中国科学 : 信息科学 2026 年 第 56 卷 第 2 期: 394–408
SCIENTIA SINICA Informationis
论文

《中国科学》杂志社
SCIENCE CHINA PRESS
CrossMark click for updates

基于关键路径点的深度强化学习复合路径规划

孙超[1,2], 王江[1], 穆朝絮[1,2,3*], 邢良平[1], 孙长银[1,2*]

1. 安徽大学人工智能学院, 合肥 230601
2. 安徽大学自主无人系统技术教育部工程研究中心, 合肥 230601
3. 天津大学电气自动化与信息工程学院, 天津 300192
* 通信作者. E-mail: cxmu@tju.edu.cn, cysun@ahu.edu.cn

收稿日期: 2025–03–04; 修回日期: 2025–04–07; 接受日期: 2025–06–04; 网络出版日期: 2026–02–03

国家自然科学基金 (批准号: 62495083, 62503005) 和安徽省自然科学基金青年项目 (批准号: 2408085QF208) 资助

摘要 本文提出了一种结合 Voronoi 图和深度强化学习的机器人路径规划方法. 在全局规划方面, 通过 Voronoi 图构建 “安全区域”, 利用 A* 算法和关键路径点提取算法优化路径点, 为局部规划提供目标. 局部规划采用融合阶段关键点与均衡回放策略的 RainbowDQN (stage-key & weight-balanced replay Rainbow deep Q-network, SW-RDQN) 算法, 通过 3 项关键设计提升复杂动态环境下的导航适应性: 首先, 采用多阶段路径分割策略对全局路径进行划分与剪枝, 提取并保留关键路径点信息; 其次, 结合轨迹落点预测与环境势场设计奖励函数, 引导智能体学习更安全有效的避障行为; 最后, 采用基于奖励与 TD 误差动态加权的优先经验回放, 通过偏移与对数归一化提升关键经验采样概率, 从而提高训练效率. 状态输入融合了卷积和多层感知机提取信息, 取若干场景帧送入长短期记忆网络模型, 最后由稠密层做出决策, 提升了机器人对动态环境的感知和响应能力. 该方法有效提升了机器人在动态环境中的路径规划精度与鲁棒性.

关键词 多阶段路径分割, Voronoi 图, SW-RDQN, 落点预测, 场景避障导航

1 引言

机器人路径规划在智能导航领域具有核心地位, 其目标是在复杂环境中实现从起点到终点的安全高效通行. 随着工业自动化与服务机器人的普及, 动态未知环境下的路径规划成为当前研究热点. 受文献 [1] 中复合路径规划方式启发, 本文提出一种基于关键路径点的深度强化学习复合路径规划框架, 通过全局 – 局部协同优化策略突破现有方法的性能瓶颈.

全局路径规划是在一个已知的环境中, 为机器人生成一条从起始点到目标点的最优路径. 传统方法通常依赖于全局地图的详尽信息在环境中计算出一条可通行路径. 文献 [2] 改进了 PSO (particle swarm optimization) 算法, 使得导航效果有一定进步. 然而, 随着实际应用场景日趋复杂, 实际导航过程中, 机器人仅能获得局部视野, 传统路径规划方法依赖全局环境先验信息, 在高度动态场景中难以应对障碍物的不确定性. 并且传统算法在路径规划过程中需要处理大量的数据和复杂的计算, 会消耗巨大的计算资源, 难以满足实时性要求.

引用格式: 孙超, 王江, 穆朝絮, 等. 基于关键路径点的深度强化学习复合路径规划. 中国科学: 信息科学, 2026, 56: 394–408, doi: 10.1360/SSI-2025-0101
Sun C, Wang J, Mu C X, et al. Deep reinforcement learning-based composite path planning with key path points. Sci Sin Inform, 2026, 56: 394–408, doi: 10.1360/SSI-2025-0101

© 2026《中国科学》杂志社

https://www.sciengine.com/doi/10.1360/SSI-2025-0101

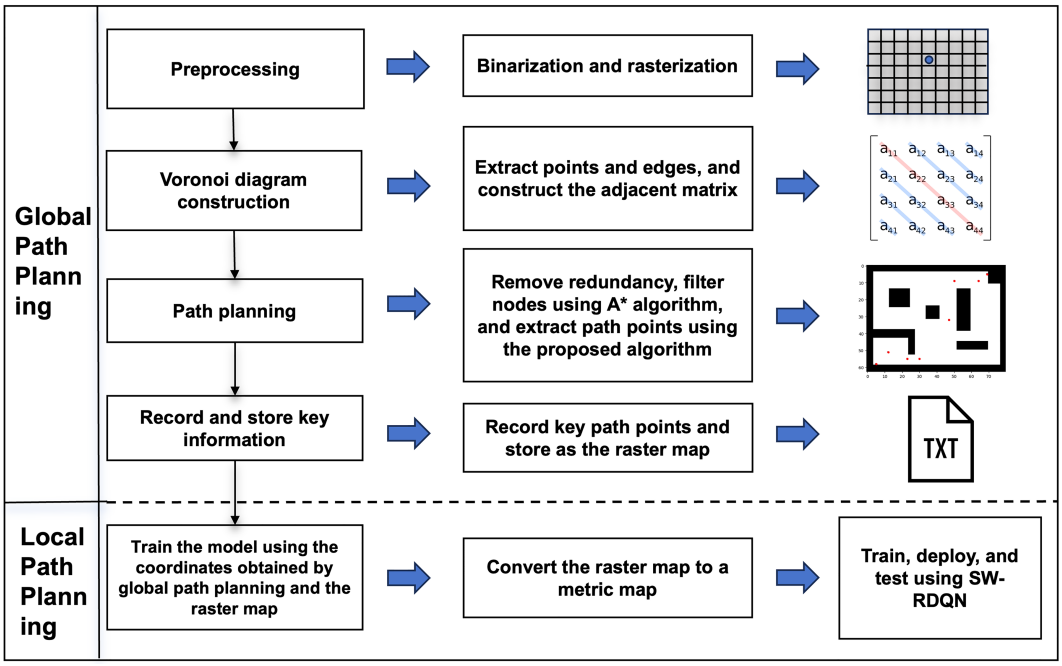


图 1　整体算法流程图.

Figure 1　Overall algorithm flowchart.

局部路径规划聚焦于动态环境中的灵活调整. 近年来兴起的基于强化学习的方法在路径优化和障碍物避让方面取得了显著的进展, 但随着机器人导航环境的逐渐复杂, 这些方法仍然面临着诸多挑战, 如训练效率低、场景泛化性不足等. 文献 [3] 提出在 Voronoi 图规划的全局路径上使用 DWA 进行局部避障, 算法提高了移动机器人运动的稳定性及移动效率; 文献 [4] 采用了基于 DDPG 的强化学习局部路径规划方式, 提高了机器人对于动态障碍物的避障性能; 文献 [5] 采用改进的深度双 Q 网络学习算法进行机器人路径规划, 提高了机器人在环境中的动态避障能力. 虽然上述文章都对路径规划方式进行了一定改进, 但在环境设置上均过于简单, 无法验证算法在复杂环境内的导航性能. 为了提升机器人实际复杂场景中的导航性能, 本文创新性地整合从全局地图筛选关键路径点、基于深度强化学习设计局部规划器两大策略, 相较于已有算法, 在多方面实现显著性能提升.

(1) 全局路径规划的关键路径点筛选创新. 本文提出了一种基于 Voronoi 图的全局路径筛选策略, 摒弃了传统的 “单一路径规划” 的固化框架, 转而聚焦于关键路径点的寻找. 通过在栅格地图的基础上, 结合 Voronoi 图对环境进行划分并提取出关键路径点, 优化了全局路径的安全性和稳定性.

(2) 全局与局部路径规划的结合. 本文将全局关键路径点规划与局部路径调整相结合, 通过深度强化学习对路径规划进行进一步的精细化优化. 在实际应用中, 这种结合方式既显著提高了路径规划的效率, 又能根据动态障碍物的实时变化对路径进行灵活的调整, 确保了路径的安全性和灵活性.

(3) 局部路径规划基于 RainbowDQN [6] 算法, 将前文筛选出的关键路径点作为阶段划分依据, 基于势场以及预测的轨迹落点, 设计出具有独特性的状态函数与奖励函数. 同时, 充分考虑机器人与动态障碍物前后移动的时序关联特性, 对神经网络展开设计与优化, 并完善回放池管理机制. 通过依据 TDerror 与 Reward 的权重比例, 动态调整回放权重, 保证重要场景在训练进程中得以多次地运用.

整体算法流程如图 1 所示. 对实际地图进行拍摄并转为训练环境所需的栅格地图, 在栅格地图的基础上依据 Voronoi 图建立对应点集边集的邻接矩阵, 创建实验环境度量地图, 筛选出关键路径点, 最后采用本文提出的融合阶段关键点与均衡回放策略的 RainbowDQN (stage-key & weight-balanced replay Rainbow deep Q-network, SW-RDQN) 算法进行训练及项目部署.

https://www.sciengine.com/doi/10.1360/SSI-2025-0101

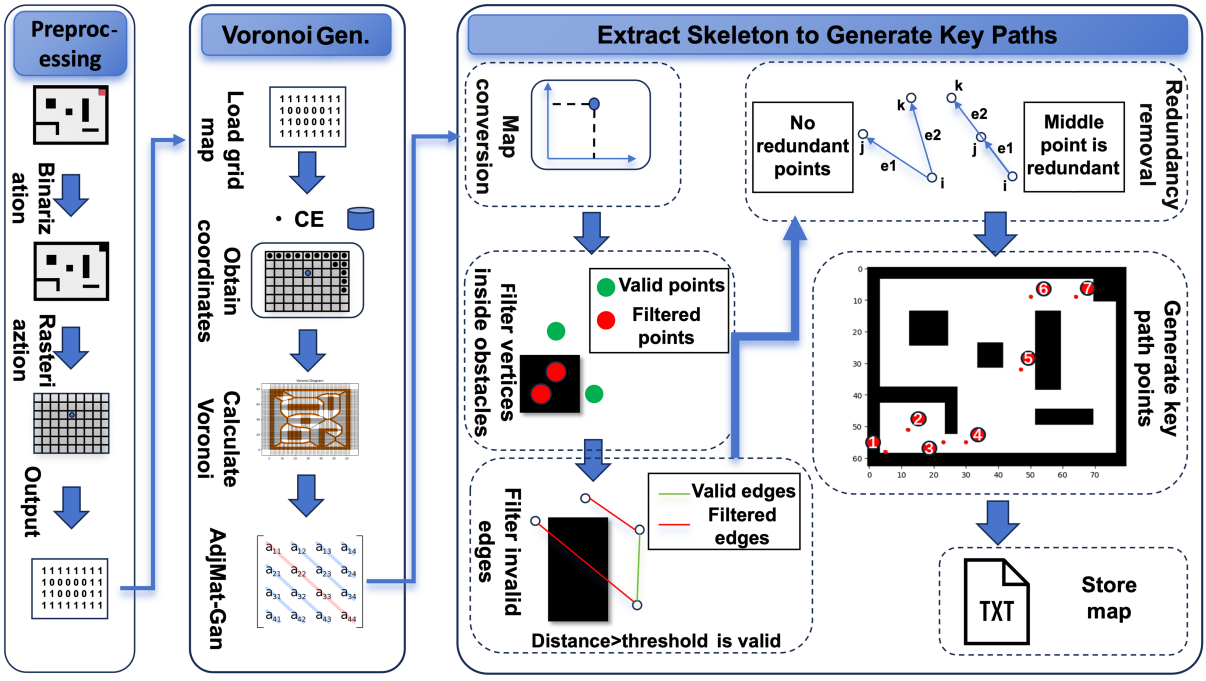


图 2　全局路径规划.

Figure 2　Global path planning.

2　全局路径规划过程

2.1　预处理实际地图并生成 Voronoi 图

2.1.1　由实际地图转为栅格地图

全局路径规划的整体步骤如图 2 所示, 为实现地图的数字化处理, 本文采用以下预处理步骤: 设原始地图为图像矩阵 $\mathbf{M} \in \mathbb{R}^{m\times n}$, 其中 m 和 n 分别表示图像的行数和列数. 采取二值化处理, 定义阈值函数 $T : \mathbb{R} \to \{0,1\}$, 对于给定的阈值 $\tau \in \mathbb{R}$, 有

$$T(x) = \begin{cases} 1, & x \geqslant \tau, \\ 0, & x < \tau, \end{cases} \tag{1}$$

对原始地图 $\mathbf{M}$ 中的每个元素 M_{ij} $(i = 1,2,\ldots,m;\ j = 1,2,\ldots,n)$ 应用阈值函数 T, 得到二值图像矩阵 $\mathbf{B} \in \{0,1\}^{m\times n}$, 其元素定义为 $B_{ij} = T(M_{ij})$. 通过此操作, 将原始地图 $\mathbf{M}$ 转换为二值图像 $\mathbf{B}$. 对二值化后的图像栅格化处理: 设缩减比例系数为 $\alpha \in (0,1)$, 对二值化后的地图 $\mathbf{B}$ 进行等比例的横纵向缩减. 将原始的 $m \times n$ 栅格划分为 $\lfloor \alpha m \rfloor \times \lfloor \alpha n \rfloor$ 个新的栅格区域, 其中 $\lfloor \cdot \rfloor$ 表示向下取整函数. 对于每个新的栅格区域 R_{pq} $(p = 1,2,\ldots,\lfloor \alpha m \rfloor;\ q = 1,2,\ldots,\lfloor \alpha n \rfloor)$, 采用特征保留策略来确定该区域的栅格值. 定义代表性障碍物特征判断函数 $F : \{0,1\}^{k\times l} \to \{0,1\}$ (其中 k 和 l 是新栅格区域对应的原始栅格子块的尺寸), 若 $F(\mathbf{B}_{R_{pq}}) = 1$, 则新栅格地图 $\mathbf{G} \in \{0,1\}^{\lfloor \alpha m \rfloor \times \lfloor \alpha n \rfloor}$ 中对应位置的元素 $G_{pq} = 1$, 否则 $G_{pq} = 0$. 通过这种方式在缩减栅格数量的同时保留具有代表性的障碍物特征, 从而降低后续路径规划的计算复杂度. 经过上述两个预处理步骤, 将实际地图 $\mathbf{M}$ 转化为便于计算分析和处理的栅格地图 $\mathbf{G}$.

2.1.2　使用栅格地图进行 Voronoi 图构建

沃罗诺伊图 (Voronoi diagram) 是一种在平面上根据确定点集将空间划分为若干区域的几何结构, 每个区域内的所有点到该区域内点的距离小于到其他区域内点的距离. 加载预处理完成的栅格地图数据矩阵 $\mathbf{G} \in$

https://www.sciengine.com/doi/10.1360/SSI-2025-0101

$\{0,1\}^{m\times n}$, 其中 m 和 n 分别为栅格地图的行数和列数. 若 $G_{ij}=1$, 则表示该位置 (i,j) 为障碍物. 通过遍历矩阵 $\mathbf{G}$, 提取所有满足 $G_{ij}=1$ 的坐标 (i,j) , 记障碍物坐标集合为 $\mathcal{O}=\{(i,j)|G_{ij}=1,i=1,\ldots,m;j=1,\ldots,n\}$. 对于集合 $\mathcal{O}$ 中的每个障碍物坐标 (x,y), 通过在原始坐标基础上分别增加 0.5 个单位长度, 得到障碍物的中心坐标. 设障碍物中心坐标集合为 $\mathcal{C}$, 则 $\mathcal{C}=\{(x+0.5,y+0.5)|(x,y)\in\mathcal{O}\}$. 受 Fortune [7] 算法的启发, 将 $\mathcal{C}$ 点集作为算法的点事件集构建 Voronoi 图.

为了表征 Voronoi 图中顶点间的连接性, 构建邻接矩阵 $\mathbf{A}\in\mathbb{R}^{t\times t}$, 其中 t 为 Voronoi 图中顶点的数量, 即记障碍物坐标集合 $\mathcal{C}$ 中的数量. 邻接矩阵 $\mathbf{A}$ 的元素 e_{ij} 定义如下:

$$e_{ij}=\begin{cases}\sqrt{(x_{v_i}-x_{v_j})^2+(y_{v_i}-y_{v_j})^2}, & \text{若顶点 } v_i \text{ 和 } v_j \text{ 之间存在边连接,}\\ 0, & \text{若顶点 } v_i \text{ 和 } v_j \text{ 之间不存在边连接.}\end{cases}\tag{2}$$

2.2　对 Voronoi 图进行对应骨架提取

(1) 网格坐标转换. 通过式 (3) 将 Voronoi 图的二维顶点坐标转换为网格坐标. 给定点 (x,y) 和网格的左下角 (m_{xl},m_{yl}) 及分辨率 resolution, 转换为网格坐标 (x_g,y_g):

$$x_g=\left|\frac{x-m_{xl}}{\text{resolution}}\right|,\quad y_g=\left|\frac{y-m_{yl}}{\text{resolution}}\right|.\tag{3}$$

(2) 过滤障碍物顶点. 设 $\mathbf{G}$ 为网格地图, 其中 $G(i,j)$ 表示网格地图中坐标为 (i,j) 的网格值, 当 $G(i,j)=1$ 时表示该网格为障碍物区域. 设 $\mathbf{A}$ 为 Voronoi 图顶点的邻接矩阵, 初始时其元素 A_{ij} 表示顶点 v_i 与 v_j 之间的连接情况. 遍历 Voronoi 图的顶点集合 V, 对于每个顶点 $v=(x,y)\in V$, 通过网格坐标转换函数 (3) 得到其网格坐标 (x_g,y_g). 若 $G(x_g,y_g)=1$, 则删除对应顶点在邻接矩阵中所对应的行列.

(3) 过滤 Voronoi 边. 遍历 Voronoi 图中的边集, 判断其是否穿过障碍物. 通过计算障碍物与边之间的最短垂直距离, 若该距离小于设定的阈值, 则认为该边与障碍物相交, 设置其邻接矩阵中对应的矩阵元素为 0. 若距离满足要求, 则保留邻接矩阵中对应的矩阵元素.

(4) 移除冗余节点. 设 3 个共线顶点为 $v_i=(x_i,y_i)$, $v_j=(x_j,y_j)$ 和 $v_k=(x_k,y_k)$, 计算向量 $\overrightarrow{v_iv_j}=(x_j-x_i,y_j-y_i)$ 和 $\overrightarrow{v_iv_k}=(x_k-x_i,y_k-y_i)$ 的叉积如下:

$$\overrightarrow{v_iv_j}\times\overrightarrow{v_iv_k}=(x_j-x_i)(y_k-y_i)-(x_k-x_i)(y_j-y_i),\tag{4}$$

若叉积结果为零, 如下所示:

$$\overrightarrow{v_iv_j}\times\overrightarrow{v_iv_k}=0,\tag{5}$$

则顶点 v_i 为冗余顶点, 删除对应顶点在邻接矩阵中所对应的行列.

2.3　关键路径点的选取

A* 算法是基于启发式搜索的路径规划方法, 通过综合实际成本和启发式估算寻找最短路径. 对筛选后的 Voronoi 图顶点应用 A* 算法, 生成从起点到终点的初步优化路径. 在此基础上, 本文提出的算法旨在从 A* 路径中剔除冗余节点, 保留避障或影响路径形状的关键节点.

设 A_{path} 为 A* 初步路径点集, map 为地图数据, d_{th} 为安全距离阈值, vor 为 Voronoi 图点集, T_{path} 为关键路径点集. 路径段与障碍物的最小距离通过以下公式计算:

$$d_{\min}=\min\left(\frac{\|\det(S,j_v,p_i)\|}{\|S-j_v\|}\right),\tag{6}$$

其中 $\det(S,j_v,p_i)$ 是由点 S, j_v 和障碍物中心点 p_i 组成的行列式, 表示点到路径的有向面积; $\|S-j_v\|$ 为路径段长度, 该公式计算了障碍物到路径的垂直距离. 若 $d_{\min}>d_{\text{th}}$, 则路径安全, 可以继续检查下一个节点; 否则将当前节点设为关键点. 具体算法流程如算法 1 所示.

该算法具有以下性质.

https://www.sciengine.com/doi/10.1360/SSI-2025-0101

算法 1 从路径中提取关键点.

Require: Apath, map, dth, vor.
Ensure: Tpath.
Tpath ← 空列表;
$i \leftarrow 0$;
while $i <$ len(Apath) $- 1$ **do**
　将 Apath[i] 添加到 Tpath;
　获取当前关键点 S 的坐标;
　$j \leftarrow i + 1$;
　while $j <$ len(Apath) **do**
　　获取下一个节点 jv 的坐标;
　　使用式 (6) 计算路径段和障碍物之间的最小距离;
　　if 最小距离小于 d_{th} **then**
　　将 Apath[$j - 1$] 添加到 Tpath;
　　$i \leftarrow j - 1$;
　　跳出循环;
　　end if
　　更新 $j \leftarrow j + 1$;
　end while
　更新 $i \leftarrow i + 1$;
end while
将 Apath[-1] 添加到 Tpath.

定理1 (安全距离保证性)　算法 1 确保生成的任意连续关键点间路径段与障碍物保持最小安全距离. 给定 A* 路径点集 A_{path}、地图 $\mathcal{M}$ 和距离阈值 $d_{\text{th}} > 0$, 对任意连续关键点 t_i 和 t_{i+1}, 路径段 $\overline{t_i t_{i+1}}$ 满足 $d_{\min}(t_i, t_{i+1}, \mathcal{O}) \geqslant d_{\text{th}}$.

证明　算法通过双重循环遍历路径点, 当检查到路径段 $S \to j_v$ 与障碍物距离小于 d_{th} 时, 将前一节点 $j_v - 1$ 设为新关键点. 由于每次循环仅当路径段 $S \to A_{\text{path}}[j-1]$ 满足 $d_{\min} \geqslant d_{\text{th}}$ 时才会继续, 故任意连续关键点间路径段均满足安全距离要求.

定理2 (算法终止性与复杂度)　算法 1 在有限步内必然终止, 时间复杂度为 $O(|A_{\text{path}}|^2 \cdot |\mathcal{O}|)$, 空间复杂度为 $O(|A_{\text{path}}|)$.

证明　外层循环变量 i 严格递增且有界, 内层循环变量 j 同样严格递增且有界, 因此算法必然终止. 距离计算需遍历所有障碍物, 故总时间复杂度为 $O(|A_{\text{path}}|^2 \cdot |\mathcal{O}|)$.

算法 1 输出的关键路径点集 T_{path} 满足以下特性: (i) $T_{\text{path}} \subseteq A_{\text{path}}$ 且保持原有顺序; (ii) 保持路径连通性: 起点和终点必然包含在 T_{path} 中; (iii) 最小化关键点数量: $|T_{\text{path}}| \leqslant |A_{\text{path}}|$, 通常 $|T_{\text{path}}| \ll |A_{\text{path}}|$; (iv) 保证路径安全性: 所有连续关键点间直线路径不与障碍物碰撞.

3　使用全局关键路径点进行局部路径规划

3.1　算法框架

3.1.1　本文局部路径规划算法流程

如文献 [8] 所述, 基于值函数的算法目标是学习一个最优的值函数 $Q^*(s,a)$, 本文算法通过逼近多步 (n-step) 最优贝尔曼 (Bellman) 方程 (7) 来优化策略.

$$Q^*(s,a) = \mathbb{E}\left[\sum_{i=0}^{n-1} \gamma^i r_{t+i+1} + \gamma^n \max_{a'} Q^*(s_{t+n}, a') \mid s_t = s, a_t = a\right], \tag{7}$$

https://www.sciengine.com/doi/10.1360/SSI-2025-0101

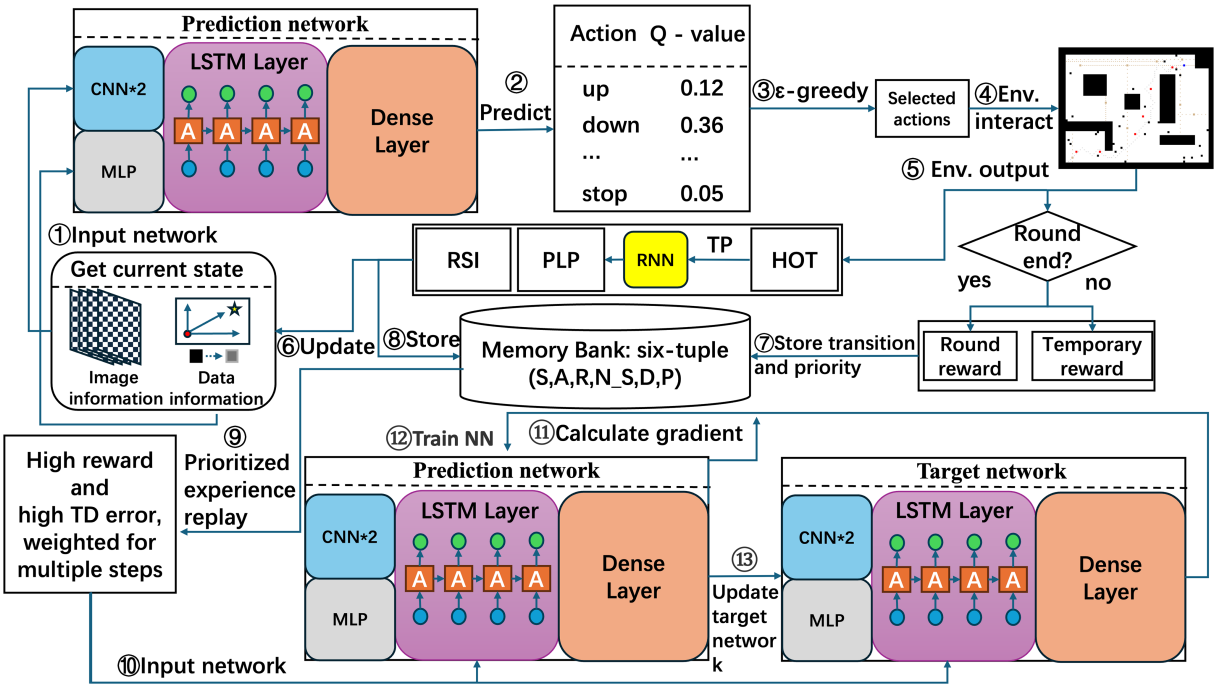


图 3　局部路径规划.

Figure 3　Local path planning.

经过每一回合的采样, 机器人从初始状态 s_0 开始, 根据行为策略生成状态 – 动作序列, 通过不断与环境交互, 机器人收集一个长度为 n 步的轨迹片段 (trajectory segment). 在每一步 t, 算法利用从 t 到 $t+n$ 的采样数据, 通过 n 步回报来估计当前状态 – 动作对的价值, 并按式 (8) 更新 Q 值.

$$Q(s_t,a_t)\leftarrow Q(s_t,a_t)+\alpha\left[\left(\sum_{i=0}^{n-1}\gamma^i r_{t+i+1}\right)+\gamma^n\max_{a'}Q(s_{t+n},a')-Q(s_t,a_t)\right], \tag{8}$$

目标策略根据 Q 值最优化策略, 采取优化行动, 使机器人在环境中能做出更好的决策. 式 (9) 中 k 表示迭代轮次, 在每一时间步, 算法都会利用未来最多 n 步的轨迹数据来计算回报并更新 Q 值; 当所剩时间步数不足 n 步时. 则采用截断方式进行更新.

$$\pi_{(k+1)}(a\mid s)=\begin{cases}1, & \text{if } a=\arg\max_{a'}Q_{k+1}(s,a'),\\ 0, & \text{otherwise.}\end{cases} \tag{9}$$

如图 3 所示, 本文算法是基于在线强化学习 (online RL) 范式, 通过不断让机器人与环境进行交互, 利用收集到的数据同步进行训练, 最后将训练模型完成迁移部署, 主要流程如下.

(1) 本文采用文献 [9] 中提出的局部探测信息方法进行环境采样以获取信息, 将其作为状态空间 s_t. 文献 [10] 使用距离传感器进行采样, 从不同角度收集周围环境信息. 本文对其采样方式进行了改进, 将探测内容拓展为 image information 和 data information 两个维度. Image information 指机器人周边半径 R 的图像信息; data information 包括由 RNN [11] 模块生成的动态障碍物预测位置 (predicted landing points, PLP), 以及当前阶段目标点相对坐标. 将观测到的当前状态输入预测网络, 该预测网络采用多模态特征融合设计, 包含 3 个核心组件: (i) CNN+MLP 模块处理视觉与数值信号; (ii) 长短期记忆网络 (LSTM) 层提取时序特征; (iii) 全连接层输出动作价值估计.

(2) 对预测网络中稠密层输出进行动作 a_t 映射.

https://www.sciengine.com/doi/10.1360/SSI-2025-0101

表 1　环境输出情况.

Table 1　Environmental output.

Condition	Episode status	Action
Robot is in a traversable area	Continues	Assign step reward
Robot reaches a subgoal	Continues	Proceed to next stage, assign intermediate reward
Robot collides with an obstacle	Terminates	Assign final episode reward
Robot reaches the terminal goal	Terminates	Assign final episode reward
Step limit exceeded	Terminates	Assign final episode reward

(3) 将网络输出的若干动作期望通过 ϵ-greedy 策略进行具体行为选取, 如下所示:

$$a_t = \begin{cases} \text{随机选择}, & \text{if } \epsilon > \text{rand}(0,1), \\ \arg\max_a Q(s_t, a), & \text{otherwise}, \end{cases} \tag{10}$$

其中 ϵ 随训练回合数呈指数衰减.

(4) 执行动作 a_t 与环境进行交互.

(5) 环境根据机器人位置输出即时奖励及回合终止信号, 判断依据和输出条件如表 1 所示.

(6) 在每个交互步骤后, 系统会整合环境反馈信息以构建下一时刻的状态 s_{t+1}. 该过程首先采集机器人在新位置下的视觉与数值信号, 将其作为原始状态信息 (raw state information, RSI). 同时, 系统将探测到的障碍物历史轨迹 (history of obstacle trajectories, HOT) 送入一个以 RNN 模块为核心的时序处理模块, 该模块负责对动态障碍物的未来位置进行预估, 生成 PLP. 最终, 这些预测信息、原始状态信息 (RSI) 以及机器人当前的目标点 (target point, TP) 被共同组合成一个完整的状态, 并送入预测网络进行训练与下一轮决策.

(7) 计算当前动作执行获得的奖励, 使用 Reward 与 TDerror 计算出优先级后进行记录.

(8) 将状态、动作、奖励、下一状态、回合结束标识、优先级组合成六元组送入记忆库存储.

(9) 从记忆库中取 n 步经验组合.

(10) 通过经验回放机制将样本输入预测网络进行训练, 预测网络和目标网络的参数分别记为 ω_{now} 和 ω^-_{now}. 网络更新过程如下: 预测网络通过前向传播计算得到 $\hat{q}_j = Q(s_j, a_j; \omega_{\text{now}})$. 根据预测网络输出结果, 选择具有最大期望值的动作 $a^* = \arg\max Q(s_{j+1}, a_j; \omega_{\text{now}})$. 在环境中执行动作 a^*, 获得下一步状态 s_{j+1}, 并将新的状态 – 动作对 s_{j+1} 输入目标网络, 计算 $\hat{q}_{j+1} = Q(s_{j+1}, a^*; \omega^-_{\text{now}})$. 基于 $\hat{q}_j$ 与 $\hat{q}_{j+1}$ 计算 TD 目标 $\tilde{y}_j = r_j + \gamma \cdot \hat{q}_{j+1}$ 和 TD 误差 $\delta_j = \hat{q}_j - \tilde{y}_j$.

(11) 对预测网络做反向传播, 得到梯度 $\nabla_\omega Q(s_j, a_j; w_{\text{now}})$.

(12) 梯度下降更新 $\omega_{\text{new}} \leftarrow \omega_{\text{now}} - lr \cdot \delta_j \cdot \nabla_\omega Q(s_j, a_j; \omega_{\text{now}})$, 其中 lr 为学习率.

(13) 加权更新目标网络中的参数 $\omega^-_{\text{new}} \leftarrow \tau \cdot \omega_{\text{new}} + (1-\tau) \cdot \omega^-_{\text{now}}$, 其中 τ 为需要自身设定的超参数.

3.1.2　状态空间设计

(1) 目标特征 (fet). 参考文献 [12] 中设置的机器人与目标点之间的关系设置状态向量, 其中包含当前机器人到目标点的欧氏距离 $d = \sqrt{(x_g - x_a)^2 + (y_g - y_a)^2}$, 方位角 $\theta = \arctan2\,(y_g - y_a, x_g - x_a)$.

(2) 环境感知 (env). 通过图像编码半径 R 范围内的静态、动态障碍物分布, 动态障碍物预测落点以特殊编码标注.

(3) 时序特征. 堆叠最近 n 帧状态构成状态序列 st, 维度为 (n, st). 其中 $\text{st} = [\text{fet}, \text{env}]$.

3.1.3　动作空间设计

动作空间 A 采用 8 邻域的机器人离散移动方式进行定义, 如下所示:

$$A = \{(\Delta x, \Delta y) \mid \Delta x, \Delta y \in \{-1, 0, 1\} \cup \{-1, 0, 1\}\}. \tag{11}$$

https://www.sciengine.com/doi/10.1360/SSI-2025-0101

表 2　奖励函数参数设置.

Table 2　Reward function parameter settings.

Variable	Value	Variable	Value	Variable	Value	Variable	Value	Variable	Value
r_a	0.6	r_c	-10.0	r_e	-0.2	r_1	-2.0	r_3	-0.2
r_b	-0.4	r_d	-3.0	α	0.5	r_2	-1.5	r_4	-0.01

3.1.4　奖励函数设计

本文参考文献 [13] 在每个时间步中细化奖励, 通过环境交互最终设置奖励函数如下.

(1) **距离激励项** R_{goal}. 构建基于相对距离变化的奖惩函数 (12), 其中 d_t 表示 t 时刻与当前目标点的欧氏距离. 当机器人位姿满足 $\parallel d_t \parallel \leqslant \epsilon$ (ϵ 为位置容差阈值) 时, 系统立即授予阶段性完成奖励 r_c, 并更新目标点序列 $G \in \{g_{k+1}, \ldots, g_n\}$.

$$R_{\text{goal}} = \begin{cases} r_a, & \text{if } d_t < d_{t-1}, \\ r_b, & \text{otherwise.} \end{cases} \tag{12}$$

(2) **方向对齐奖励** R_{align}. 通过点积评估动作方向与目标方向的对齐程度, 其中 $\mathrm{d}x$ 和 $\mathrm{d}y$ 分别表示动作的水平和垂直位移量, (x_a, y_a) 表示动作的起始点, (x_g, y_g) 表示目标点的坐标, α 为奖励缩放因子.

$$r_{\text{align}} = \alpha \cdot \frac{\mathrm{d}x \cdot (x_g - x_a) + \mathrm{d}y \cdot (y_g - y_a)}{\sqrt{\mathrm{d}x^2 + \mathrm{d}y^2} \cdot \sqrt{(x_g + x_a)^2 + (y_g + y_a)^2}}. \tag{13}$$

(3) **避障奖励** R_{avoid}. 机器人与障碍物之间距离小于预警距离或与障碍物发生碰撞时给予惩罚.

$$R_{\text{avoid}} = \begin{cases} r_d, & d_{\text{obstacle}} = 0, \\ r_1, & 0 < d_{\text{obstacle}} \leqslant 1.0, \\ r_2, & 1.0 < d_{\text{obstacle}} \leqslant 2.0, \\ r_3, & 2.0 < d_{\text{obstacle}} \leqslant 3.0, \\ r_4, & \text{otherwise.} \end{cases} \tag{14}$$

(4) **预测落点靠近惩罚** R_{pre}. 机器人在前序动作执行后产生的后续位姿超出安全阈值逼近动态障碍物的预期轨迹点时, 系统将施加与相对距离相关的惩罚函数.

$$R_{\text{pre}} = \begin{cases} -1.25 \times \text{dis}, & \text{if dis} < 2, \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

(5) **步数惩罚** R_{step}. 每步给予小幅负奖励 r_e, 以鼓励更高效的策略.

回合总奖励 R_{round} 定义如式 (16) 所示, 其中奖励函数内定义的部分参数使用信息如表 2 所示.

$$R_{\text{round}} = R_{\text{goal}} + R_{\text{align}} + R_{\text{avoid}} + R_{\text{pre}} + R_{\text{obs}}. \tag{16}$$

3.2　模型结构

网络架构如图 4 所示, 包含 3 个核心模块: 输入层、LSTM 层和密集层. 输入为三维张量, 其维度定义为 (batch_size, time_steps, input_features) 其中 batch_size 表示回放池中单次取出批量大小, time_steps 表示时间序列的长度, input_features 表示单时间步状态的特征维度.

(1) 输入层通过多模态特征融合机制处理环境信息与轨迹预测数据, 具体包含如下.

https://www.sciengine.com/doi/10.1360/SSI-2025-0101

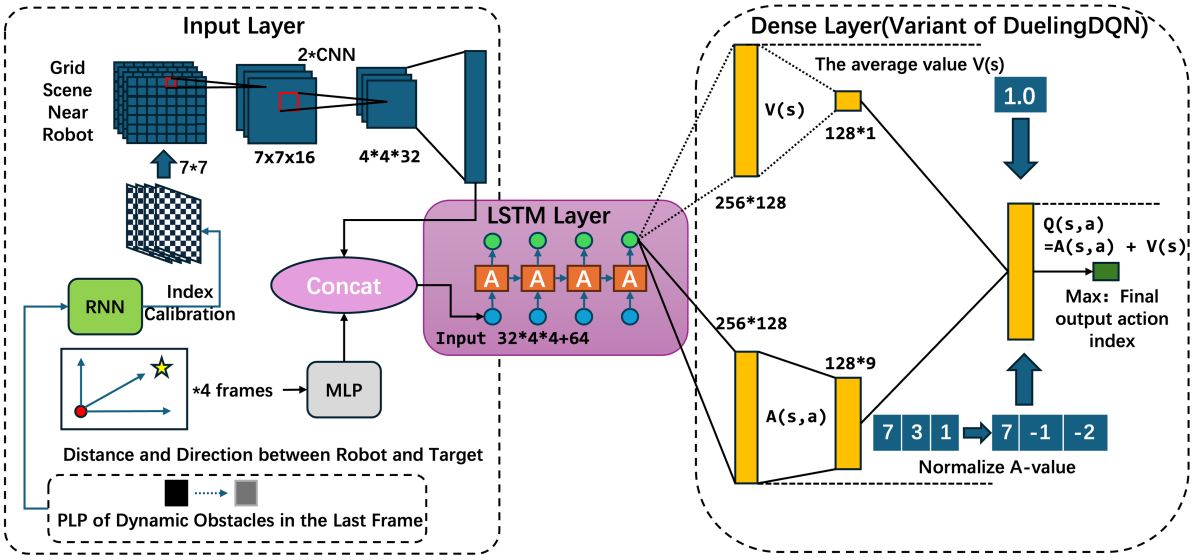


图 4　网络模型结构图.

Figure 4　Network structure.

• 空间关系编码. 机器人与目标点之间的相对距离和航向角经全连接层映射至 64 维特征空间, 形成空间关系特征向量.

• 场景特征提取模块. 以机器人为中心截取半径 R 的局部环境栅格图, 叠加基于预训练 RNN 的动态障碍物轨迹预测模块获取的落点概率分布, 构建输入张量. 通过双层卷积架构进行特征抽象: 首级采用 5×5 卷积核提取障碍物空间分布及动态落点坐标特征, 次级 3×3 卷积层通过步长 2 的下采样操作获得空间表征.

• 特征融合. 通过通道拼接将 CNN 输出特征与空间关系特征整合为 576 维混合特征向量作为下游网络的输入表征.

(2) 长短时记忆层采用 LSTM 层建模时序动态. 文献 [14] 论述通过自我监督预测进行好奇心驱动的探索, 本文将这种自我监督作用于迷宫环境下的机器人路径规划, 通过 LSTM 模块提取环境互动中的一切动态要素, 在 LSTM 模块中, 拼接后的特征向量作为每个时间步的输入, 经过 LSTM 处理后输出最终隐藏状态, 包含了关于机器人运动轨迹及环境状态的时间信息. 网络结构结合了时间序列建模能力, 通过 LSTM 层对输入序列进行建模, 从而能够捕捉长期依赖信息, 适用于部分可观测马尔可夫 (Markov) 决策过程.

(3) 稠密层采用 Dueling DQN 分支结构, 包含价值分支和优势分支: 价值分支估计状态价值 $V(s)$, 表示从当前状态出发的预期回报; 优势分支估计动作相对优势 $A(s,a)$, 表示动作相对于平均水平的优劣. 通过式 (17) 组合分支输出计算动作 Q 值:

$$Q(s,a) = V(s) + \left(A(s,a) - \frac{1}{|A|} \sum_{a'} A(s,a') \right). \tag{17}$$

3.3　SW-RDQN 的实现

(1) 阶段关键点机制. 文献 [15] 指出稀疏奖励会导致机器人学习效率低下 (如迷宫问题中代理易迷失), 因此本文引入阶段关键点 (stage-keys) 将复杂任务分解为多阶段子目标. 假设任务 T 可分解为 $T_1, T_2, \ldots, T_N$, 对应关键点序列 $K = \{k_1, k_2, \ldots, k_N\}$, 则 $T = T_1 \cup T_2 \cup \cdots \cup T_N$. 每个子任务 T_i 的奖励为 $R_i = r(k_i, k_{i+1})$, 通过阶段性反馈解决稀疏奖励问题. 任务分解后, Q 值函数转化为

$$Q_T(s,a) = \sum_{i=1}^{N} \gamma^i r(k_i, k_{i+1}), \tag{18}$$

https://www.sciengine.com/doi/10.1360/SSI-2025-0101

关键点依据状态空间特定区域定义, 将复杂目标细化为子策略, 使机器人聚焦于单阶段目标, 降低 Q 值函数复杂度. 阶段关键点机制通过任务分解将稀疏奖励转化为密集奖励, 使值函数估计方差降低 $O(N)$ 倍. (i) 原始任务方差: 若仅终点给出奖励, Q 值函数方差为

$$\mathrm{Var}[Q(s,a)] = R_{\mathrm{final}}^2 \left(\mathbb{E}[\gamma^{2T}] - \left(\mathbb{E}[\gamma^T] \right)^2 \right),$$

其中 T 为当前状态到终点步数, 方差随 T 指数增长. (ii) 分解后方差: 每个子任务 T_i 奖励延迟 $t_i \leqslant T/N$, 方差满足

$$\sum_{i=1}^{N} \mathrm{Var}[Q_{T_i}(s,a)] \leqslant N R_{\max}^2 \gamma^{2T/N},$$

因 $\gamma \in (0,1)$, 当 $N \geqslant 1$ 时, $\gamma^{2T/N} \gg \gamma^{2T}$, 方差显著降低.

(2) 状态空间改进. 在 SW-RDQN 网络基础上, 提出了一种改进方法, 引入了预先训练好的 RNN 模型用于动态障碍物轨迹预测. 该方法基于动态障碍物的历史位移数据进行训练, 实现了对障碍物未来位置的精确预测, 从而提高了路径规划的准确性和避障性能. 具体实现包括以下两个关键模块. 轨迹预测模块: 输入为动态障碍物的历史位移数据序列, 每个时间步的位移信息表示为二维向量 (x, y), 输出为未来若干时间步的障碍物位置预测结果, 并将其标注在图像信息中. 阶段特征嵌入: 在状态空间中引入阶段信息特征, 并根据机器人行为动态调整关键点位置. 网络能够实时感知任务阶段的进展, 从而优化决策策略, 提升任务完成的准确性和效率.

(3) 回放机制. 本文提出一种改进的经验回放池, 通过优先经验回放和经验优先级动态计算优化强化学习模型训练. 为验证改进有效性, 对比两种采样策略: (i) 传统均匀采样中, 每条经验采样概率相等, 若回放池有 N 条经验, 则采样概率为 $P_{\mathrm{uniform}} = 1/N$, 但无法根据经验重要性调整; (ii) 本文优先级采样按经验优先级加权计算采样概率 $P(i)$, 如式 (19) 所示, 其中 w_{reward} 和 $w_{\mathrm{TD\ error}}$ 分别为奖励和 TD 误差的加权参数. 通过对 TD error 和即时奖励进行偏移处理 (使其值 > 0) 并对数归一化, 保证数据平滑性, 使高 TD 误差和关键经验更易被采样, 加速模型收敛.

$$P(i) = \frac{w_{\mathrm{reward}} \cdot \log(\mathrm{reward}_i + |\min(\mathrm{reward}_i, 0)| + 1) + w_{\mathrm{TD\ error}} \cdot \log(|\mathrm{TD\ error}_i| + 1)}{\sum_{j=1}^{N} \left(w_{\mathrm{reward}} \cdot \log(\mathrm{reward}_j + |\min(\mathrm{reward}_j, 0)| + 1) + w_{\mathrm{TD\ error}} \cdot \log(|\mathrm{TD\ error}_j| + 1) \right)}. \tag{19}$$

式 (19) 定义的 $P(i)$ 满足合法概率分布. 非负性: 奖励项 $\mathrm{reward}_i + |\min(\mathrm{reward}_i, 0)| = \max(\mathrm{reward}_i, 0) \geqslant 0$, 加 1 后对数非负; TD 误差项 $|\mathrm{TD\ error}_i| + 1 \geqslant 1$, 对数非负; 权重 $w_{\mathrm{reward}}, w_{\mathrm{TD\ error}} > 0$, 故 $P(i) \geqslant 0$. 归一性: 设分子单项为 p_i, 则 $\sum_{i=1}^{N} P(i) = \frac{\sum_{i=1}^{N} p_i}{\sum_{j=1}^{N} p_j} = 1$, 满足概率分布归一性.

4　算法测试实验和分析

4.1　全局路径规划算法对比

图 5 展示了不同路径规划算法在避障及路径优化方面的表现. 红色实线为文献 [16] 所提出的具有安全距离的 A* 算法路径, 黄色虚线为文献 [17] 所提出的 RRT 路径, 蓝色虚线为文献 [18] 所提出的 Dijkstra 路径, 在原有基础上加上了安全距离机制, 绿色虚线为文献 [19] 所提出的双向 A* 路径 (Bi-A*), 紫色实线为本文算法 (our algorithm) 求解出的全局路径. 为了验证本文全局规划算法的优越性, 设置如图 6(a) 所示大小为 (80, 68) 像素的场景地图, 额外添加 30 个静态障碍物和 10 个动态障碍物, 将多种全局路径规划算法的拐点作为关键路径点, 使用本文提出的局部路径规划算法进行验证测试, 每类关键路径点引导下导航成功次数记为 K, 平均导航所需时间记为 T. 从数据可以看出, 本文算法在成功次数 (K) 和导航时间 (T) 方面均表现优异. 具体而言, 本文算法的成功次数为 79 次 (每 100 次实验中的成功次数), 较其他对比算法表现出更高的成功率. 例如, Dijkstra 算法的成功次数为 72 次, A* 算法为 73 次, 而本文算法则以较高的成功率证明了其在复杂环境中的鲁棒性. 此外, 本文算法的平均导航时间为 75 s, 相较于 Dijkstra 算法的 85 s, A* 算法的 80 s 等, 显示出在保证高成功率的同时,

https://www.sciengine.com/doi/10.1360/SSI-2025-0101

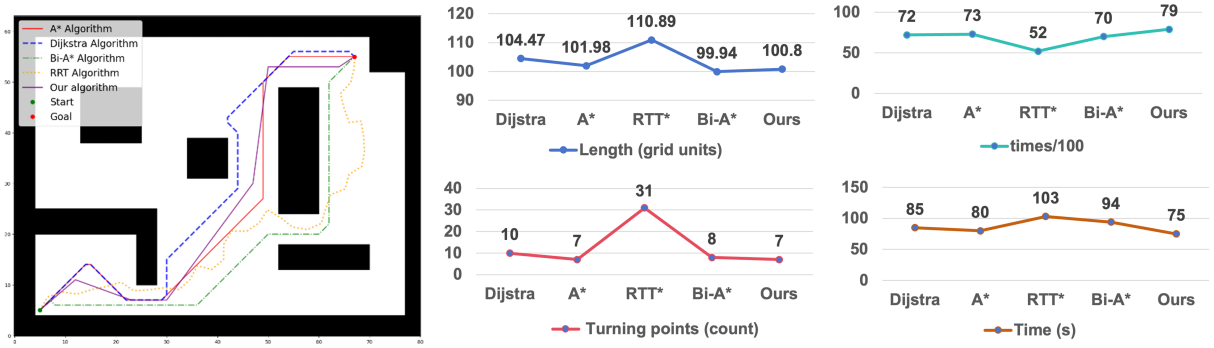


图 5　全局路径规划曲线.

Figure 5　Global path planning curves.

表 3　参数设置.

Table 3　Parameter settings.

Parameter	Value	Description	Parameter	Value	Description
r	0.95	Discount factor	lr	5×10^{-4}	Learning rate
$w_{\text{TD error}}$	0.85	TD error weight	stepn	3	N-step size
maxlen	4	Frame stack size	stepmax	50000	Max steps/episode
w_{reward}	0.15	Reward weight	episode	1000	Training episodes

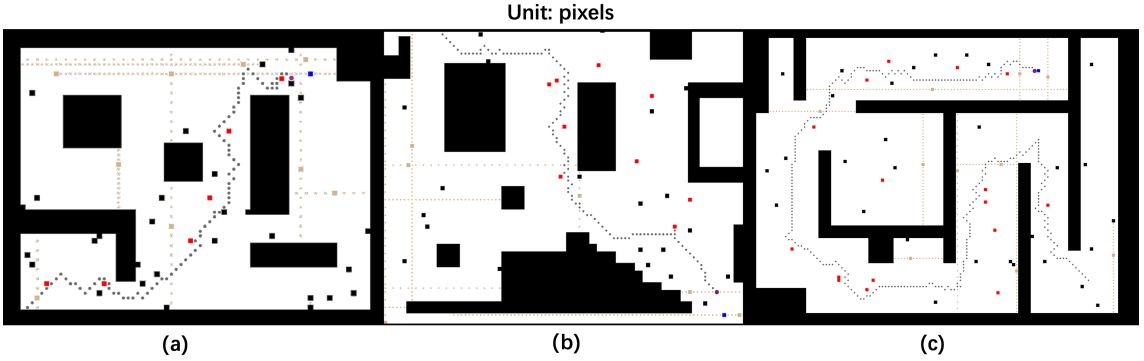


图 6　多种复杂场景下路径规划曲线. (a) 室内场景 (80×68); (b) 开阔场景 (120×100); (c) 迷宫场景: 多拐点 (100×80).

Figure 6　Path planning curves in complex environments. (a) Interior scene (80×68); (b) open scene (120×100); (c) maze scene: multiple inflection points (100×80).

其计算效率也显著高于传统路径规划算法. 本文算法相较于传统的 Dijkstra 算法和 A* 算法, 能够在更短的时间内完成路径规划任务. 尤其在动态障碍物存在的情况下, 本文算法的表现依然优异, 能够在复杂的动态环境中保证较高的规划精度与较低的计算开销, 这对于实时路径规划系统至关重要. 综上所述, 本文提出的全局路径规划算法在复杂环境中不仅能够保证高成功率, 还能够有效降低计算时间, 具备较强的实时性和实用性, 具有较大的应用潜力.

4.2　局部路径规划实验设计

为了评估训练好的机器人模型在环境中的表现, 将使用本文算法 SW-RDQN 进行若干实验. 表 3 为本次实验中使用本文方法设置的部分训练参数.

https://www.sciengine.com/doi/10.1360/SSI-2025-0101

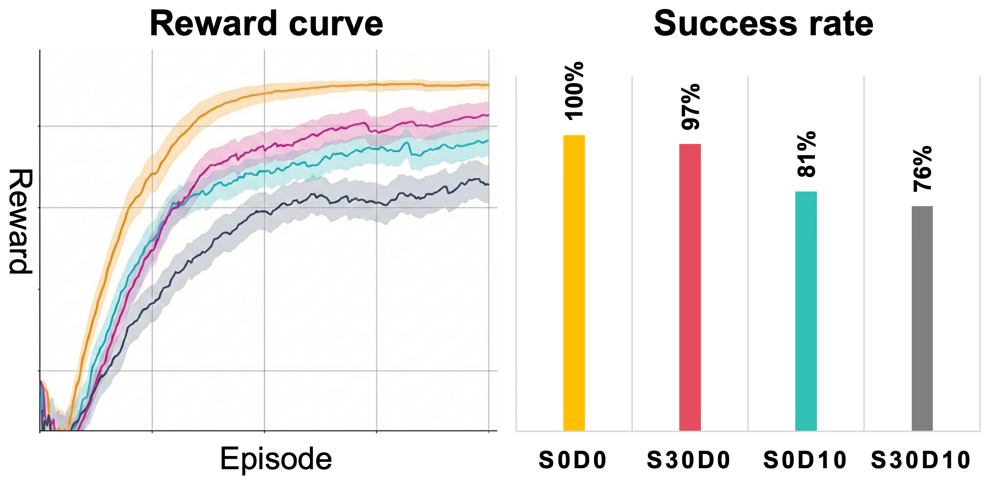


图 7　不同障碍物数量下避障性能.

Figure 7　Obstacle avoidance performance in scenarios with different numbers of obstacles.

4.2.1　交互性测试分析

图 6 展示了机器人在多种复杂场景下进行导航避障的路径规划曲线. 场景 (a) 为长宽比例较小的室内模拟环境, 场景 (b) 为长宽比例较大的开阔模拟环境, 场景 (c) 则为关键路径点规划较多的迷宫环境. 在场景信息方面, 白色区域表示可通行区域, 蓝色区域为机器人目标区域, 黑色区域为障碍物, 红色区域为机器人可实时设置障碍物的区域, 黄色区域为动态障碍物, 紫色圆点为机器人当前位姿. 在路径规划方面, 灰色点集表示机器人行走路径, 黄色点集则为动态障碍物的移动轨迹. 从图中可以看出, 机器人在这些复杂环境下, 能够有效避让各类障碍物并完成避障操作, 最终规划的路径点均达到关键路径点集中的最终目标点, 表明该算法在复杂环境下的避障效果显著.

4.2.2　局部路径算法性能展示

图 7 展示了机器人避障算法在不同障碍物配置条件下的训练性能对比. 图 7 中, 横坐标轴表示训练迭代轮次, 纵坐标轴反映的是累积奖励值. 黄线对应基准场景, 此时环境完全无动态障碍物 ($D = 0$) 且无静态障碍物 ($S = 0$). 红线表示环境仅包含静态障碍物 ($S = 30$)、无动态障碍物 ($D = 0$) 的情况; 青绿色线则对应环境仅包含动态障碍物 ($D = 10$)、无静态障碍物 ($S = 0$) 的场景; 黑线描绘的是环境同时存在静态障碍物 ($S = 30$) 和动态障碍物 ($D = 10$) 的复杂情况. 相较于仅增加静态障碍物的场景, 动态障碍物引入对算法性能的影响更为显著, 主要体现在奖励曲线的峰值降低, 曲线波动增大. 然而, 即使是在大量障碍物的复杂场景中, 该算法依然展现出良好的适应性和稳定性.

4.2.3　局部路径算法对比实验

在对比实验中, 所有算法采用的环境地图障碍情况与图 6(a) 一致, 环境包含固定障碍物、30 个随机静态障碍物和 10 个随机动态障碍物.

参考文献 [20] 中实验对比方法, 采用本文算法与若干传统算法进行比对. 如图 8 所示, 此环境下, 端到端移动模型的成功训练标志是平滑后的奖励曲线应维持在 60 或更高. 如图 8(a) 所示, 深度 Q 网络算法在复杂环境中无法收敛即使经过平滑处理, 其奖励曲线仍呈锯齿状, 不满足端到端模型成功训练所需的奖励曲线要求. 经过一千次测试, 端到端导航成功次数为 5 次. 文献 [21] 提出的 soft actor-critic (SAC) 算法可以作用于离散环境. 是一种强大的强化学习算法, 通过最大熵目标和双 Q 网络的设计, 能够在复杂的连续控制任务中实现高效、稳定的性能表现, 是目前强化学习中广泛应用的算法. 如图 8(b) 所示, 使用 SAC 算法进行训练时, 奖励曲线最终能够收敛到端到端模型成功训练所需的奖励曲线要求. 然而, 该曲线的平滑度和最终奖励均不如本文提出的局部

https://www.sciengine.com/doi/10.1360/SSI-2025-0101

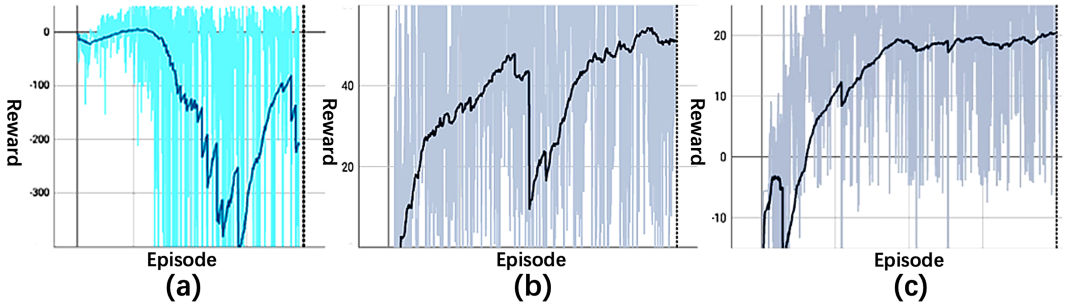


图 8　不同强化学习算法奖励曲线. (a) DQN; (b) SAC; (c) TD3.

Figure 8　Reward curves of different reinforcement learning algorithms. (a) DQN; (b) SAC; (c) TD3.

表 4　不同导航算法对比.

Table 4　Comparison of different navigation algorithms.

Algorithm	Success rate (%)	Average time (ms)	Average path length (grid units)
Ours	76.1	291.0	349.15
IPSO	80.1	851.3	663.2
IGA	74.2	683.1	398.15

路径规划算法. 此外, 经过一千次模拟, 端到端成功率为 480/1000. 将动作空间转为 8 邻域连续空间, 对环境采用 TD3 算法进行训练, 算法具体内容如文献 [22] 所述, 由于 TD3 算法对参数和环境的强要求, 并不能很好地完成训练要求. 如图 8(c) 所示, TD3 算法训练所能达到的最大值仅在 30 左右, 平滑后的曲线维持在 20 上下, 不满足此环境下端到端模型成功训练的奖励曲线要求.

4.2.4　算法对比实验

实验在标准室内环境 (80×60 栅格) 中进行, 障碍物密度为动态 30 个、静态 10 个, 对比结果如表 4 所示. 改进遗传算法 (IGA)[23] 是基于自然选择原理, 通过势场引导的个体生成、自适应适应度函数和智能变异操作, 在复杂动态环境中搜索全局最优路径. 改进粒子群优化 (IPSO)[24] 是通过模拟鸟群觅食行为, 利用粒子的速度和位置更新机制, 结合动态障碍物斥力和时间同步碰撞检测实现实时高效的路径规划, 同时基于历史数据实现动态障碍物轨迹预测, 并引入置信度评分机制以加强动态障碍物避障能力. 根据表 4 的数据, 在成功率方面, IPSO 以 80.1% 的成绩位居第一, 显示出最高的路径寻找能力. 本文算法 (Ours) 以 76.1% 的成功率紧随其后, 而 IGA 的成功率为 74.2%. 然而, 在效率指标上则呈现出完全不同的情况. 本文算法在执行时间 (291.0 ms) 和路径长度 (349.15 grid units) 上均是最优的. 相比之下, 群体智能算法的计算成本极高. IPSO 虽然成功率最高, 但其执行时间长达 851.3 ms, 路径长度也为最长, 为 663.2 grid units. IGA 的效率稍好于 IPSO, 但其时间和路径长度 (683.1 ms 和 398.15 grid units) 也远不及本文算法. 综上所述, IPSO 算法虽然最可靠, 但其巨大的时间开销和极低的路径效率使其难以应用于实时场景. IGA 表现居中. 而本文算法 (Ours) 通过全局路径点筛选与深度强化学习的结合, 在高成功率、最快执行时间和最短路径长度之间实现了最佳的综合性能与平衡, 验证了该复合规划框架在复杂动态环境中的高效性与实用性.

4.2.5　关键架构消融实验

本文提出的综合性方法显著提高了复杂环境中的避障性能. 通过表 5 所示, 即便移除部分核心机制, 系统仍能履行基本功能, 但成功率明显下降, 但完全简化为 DQN 后, 系统陷入故障.

- 落点预测模块 (No_predict). 移除该模块后, 模型的训练时间缩短, 但成功率明显下降, 均表明其对模型性能的精确度有极大影响.

https://www.sciengine.com/doi/10.1360/SSI-2025-0101

表 5　模块消融实验.

Table 5　Module ablation experiment.

	ALL	No_predict	No_TDR	No_Due	S_scene	One & S	No_all
Training time	1 h 13 min	55.94 min	53 min	53 min	50.21 min	55.31 min	1 h 54 min
Effectiveness	✓	✓	✓	✓	✓	✓	×
Success rate (%)	86.1	65	59	58	57	56	0

• TDR 权重 (No_TDR). 移除后, 训练时间大幅缩短至分钟级别, 但成功率下降至 59%. 这表明 TDR 权重对提升模型性能至关重要, 其移除显著降低了模型的成功率.

• Dueling DQN (No_Due). 移除后, 训练时间缩短至 51.25 min, 成功率进一步下降至 58%.

• 去除前后记忆输入 (S_scene). 移除后, 模型的训练时间为 50.21 min, 成功率为 57%, 与不去除记忆输入时相比, 成功率有所下降.

• 采用单步回放且去除前后记忆 (One & S). 训练时间增加至 55.31 min, 成功率降至 56%. 表明单步回放策略在此实验设置下的效果不如多步回放, 限制了模型的探索能力.

• 去除所有模块 (No_all). 移除所有模块后, 训练时间显著增加至 1 h 54 min, 但成功率降至 0%. 这表明各个模块的组合对模型的成功率至关重要, 去除所有关键模块会导致模型性能崩溃.

5　总结

本研究提出了一种融合 Voronoi 图关键路径点分割与深度强化学习的机器人路径规划系统, 旨在实现复杂多变环境下的高效全局路径规划与局部避障能力. 通过一系列实验验证, 本文所提出的算法在复杂环境适应性、路径生成效率及避障策略等方面均显著优于现有深度强化学习路径规划与导航方法. 然而, 本文算法仍存在一定的优化空间和改进潜力. 首先, 从实际机器人部署需求出发, 当前算法的动作空间设计尚未全面考虑机器人在连续空间中的运动灵活性和动力学特性, 这限制了算法在运动自然性和路径平滑性方面的表现. 其次, 本研究主要针对单机器人的动态避障问题进行了实验验证, 而现实环境中的避障场景通常涉及多机器人的交互与协作. 因此, 未来研究将致力于扩展当前算法, 以提升机器人在双机器人或多机器人环境中的避障能力, 并进一步增强其动态环境适应性.

参考文献

1　Qiu H. Multi-agent navigation based on deep reinforcement learning and traditional pathfinding algorithm. 2020. ArXiv:2012.09134

2　Yu Z, Si Z, Li X, et al. A novel hybrid particle swarm optimization algorithm for path planning of UAVs. IEEE Int Things J, 2022, 9: 22547–22558

3　Jiang L, Li J, Ma X Z, et al. Voronoi path planning based on improved skeleton extraction. J Mech Eng, 2020, 56: 138–148 [蒋林, 李峻, 马先重, 等. 一种改进骨架提取的 Voronoi 路径规划. 机械工程学报, 2020, 56: 138–148]

4　Wu X, Yang J J, Tang K, et al. Hierarchical path planning for mobile robots based on hybrid maps. China Mech Eng, 2023, 34: 563–575 [武星, 杨俊杰, 汤凯, 等. 面向复合地图的移动机器人分层路径规划. 中国机械工程, 2023, 34: 563–575]

5　Zhang L, Mu Y S, Pan Q. Mobile robot path planning algorithm with improved deep double Q networks. Inform Control, 2024, 53: 365–376 [张磊, 母亚双, 潘泉. 基于改进深度双 Q 网络的移动机器人路径规划算法. 信息与控制, 2024, 53: 365–376]

6　Hessel M, Modayil J, van Hasselt H, et al. Rainbow: combining improvements in deep reinforcement learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2018

7　Fortune S. A sweepline algorithm for Voronoi diagrams. In: Proceedings of the 2nd Annual Symposium on Computational Geometry (SCG'86), New York, 1986. 13–322

8　Yuan L, Zhang Z Q, Li L H, et al. Progress on cooperative multi-agent reinforcement learning in open environment. Sci Sin Inform, 2025, 55: 217–268 [袁雷, 张子谦, 李立和, 等. 开放环境下的协作多智能体强化学习进展. 中国科学: 信息科学, 2025, 55: 217–268]

9　Zhang C G, Xi Y G. Robot rolling path planning based on local detection information. J Autom, 2003, 29: 38–44 [张纯刚, 席裕庚. 基于局部探测信息的机器人滚动路径规划. 自动化学报, 2003, 29: 38–44]

https://www.sciengine.com/doi/10.1360/SSI-2025-0101

10 Xin J, Zhao H, Liu D, et al. Application of deep reinforcement learning in mobile robot path planning. In: Proceedings of Chinese Automation Congress (CAC), Jinan, 2018
11 Lipton Z C, Berkowitz J, Elkan C. A critical review of recurrent neural networks for sequence learning. 2015. ArXiv:1506.00019
12 Sartoretti G, Kerr J, Shi Y, et al. PRIMAL: pathfinding via reinforcement and imitation multi-agent learning. IEEE Robot Autom Lett, 2019, 4: 2378–2385
13 He Z, Dong L, Sun C, et al. Asynchronous multithreading reinforcement-learning-based path planning and tracking for unmanned underwater vehicle. IEEE Trans Syst Man Cybern Syst, 2022, 52: 2757–2769
14 Pathak D, Agrawal P, Efros A A, et al. Curiosity-driven exploration by self-supervised prediction. In: Proceedings of the 34th International Conference on Machine Learning, 2017. 2778–2787
15 Ladosz P, Weng L L, Kim M, et al. Exploration in deep reinforcement learning: a survey. Inf Fusion, 2022, 85: 1–22
16 Duan S Y, Wang Q F, Han X, et al. Improved A-star algorithm for safety insured optimal path with smoothed corner turns. J Mech Eng, 2020, 56: 205–215 [段书用, 王启帆, 韩旭, 等. 具有确保安全距离的 A* 路径优化方法. 机械工程学报, 2020, 56: 205–215]
17 LaValle S M, Kuffner J J. Randomized kinodynamic planning. In: Proceedings of IEEE International Conference on Robotics and Automation, Detroit, 1999. 473–479
18 Dijkstra E W. A note on two problems in connexion with graphs. Numer Math, 1959, 1: 269–271
19 Ikeda T, Hsu M Y, Imai H, et al. A fast algorithm for finding better routes by AI search techniques. In: Proceedings of Vehicle Navigation and Information Systems Conference, 1994. 291–296
20 Pathak S, Shrestha B, Pahi K. Enhanced deep Q-learning for 2D self-driving cars: implementation and evaluation on a custom track environment. 2024. ArXiv:2402.08780
21 Christodoulou P. Soft actor-critic for discrete action settings. 2019. ArXiv:1910.07207
22 Fujimoto S, Hoof H V, Meger D. Addressing function approximation error in actor-critic methods. 2018. ArXiv:1802.09477
23 Receveur J B, Victor S, Melchior P. Autonomous car decision making and trajectory tracking based on genetic algorithms and fractional potential fields. Intel Serv Robotics, 2020, 13: 315–330
24 Zhang Y, Wang S, Ji G. A comprehensive survey on particle swarm optimization algorithm and its applications. Math Problems Eng, 2015, 2015: 1–38

Deep reinforcement learning-based composite path planning with key path points

Chao SUN[1,2], Jiang WANG[1], Chaoxu MU[1,2,3*], Liangping XING[1] & Changyin SUN[1,2*]

1. *School of Artificial Intelligence, Anhui University, Hefei 230601, China*
2. *Autonomous Unmanned System Technology Ministry of Education Engineering Research Center, Anhui University, Hefei 230601, China*
3. *School of Electrical Automation and Information Engineering, Tianjin University, Tianjin 300192, China*
* Corresponding author. E-mail: cxmu@tju.edu.cn, cysun@ahu.edu.cn

Abstract This article proposes a robot path planning method that combines Voronoi diagrams and deep reinforcement learning. In terms of global planning, a "safe zone" is constructed through Voronoi diagrams, and path points are optimized using the A* algorithm and a critical path point extraction algorithm to provide objectives for local planning. During the local planning period, we propose the SW-RDQN (stage-key & weight-balanced replay Rainbow deep Q-network) algorithm, which improves navigation adaptability in complex dynamic environments through three key designs. First, a multi-stage path segmentation strategy is employed to partition and prune the global path, extracting and preserving critical key-point information. Second, a reward function is designed by integrating predicted trajectory landing points with an environmental potential field, guiding the agent to learn safer and more effective obstacle-avoidance behaviors. Third, we adopt a prioritized experience replay scheme with dynamically weighted sampling based on rewards and TD errors. By applying offsetting and logarithmic normalization, the scheme increases the sampling probability of critical experiences, thereby improving sample efficiency. The state input integrates a convolutional layer and a multi-layer perceptron to extract information, takes several scene frames, sends them to a long short-term memory model, and finally makes decisions using a dense layer, thereby improving the robot's perception and response capabilities in dynamic environments. This method effectively improves the path planning accuracy and robustness of robots in dynamic environments.

Keywords multi-stage path segmentation, Voronoi diagram, SW-RDQN, landing point prediction, scene-based obstacle avoidance navigation

https://www.sciengine.com/doi/10.1360/SSI-2025-0101