1－10.

[3] LaValle S M. Planning algorithms[M]. Cambridge University Press，2006.

[4] Koren Y，Borenstein J. Potential field methods and their inherent limitations for mobile robot navigation[C]//1991 IEEE International Conference on Robotics and Automation. IEEE，1991：1398－1404.

[5] Clerc M，Kennedy J. The particle swarm-explosion，stability，and convergence in a multidimensional complex space[J]. IEEE transactions on Evolutionary Computation，2002，6(1)：58－73.

[6] Li Y. Deep reinforcement learning：An overview[EB]. arXiv：1701.07274，2017.

[7] Mnih V，Kavukcuoglu K，Silver D，et al. Human-level control through deep reinforcement learning[J]. Nature，2015，518(7540)：529－533.

[8] Luong N C，Hoang D T，Gong S，et al. Applications of deep reinforcement learning in communications and networking：A survey[J]. IEEE Communications Surveys & Tutorials，2019，21(4)：3133－3174.

[9] Li J，Monroe W，Ritter A，et al. Deep reinforcement learning for dialogue generation[EB]. arXiv：1606.01541，2016.

[10] 张福海，李宁，袁儒鹏，等. 基于强化学习的机器人路径规划算法[J]. 华中科技大学学报(自然科学版)，2018，46(12)：65－70.

[11] 陈杰，程胜，石林. 基于深度强化学习的移动机器人导航控制[J]. 电子设计工程，2019，27(15)：61－65.

[12] Sutton R S，Barto A G. Reinforcement learning：An introduction[M]. MIT press，2018.

[13] Fan T，Long P，Liu W，et al. Fully distributed multi-robot collision avoidance via deep reinforcement learning for safe and efficient navigation in complex scenarios[EB]. arXiv：1808.03841，2018.

[14] Lillicrap T P，Hunt J J，Pritzel A，et al. Continuous control with deep reinforcement learning[EB]. arXiv：1509.02971，2015.

[15] Schulman J，Levine S，Moritz P，et al. Trust region policy optimization[C]//International Conference on Machine Learning(ICML)，2015：1889－1897.

[16] Schulman J，Wolski F，Dhariwal P，et al. Proximal policy optimization algorithms[EB]. arXiv：1707.06347，2017.

[17] Haarnoja T，Zhou A，Abbeel P，et al. Soft actor-critic：Off-policy maximum entropy deep reinforcement learning with a stochastic actor[EB]. arXiv：1801.01290，2018.

[18] Haarnoja T，Zhou A，Hartikainen K，et al. Soft actor-critic algorithms and applications[EB]. arXiv：1812.05905，2018.

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

[20] Tobin J，Fong R，Ray A，et al. Domain randomization for transferring deep neural networks from simulation to the real world[C]//2017 IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS). IEEE，2017：23－30.

[21] Karaman S，Frazzoli E. Sampling-based algorithms for optimal motion planning[J]. The International Journal of Robotics Research，2011，30(7)：846－894.

[22] 吴运雄，曾碧. 基于深度强化学习的移动机器人轨迹跟踪和动态避障[J]. 广东工业大学学报，2018，36(1)：42－50.

[23] Trott A，Zheng S，Xiong C，et al. Keeping your distance：Solving sparse reward tasks using self-balancing shaped rewards[J]. Advances in Neural Information Processing Systems，2019，32：10376－10386.

[24] François-Lavet V，Henderson P，Islam R，et al. An introduction to deep reinforcement learning[EB]. arXiv：1811.12560，2018.