扬 州 大 学 毕 业 设 计（论 文）

对抗环境下多机器人协作追捕路径生成系统

学　　号: 141405157

姓　　名: 颜文豪

专　　业: 软件工程

指导老师: 朱俊武

二〇一八年五月

揚州大學

本科学位论文

论文题目：对抗环境下多机器人协作追捕路径生成系统

论文编号：

摘　　要

本文提出了一种解决在对抗性多机器人追逃环境下协作追捕进行路径选择的途径——基于深度Q学习网络算法的协作追捕路径生成系统。

该系统采用一种将深度学习与增强学习相结合的路径生成算法。它通过在增强学习中引入神经网络的形式，可以较好地解决传统方法所不可避免的收敛困难和维度灾难。以该方法为理论基础建立的路径生成系统，相比较于目前被发布使用的系统，这种系统具有更加先进的前沿性与更加广阔的适用性。

本文的大致内容结构如下：

1. 介绍了机器人的追捕与逃逸及追捕路径生成问题的研究价值、用途和目前研究的一系列成果，以及深度学习与增强学习的研究情况；

2. 介绍了目前主流的一些增强学习算法，探讨他们的基本原理和优缺点以及在本问题下应用；

3. 介绍了深度Q学习网络算法的思路、流程、原理和仿真方式；

4. 描述了本文建立的协作追捕路径生成系统的开发环境，程序结构与运行流程；

5. 对开发完成的系统进行测试；

关键词：追捕&逃逸； 路径生成； 深度Q学习网络； 增强学习； 深度学习；

Abstract

This paper proposes a new solution towards the pursuit-evasion problem of multi-robots system. Cooperative Path Generation System Based on the algorithm of Deep Q Network.

This system uses a combinatorial algorithm of both Deep learning and Reinforcement learning. By drawing neural network into reinforcement learning algorithm, it solves the difficulties of convergence and dimension disaster of traditional methods successfully. The path generation system based on this theory, offers highly-advanced progressiveness and much more widely application compared to similar system under use nowadays.

The main contents of this paper are as follows:

1. Introduces the research value of the pursuit-evasion problem, uses and a series of results of the current study, together with something about deep learning and reinforcement learning;

2. Introduced the current mainstream of the reinforcement learning algorithm, to explore their basic principles and advantages and disadvantages.

3. Describe the designed algorithm of DQN, and its ideas, procedures, principles and the relatives;

4. Describe the developing environment of the system this paper proposes, along with the code structures and technological running process.

5.Test the System.

Keywords: Pursuit-Evasion; Path generation; Deep Q Network; Reinforcement learning; Deep learning

目 录

第一章 引言

1.1 课题背景与意义

1.1.1 关于追捕-逃跑问题

多机器人系统中的追捕-逃跑问题一直被研究多智能体自学习的学者关注，同时它在自动控制领域也是一个经久不衰的热点问题。追捕-逃逸问题描述的是自然界、工业界以及军事界中普遍存在的多智能体协作与博弈问题。在自然界中，狩猎者群体通过合理的路径规划和手段将猎物捕获，而猎物则尝试逃离追捕。如果出现多个狩猎者，狩猎者之间会互相得知各自的位置，并在某些情况下，狩猎者会进行相互交流，达到合作追捕目标的目的。多智能体领域的研究人员经常通过模拟追捕逃逸问题测试各类多机器人系统，因为追捕逃逸的效率高低直接反映了机器人或智能体的学习效果。在对实时性要求较高的工业或军事领域中，追捕-逃逸问题的研究成果受到了更加广泛的应用。此外，有许多博弈论的研究者会聚焦于追捕智能体与逃逸智能体之间的对抗性决策的研究。因此，追逃问题因其独有的一些性质，理所当然地成为了自动控制理论、增强学习和多机器人系统等领域学者们首选的研究平台。

追捕-逃逸问题经常在多智能体理论研究中被人们对其进行反复的讨论和改进，Benda 等人聚焦研究单猎物合作追捕算法，他们用四个追捕机器人对一个逃逸机器人在栅格环境内进行追捕，并收到了良好的效果。Korf 提出允许斜向运动的动作空间设定代替改进了原本被广为使用的直角近似方案，并利用贪心思路将可满足的最多机器人联合追捕猎物的机器人数量增加到 8 个。Kopparty 详细阐述了追捕者必然抓获逃逸者的条件，他通过计算几何颇为详细地证明了他提出的结论，但是他建立的环境中赋予追捕者全局整体视野，这并不是一个常见的情形。Stephens 和 Merx 研究了在无通信条件下多个追捕者的协作追捕策略，发现这种条件对追捕者的效率的影响十分明显，由于相互间通信的缺乏，追捕者频繁出现干扰到其他追捕者的移动方向的情况。这对追捕的效率造成了很大的负面影响。

追捕-逃跑问题的研究成果被广泛地应用在各个领域。在军事领域中，弹道导弹要通过合理的路径规划对目标实施追踪和精确打击，这就需要相当复杂的自学习改进策略机制；我国在南海区域使用大量的无人机群进行巡航，用来围捕从国内驶往东南亚等地区的自私民船，以及将外来的具有恶意动机的船只阻截后送回公海区域，这就需要无人机群能够自主的完成队形控制、目标确定、路线规划等需要无人机之间相互

协调，保持通讯的任务。

1.1.2 强化学习

自从 80 年代末到 90 年代初期，强化学习开始受到人们的关注，它的相关成果被广泛地应用在分析预测、行为模拟以及多机器人控制等领域。强化学习是一种允许智能体进行在线学习的算法模式，智能体在实际或模拟的环境中进行所有可能的尝试动作并从环境中获取符合环境规则的奖励值，以此逐步调节自身的行为模式，直至找到最优解或局部最优解。在模型训练的初始阶段，智能体对周边环境完全未知，这时它采取的决策完全通过“天性”（被赋予的初始值）获得，之后通过与环境交互的训练过程，逐步迭代更新并改变智能体自身的行为方式，使自己能够从环境中获得更高的奖励值，因此智能体的行为也会变得越来越好。

强化学习中有许多算法在机器人领域被广泛应用，其中就包括 Q 学习、TD 学习、状态行动奖励状态行动算法和策略梯度算法等。强化学习被认为一种无监督学习过程，或者说它在训练过程中受到环境的监督（奖励），不需要通过人工的手段来标记样本，系统会通过从环境中获得的 reward 值来调整自己以适应所处的环境。训练好控制程序的无人机可以轻松地完成在空中规避障碍物和组队完成地面搜索等任务。另外，由于在线学习的特性，强化学习也适用于一些实时性要求较高的工业需求。

然而，强化学习也会不可避免地因为学习规模的扩大而出现收敛速度变慢的情况，这在几乎所有人工智能领域十分常见，由于多智能体的学习状态组合维度过高而出现组合爆炸，其计算复杂度呈指数增长，这就是所谓的维度灾难问题。

研究者们提出了许多基于传统强化学习算法的改良方法来解决维度灾难的问题，其中就包括分层强化学习理论。分层强化学习理论改变了经典强化学习中对所有状态平等对待的做法，通过建立高层的抽象状态以达到简化学习的目的。而对于无法进行分层的情况，研究者们则选择使用神经网络来取代 Q 表格进行 Q 学习模式，即 Deep Q Network 算法（DQN）。

1.2 开发方法与使用技术

强化学习是机器学习的重要组成部分，这一领域拥有区别于其他机器学习领域的理论体系，同时强化学习也是一个实战性很强的领域，这就需要研究者学会使用强有力的开发工具。Python 是一种面向对象、解释型高级程序设计语言。作为一种开源的脚本语言，其在机器学习领域具有其他高级语言无法获得的优势。

1.2.1 开发方法

Python 语言是一种完全开源的脚本语言，受到了各行业算法研究者的青睐，众多鼎鼎大名的科学计算库都为 Python 语言提供了的调用接口，这也赋予了 Python 语言极强的可移植性。这为强化学习程序开发过程中数据组织、数据预处理、数据可视化、数据分析建模以及模型验证的过程提供了非常便捷的途径。

Python 语言的语法规则十分简洁，其开发出的程序代码具有极佳的可读性，这也方便了未来对程序的维护。

Tkinter 模块是 Scriptics 的 GUI 工具包（以前由 Sun Labs 开发）的标准 Python 接口。也是 Python 语言在 3.0 之后的版本默认包含的 GUI 开发工具包。Tkinter 可以在大多数 Unix 平台以及 Windows 和 Macintosh 系统上使用。 且从 8.0 版本开始，Tk 在所有平台上提供本地外观。 Tkinter 由许多模块组成。 Tk 接口由名为_tkinter 的二进制扩展模块提供。 该模块包含 Tk 的底层接口，不应直接由应用程序员使用。它通常是共享库（或 DLL），但在某些情况下可能会与 Python 解释器静态链接。由 Tkinter 开发出的 GUI 程序，对于任何系统平台都拥有足够的兼容性和可移植性。

本文提出的多机器人协作追捕路径生成系统选择 Python 语言作为开发语言，其丰富的第三方科学计算库极大的为算法的开发实现提供了便利，也是机器学习研究的首选编程语言。

1.2.2 技术进展

TensorFlow 脱胎于大名鼎鼎的 Google Brain 项目。 它的第一代正式版发布于 2017 年 2 月 11 日。而且不止时可以在单个设备上运行，TensorFlow 可以同时在多个 CPU 和 GPU 上运行。 TensorFlow 拥有面向 64 位 Linux，macOS，Windows 以及各种移动计算平台的发行版，即使在 Android 和 iOS 上也可以运行。 TensorFlow 计算表示为有状态数据流图。 TensorFlow 这个名字源于这种神经网络在多维数据阵列上执行的操作。这些阵列被称为“张量”。

从发布的第一个版本起，TensorFlow 就开放了 Python 语言的编程接口，使用 Python 的开发者和研究人员可以通过 TensorFlow 构建出复杂的神经网络模型。

1.3 本文结构与内容

追逃问题并不是一种新的问题，它有多年的研究历程。经过多年的发展，研究者们提出了大量的关于追逃问题的策略方法，但是，强化学习在被提出后，取代这些方法成为解决追逃问题的主要思路。接下来，本文会分别对一些主流强化学习方法进行简单的介绍。

本篇论文将着重讨论解决追逃问题的算法。现如今几种主流的算法都有其各自独特的优缺点，本文会对其进行较为详细的分析，并选取一种可行的思路，构建多机器人协作追捕路径生成系统，来对多机器人协作追捕问题提出一种全新的解决途径，以及对这一新系统的测试与验证。

第二章 深度强化学习

2.1 强化学习概述

首先，强化学习（reinforcement learning），又称为增强学习或激励学习，它的主要思路来自条件反射论和动物学习理论，是一种在学习训练过程中受到来自环境因素的奖励或惩罚而启发自身的仿生算法。是机器学习理论的重要组成部分。智能体对环境做出各种试探性动作，获得环境对这个动作优劣评价的奖励值，从而改变自身选择动作的策略以获得更高的奖励，并不断地重复这一“试探--奖励”的过程。

强化学习算法是一种具有普适性的算法，因此除了在计算机科学领域，强化学习在其他许多领域都有研究，例如博弈论、控制科学、运筹学、仿真优化、多主体系统学习、群论、数理统计学以及遗传进化算法。在运筹学和控制理论研究的语境下，强化学习被称作“近似动态规划”（approximate dynamic programming，ADP）。在最优控制理论中也有研究这个问题，虽然大部分的研究是关于最优解的存在和特性，并非是学习或者近似方面。在经济学和博弈论中，强化学习被用来解释在有限理性的条件下如何出现平衡。

强化学习和标准的监督式学习之间的区别在于，它并不需要出现正确的输入/输出对，也不需要精确校正次优化的行为。强化学习更加专注于在线规划，需要在探索（在未知的领域）和遵从（现有知识）之间找到平衡。强化学习中的“探索-遵从”的交换，在多臂老虎机问题和有限 MDP 中研究得最多。

2.2 Markov 决策过程

其次，目前主流的强化学习方法的研究都是建立在可被规范为马尔可夫决策过程（Markov Decision Process）的环境中。强化学习并不只适用于马尔可夫决策过程，但离散、状态有限的马尔可夫决策过程是许多经典强化学习算法的基础。

完整的马尔可夫决策过程由四个要素构成，研究者通常将其描述为四元组<S,A,T,R>，在这其中：

S：智能体可能处在的有限的状态集合；

A：有限的动作选择空间集合；

T：状态转移函数；

R：奖励函数。

在 Markov 决策过程中，智能体与环境交互的过程可被分为以下的步骤。

（1） 智能体获得当前所处的状态 $s_i \in S$；

（2） 根据算法执行一个动作 $a_i \in A$，并从环境中获得奖励值 r；

（3） 根据奖励值 r 更新自身参数 ；

（4） 根据状态转移函数：T: S×A→S，状态更新为 s_{i+1}

马尔可夫决策过程的本质就是当前状态向下一个状态转移的概率和奖励值只取决于当前的状态和所选择的动作，与历史选择的状态和动作。

2.3 Q 学习

Q 学习算法是强化学习发展的里程碑算法，在 1989 年被提出并推广。

Q 学习是一种经典的无模型强化学习算法，是一种通过智能体在环境中所经历的动作序列来选择当前最优动作的算法。在 Q 学习对智能体进行训练的过程中，通过不断的在每一步估计当前状态的奖励值函数，直接优化一个可迭代计算的 Q 函数来判别最优策略，这是一种增量式的在线学习。

Q 学习是强化学习算法中应用最普遍的算法之一，它迭代时采用状态与动作二元组的 Q（s, a）函数，在智能体的每一次学习迭代时考察每一个状态与动作，直到学习过程收敛。

2.3.1 Q 学习算法过程

Q 学习首先初始化 Q（s, a）的值：智能体在初始状态 s，根据某种策略选择一个动作 a，例如 ε-greedy 或 Boltzmann 策略，得到下一个状态 s'，立即获得回报 r，根据更新规则修改 Q 值，当智能体访问到目标状态或满足退出条件时，算法完成一次迭代循环，直到学习过程收敛。

Q 学习过程中通过优化可迭代计算的 Q（s, a）函数值来逼近最优值函数，基本的更新规则如下：

$$Q(s,a) = r + \gamma \max Q(s',a')$$

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha\left[r + \gamma \max Q(s',a')\right]$$

Q 学习过程中包括很多个情节（episode），每个情节重复上述步骤，直到算法退出。

2.3.2 Q 学习算法的收敛性分析

Q 学习算法通常是在一张多维表格内“搜索”每个状态-动作对有关的 Q 值，对

此，Q 学习算法的提出者 Watkins 已证明当学习率α满足一定条件时 Q 学习过程一定是收敛的。收敛条件如下：

（1） 奖励函数一定有界，

即：

∀（s, a），$|r(s,a)| \le c$ ；

（2） 用查询表来表示 Q 函数；

（3） 使用折扣因子γ， $0 \le \gamma \le 1$；

（4） 每个状态-动作对都可以被无限次访问。

（注）收敛定理：给定有界强化信号$|r_n| \le C$和学习率a_n（$0 \le a_n \le 1$），对所有的 s, a 满足：

$$\sum_{n=1}^{\infty} a_n = \infty$$

$$\sum_{n=1}^{\infty} [a_n]^2 = \infty$$

则当 n→∞时，对于任意的 s 和 a，$Q_n(s,a)$会以概率 1 收敛于最优 Q（s, a）。

2.4 深度学习与神经网络

深度学习（deep learning）就是通过构建具有很多隐层的机器学习模型和海量的训练数据，以此学习到更有用的特征，从而学习到更有用的特征，从而最终提升分类或回归的准确性。深度学习区别于传统的潜层机器学习在于：

（1） 强调了模型结构的深度，一般而言深度学习模型有 10 层以上的隐层节点。

（2） 明确突出了特征学习的重要性。

神经网络（neural network）是人工智能和生物科学的交叉部分，是具有适应性的神经元感知器组成的广泛的并行连接的网络结构，它的组织结构能够模拟生物的神经系统对真实外在环境所做出的反应。神经网络中最基本的组成部分就是神经元模型，图 2.1 所示为最简单的“M-P 神经元模型”。

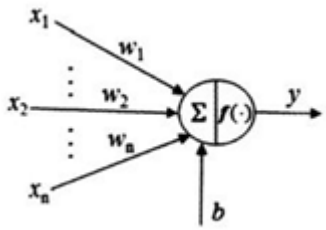

图 2.1 M-P 神经元模型

在 M-P 模型中，神经元通过带权重的系数与其他 n 个神经元相连接，输入信号的加权输入总值与阈值比较后，通过激励函数后作为神经元的输出。通常的激励函数包括且不限于 sigmoid 函数，tanh 函数，relu 函数。在本文所提出的算法中，神经元的激励函数为 relu’函数。

relu 函数、tanh 函数和 sigmoid 函数的图像如图 2.2、图 2.3 和图 2.4 所示。

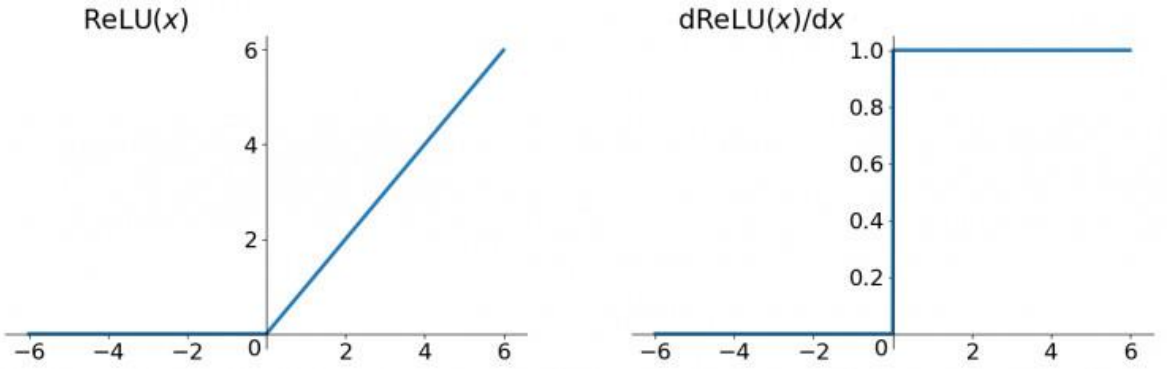

图 2.2 Relu 函数图像与其导数的图像

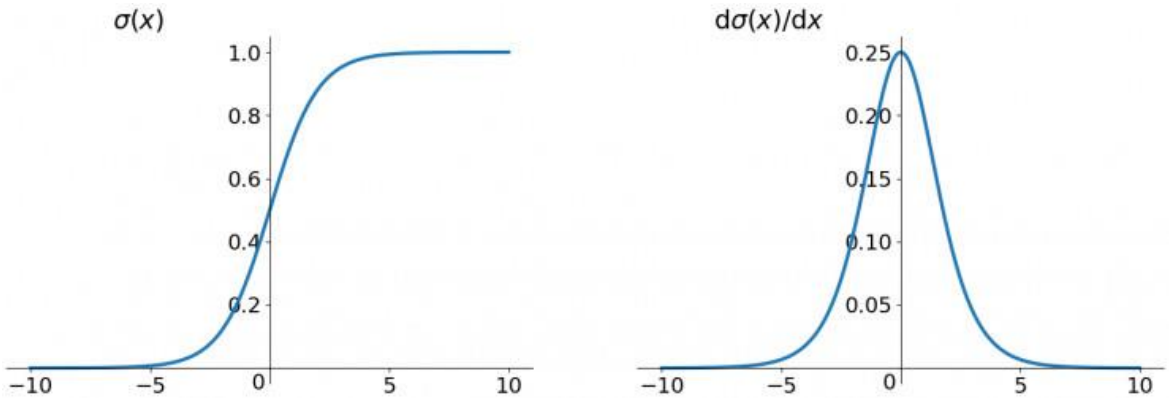

图 2.3 Sigmoid 函数图像与其导数的图像

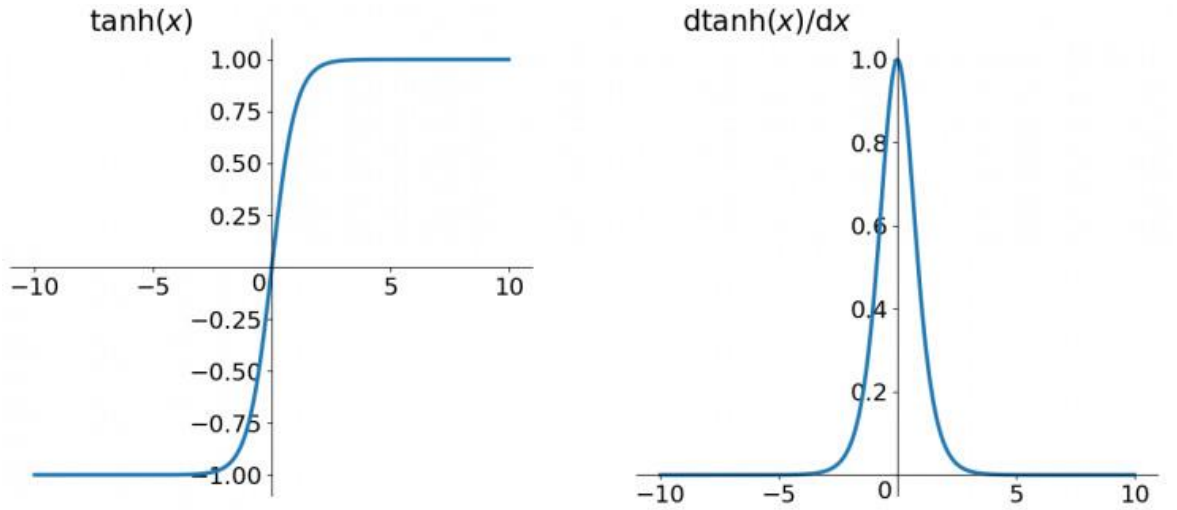

图 2.4 tanh 函数图像与其导数的图像

2.5 深度强化学习

深度学习（deep learning）作为当今人工智能领域最引人关注的研究热点之一，已经在图象处理、语音识别等领域取得了举世瞩目的成功。深度学习的基本思想是通过多层网络结构和非线性变换，发现数据的特征表示。强化学习作为机器学习领域的另一个热点，则侧重于学习解决任务的策略方法。在机器学习付诸于实用的越来越复杂的现实场景中，人们采用了利用深度学习来学习输入的海量数据的抽象特征，再以这些特征为依据训练自我激励的强化学习模型，以优化解决问题的策略。这就形成了人工智能领域新的研究热点，即深度强化学习（deep reinforcement learning）。

深度强化学习是一种端对端的感知与控制系统，具有极强的适应性和通用性，其学习过程可以描述为：

（1） 智能体与环境交互得到一个高维度的环境观察值 s，用深度学习的方式来对其做感知，以得到当前环境的特征表示；

（2） 基于预期回报来评价各个动作的奖励值，并通过某种策略将状态 s 映射为相应的动作；

（3） 环境对智能体的动作做出响应，并得到对下一个状态的环境观察值 s'，循环上述过程。

由此在一定次数的迭代后，最终可以得到实现目标任务的最优策略，深度强化学

习的原理如图 2.3 所示 ：

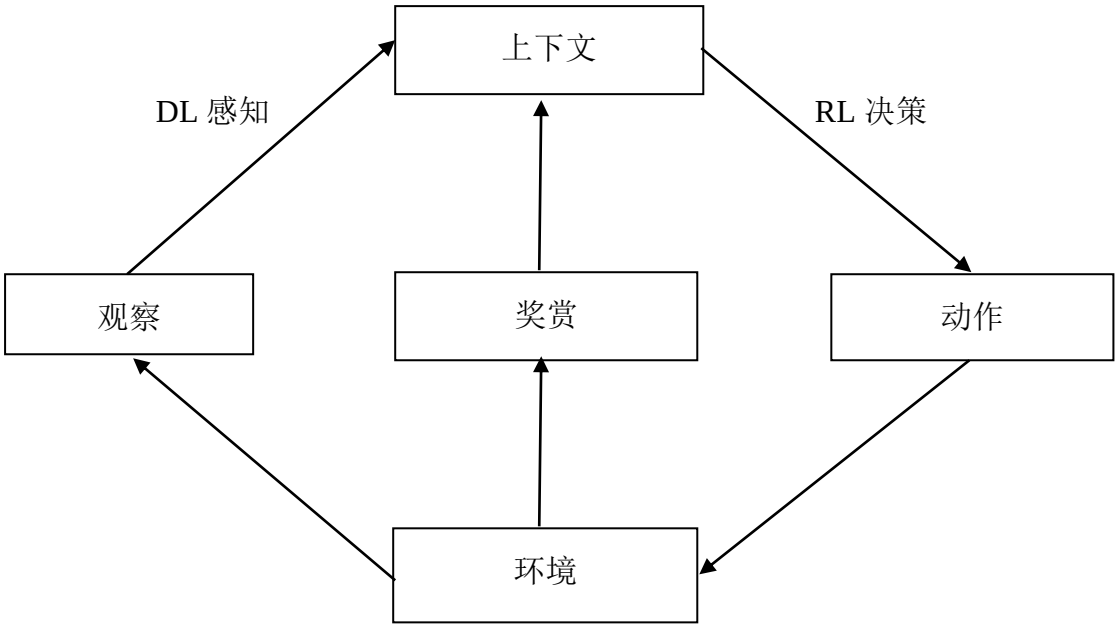

图 2.3 深度强化学习原理图

第三章 基于 DQN 模型的协作追捕路径生成算法

本文研究的是三个追捕机器人追捕同一个猎物，猎物采取随机逃跑的策略，三个追捕者通过合作来追捕猎物，因此本文提出的路径生成系统模拟的是多机器人协作追捕的问题。

3.1 状态空间以及机器人动作的离散化

真实环境中的追捕-逃逸问题是在连续的时空环境下，而深度强化学习的算法要求环境满足 Markov 决策模型，因此需要对机器人的运动过程做好恰如其分的划分。本文研究的追捕-逃逸环境设定在二维的栅格环境中，追捕者与逃逸者都可以执行“上”、“下”、“左”、“右”四个动作，且在每个回合内按照“逃逸者——追捕者 1 号——追捕者 2 号——追捕者 3 号”的顺序依次执行。彼此联盟的机器人可以实时获取盟友机器人与自己的相对位置，所有追捕机器人都可以实时获取逃逸机器人的与自己的相对位置。

3.2 追捕成功条件

在以往的追捕-逃逸研究中，大部分都是在栅格环境下做的仿真，在栅格环境下，围捕成功的条件定义为如果追捕者占据了逃跑者所有临近栅格，那么逃跑者无处可逃从而被抓获。但是虽然本文因为算法特性的原因也不可避免的使用了基于栅格的离散型追捕-逃逸环境，但是为了实现更加贴近真实的连续环境仿真，本文选择采用更加复杂的通过角度和距离的追捕成功条件定义。

假设 t 时刻追捕者 p_i 的坐标是 $x_{pi}(t)$，逃跑者 e 的坐标为 $x_e(t)$，$\theta_{\max}$ 表示参与包围 e 的所有 p_i 中的最大相邻角，追捕成功条件如下：

追捕成功条件：如果包围 e 的 p_i 形成的包围圈严格包含 e，且 p_i 均匀分布在包围圈上，即存在 ε 和时刻 γ 以及某个定常数 C，使得下两式同时成立：

$$\max_{p_i \in P} \left| x_{p_i}(r) - x_e(r) \right| < \varepsilon$$

$$\left| \theta_{\max}(\gamma) - \frac{2\pi}{k} \right| < c$$

则 e 在时刻 γ 被 P 抓住。其中，$x_{pi}(t)$、$x_e(t)$ 分别是 t 时刻 p_i 和 e 的坐标，$\theta_{\max}$ 表示 p_i 与 e 之间连线夹角的最大值，k 为参与围捕 e 的追捕者的个数。

3.3 深度 Q 网络算法

3.3.1 深度 Q 网络算法的优势

深度 Q 网络（DQN）的训练算法也叫做深度 Q 学习，这个算法在 Q 学习的基础上做了一定的改进，改进主要体现在：

（1） 建立了经验回放机制；

（2） 固定目标 Q 值网络；

（3） 减小了 reward 值的范围。

具体的改进为：

（1）使用经验回放机制。在每个单位时间 t 中将智能体与环境交互得到的转移样本$e_i=(s_i,a_i,r_i,s_{i+1})$存储到回放记忆单元 D 中。训练时从 D 中随机抽取固定数量的一批转移样本，并使用随机梯度下降(SGD)算法进一步优化神经网络中的 Weight 和 Biase 参数。在训练深度神经网络时，通常要求各个转移样本之间是符合独立性假设的。因此上述随机采样的方式，去除了样本之间的关联性，从而提升了训练网络模型时的稳定性和性能。

（2）固定目标 Q 值网络。深度 Q 网络除了使用深度神经网络逼近状态动作值函数之外，还单独使用另一个深度神经网络来产生目标 Q 值。其中，$Q(s,a|\theta)$代表当前值网络的输出，用来评估当前状态动作对的值函数;$Q(s',a'|\theta^-)$代表目标值网络的输出。在深度 Q 学习算法中，通常采用$Y=r+\gamma\max^{Q}(s',a'|\theta^-)$项来近似表示值函数的优化目标，即目标 Q 值。当前值网络的权重θ保持实时更新，并且每经过 C 个时间步，将当前值网络的权重复制给目标值网络$\theta^-\leftarrow\theta$。通过最小化值网络输出 Q 值和目标 Q 值之间的均方误差来更新网络权重。构造的误差函数为:

$$L(\theta)=E_{s,a,r,s'}\left[\left(Y-Q(s,a|\theta_i)\right)^2\right]$$

（3）缩小误差项的范围。在不同的任务场景中，奖励的量级可能是大不相同的。而太大的奖励设置会导致梯度想过大，从而导致学习的不稳定。因此在深度 Q 学习算法中将误差项缩减到小区间[-1,1]内，以保证梯度项的大小处于合理的范围内。显著提高算法稳定性。

3.3.2 深度 Q 网络算法的过程

1：初始化：回放记忆单元 D 的容量为 N，Q 值函数的初始权重为θ，目标 Q 值函数的初始权重$\theta^-=\theta$

2：Repeat（对于每一个 episode）：

3： 初始化状态序列为$s_1=\{o_1\}$，并经过预处理后得到$\phi_1=\phi(s_I)$

4： Repeat（对于 episode 中的每一个 step）：

5： 以概率 ε 选择随机动作 a_i，否则选择动作 $a_i = \arg\max^{Q}\left(\phi(s_i), a|\theta\right)$

6： 执行动作 a_i，并得到奖励 r_i 和下一个 O_{i+1}

7： 设置 $s_{i+1} = s_i, a_i, o_{i+1}$，并经过预处理得到 $\phi i+1 = \phi\left(s_{j+1}\right)$

8： 存储转移序列 $\left(\phi i, a_j, r_j, \phi i+1\right)$ 到 D 中

9： 从 D 中随机采样固定数量的转移样本 $\left(\phi i, a_j, r_j, \phi i+1\right)$

10： 如果达到终止状态，则优化目标设置为：$Y_i = r_i$

11： 否则设置为： $Y = r + \gamma \max^{Q}\left(s', a'|\theta^{-}\right)$

12： 对 $L(\theta) = E_{s,a,r,s'}\left[\left(Y - Q\left(s, a|\theta_i\right)\right)^2\right]$ 式关于权重 θ 梯度下降，更新 θ

13： 每隔 C 时间步重置目标网络的权重 $\theta^{-} = \theta$

14： Until 到达终止状态

15：Until episode 数目到达上限次数 M

3.4 基于 DQN 的协作追捕路径生成算法

3.4.1 DQN 抽象

由上文所提到的约束条件可得知，每次需要输入 DQN 模型的环境值的维度共有六维，即追捕者 p_i 与逃跑者 e 的相对位置矢量（2 维）以及与其他盟友追捕者之间的相对位置矢量。而动作则有“上下左右”四种。

即

s = [$x_e(t) - x_{pi}(t)$），$x_{pj}(t) - x_{pi}(t), \mathrm{x}_{pk}(t) - \mathrm{x}_{pi}(t)$]

a = {“u”、“d”、“l”、“r”}

输出值为“上”、“下”、“左”、“右”四个动作的 Q 估计值。智能体将有 ε 的概率执行 Q 值最高的动作，此外随机执行任意动作。

其抽象结构如图 3.1 所示：

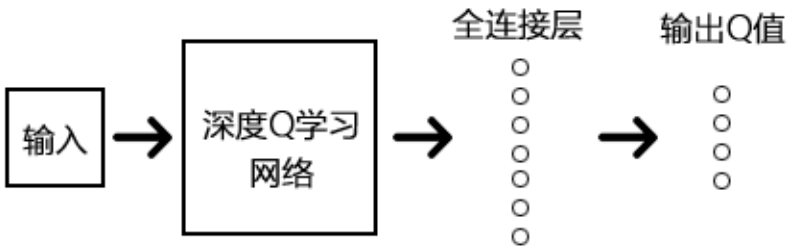

图 3.1 DQN 抽象结构图

3.4.2 关于联盟机器人的处理方法

因为在神经网络训练中使用缺失值会使训练效果出现较大的误差，而本文的约束条件又要求所有追捕者不可获知非盟友追捕者的相对位置。因此在给模型输入非盟友追捕者的相对位置时使用追捕-逃逸环境的长度与宽度取代相对位置矢量，即认为在很远的地方“存在一个盟友”，无法对目前的追捕情况造成客观的影响。

3.4.3 奖励函数

追捕成功	所有追捕者奖励值为 1
追捕者远离了逃逸者	该追捕者奖励值为-1
追捕者靠近了逃逸者	该追捕者奖励值为 1
追捕者“撞墙”	该追捕者奖励为-1

3.5 算法仿真

为验证 DQN 算法在追逃问题中的实用性，需要把算法在尽可能接近实际环境的情境下进行仿真，确定出符合实际情况的参数。然后给出性能评价。

3.5.1 仿真环境

本文的提出的多机器人协作追捕路径生成系统，所使用的环境自然是生成系统所带有的模拟环境。然而在算法研究阶段，生成系统尚未实现完成，为了优化算法模型以及为生成系统的环境开发提供参考，本文在研究过程中采用 TeamBots 作为临时的仿真环境。

TeamBots 是美国卡耐基梅隆大学和佐治亚理工学院共同开发的多机器人仿真平台，它是基于 Java 语言开发的一组多移动机器人实验软件。由于 Java 语言的可移植性，因此该仿真平台可以在 Windows、Linux、Mac OS 等多种操作系统下运行。采用 TeamBots 开发的机器人控制系统，既可以通过软件包中的 TBSim 仿真程序进行仿真，也可以通过 TBHard 执行程序来控制实际的机器人系统。TeamBots 系统的完整功能框架如图 3.2 所示。

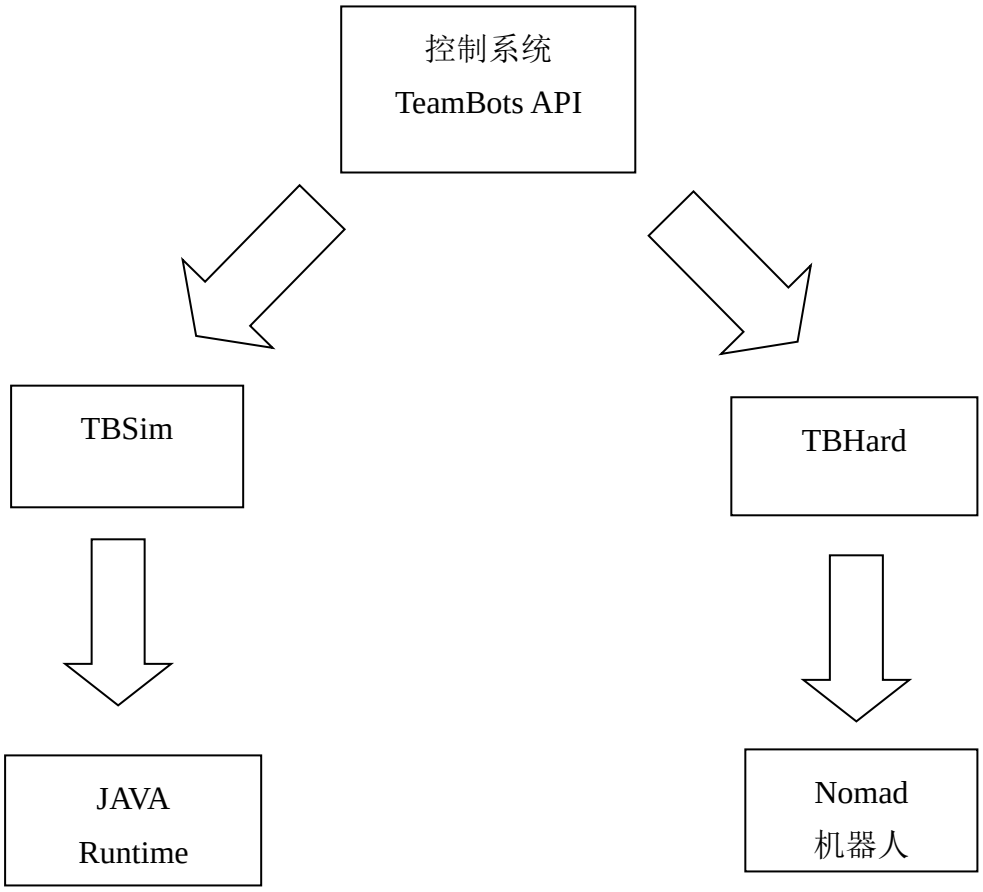

图 3.2 TeamBots 仿真环境

在 TeamBots 仿真平台上，用户可以根据需要编写环境配置文件，来定制环境、包括各种障碍、目标物体、机器人类型等等。另一方面，可以对机器人控制系统进行编程开发，编写自己的控制策略，并通过环境配置文件中的机器人类型，载入仿真平台运行，以检测控制策略的性能。

3.5.2 机器人避障策略

在移动机器人在运动的过程中，既要避开环境中的静态障碍物，又要避免与其他机器人相碰撞。在移动机器人的研究中，最为常用的避障策略是人工势场法。人工势场法的核心思想是在环境中构建一个虚拟的势场，使用势场合力的矢量方向对移动机器人的运动造成影响。比如，让障碍物产生斥力，目标点产生引力，机器人离障碍物越近，斥力越大，反之则斥力越小；机器人离目标点越远，引力越大。机器人总能从势能高的地方向势能低的地方运动，这样，机器人就能在势场的引导下避开障碍物而

移动到目标点。人工势场法应用在移动机器人碰撞中的区域划分原理如图 3.3 所示。

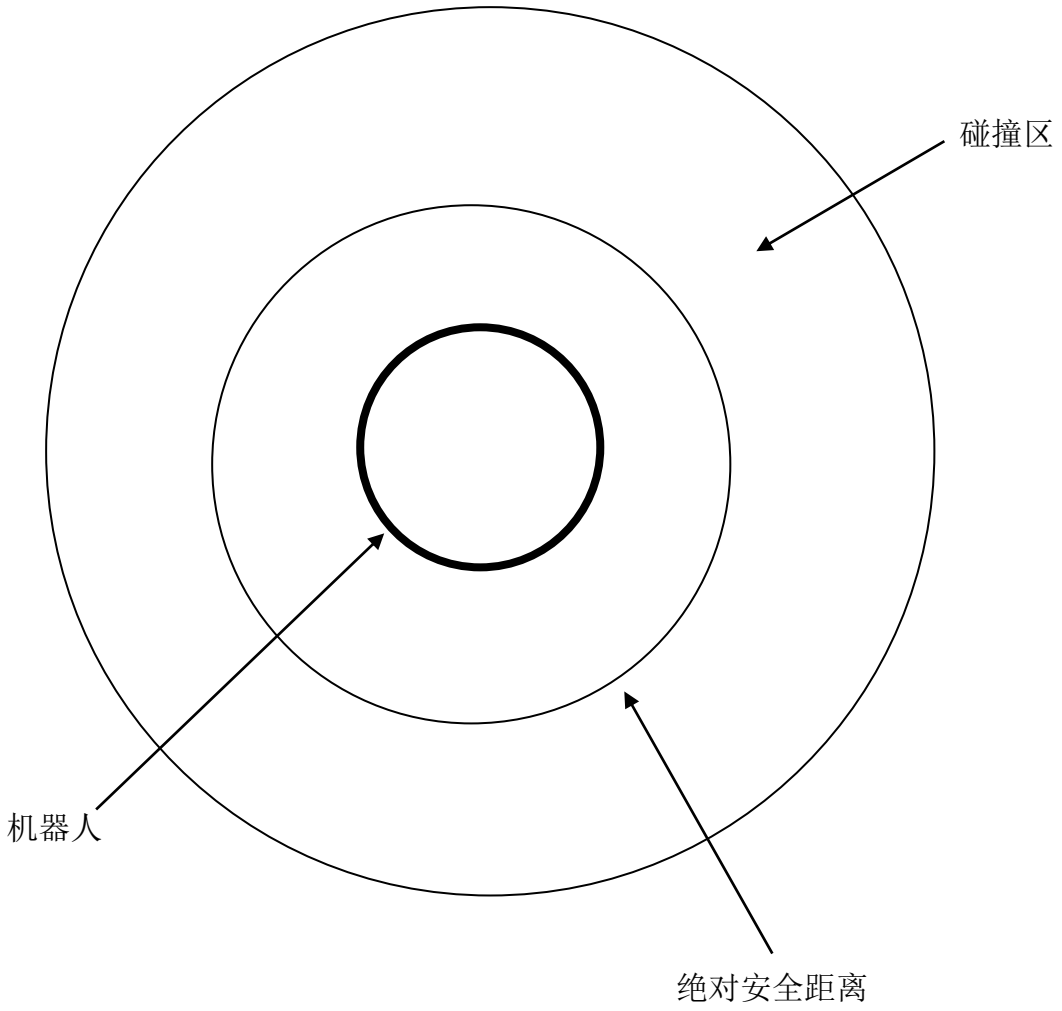

图 3.3 机器人碰撞区域划分

但是，传统的人工势场法存在一些缺陷：

（1） 容易陷入局部最小值。

（2） 当目标点距离障碍物太近时，由于障碍物的斥力原因使得目标点不可达。

（3） 容易出现震荡现象。

在本文设定的围捕任务比较复杂，追捕者要想成功地捕获逃跑者，必须在保持避开障碍物的同时追踪逃跑者。如果追捕机器人将逃跑者看成目标，那么有可能撞到逃跑者，如果看成障碍物，那么刚要靠近目标就被斥力反弹回来。另外，追捕者之间也不能简单的看成障碍物，否则难以形成有效的包围。显然，我们希望机器人之间可以稍微靠得近一点但不能太近，并且在某些距离不能受到互相斥力的影响。因此，追捕机器人之间以及追捕机器人和目标机器人之间也应该存在碰撞的关系，但与障碍物的

性质不同。本文认为可以通过设定不同的碰撞距离和安全距离来尽可能减少干扰的影响。

	碰撞距离	安全距离
障碍	1	0.2
追捕者	0.5	0.01
逃跑者	0.5	0.01

3.5.3 行为综合

追捕机器人在追踪猎物的时候可能会突然发现前方有一个障碍，如果机器人先采取避障行为，有可能完成避障之后发现逃跑者逃出了感知范围。因此，在某一个确定的时刻，机器人不仅要执行主要的学习行为（比如在追捕过程中学习追踪），同时需要增加一些行为来让机器人能应付各种情况，比如增加噪声模拟环境中的噪声等。为解决上述问题，本文采取将各个子行为进行综合，如下图所示。

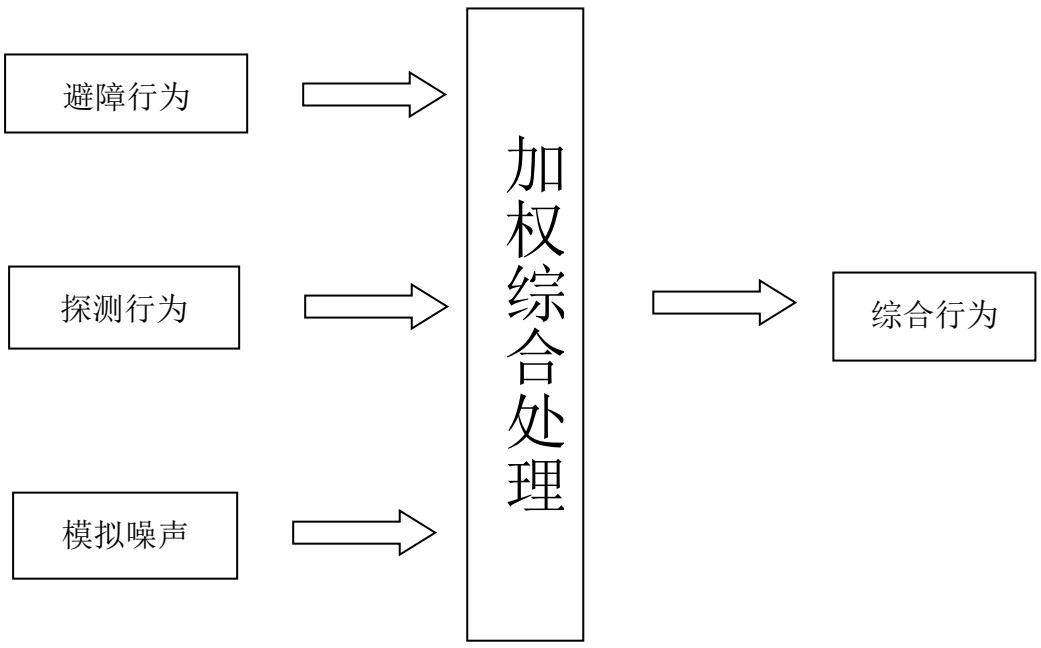

图 3.4 机器人的行为综合过程

行为的综合通过加权处理，假设要将 m 个行为（$a_1, a_2 \ldots \cdots a_m$）进行融合，可设

定向量 $w=(w_1,w_2,\ldots,w_m)$ 且令 $\sum_{j=1}^{m} w_i=1$，则其综合行为 A 可由下式获得：

$$A=a_1w_1+a_2w_2+\ldots+a_mw_m=\Sigma^m(a_iw_i)$$

其中 w_i 称为权值系数，a_i 为机器人需要考虑的第 i 个行为，w_i 的值越大，机器人执行 a_i 的偏好就越大。

第四章 系统结构与代码分析

4.1 系统整体设计

系统设计就是针对开发出的系统如何实现用户所需要的功能需求。也就是说系统设计就是设计师在给定的条件（用户需求）下，设计出满足所提出的约束条件的最佳实践方案。在这个阶段，要根据各种实际环境中存在的因素（社会因素、经济因素、技术因素），确定系统的开发方案，即软件模型。要对系统的整体结构做出宏观的解读，首先对各个模块的具体设计思想做出阐释，然后说明模块设计的数据结构以及具体功能的实现方法。

本文提出的多机器人协作追捕路径生成系统，并不是一个具有复杂增删改查功能的信息系统，因此将系统的逻辑结构抽象成环境、训练模型、用户接口三个模块。环境模块可以根据模拟情况的需要进行封装和替换，以适应对不同情况的模拟与路径生成需要。训练模型模块内部封装有深度 Q 网络的模型以及模型参数的修改接口，在每次实用过程中会使用训练模块对深度 Q 网络进行实例化，并在实例化过程中设置好包括学习速率，贪心概率在内的模型参数。用户接口主要用于生成路径与模型参数的显示和结果的反馈。

系统的整体结构如图 4.1 所示：

图 4.1 系统整体结构图

本章内容主要讨论三个模块的具体功能以及关键代码实现。

4.2 模拟环境模块

本文提出的多机器人协作追捕路径生成系统，模拟的追逃环境为一个 12×12 的栅格空间，依赖于 Python3 自带的 GUI 库——Tkinter 开发，并利用第三方库 numpy 进行与训练模型模块的信息读取和返回的科学计算任务，同时利用 sys 库完成动作时序调节与程序调试任务，模拟环境模块引入的库如下段所示。

```
import numpy as np
import time
import sys

if sys.version_info.major == 2:
    import Tkinter as tk
else:
    import tkinter as tk
```

注：在 Python2.x 中，Tkinter 库的名字叫作 tkinter

模拟环境模块在系统中主要起到了充当追逃模拟环境、执行追捕者动作并反馈结果、执行逃跑者动作并反馈结果，构建可视化 GUI 环境等功能。在项目根目录下创建 env.py 文件，在文件中创建 Env 类，并实现以下函数：

build_env(width, height, initial_position);

reset(self);

step(self, num, action)

do_escape(self)

render()

上述函数分别实现创建环境、重置追逃状态、执行追捕者动作、执行逃逸者动作、刷新追逃状态的功能。

由于Python语言在类与对象方面的特性和机制，一个对象实例在初始化的时候，需要执行对应类中的__init__( )函数，根据功能要求，需要在实例化时确定模拟栅格环境的长度与高度、追捕者与逃逸者动作空间等环境参数，一次本系统用例测试使用的__init__( )如图 4.3 所示。

```
def __init__(self):
    super(Mz, self).__init__()
    self.action_space = ['u', 'd', 'l', 'r']
    self.n_actions = len(self.action_space)
    self.n_features = 9
    self.escapeOrigin = np.array([6, 6])
```

```
self.angentOriginList = [np.array([5, 8]), np.array([8, 4]), np.array([5, 3])]
self.angentList = []
self.title('Pursuit-Escape')
self.geometry('{0}x{1}'.format(MAZE_W * UNIT, (MAZE_H+3) * UNIT))
self._build_maze()
```

初始化成功的环境如图 4.2 所示。

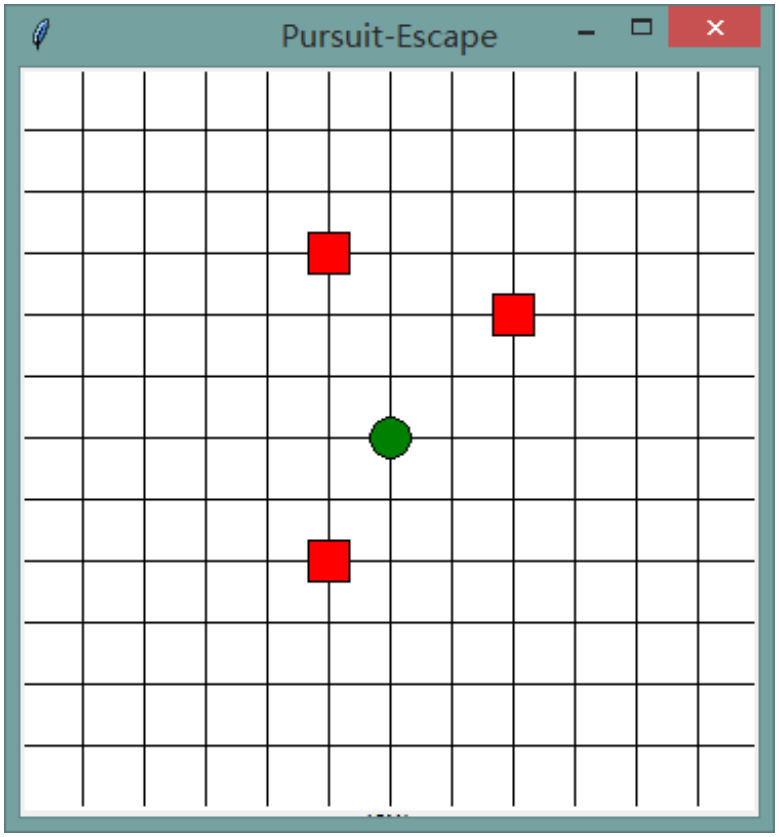

图 4.2 初始化成功的环境

4.3 训练模型模块

本文提出的多机器人协作追捕路径生成系统，所采用的是基于 DQN 算法的神经网络模型，具体算法已经在第三章中做了详细描述，这里只对一些具体实现细节做一定解释。

训练模型模块同样在项目的根目录下创建，命名为 DRL_Brain.py，依赖于 numpy 和 tensorflow 两个第三方库。其中，numpy 主要用于数据的预处理和部分科学计算；tensorflow 主要用于神经网络模型的建立、训练和优化。引入依赖包的相关代码如下

段代码所示:

```
import numpy as np
import tensorflow as tf
np.random.seed(1)
tf.set_random_seed(1)
```

在 DeepQNetwork 类中，__init__( )函数需要包括对 DQN 模型整体参数的确定，以及神经元参数的初始化，包括学习速率，贪心概率，记忆空间容量等，相关程序如下段代码所示：

```
def __init__(
        self,
        n_actions,
        n_features,
        learning_rate=0.01,
        reward_decay=0.9,
        e_greedy=0.9,
        replace_target_iter=300,
        memory_size=500,
        batch_size=32,
        e_greedy_increment=None,
        output_graph=False,
):
    self.n_actions = n_actions
    self.n_features = n_features
    self.lr = learning_rate
    self.gamma = reward_decay
    self.epsilon_max = e_greedy
    self.replace_target_iter = replace_target_iter
    self.memory_size = memory_size
    self.batch_size = batch_size
    self.epsilon_increment = e_greedy_increment
```

```
self.epsilon = 0 if e_greedy_increment is not None else self.epsilon_max

# total learning step
self.learn_step_counter = 0

# initialize zero memory [s, a, r, s_]
self.memory = np.zeros((self.memory_size, n_features * 2 + 2))

# consist of [target_net, evaluate_net]
self._build_net()
t_params = tf.get_collection('target_net_params')
e_params = tf.get_collection('eval_net_params')
self.replace_target_op = [tf.assign(t, e) for t, e in zip(t_params, e_params)]

self.sess = tf.Session()

if output_graph:
    # $ tensorboard --logdir=logs
    # tf.train.SummaryWriter soon be deprecated, use following
    tf.summary.FileWriter("logs/", self.sess.graph)

self.sess.run(tf.global_variables_initializer())
self.cost_his = []
```

另外值得一提的是，假如在这个 batch 中，我们有 2 个提取的记忆，根据每个记忆可以生产 3 个 action 的值:

```
q_eval =
        [[1, 2, 3],
        [4, 5, 6]]
q_target = q_eval =
```

[[1, 2, 3],

[4, 5, 6]]

然后根据 memory 当中的具体 action 位置来修改 q_target 对应 action 上的值，比如在记忆 0 的 q_target 计算值 -1，而且我用了 action 0；记忆 1 的 q_target 计算值是-2，而且我用了 action 2:

q_target =

[[-1, 2, 3],

[4, 5, -2]]

所以 (q_target - q_eval) 就变成了:

[[(-1)-(1), 0, 0],

[0, 0, (-2)-(6)]]

最后我们将这个 (q_target - q_eval) 当成误差，反向传递会神经网络。所有为 0 的 action 值是当时没有选择的 action,，之前有选择的 action 才有不为 0 的值。我们只反向传递之前选择的 action 的值,

4.4 用户接口模块

用户接口模块主要功能是向本系统的用户展示所需要反馈的信息，包括初始状态、生成路径、神经网络相关参数等，以及提供操作接口，让用户可以控制生成路径的过程的开始和停止。

操作界面如下图 4.3 所示:

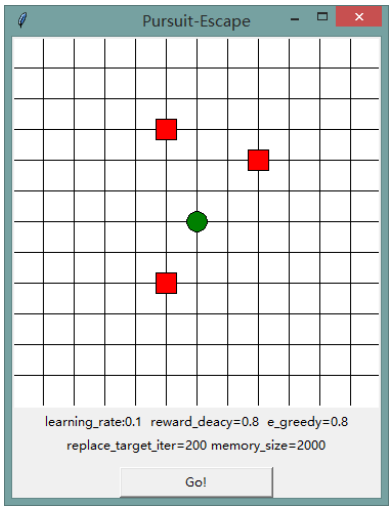

图 4.3 程序界面

第五章 系统测试

5.1 系统测试概述

因为深度强化学习是关于量化数据和决策理论的算法，想要全面覆盖地测试一种深度强化算法是一件相当困难的事情，这其中当然也包括 DQN 算法。由于时间因素，目前本人只能采用有限的数据与状态使用一些简单的方法来测试本系统。

一些高维数据的最优决策结果很难用直接的方法得出，所以这里会降低对于最优性的测试，转而更多的聚焦于对于计算模型收敛性、临时垃圾回收、计算效率等方面的测试。

5.2 测试方法介绍与调整

纯度计算测试方法，即 Purity 测试方法。原理十分简单，即运算正确决策的动作次数占到总尝试次数集的比重，输出一个 0 到 1 的数值，数值越高越好。其中 Ω（ω）为深度强化学习算法得到的结果，C（c）表示在该种初始情况实际上应该得到的决策结果。

由此易知，P 代表精确度，表示本系统所得到的最优协作追捕路径与实际最优路径的重合比率；R 代表召回率，为正确决策在每一次动作选择中被选择的比率。最后得出结果越大，代表算法决策越优秀。

在本文中，由于 P 与 R 都难以求的，则根据深度强化学习算法的损失函数以及损失函数达到收敛状态所需的步骤数作为测试依据。

5.3 测试数据

Python3 拥有 matplotlib 这一优秀的数据绘图工具，开发者可以凭借这个工具轻易地生成绘图，直方图，功率谱，条形图，错误图，散点图等图像。用户可在模型中即时自动记录下每一次迭代的 loss 值，并在退出迭代后绘制出损失函数图像，当在大部分情况下损失函数都可以收敛后，则认为本系统可行。模型的收敛速度也是评价算法的重要依据。

本系统在某次测试中的损失函数如图 5.1（下页）所示

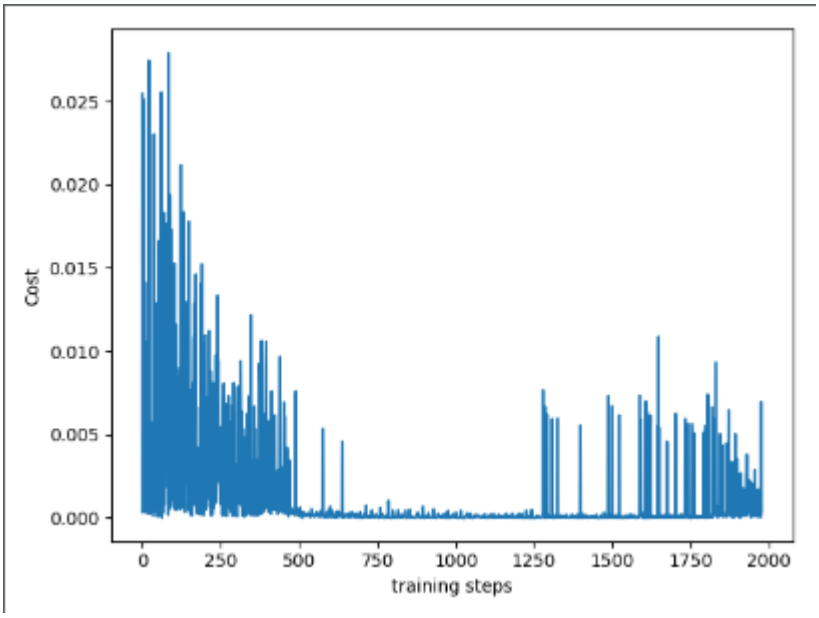

图 5.1 cost 函数

完整的测试用例数据如下表所示：

（注：加粗的追捕者表示）

序号	逃逸者坐标	追捕者坐标	能否收敛	步骤数
1	（1，3）	（1，1）（5，6）（7，7）	能	213
2	（2，8）	（6，3）（3，3）（6，8）	能	426
3	（4，5）	（4，3）（5，6）（6，9）	能	306
4	（5，6）	（2，2）（3，9）（10，10）	能	501
5	（3，3）	（4，3）（1，8）（6，6）	不能	-
6	（3，7）	（9，5）（5，9）（6，5）	能	493
7	（8，9）	（6，8）（7，9）（11，12）	能	606
8	（10，11）	（1，1）（5，6）（12，12）	能	358
9	（6，6）	（4，8）（8，9）（12，4）	不能	-
10	（8，6）	（1，2）（5，6）（9，8）	能	621

第六章 总 结

6.1 现有成果总结

至此，整篇论文即将结束。 可以说本系统的开发与相关算法的设计基本满足了指导老师之前所提出的要求。基于深度强化学习的协作追捕路径生成算法是目前多机器人追捕-逃逸问题中的一个新兴的解决思路，多种学者与相关研究人员对此的讨论层出不穷。本次论文所提出的多机器人协作追捕路径生成系统相比其他类似的软件系统有以下几个优点：

1、使用了基于深度强化学习的 DQN 的算法模型，极大的解决了高维数据所带来的高维灾难；

2、利用开源语言 Python 作为开发语言，所用的依赖库也是在开源界赫赫有名的 numpy 和 tensorflow，系统作品完全开源，程序代码可读性良好，适合其他的研究者在此工作基础上进行二次开发；

3、用户界面简洁直观；

4、系统设计思想简单易懂。

当然还是不可避免的有以下缺点：

1、依然没有摆脱对参数的依赖，DQN 本质上也是一种神经网络，参数数量很多，在学习速率、记忆空间容量、贪心概率等学习参数上依然离不开人工手动调参来使模型达到最大效率；

2、在维度较低，环境状态空间较小的情况下效率低于 Q 学习算法和策略梯度算法等经典的强化学习方法；

3、对于一些超高维数据（数百维以上），依然为因为模型效率的不足而显得无能为力难以适应。

但是总体来说，我本人对于我此次的毕业设计的完成情况感到十分的满意，在这半年的时间里，我接触到了很多在以往的课堂上难以遇到的情况，阅读了不少高水平的强化学习与深度学习领域的学术文献，也在尝试用自己已经掌握的知识解决高深的问题，我本人对自己在此次毕业设计中付出的努力与获得的成果感到自豪与满满的成就感。

6.2 未来展望

本次算法的设计、系统的开发以及论文的撰写工作，由于时间问题、知识储备

等等多方面的原因，还有一定工作未能完成，我将会在未来条件成熟之后继续进行，加以完善多机器人协作追捕路径生成系统。

1、由于知识水平所限，只对开发完成的系统进行了算法的收敛性测试，未能加入其他对精准度和可用性的测试。

2、测试中没有加入与其他算法的比较，不能直观展现出各算法之间的优劣；

总之，本次毕业论文远不是终点，想要继续完善本算法理论、提高本算法性能，还需要进行大量的工作，在将来离开母校的时光里，我还会投入大量精力在这个领域之中。

致 谢

首先要感谢我的指导老师朱老师，从选题到研究过程都给予了我很大的帮助。让我明白了测试方法、研究他人算法的思路以及论文撰写的注意点。在治学方面朱老师的精神值得我深刻学习。

同时，我也要感谢身边的同学、舍友和朋友。在四年学习的时间中大家互相促进，互相帮助，你们是我大学生涯中不可磨灭记忆。

当然，我也要感谢父母的支持，不论是对我生活的关心还是对我学习的支持，他们都做了很多，大学四年学业上的顺利离不开他们的帮助。

转眼即要毕业了，进入扬州大学是一种缘分，感谢扬大给予我四年充实的学习生活。扬大的老师们让我对这个专业有了新的认识，在此向他们致敬！

参考文献

[1] 韩韬，郝矿荣，丁永生，唐雪嵩. 基于深度 LSTM 神经网络的人体服装压力信息预测，2016.

[2] 王艳.，多机器人追捕-逃跑问题对策模型的研究，2014.

[3] Sutton S. Learning to predict by the methods of temporal difference [J]. Machine Learning, 1988 (3): 9-44.

[4] 张旭，贾磊磊，陈群．关于多机器人围捕协调路径策略研究.2016.

[5] 刘全，翟建伟，章宗长，钟珊，周倩，章鹏，徐进. 深度强化学习综述，2018.

[6] Watkins C. Q-Learning [J]. Meachine Learning, 1992,8 (3): 279-292.

[7] 黎育权. 基于集成学习改进的卷积神经网络的手写字符识别，2018.

[8] Sutton R S.Barto A G.Reinforcement learning: An Introduction[M]. MIT Press, Canbridge, MA, 1998.

[9] S P Singh. Transfer of learning by composing solutions of elemental sequential tasks. Machine Learning,8(3),May 1992.

[10] Drew Fudenberg，Jean Tirole．博弈论．中国人民大学出版社，2010.

[11] Martin Ester，Hans-Peter Kriegel. A Density-Based Algorithm for Discovering Clusters in Large Spatial Database with Noise. Proceeding of 2nd International Conference on Knowledge Discovery and Data Mining. 1996，KDD-96.

[12] Howard M.Schwartz. 多智能体机器学习：强化学习方法 第一版. 机械工业出版社，2017.

[13] G J Teausro. TD-gammon, self-teaching backgammon program, achieves master-level play [J]. Neural computation,1994,6(2): 215-219.

[14] Balch. Integrating RL and Behavior-based Control for Soccer[C]. In:IJCAI Workshop on Robocup, Nagoya, Japan, 1997.

对抗环境下多机器人协作追捕路径生成系统

【PDF报告-大学生版】

报告编号：54d7d7d0d580a4b2　　检测时间：2018-05-27 04:57:57　　检测字数：15,941字

作者名称：颜文豪　　所属单位：

检测范围：

◎ 中文科技期刊论文全文数据库
◎ 中文主要报纸全文数据库
◎ 中国专利特色数据库
◎ 博士/硕士学位论文全文数据库
◎ 中国主要会议论文特色数据库
◎ 港澳台文献资源
◎ 外文特色文献数据全库
◎ 维普优先出版论文全文数据库
◎ 互联网数据资源/互联网文档资源
◎ 高校自建资源库
◎ 图书资源
◎ 古籍文献资源
◎ 个人自建资源库
◎ 年鉴资源
◎ IPUB原创作品

时间范围：1989-01-01至2018-05-12

检测结论：

全文总相似比：22.67%　（总相似比=复写率+他引率+自引率）

自写率：77.33%　（原创内容占全文的比重）

复写率：22.67%　（相似或疑似重复内容占全文的比重,含专业用语）

他引率：0%　（引用他人的部分占全文的比重，请正确标注引用）

自引率：0%　（引用自己已发表部分占全文的比重，请正确标注引用）

专业用语：0.00%　（公式定理、法律条文、行业用语等占全文的比重）

总相似片段：115

期刊：15 博硕：49 外文：0 综合：2 自建库：0 互联网：49
