中国科学院大学

University of Chinese Academy of Sciences

基于深度强化学习的 EP 机器人导航与对抗

院系名称： 人工智能学院

姓　　名： 李　超

刘晓东

桑　明

培养单位： 自动化研究所

深圳先进技术研究院

二〇二二年六月

摘 要

本文的研究内容是 EP 机器人的自主导航规划与对抗。

在第一阶段的导航规划中，我们要完成的任务是按照固定顺序找到五个目标点。该问题的本质是避障和寻径，是一个路径规划问题，通过构图和 A^* 算法在理论上可以得到最优路径，但是由于在仿真阶段传感器对环境的感知存在噪声，导致机器人无法正确按照路径移动，为了解决该问题，本文采用强化学习方法对该问题进行建模，由浅入深，由简及难地进行了了如下尝试：首先尝试了利用基于离散动作集的 DQN 算法进行机器人自主导航规划。在得到可行的结果之后，利用基于 Actor-Critic 架构的 DPG 算法对机器人进行连续动作的控制，进一步地我们利用了训练过程中的数据和 GMM 模型对固定障碍物的分布进行了估计，并且用来修正机器人与目标点的距离。此外，我们设计了一些后处理操作进一步解决我方机器人在导航过程中陷入局部极值点的问题。

在第二阶段的任务中，我们要进行我方机器人与敌方机器人的对抗。其中，敌方机器人的朝向始终指向我方，并且，当我方机器人出现在敌方机器人的射程内时会对我方机器人进行射击（攻击机制），敌方机器人在运动时会产生随机动作以尽可能的防止我方机器人瞄准（防守机制）。因此，本文将从攻击机制和防守机制的设计两个方面完成敌我双方机器人对抗的策略：在攻击策略中，我们基于阶段一中的导航算法与敌方机器人保持安全距离，并训练了 DQN 网络来决策射击时机；在防守策略中，我方机器人的运动是以敌方机器人为圆心，敌我双方机器人地距离为半径的弧形运动，以尽可能躲避敌方的进攻。

我们实现了上述算法，最终效果是基本可以稳定找到五个目标点，并在对抗阶段给予敌方机器人一定的伤害。

关键词：深度强化学习，EP 机器人，导航，对抗

目 录

图形列表

表格列表

第 1 章　研究内容

在仿真游戏任务中，越来越多的 AI 算法超越了人类玩家。然而在实际环境中，AI 算法的应用受到很多限制。本文通过仿真考察动态环境中机器人定位、导航、对抗能力。本文基于的问题背景是，在 $8.08 \times 4.48(m)$ 的矩形区域内，存在固定障碍物和动态障碍物。固定障碍物每轮游戏固定不变，如图1.2中白色矩形所示，动态随机障碍物（$30cm \times 30cm \times 40cm$）为 5 个位置随机的目标块和敌方机器人，分别如图中星状标志和红色矩形所示。机器人在 3 分钟内需要完成的两个任务：

1. 依次激活五个目标点；
2. 与防守机器人进行射击对抗。

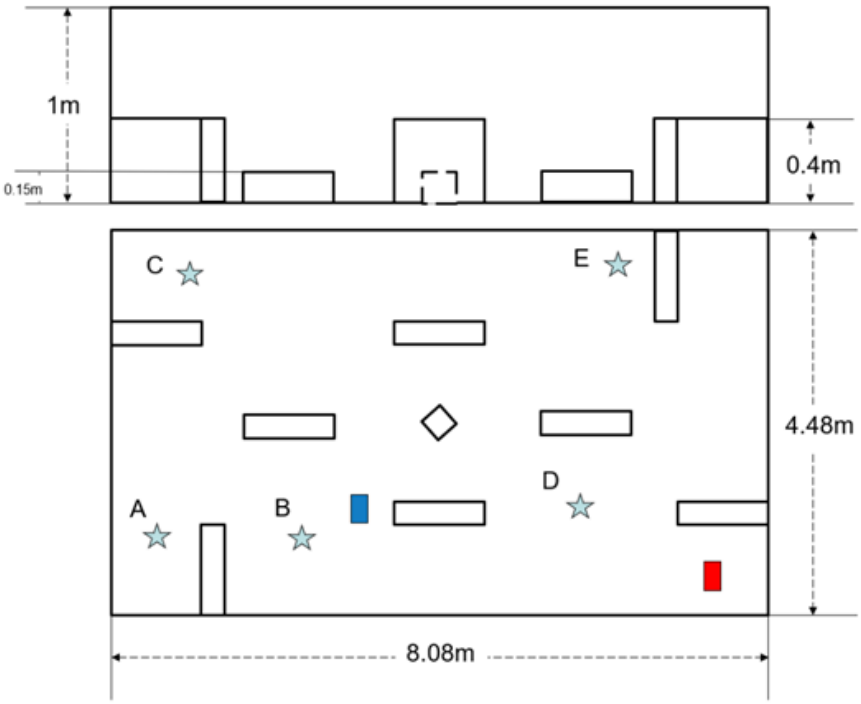


图 1.1 场地示意图

机器人在完成任务同时，需躲避固定障碍物和动态障碍物。过程中机器人需要按照顺序依次激活目标块，如图1.2所示，目标快被激活需要满足下述所有条件：

1. 机器人与目标块的距离小于 $1m$

2. 机器人与目标块之间没有障碍物

3. 机器人在世界坐标系下的朝向与机器人与目标块的连线所构成的夹角小于 30 度

4. 目标块是按照 $ABCDE$ 的顺序激活 (例如抵达目标块 C 之前，依次抵达过 AB，否则目标块无法激活)

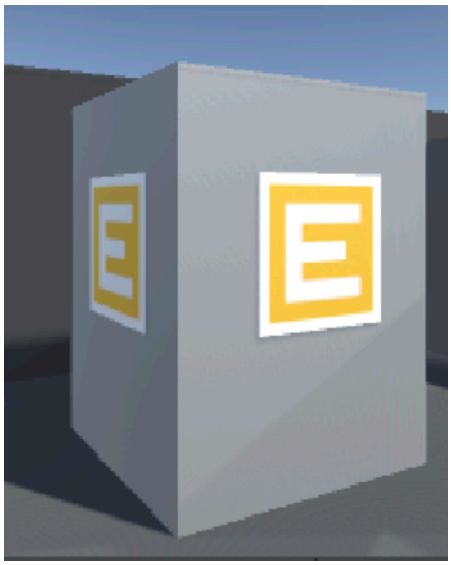


图 1.2 目标样式示意图

只有在五个目标点的按顺序激活后，防守机器人才会被激活并开始向我方机器人射击。当敌方机器人处于静默状态的时候，敌方机器人不会自主移动或向我方机器人射击。对抗开始时，防守机器人和我方机器人的初始血量为 800，双方机器人的射击频率为 $1Hz$，被击中一次掉血为 100，每个机器人可射击次数为 24 次，当下发射击指令多于 24 次时，射击指令将不会被执行。机器人的得分 $Score = 60 \times N + 0.5 \times (D + H) - T - 10K$，考虑以下四个部分:

1. 寻找目标的得分: $60 \times N$，按顺序激活一个目标获得奖励 60，N 为成功激活目标的个数, 最高得分为 300

2. 防守机器人对抗的得分:$0.5 \times (D + H)$，敌我双方初始血量为 800，比赛结束时防守机器人的伤害为 D=800-防守机器人血量剩余，我方机器人的血量剩余为 H

3. 比赛总用时:T(s)，完成任务用时越长，得分越低，最多扣减为 180

4. 碰撞惩罚: $K = 2 \times T_k$，T_k 为连续碰撞时间 (单位为秒)，碰撞时间越长，得分越低，最多扣减 3600

本报告采用的是信息完备的导航和对抗，即得到的已知信息包括我方机器人的机器人车载相机拍摄图像，激光雷达扫描数据，参赛机器人在地图中的位置，血量和剩余子弹数量，目标块的位置和激活状态，敌方机器人的激活状态，位置，血量和剩余子弹数量，以及碰撞相关信息。

第 2 章　机器人自主导航规划

在该阶段的任务中，我们要完成的任务是按照固定顺序找到五个目标点。该问题的本质是避障和寻径，是一个路径规划问题，通过构图和 A^* 算法在理论上可以得到最优路径，但是由于在仿真阶段传感器对环境的感知存在噪声，导致机器人无法正确按照路径移动，为了解决该问题，本文采用强化学习方法对该问题进行建模，由浅入深，由简及难地进行了了如下尝试：首先尝试了利用基于离散动作集的 DQN 算法进行机器人自主导航规划。在得到可行的结果之后，利用基于 Actor-Critic 架构的 DPG 算法对机器人进行连续动作的控制，进一步地我们利用了训练过程中的数据对固定障碍物的分布进行了估计，并且用来修正机器人与目标点的距离。

2.1　基于 DQN 的机器人导航规划

2.1.1　问题建模

在该阶段中，我们将问题描述为一个连续状态空间，离散动作空间的强化学习问题。

系统状态

该问题是一个连续状态控制的问题，官方给出的环境的连续状态空间为

$$x \in (0, 8.48m);\ y \in (0, 4.48m);\ \alpha \in [-\pi/2, 3\pi/2)$$

考虑到仅以小车的位置和朝向为系统状态是任务无关的，该阶段的问题在于对目标物的寻找，因此本文考虑将目标物的位置同样作为系统状态，我们定义的状态向量为：

$$state = (self_x, self_y, goal_x, goal_y, dist, \theta)$$

其中 $(self_x, self_y)$ 表示我方机器人的坐标；$(goal_x, goal_y)$ 代表当前目标的坐标；$dist$ 代表我方机器人与目标之间的欧氏距离；θ 代表我方机器人的朝向与目标方向的夹角，设目标方向与 x 方向的夹角为 β，则有：

$$tan\beta = \frac{goal_y - self_y}{goal_x - self_x} \tag{2.1}$$

我方机器人的朝向为 $\alpha \in [-\frac{\pi}{2}, \frac{3\pi}{2})$, 故有我方机器人的朝向与目标方向的夹角 θ 的计算方式如下:

$$\theta = \begin{cases} \beta - \alpha & if\, goal_x - self_x > 0 \\ \beta - \alpha + \pi & if\, goal_x - self_x < 0 \end{cases}$$

系统动作

机器人输入的控制量是一个四维向量，包括左右移动，前后移动，方向旋转，是否射击。控制向量前三个分量的输入范围是 [-1,1]，正方向分别是向左移动，向前移动，逆时针旋转；最后一个分量是一个 bool 量，0 代表不射击，1 代表射击。为了简化问题，本文首先以离散动作控制的方式解决该问题，不失一般性，采用前后左右四个方向的移动，并令机器人的朝向指向目标，并直接将该角度归一化后作为角速度的输入量，所以离散的动作集定义为:

$$A = \left\{ \left(1, 0, \frac{\theta}{\pi}, 0\right), \left(-1, 0, \frac{\theta}{\pi}, 0\right), \left(0, 1, \frac{\theta}{\pi}, 0\right), \left(0, -1, \frac{\theta}{\pi}, 0\right) \right\}.$$

奖励函数

环境的奖励函数函数为当我方机器人找到一个目标则给予奖励，而其他状态都不给予奖励，该稀疏奖励不利于策略的学习，因此本文设计直觉性的奖励，由于机器人的任务是尽可能快的找到目标，因此本文设定距离惩罚和行动惩罚，其中行动惩罚为固定数值-10，距离惩罚为机器人到目标的距离，该惩罚随着机器人与目标的靠近逐渐降低。考虑到机器人在寻找目标的同时需要积极避障，因此当机器人产生碰撞时给予碰撞惩罚。

奖励函数的计算公式为:

$$R = Reward - 10 - dist(robot, goal) - 70 \times count\_coll - 30 \times time\_coll \quad (2.2)$$

其中, $Reward\_goal$ 指每找到一个目标点便给定 100 的奖励; $dist(robot, goal)$ 指我方机器人与目标点的欧式距离; $count\_coll$ 表示碰撞次数; $time\_coll$ 表示碰撞持续的时间。

折扣因子

折扣因子 γ 设定为 0.98，选择较大的折扣因子是为了提高目标点附近奖励在初始时刻状态价值的重要性。

2.1.2　DQN 的原理

Deep Q Network(DQN) 是利用神经网络进行价值函数的估计。

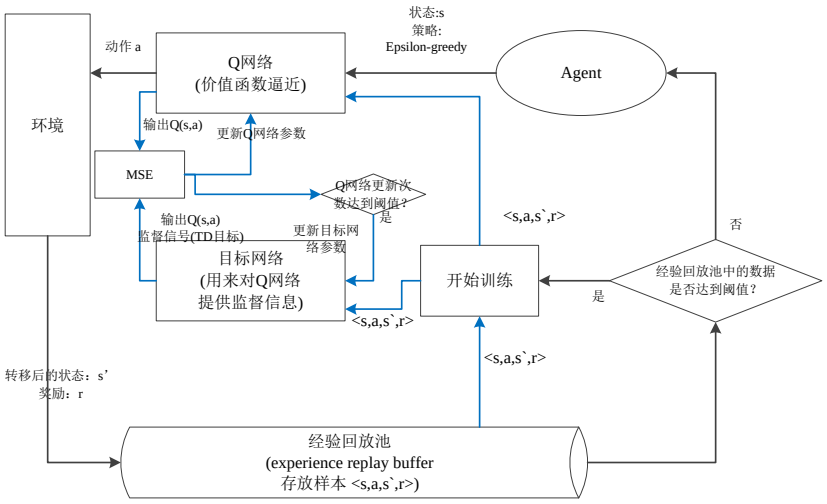


图 2.1 DQN 的算法流程图

在图 2.1中，黑色的流程代表训练样本的采集的过程，也是探索的过程。只有经验回放池中的样本数量达到一定阈值才会开始训练。在该问题的探索过程中，探索策略为随机动作探索。

蓝色的流程代表训练的过程，其中：Q 网络是状态-动作价值 Q 函数的逼近器，输出层是三个神经元，分别代表不同动作 a 的 $Q(s,a)$，然后通过经验回放池的样本 $< s, a, r', s' >$ 来选择对应的 $Q(s,a)$。目标网络与 Q 网络的网络结构如表 2.1所示。综上，Q 网络与目标网络的网络结构如表 2.1. 目标网络的输出是下一个状态 s' 的三个 $Q(s',a')$。DQN 是以 TD 误差监督信号的，所以需要计算 TD 目标，公式如下:

$$\delta = R_s^a + \max_{a'} Q_{target}(s',a') - Q(s,a) \tag{2.3}$$

Loss 是对这一个 batch 的样本求 MSE，公式如下:

$$Loss = \frac{1}{N}\sum\left[R_s^a + \gamma \max_{a'} Q_{target}(s',a') - Q(s,a)\right] \tag{2.4}$$

将 Loss 产生的梯度反传给 Q 网络的参数进行更新，当 Q 网络的更新次数达到一定阈值，再将目标网络与 Q 网络的参数同步，在下一次同步前，目标网络的参数不发生变化。这样延迟更新的目的是为了让训练的过程更加稳定。

2.1.3 算法设计细节

表 2.1 Q 网络与目标网络的结构

Q 网络	Target 网络
输入层 6 维	输入层 6 维
全连接层 128 维	全连接层 128 维
ReLU	ReLU
全连接层 256 维	全连接层 256 维
ReLU	ReLU
输出层 4 维	输出层 4 维
tanh	tanh

训练方式：在训练过程中，我们将一局 (Episode) 设定为依次找到五个目标点。并将每一步的样本放入经验回放池中，进行在线式 (Online) 的训练。

2.2 基于 DPG 的机器人导航规划

在离散动作集上取得初步效果之后，我们尝试了适用于连续动作空间的基于确定策略梯度 (Determinate Policy Gradient, DPG) 的机器人路径规划的方法。

2.2.1 DPG 的基本原理

DPG 是策略梯度方法的一种。策略梯度的方法是用一个参数化的概率分布来表示策略，作为策略逼近器，并且由于策略是一个概率分布，动作是根据策略随机选取的，所以可以适用于连续动作空间。与一般的策略梯度不同，DPG 的策略是确定性的，即 $\pi_\xi(s)$，并采用 Actor-Critic 架构。在这里，我们用神经网络对动作状态价值函数 $Q_w(s,a)$ 和策略 $\pi_\xi(s)$ 进行逼近，并采用时间差分方法进行学习。

DPG 中的 Critic 网络拟合动作价值函数，目标函数是最小化 TD 误差，即

$$\delta = r_{t+1} + \gamma \max_{a'} Q_w(s_{t+1}, a') - Q_w(s_t, a_t) \tag{2.5}$$

Critic 网络的参数更新公式是：

$$w = w + \eta_1 \delta \nabla_w Q_w(s_t, a_t) \tag{2.6}$$

Actor 网络拟合策略函数，目标是对确定状态得到使动作价值最大的动作，故其

优化目标为:

$$\max_{\pi(s_t)} Q(s_t, \pi(s_t)) \tag{2.7}$$

Actor 网络的更新公式是

$$\xi = \xi + \eta_2 \nabla_a Q_w(s_t, a)|_{a=\pi_\xi(s_t)} \nabla_\xi(s_t) \tag{2.8}$$

2.2.2　算法细节

状态设置

在该阶段，状态设置与 2.1中相似。该部分的状态向量包括:

$$state = (self_x, self_y, goal_x, goal_y, dist, \theta) \tag{2.9}$$

系统动作

该部分我们没有考虑我方机器人是否射击，但要注意的是，在该部分，我们没有固定我方机器人的朝向使之始终指向目标，而是作为系统动作中的变量。所以该部分机器人的控制向量为:

$$action = (move\_x, move\_y, rotate\_cc, 0) \tag{2.10}$$

其中，*move_x* 代表机器人水平方向移动，*move_y* 代表机器人垂直方向移动，*rotate_cc* 代表转动角度。

奖励函数的设计:

我们的目标是要让我方机器人实现避障与路径规划的功能，在奖励函数的设计上类似离散动作控制的做法，本文考虑目标奖励、距离奖励、碰撞惩罚，由于机器人的朝向并非始终指向目标，而目标激活的条件中需要机器人的朝向与其夹角不大于 30 度，因此，在奖励函数中考虑角度奖励：定义我方机器人的朝向与我方机器人-目标点连线之间的夹角变小时，给予奖励；当我方机器人的朝向与我方机器人-目标点连线之间的夹角变大时，给与惩罚。在这里，我们用如下公式表示。

$$Reward\_angle = 10 \times \left[sign(abs(\theta_{t+1}) < abs(\theta_{t+1})) \right] \tag{2.11}$$

奖励函数的公式如下:

$$R = Reward - 10 - dist(robot, goal) - 70 \times count\_coll - 30 \times time\_coll + Reward\_angle \tag{2.12}$$

其中，$Reward, count\_coll, time\_coll$ 与 2.1定义相同。

2.2.3 实验结果

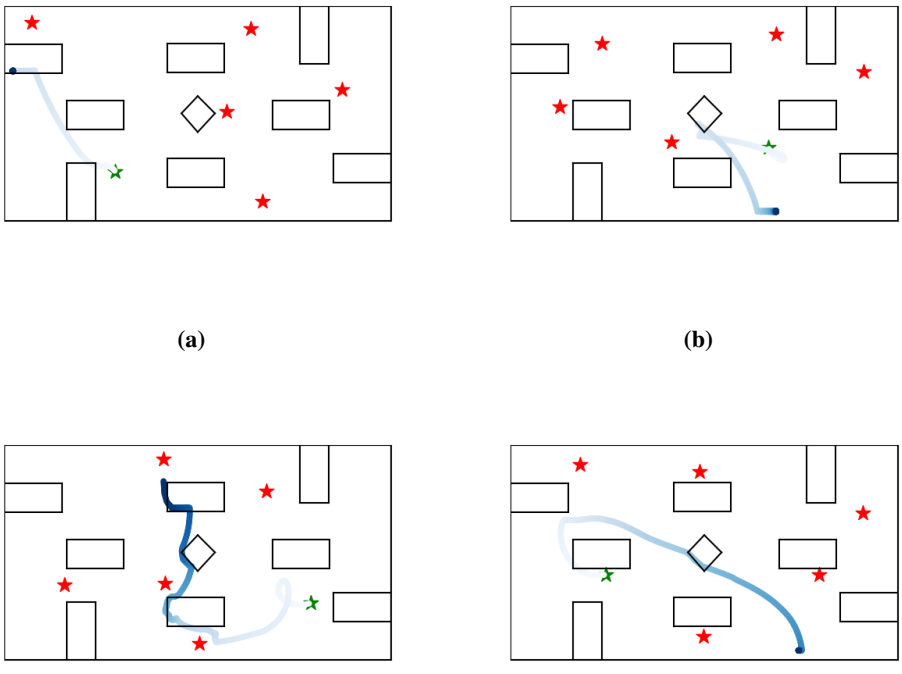
2.2.3.1 实验环境

表 2.2 实验环境

操作系统	Windows 10
GPU	Nvidia Geforce RTX 3080Ti
Python 版本	3.10
深度学习框架	PyTorch

2.2.3.2 参数设置及结果

算法设置探索总步长为 10^5 步，设定每局最大步长为 500，模型对机器人控制情况如图轨迹所示。



(a) **(b)**

(c) **(d)**

图 2.2 基于 DPG 的初步导航结果。绿色星状代表我方机器人的初始位置；红色星状代表目标的位置；蓝色代表我方机器人的行进轨迹。(a) 情况一，(b) 情况二，(c) 情况三，(d) 情况四。

2.2.3.3　结果分析

从轨迹图可以看出，机器人并没有正确的避障，并且从训练过程中的奖励可以观察出，机器人找到目标的轨迹样本是极其稀疏的，而大多数的行动轨迹中，机器人遭遇路面障碍物时行动受限，导致机器人对于环境探索不均匀且大多集中于墙体周围，这使得算法训练效率不高。

2.3　算法改进

通过对结果的分析，本文对当前算法进行改进。考虑到机器人当前的策略效果不好是机器人对环境的探索不充分导致的，即机器人遭遇墙体阻挡时产生了过多重复无效状态。因此本文考虑对探索路径进行截断，当机器人与障碍物碰撞或找到目标时重置系统状态，减少机器人阻塞于障碍物周围的状态。同时考虑到机器人的避障性能，本文对机器人的状态、奖励函数进行了改进。另外，由于本文采用的机器人到目标的距离是欧式距离，该距离没有考虑障碍物的信息，因此是具有欺骗性的，本文在之后的改进思路上引入了 GMM 模型对该距离进行修正。最后，考虑到本文采用的稠密奖励可能会使得我方机器人在目标和障碍物之间陷入局部极值点导致卡顿，本文引入后处理机制，并充分解决了这个问题。

2.3.1　状态

为了提升机器人的探索性能，本文在原有的状态基础上增加了雷达信息，且为了防止雷达信息特征过多导致训练效率低的问题，本文均匀采样了 7 个位置的雷达信息，用于解决避障问题。而寻径相关的特征则包括机器人到目标的位置和机器人朝向与目标的夹角。本文最终采用了包含上述的 9 维特征向量。

2.3.2　奖励函数

改进后的奖励函数包括如下几个部分：

- 目标奖励 (*Reward_goal*)：我方机器人每找到一个目标，便给予一个较大奖励。
- 距离奖励 (*Reward_dist*)：由于根据之前的实验结果，我们发现我方机器人容易陷入局部极值点，从而造成卡顿现象。因此我们考虑改进距离奖励来缓解这一现象：

当我方机器人朝向目标点靠近并且移动的距离较大时，给予较大的奖励；

当我方机器人朝向目标点靠近并且移动的距离较小时，给予较小的惩罚；

当我方机器人远离目标点时，给予较大的惩罚；

- 角度奖励 (*Reward_angle*)：该部分的奖励与公式 2.11一致。
- 碰撞惩罚 (*Reward_collision*)：每次检测到碰撞，给予机器人一个很大的惩罚。

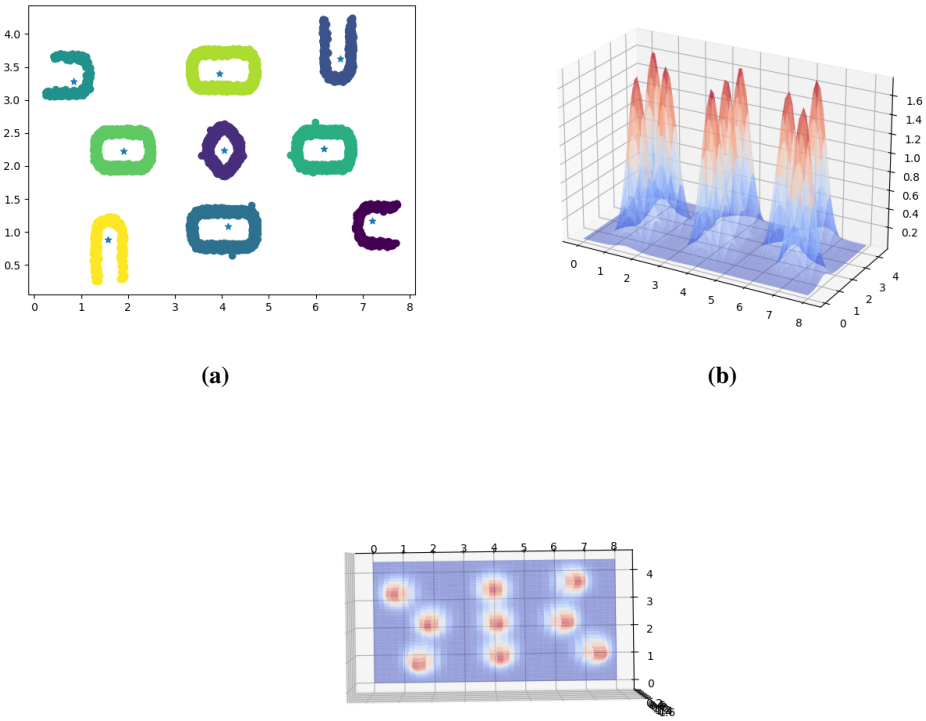
2.3.3 训练方式

我们考虑到，我们设计的奖励函数是一个比较稠密的奖励函数，因此，如果我们将每一局 (Episode) 设定为依次找到 5 个目标点，可预见的是在训练过程中我方机器人在规划过程中容易陷入局部极值点，从而造成我方机器人的卡顿。机器人的卡顿会导致在训练过程中多了很多无效探索，即由于卡顿，造成放入经验回放池中的样本是无意义的。所以，造成训练过程效率很多，我方机器人对环境的探索也不够充分，效果也不尽人意。因此，我们改变了训练方式，把任务拆分，即将每一局设定为找到一个目标点或者碰到障碍物。每找到一个目标点或发生碰撞就将环境重置 (Reset)，这种方式在一定程度上缓解了上述问题，我们通过实验也验证了该训练方式的可行性。

2.3.4 基于 GMM 的固定障碍物分布的拟合

为了更好地实现避障的功能并且尽可能准确地获得我方机器人与目标之间的距离，我们对环境中的固定障碍物的分布进行了估计。

我们在训练过程中记录下发生碰撞时我方机器人的坐标，代表障碍物的坐标。随着训练过程的增加，可以大致获得障碍物的外围坐标。之后通过 DBSCAN 的方法对获得到的障碍物的坐标进行聚类，并且利用高斯混合模型 (Gaussian Mixture Model, GMM) 对障碍物分布进行拟合。固定障碍物的分布以及 GMM 的拟合情况如图 2.3所示。



(a)　　**(b)**

(c)

图 2.3 固定障碍物的分布以及 GMM 的拟合情况。(a)DBSCAN 聚类结果，(b) GMM 对固定障碍物分布的拟合情况，(c)GMM 对固定障碍物分布的拟合俯视图。

在得到障碍物的分布后，我们修正了奖励函数中我方机器人与目标之间的距离 ($dist(robot, goal)$)。修正后的距离用我方机器人与目标点之间的连线在 GMM 表面上的曲线积分表示。当机器人与目标点之间恰好隔着一个固定障碍物时，改进前的欧式距离会误导机器人认为距离目标很近，但是训练的网络又不允许机器人在当前状态下直接沿着最短欧氏距离的方向前进 (会与固定障碍物发生碰撞)，从而造成该情况下的机器人卡顿。修正距离可以让机器人在上述情况下认为距离目标较远，从而采取绕开障碍物的动作。修正后的距离的示意图是如图 2.4所示：

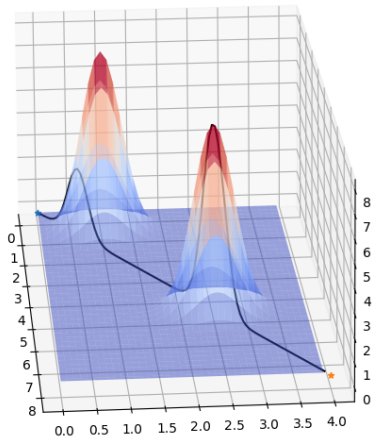
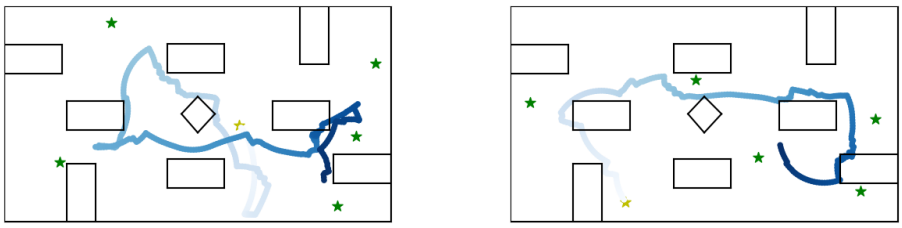


图 2.4 曲线积分示意图

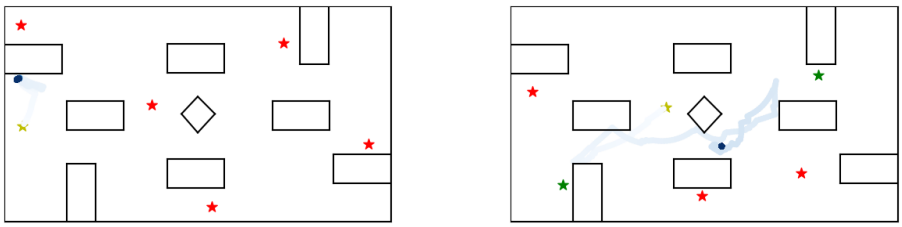
2.3.5 后处理操作

经过上述的改进，虽然导航的性能有所提高，如图 2.5所示。但是，仍然存在在导航途中机器人陷入局部极值点而卡顿的现象。如图 2.6所示。



(a) **(b)**

图 2.5 基于 DPG 的初步改进后的导航结果。绿色星状代表我方机器人的初始位置；红色星状代表目标的位置；蓝色代表我方机器人的行进轨迹。(a) 情况一，(b) 情况二。



(a)　　(b)

图 2.6 基于 DPG 的导航方法的卡顿情况。绿色星状代表我方机器人的初始位置；红色星状代表目标的位置；蓝色代表我方机器人的行进轨迹。(a) 情况一，(b) 情况二。

为了缓解这一现象，我们对我们的算法添加了后处理机制。具体的处理步骤如下：

1. 通过队列记录当前时刻 T 以及之前 N 步的轨迹。即 $T-N \sim T$ 时刻的我方机器人的坐标。

2. 计算 $T-N \sim T$ 时刻我方机器人的坐标的方差 var。

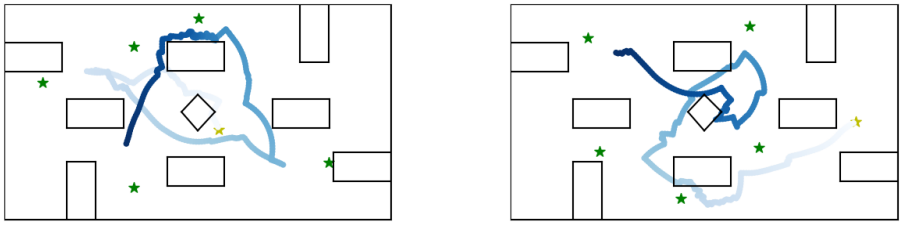
3. 如果 var 小于某一阈值，则认为我方机器人处于卡顿状态，并触发后处理机制。

4. 后处理操作 1：根据我方机器人的雷达信息，找到最大雷达距离的方向，并朝着该方向运行 a 步。

5. 后处理操作 2：在机制 1 之后，计算每个方向上雷达信息的向量 $\vec{n}$ 与我方机器人-目标点连线向量 $\vec{g}$ 的点积，找到最大点积的方向，并朝着该方向运行 b 步。

2.3.6　最终实验结果

整体的改进完成后，基于 DPG 的最终的导航结果如图 2.7所示。



(a) (b)

图 2.7 基于 DPG 的最终的导航结果。绿色星状代表我方机器人的初始位置；红色星状代表目标的位置；蓝色代表我方机器人的行进轨迹。(a) 情况一，(b) 情况二。

我们将改进前后的方法分别进行了 20 次实验，并分别记录了平均找到目标点的个数和环境提供的分数 (Score，仿真环境提供)。最终结果如图 2.3所示。

表 2.3 最终导航实验结果

采用的方法	平均找到的目标个数	平均得分
原始 DPG	0.5	-
改进 DPG(无后处理)	4	175.56
改进 DPG(有后处理)	5	363.16

第 3 章　机器人对抗

在该阶段的任务中，我们要进行我方机器人与敌方机器人的对抗。其中，敌方机器人的朝向始终指向我方，并且，当我方机器人出现在敌方机器人的射程内时会对我方机器人进行射击（攻击机制），敌方机器人在运动时会产生随机动作以尽可能的防止我方机器人瞄准（防守机制）。因此，本文将从攻击机制和防守机制的设计两个方面完成敌我双方机器人对抗的策略。

3.1　攻击机制

在设计机器人的攻击策略时，本文考虑靠近敌方机器人和对敌方机器人射击两个方面。

在靠近敌方机器人方面，本文的目标是让机器人在尽可能躲避障碍物的基础上，找到地方机器人的位置，将朝向对准敌方机器人，并与其保持一段距离。此问题与机器人导航问题有极大的相似性，在问题一中，本文已经得到了一个具有优越性能的避障并寻找目标的策略，而该策略与本问题的区别在于机器人需要与目标保持一定的距离的安全区域而非始终靠近目标，因此，本文通过更改 2.3.1状态中机器人与目标的距离，将该值减少安全区域的半径大小。

更改后的状态向量中的距离项为：

$$dist_e = d \times sigmod(d - r) \tag{3.1}$$

其中，d 代表我方机器人与敌方机器人之间的距离，r 为我们设定的与敌方机器人应保持的距离半径。

此外，为了让我方机器人更快地对准敌方机器人，我们对状态向量中的角度项也进行了修正，修正后的角度项为：

$$\theta_e = \begin{cases} \frac{\pi \times \log(\theta+1)}{\log(\pi+1)}, & \theta \leq 0 \\ \frac{-\pi \times \log(1-\theta)}{\log(\pi+1)}, & \theta < 0 \end{cases} \tag{3.2}$$

在对敌方机器人射击方面，由于靠近机器人时包括机器人的行动策略（包括水平、垂直移动及转动角度）和对抗策略，而行动策略可以通过已有的模型转换

得到，因此本文采用分层强化学习的思想，将对抗任务由两部分策略完成，行动策略和射击策略，其中射击策略给出在机器人移动的过程中是否设计。机器人的射击动作是一个离散动作，因此本文采用 DQN 求解机器人的射击策略。综上，该阶段我们定义的动作向量为：

$$action = (move\_x, move\_y, rotate\_cc, DQN_{shooter}) \tag{3.3}$$

其中，$move\_x$ 代表机器人水平方向移动，$move\_y$ 代表机器人垂直方向移动，$rotate\_cc$ 代表转动角度，$DQN_{shooter}$ 表示 DQN 的输出决定是否射击。

对于射击策略的学习，本文将其形式化为一个强化学习问题，采用 DQN 算法。由于该策略不需要考虑避障的问题，因此定义系统状态为敌我机器人之间的距离、我方机器人的朝向和目标方向的夹角以及 0 度雷达信息共三维的特征。奖励函数设计为防守机器人对抗的得分:$0.5 \times (D + H)$，其中敌我双方初始血量为 800，结束时防守机器人的伤害为 D=800-防守机器人血量剩余，我方机器人的血量剩余为 H。同时为了提高子弹的利用率防止子弹提前用尽的情况，当子弹数减少时会给予不大于击中目标获取的奖励数值的惩罚。

3.2 防守机制

在防守方面，我们的目标是尽可能地移动来干扰敌方机器人，同时尽可能保持炮台对准地方机器人。我方机器人在与敌方机器人保持一定距离之后，开始进行防守机制，这时，我方机器人的运动是以敌方机器人为圆心，敌我双方机器人地距离为半径的弧形运动。与公式 3.3相似，弧形运动采取的动作可以如下表示：

$$action = (0, move\_y, -\frac{move\_y}{d_e}, DQN_{shooter}) \tag{3.4}$$

其中，因为每一个 step 移动的距离很小，所以我们近似 $move\_y$ 为我方机器人运动的弧长，$-\frac{move\_y}{d_e}$ 近似表示我方机器人为对准敌方机器人应转的角度。

此外，为了获得较大的躲避范围，我们利用将动作 3.4重复执行 N 步。

3.3 实验结果

在这里我们对整个任务 (任务一 + 任务二) 实验了 20 次，并记录了最终系统反馈的得分情况，详细数据如下表所示。我们在附件中的视频提供了具体的实验效果。

表 3.1 最终导航对抗实验结果

实验次数	1	2	3	4	5	6	7	8	9	10
得分	454.9	218.02	-98.28	181.00	470.56	416.58	416.22	714.7	322.44	620.62
实验次数	11	12	13	14	15	16	17	18	19	20
得分	434.98	247.26	209.72	433.82	310.20	496.78	505.3	671.82	430.98	224.1
平均得分	384.09									