**学号 19029216**

**北京工业大学信息学部**

**《机器人系统仿真》**

**课程论文**

**中文题目**

**基于多种深度学习算法的机器人导航仿真研究**

**英文题目**

**Robot navigation simulation research based on multiple deep learning algorithms**

**学生姓名 张亦非**

**班 级 190292**

**成 绩**

**2022年 6 月 7 日**

# 摘要

移动机器人技术目前得到了广泛应用，对于应用在各个领域移动机器人来说，导航控制都是其一项非常基本和关键的技术。移动机器人导航研究的一个基本目标就是，希望建立一套可以在复杂环境下进行自主推理、规划和控制的移动机器人系统。

目前室内导航移动机器人主要采用基于已知室内环境模型的导航方法，该类方法导航控制的精确度取决于对导航环境所建立模型的精确度，如果环境模型和实际环境存在较大的误差会导致在导航过程中出现无法估计的影响因素，而且对环境建模需要耗费额外的资源。

传统的路径规划算法应用到未知复杂的环境时,要寻到一条全程无碰撞的路径是很困难的。而深度强化学习可以使智能体在探索环境的同时,学习到相关经验、避障能力以及趋向目标点的能力,使机器人通过不断“试错”的方式,获得一条最优路径。

因此，本小组尝试开展了基于多种机器学习算法的路径规划算法研究问题,论文主要研究如下:

在ROS和Gazebo平台搭建基于fira平台的三维有障碍环境,通过Turtle Bot移动机器人平台进行仿真实验,从实验计时结果分析，实现基于多种算法模型能稳定且智能体能学习到目标趋向能力和避障能力,能有效完成路径规划任务。

**关键词：**机器学习；避障；ROS；Gazebo；仿真

# Abstract

 At present, mobile robot technology has been widely used. Navigation control is a very basic and key technology for mobile robots in various fields.  A basic goal of mobile robot navigation research is to build a mobile robot system that can carry out autonomous inference, planning and control in complex environment.

At present indoor mobile robot navigation is mainly based on the known model of indoor environment navigation methods, the accuracy of the navigation control method depends on the accuracy of the established model of navigation environment, if the environment model and the actual environment can lead to large error exists in the process of navigation appear unable to estimate the influence of factors, and modeling requires additional resources to the environment.

 It is difficult to find a collision-free path when the traditional path planning algorithm is applied to the unknown and complex environment.  Deep reinforcement learning enables the intelligent body to learn relevant experience, obstacle avoidance ability and the ability to approach the target point while exploring the environment, so that the robot can obtain an optimal path through continuous "trial and error".

 Therefore, this group tried to carry out path planning algorithm research based on a variety of machine learning algorithms. The main research of this paper is as follows:

 Based on ROS and Gazebo platform, a 3D obstacle environment based on FIRA platform was built. Simulation experiments were carried out by Turtle Bot mobile robot platform. From the analysis of experimental timing results, the intelligent body based on a variety of algorithm models can be stable and learn the ability to target orientation and obstacle avoidance, which can effectively complete the path planning task.

**Key words:** Machine learning; Obstacle avoidance;  ROS;Gazebo；The simulation

目录

[摘要 2](#_Toc29711)

[Abstract 3](#_Toc21553)

[1. 绪论 7](#_Toc969)

[1.1 研究的背景和意义 7](#_Toc25977)

[1.2 导航方法现状 7](#_Toc3505)

[1.3 基于深度学习的导航方法研究现状 8](#_Toc4383)

[1.4 本文主要工作 8](#_Toc9921)

[2. 机器学习基础 8](#_Toc2444)

[2.1 监督学习算法 8](#_Toc11205)

[2.1.1 BP神经网络 8](#_Toc1083)

[2.1.2 支持向量机 9](#_Toc8363)

[2.1.3 最近邻居法 9](#_Toc28754)

[2.1.4 朴素贝叶斯法 9](#_Toc3879)

[2.1.5 决策树 10](#_Toc21308)

[2.1.6 随机森林 11](#_Toc1908)

[2.2 强化学习算法 11](#_Toc725)

[2.2.1 DQN算法 11](#_Toc15432)

[2.2.2 DDPG算法 14](#_Toc29217)

[2.3 深度学习基础 15](#_Toc15148)

[2.3.1 深度神经网络 15](#_Toc11442)

[2.3.2 卷积神经网络 18](#_Toc3293)

[2.4 强化学习基础 21](#_Toc15896)

[2.4.1 强化学习原理 21](#_Toc9565)

[2.4.2 马尔科夫决策过程 22](#_Toc28190)

[2.4.3 蒙特卡洛算法 23](#_Toc2521)

[2.4.4 时间差分算法 25](#_Toc16287)

[2.4.5 深度强化学习 28](#_Toc668)

[2.5 本章小结 31](#_Toc20267)

[3. 机器人导航 3D 仿真系统搭建 32](#_Toc8601)

[3.1 3D仿真场景搭建 32](#_Toc22505)

[3.1.1 Gazebo 编辑器简介 32](#_Toc27229)

[3.1.2 基于fira平台导航场景简化建模 32](#_Toc6216)

[3.2 仿真机器人搭建 33](#_Toc3679)

[3.2.1 仿真 TurtleBot 机器人搭建 33](#_Toc23939)

[3.2.2 TurtleBot 机器人仿真效果 33](#_Toc2316)

[3.3 场景和机器人的联合仿真 34](#_Toc29703)

[4. 机器学习模型实现 34](#_Toc258)

[4.1 视觉特征提取 34](#_Toc27703)

[4.1.1 图像获取 34](#_Toc16473)

[4.1.2 图像处理 35](#_Toc24100)

[4.1.3 获取特征 38](#_Toc25572)

[4.2 样本集制作 39](#_Toc27697)

[4.3 机器学习算法实现与测试 41](#_Toc8659)

[4.3.1 决策树 41](#_Toc962)

[4.3.2 随机森林 42](#_Toc18979)

[4.3.3 KNN算法 42](#_Toc5216)

[4.3.4 贝叶斯分类器 44](#_Toc30648)

[4.3.5 投票算法 45](#_Toc30967)

[4.3.6 支持向量机 47](#_Toc24340)

[4.3.7 bp神经网络 48](#_Toc27679)

[4.3.8 DDPG算法 51](#_Toc20491)

[4.4 算法测试与对比 52](#_Toc26507)

[4.4.1 决策树与随机森林算法准确度对比 52](#_Toc27117)

[4.4.2 Bp神经网络测试结果 53](#_Toc4117)

[5. gazebo仿真测试 53](#_Toc15848)

[5.1 测试方案 53](#_Toc6909)

[5.2 测试结果 53](#_Toc14785)

[5.3 测试结果分析 55](#_Toc15961)

# 绪论

## 研究的背景和意义

移动机器人技术目前得到了广泛应用，对于应用在各个领域移动机器人来说，导航控制都是其一项非常基本和关键的技术。移动机器人导航研究的一个基本目标就是，希望建立一套可以在复杂环境下进行自主推理、规划和控制的移动机器人系统。

基于对环境的信息的掌握程度将传统的导航方法分为两类：一种是环境信息已知的全局路径规划导航方法，即导航的环境、机器人初始位姿及导航目标都已知，该类方法有 Dijkstra 算法、人工势场法、PRM 随机路图算法和 SLAM 技术等，利用 SLAM 技术进行导航需要先在初始阶段构建机器人所处环境的地图，然后基于该地图对机器人位置进行推断，进而进行路径规划导航；另一种是用于部分或完全未知动态环境的导航，像 A\*搜索算法、D\*算法以及快速扩展随机树法 RRT 等。

目前室内导航移动机器人主要采用基于已知室内环境模型的导航方法，该类

方法导航控制的精确度取决于对导航环境所建立模型的精确度，如果环境模型和

实际环境存在较大的误差会导致在导航过程中出现无法估计的影响因素，而且对

环境建模需要耗费额外的资源。机器人导航的环境可能是比较复杂、动态多变(各

种动态因素)甚至是未知的，以及机器人所需承担的导航任务日趋复杂，综合以

上各种因素，对移动机器人的自主性提出了更高的要求，越来越需要机器人有自

主学习的能力。因此研究对环境模型依赖程度低或免模型的、能通过自主学习适

应室内复杂环境的导航方法成为室内移动机器人导航研究的必然趋势。

## 导航方法现状

目前室内移动机器人导航方法中存在严重依赖环境模型和缺乏自主性的问题。

室内导航移动机器人主要采用基于已知室内环境模型的导航方法，该类方法导航控制的精确度取决于对导航环境所建立模型的精确度，如果环境模型和实际环境存在较大的误差会导致在导航过程中出现无法估计的影响因素，而且对环境建模需要耗费额外的资源。

## 基于深度学习的导航方法研究现状

深度学习中的卷积神经网络(Convolutional neural networks，CNN)在视觉图像

任务中的成功，促使将深度学习用于解决机器人视觉导航问题。

由于深度学习有较强的感知能力，所以能较好地加工和处理图像等状态信息输入，且能对各类导航任务有较好的适应性，但是其决策需要在其他算法监督下进行，所以深度学习的决策能力较弱。

## 本文主要工作

针对目前室内移动机器人导航方法中存在严重依赖环境模型和缺乏自主性的问题，本文应用多种深度学习算法训练机器人自主学习在室内环境中动静态避障和趋向任意给定目标点。并在fira平台上进行仿真测试，通过计时的方式量化训练模型的优劣。

# 机器学习基础

## 监督学习算法

### BP神经网络

BP神经网络由输入层、隐含层、输出层组成。

BP神经网络的计算过程由正向计算过程和反向计算过程组成。

正向传播过程，输入模式从输入层经隐单元层逐层处理，并转向输出层，每一层神经元的状态只影响下一层神经元的状态。如果在输出层不能得到期望的输出，则转入反向传播，将误差信号沿原来的连接通路返回，通过修改各神经元的权值，使得误差信号最小。

### 支持向量机

支持向量机（support vector machines, SVM）是一种二分类模型，它的基本模型是定义在特征空间上的间隔最大的线性分类器，间隔最大使它有别于感知机；SVM还包括核技巧，这使它成为实质上的非线性分类器。SVM的的学习策略就是间隔最大化，可形式化为一个求解凸二次规划的问题，也等价于正则化的合页损失函数的最小化问题。SVM的的学习算法就是求解凸二次规划的最优化算法。

### 最近邻居法

KNN算法是一种基于实例的学习，或者是局部近似和将所有计算推迟到分类之后的惰性学习。用最近的邻居（k）来预测未知数据点。k 值是预测精度的一个关键因素，无论是分类还是回归，衡量邻居的权重都非常有用，较近邻居的权重比较远邻居的权重大。

KNN 算法的缺点是对数据的局部结构非常敏感。计算量大，需要对数据进行规范化处理，使每个数据点都在相同的范围。

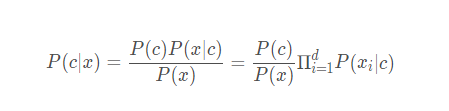
### 朴素贝叶斯法

朴素贝叶斯算法（Naive Bayes）基于概率论的贝叶斯定理，应用非常广泛，从文本分类、垃圾邮件过滤器、医疗诊断等等。朴素贝叶斯适用于特征之间的相互独立的场景，例如利用花瓣的长度和宽度来预测花的类型。“朴素”的内涵可以理解为特征和特征之间独立性强。

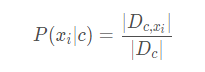
与朴素贝叶斯算法密切相关的一个概念是最大似然估计(Maximum likelihood estimation)，历史上大部分的最大似然估计理论也都是在贝叶斯统计中得到大发展。例如，建立人口身高模型，很难有人力与物力去统计全国每个人的身高，但是可以通过采样，获取部分人的身高，然后通过最大似然估计来获取分布的均值与方差。

根据朴素贝叶斯公式设计朴素贝叶斯分类器

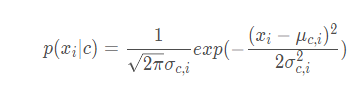
采用属性条件独立性假设。公式表达式：



在估计条件概率 时，若xi为离散属性值，计算每个属性取值占所有样本的数量比例：



若xi是连续属性值，计算概率密度函数：



### 决策树

决策树（Decision tree）是一种特殊的树结构，由一个决策图和可能的结果（例如成本和风险）组成，用来辅助决策。机器学习中，决策树是一个预测模型，树中每个节点表示某个对象，而每个分叉路径则代表某个可能的属性值，而每个叶节点则对应从根节点到该叶节点所经历的路径所表示的对象的值。决策树仅有单一输出，通常该算法用于解决分类问题。

一个决策树包含三种类型的节点：

决策节点：通常用矩形框来表示；

机会节点：通常用圆圈来表示；

终结点：通常用三角形来表示；

### 随机森林

将多个决策树结合在一起，每次数据集是随机有放回的选出，同时随机选出部分特征作为输入，所以该算法被称为随机森林算法。可以看到随机森林算法是以决策树为估计器的Bagging算法。

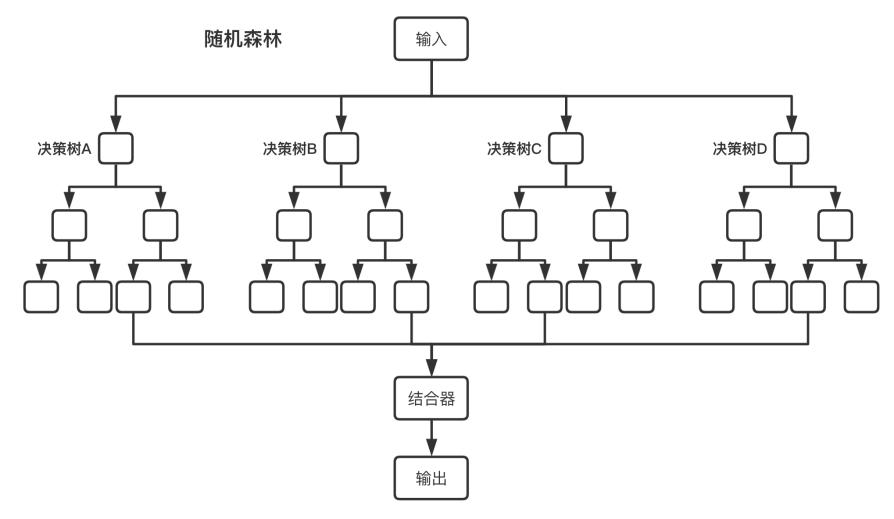


图2-1展示了随机森林算法的具体流程，其中结合器在分类问题中，选择多数分类结果作为最后的结果，在回归问题中，对多个回归结果取平均值作为最后的结果。

  使用Bagging算法能降低过拟合的情况，从而带来了更好的性能。单个决策树对训练集的噪声非常敏感，但通过Bagging算法降低了训练出的多颗决策树之间关联性，有效缓解了上述问题。

## 强化学习算法

### DQN算法

#### 2.2.1.1 强化学习

强化学习是智能体以“试错”的方式进行学习，通过与环境进行交互获得的奖赏指导行为，目标是使智能体获得最大的奖赏，是对产生动作的好坏作一种评价。由于外部环境提供的信息很少，RLS必须靠自身的经历进行学习。通过这种方式，RLS在行动-评价的环境中获得知识，改进行动方案以适应环境。

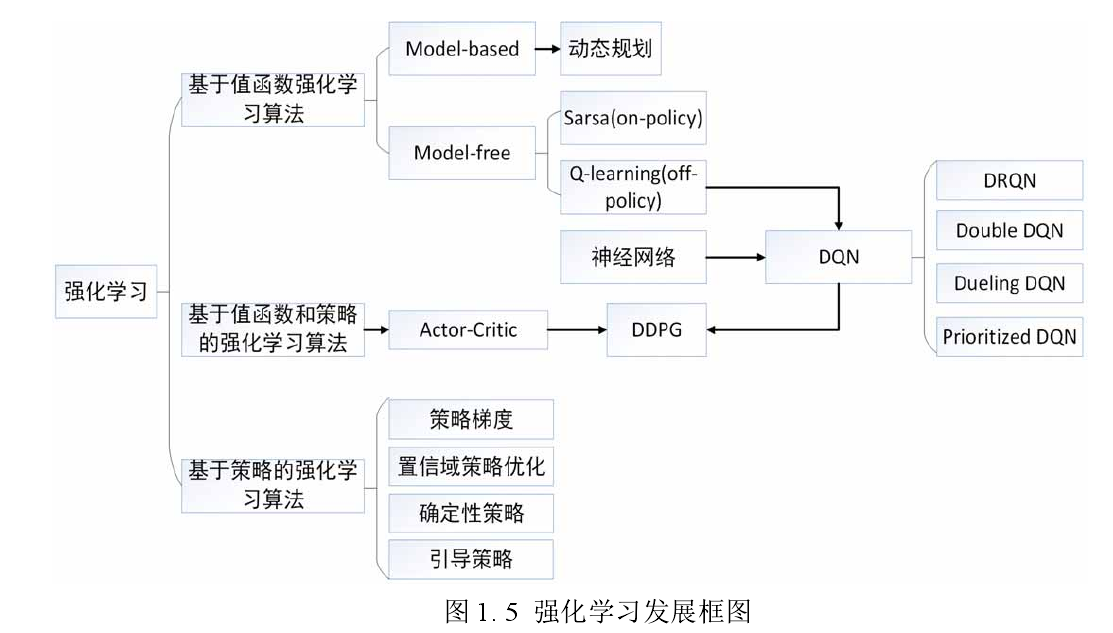


图2.2.1.1强化学习发展框图

#### 2.2.1.2Q-Learning

Q-Learning是强化学习算法中value-based的算法，Q即为Q（s，a），就是在某一个时刻的s状态下，采取动作a能够获得收益的期望，环境会根据智能体的动作反馈相应的reward奖赏，所以算法的主要思想就是将state和action构建成一张Q\_table表来存储Q值，然后根据Q值来选取能够获得最大收益的动作。

#### 2.2.1.3基于DQN的移动机器人导航方法

深度强化学习利用深度神经网络近似值函数或状态分布策略，大大提高了强化学习的适用性、自主性和学习效率，随着深度强化学习在视频游戏与围棋领域的出色表现，更多的研究者开始了对深度强化学习的研究与应用推广，不仅仅在机器控制领域，在计算机视觉[49]、自然语言处理[50]都有广泛的应用。随着智能机器人的发展，深度强化学习也被越来越多的应用到移动机器人自主导航中。T XTung 等人[51]提出了一种基于深度强化学习算法的移动服务机器人社会感知导航框架。通过激光与摄像头获取障碍物与人物状态信息，再通过奖励值的设定，将社会互动与社会规则构建到深度强化学习中，然后将移动机器人分配到一个动态的社会环境中，让移动机器人与周围人或物进行社会式互动并获得的经验，自动学习并融入环境。学习阶段完成时，移动机器人能够在社会环境中进行自主导航，通过仿真与碰撞曲线的展示，确保了移动机器人能够在保证人类安全舒适的同时进行社会可接受的行为。S H Han 等人[52]提出了一种基于深度强化学习的移动机器人导航模型，利用两种先进的 Q-learning 技术进一步完善了模型，提高了模型的性能。选取 7 个激光传感器作为数据收集，收集场景数据信息与目标坐标信息，在每一轮的训练中移动机器人的初始位置与目标的初始位置都会发成变化，在完成导航任务时，还添加了步长的设定，用来促使移动机器人在尽量短的时间内到达目标点。

在深度强化学习的训练过程中，需要移动机器人在地图中的大量探索，与自己的不断试错来更新自己，这样就会浪费大量的训练时间，如何提高移动机器人在导航中的训练速度与稳定性也是关键的一点，M Pfeiffer 等人[53]将模仿学习与强化学习相结合，利用了专家经验来训练导航策略，并用该策略来初始化强化学习的策略，从而大减少了探索所用的时间。L Xie 等人[54]将深度强化学习与现有的控制器紧密结合在一起，既利用了确定性梯度算法与深度 Q 网络的能动性的优点，又能通过控制器减少前期大量的探索所消耗的时间，通过大量的方法验证了该方法可以有效的加速和稳定移动机器人在环境中的训练。但是要经过大量轮次的训练后才能学到独立的策略，放弃对外部控制器的依赖。J Wu 等人[55]通过修改目标网络权值的更新次数，从而减少训练所需要的时间，提出了一种高效的深度Q 网络的训练算法，通过比较目标网络与实时网络的差异大小来决定更新目标网络权值的次数，从而在一定程度上减少来了训练时间。

目前，将深度强化学习应用在移动机器人导航方面，克服了强化学习的状态空间受限和维度灾难等问题，提高了移动机器人在复杂环境中的自学习与自适应能力，但面临着算法稳定性差、算法收敛速度慢、真实训练环境下成本高昂以及如何平衡探索与利用等问题，仍然需要研究与完善。针对存在的问题，本文做出了相应的改进，根据导航任务定义了一种距离奖励值，移动机器人的在每次执行完动作后，都会根据与目标点的距离信息反馈具体的奖励值，并通过大量训练确定了 DDQN 算法的网络结构与神经元数目，保证了算法在导航任务中的稳定性。并设计了定向探索策略，提高了移动机器人在前期探索的导向性，尽可能获取高奖励值样本，减少了探索环境所浪费的时间，提高了训练效率。在探索环境遇到障碍物时，利用玻尔兹曼探索策略进行试错学习，能够更为充分的学习到环境信息，避免陷入局部最优问题的同时，优化了规划路径。

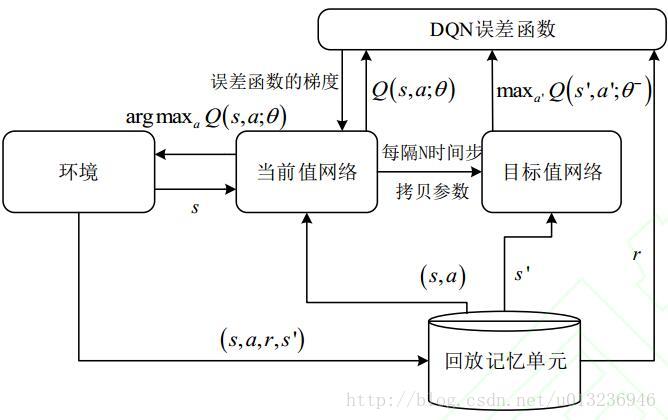


图2.2.1.3DQN算法逻辑框图

### DDPG算法

DDPG全称为Deep Deterministic Policy Gradient

Deep：首先Deep我们都知道，就是更深层次的网络结构，我们之前在DQN中使用两个网络与经验池的结构，在DDPG中就应用了这种思想。

PolicyGradient：顾名思义就是策略梯度算法，能够在连续的动作空间根据所学习到的策略（动作分布）随机筛选动作。

Deterministic : 它的作用就是用来帮助Policy Gradient不让他随机选择，只输出一个动作值。

随机性策略，∑ π ( a ∣ s ) = 1 \sum\pi(a|s)=1∑π(a∣s)=1 策略输出的是动作的概率，使用正态分布对动作进行采样选择，即每个动作都有概率被选到；优点，将探索和改进集成到一个策略中；缺点，需要大量训练数据。

确定性策略，π ( s ) S → A \pi(s) S→Aπ(s)S→A 策略输出即是动作；优点，需要采样的数据少，算法效率高；缺点，无法探索环境。然而因为我们引用了DQN的结构利用offPolicy采样，这样就解决了无法探索环境的问题。

从DDPG网络整体上来说：他应用了 Actor-Critic 形式的, 所以也具备策略 Policy 的神经网络 和基于 价值 Value 的神经网络，因为引入了DQN的思想，每种神经网络我们都需要再细分为两个, Policy Gradient 这边，我们有估计网络和现实网络，估计网络用来输出实时的动作, 供 actor 在现实中实行，而现实网络则是用来更新价值网络系统的。再看另一侧价值网络, 我们也有现实网络和估计网络, 他们都在输出这个状态的价值, 而输入端却有不同, 状态现实网络这边会拿着从动作现实网络来的动作加上状态的观测值加以分析，而状态估计网络则是拿着当时 Actor 施加的动作当做输入。

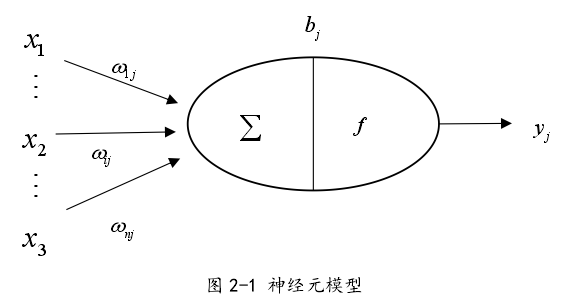
DDPG 在连续动作空间的任务中效果优于DQN而且收敛速度更快，但是不适用于随机环境问题。

## 深度学习基础

### 深度神经网络

近年来，深度学习的应用越来越多的出现在人们的生活中，如人脸识别、语言文本的转换、视频分类等。深度学习的概念是由多伦多大学的 Hinton 等人[56]在 2006 年首次提出，深度学习通过模仿人类的大脑神经结构，建立人工神经网络，对数据进行表征学习，并利用多层神经网络对数据进行层层处理，能够将最原始的输入转换成更高、更抽象的高层的特征表示。深度学习所得到的深度网络结构包含大量单一神经元，神经元之间互相连接，连接强度由权值参数表现，通过在后期的学习中修改权值参数，决定该网络的功能。根据网络结构，神经网络可以分为三类：前馈神经网络，如多层感知机[57]，卷积神经网络[58]；反馈神经网络，如反卷积神经网络[59]；双向深度网络，如深度玻尔兹曼机[60]，深度信念网络等[56]。在本小节中将对应用比较广泛的深度神经网络与卷积网络进行介绍。

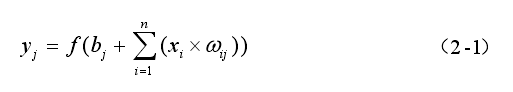
#### （1）神经元



深度学习通过建立的人工神经网络对数据进行处理与学习，而神经元作为人工神经网络中的最基本的处理单位，由输入层和输出层组成，如图 2-1 所示。

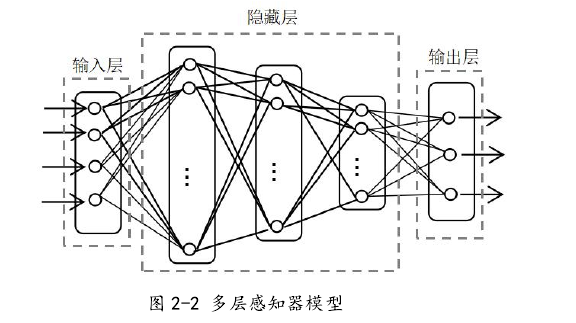
其中ix (i=1...n)分别表示输入信号的个数，ij 则是神经元i 与j 之间的连接强度，即权重参数，jb 为神经元的偏置值，jy 为该神经元的输出值。

在该神经元中，通过调整权重参数的大小决定输入量的重要程度，偏置主要是对权重调整后的输入量添加一个线性分量，除上述参数外，神经元还需要有激活函数，该激活函数为非线性函数，将输入神经元的线性分量转化为非线性信号输出，通常激活函数根据应用情况可由多种选择，如 Sigmoid 函数、Tanh 函数、ReLU 函数等。最后通过这三组参数得出输入与输出的关系式为

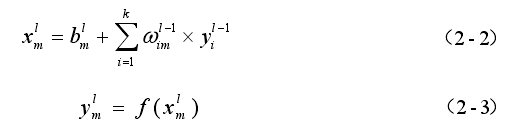


其中， f 为激活函数。

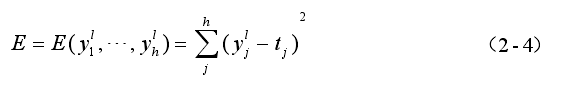
#### （2）多层感知器

深度学习在对数据进行表征学习时往往需要多层网络进行学习，所构建的神经网络模型通常指的是多层感知器(Multi-Layer Perceptron)，多层感知器是由输入层、一层或多层隐藏层与输出层组成的神经网络模型。它解决了单层感知器所存在的线性不可分问题，增强了表达能力，增加了输出层的输出结果，使得网络的应用更加灵活。

在图 2-2 中，该多层感知器模型中，该多层感知器由接收 4 输入信号的输入层，3 隐藏层和输出 3 个结果的输出层组成，层与层之间为全连接，前一层的输出会作为输入传入下一层。则对应每层的神经元输入与输出的关系式为



其中，lxm表示该网络 l 层中的第 m 个神经元的输入值，lmy 和lmb 分别表示表示该神经元的输出值和偏置参数，l1im表示前一层神经网络中第 i 个神经元与该神经元的连接权值，f为激活函数。对于神经网络的训练通常使用反向传播算法[61](backpropagation algorithm，BP)进行网络参数的更新，该算法的核心理念是通过梯度下降来降低网络的误差函数。该算法分为两步执行，分别为前向传播和反向传播两步，前向传播，则是信号由输入层传入，经过多层隐藏层后与输出层的结果进行比较，判断该结果是否满足预期，若误差函数过大则进行反向传播，将该误差信号层层传递到输入层，并通过梯度下降法对流经每层的神经元进行权值更新，最后使得该网络达到预期的输出结果，则在反向传播中损失函数为(2-4)所示：



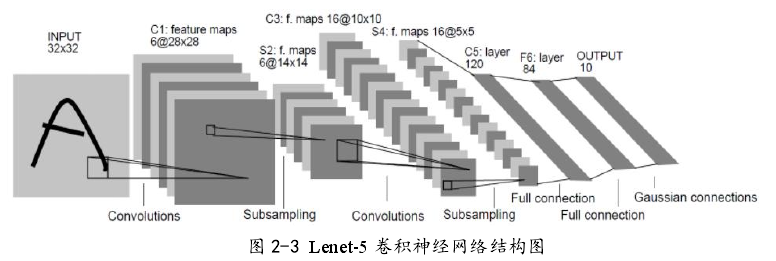
其中，第 l 层为输出层，jt 为输出层第 j 个神经元的期望值。

通过对损失函数进行一阶求导，获得网络参数的更新公式(2-5)

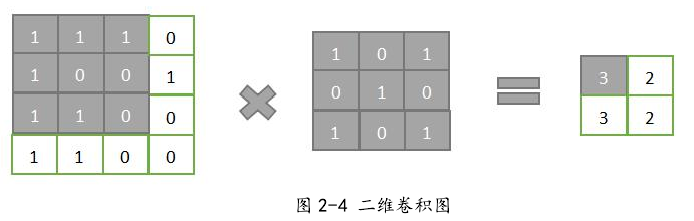


其中，η为学习率。

### 卷积神经网络

在深度神经网络中由于网络结构复杂，所训练的参数过多，而导致无法得到有效的训练，这时需要卷积网络中的卷积和池化来降低参数的个数。卷积神经网络(convolutional neural network，CNN)作为一种前馈人工神经网络，在图像分类与处理方面有着广泛的应用。该网络最早是由 Hubel 和 Wiesel 在研究猫脑皮层神经元中得到启发，进而提出的。在前一节多层感知器中几经介绍了全连接层的相关计算。所在本节利用卷积神经网络中最为经典的卷积网络 lenet-5，如图 2-3 所示，对于卷积原理，卷积层，池化层进行详细的介绍。

(1) 二维卷积原理



卷积核的计算的用如下简图 2-4 所示，灰色底表为一个卷积过程。

输入的图片为二维数组，使用二维卷积核由左到右对原图像进行卷积操作，当该行像素计算完成后切换到下一行，完成对整个图片的卷积操作。

二维卷积公式如(2-6)所示：

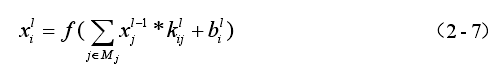


(2) 卷积层

卷积层是通过卷积核对输入的图像进行特征学习，得到高层次的特征图。在lenet 网络中，共有 C1、C3、C5 三层卷积层。通过输入层输入一张 32x32 像素大小的图片，通过 C1 层对输入的图片进行卷积操作(即二维卷积原理)，利用 5X5的卷积核进行卷积操作后，将输入图像变成 28x28 像素的特征图，从输入层到 C1层，供使用了 6 个不同卷积核进行卷积操作，不同的卷积核获得一张特征图，所以在 C1 层共有 6 张特征图。

在池化层 S2 中有 6 个特征图，这些特征图通过对应的卷积核卷积得到 C3 卷积层中得到 16 个特征图。在 C3 卷积层中每个图层都有一个偏置，对于卷积层的

计算可用如下公式(2-7)所示：



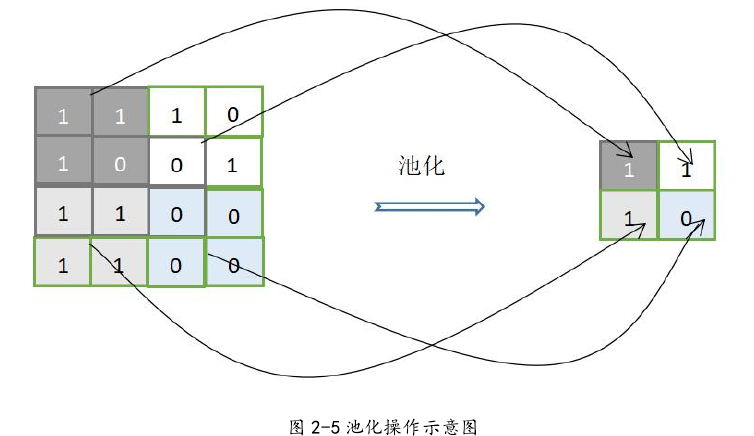
其中ljx 表示第 l 层中的第 j 个特征图。lijk 表示池化层中第 j 个特征图连接到卷积层第 k 个特征图的卷积核，lib 为该卷积层的偏置，f 为所设置的激活函数。在 C5 即第三层卷积层的计算与 C3 层一致，通过池化层 S4 后得到 120 个特征图，每个特征图与前一层的池化层 S4 中 16 张特征图相连。后面为最全连接层，输出最后的结果。

(3) 池化层

池化层用于降低特征图的维度，对特征图中的有用信息进行二次提取。在lenet5 网络中，S2，S4 为两层池化层，池化层中的特征图数量与前一层的卷积层中的数目相同且对应，池化层中的神经元根据输入的局部特征图也有特定的关联区域，不同的神经元不重叠。具体的训练为池化层中的所有元素相加在乘上可训练系数，加上可训练的偏置。池化层公式可如下(2-8)所示：



其中，lj 和ljb 为第 l 层池化层的输出的第 j 个特征图的可训练参数，D 为下采样函数， f 为激活函数。



(4) 全连接层

全连接层与多层感知器类似，全连接层作为最终的分类器，将前面卷积层与池化层所提取的特征信息映射到相应的标本空间中，由最后的输出层输出该结果如(2-9)所示：



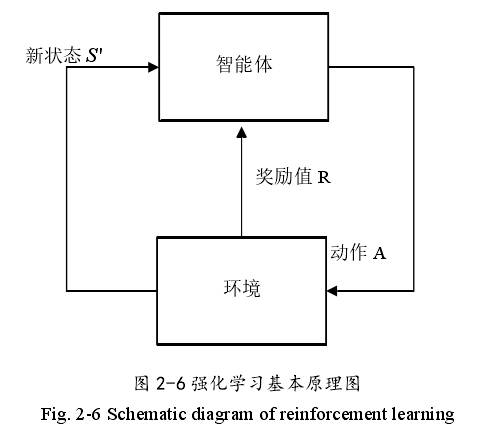
其中，Q 是全连接层的输出结果， x 未输入信号，b 为全连接层的偏置。

(5) Dropout 原理在深度强化学习模型中会经常用到 Dropout 方法。Dropout 方法是由 Hintion在 2012 年提出的，该方法主要是为了解决卷积网络中出现的过拟合现象。当面临的复杂的问题时，研究者们通常要设置更多的神经元数目来提高卷积网络的特征学习能力，但是复杂的网络模型也导致了过拟合问题的出现。除此之外，训练样本过少也容易导致过拟合。Dropout 方法原理是在前馈网络进行前向传播时，只选择网络中的一部分神经元进行特征学习，这样使的每个神经元只是用了样本集中的部分样本进行训练，即对样本集进行了采样，从而降低了样本集中数据的相关性，从而有效了降低了过拟合现象，提升模型的通用性。

## 强化学习基础

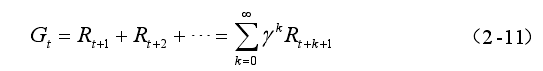
### 强化学习原理

强化学习是一种从环境状态映射到智能体动作的学习，强化学习的基本原理如图 2-6 所示，智能体在执行某项任务时，首先通过随机动 A 作与周围环境进行交互，在动作 A 和环境的作用下，智能体会产生新的状态，同时环境会反馈给智能体一个奖励值。如此循环下去，智能体与环境不断的交互从而产生很多的数据，强化学习算法则是利用产生的数据修改自身的动作策略，再次与环境进行交互，产生新的数据，并利用新的数据进一步改善自身的行为，经过多次迭代后，智能体会学习到最优的策略，从而完成相应的任务。



### 马尔科夫决策过程

马尔科夫决策过程(Markov decision process，MDP)是由 Howard 在 1960 年提出的一个理论框架，可以解决强化学习大多数问题的框架。在强化学习任务中应用时常常将其定义为一个元组(S，A，R，P，γ)，其中：S 表示智能体在环境中的所有状态集合，tS 表示智能体在该时刻的状态。A 作为智能体的动作集合，tA表示智能体在该时刻执行的动作。R 为环境的反馈奖励值， 为折扣因子，用来计算累计奖励值，]1,0[ ，用于确定未来时刻所获得的奖励值比重，由于未来奖励值的不确定性，所以从 0 到 1 从取值降低未来奖励值对当前状态的影响。累积奖励值计算公式如(2-11)所示：



P 为状态转移概率，该转移概率包含了执行的动作，公式表达如(2-10)所示：



其中，assp'表示了在tS 状态执行 a 动作到达1tS 状态的概率。

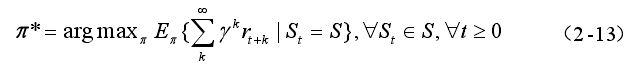
马尔科夫决策过程在应用时当满足两个条件：

(1) 马尔科夫决策的应用场景必须是完全可观测的。

(2) 当智能体达到下一状态St+1 时，该状态仅与当前状态St 有关，与之前的状态无关。定义关系式为



在公式可知，tS 包含了历史时刻的相信息，一旦当前状态确定了，历史信息就可以被抛弃。在 MDP 框架下，强化学习的最终目标则是寻找到一个最优策略\*，该策略预测了未来时刻的累计奖励值，即在未来时刻中的每个状态下都能获得最大的累积奖励值，该策略Π\*的表达式如(2-13)所示：

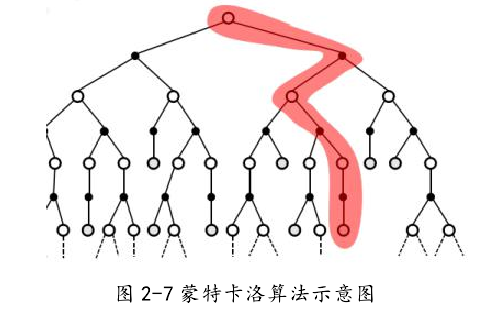


其中，γ 为折扣因子，γ∈[0,1] ， k 该策略下的状态数目，S 为状态空间。强化学习在解决问题时可以为分基于模型的和无模型的方法，下面主要对于无模型下的模特卡罗算法和时间差分算法进行介绍。

### 蒙特卡洛算法

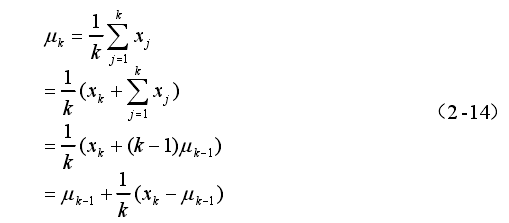
蒙特卡洛算法[62]是无模型强化学习算法中的一种，无模型问题如围棋博弈，无法预测对手的决策，即使智能体执行了相应的动作策略也无法达到预测的新状态，即状态转移概率是未知，应对该情况，蒙特卡洛算法则是利用经验平均的方法来代替随机变量的期望，从完整的状态序列中估计某一时刻下该状态的实际价值，完整的状态序列则是从某一时刻状态开始，智能体与环境进行交互直到达到终止状态，并计算获取的累计奖励值。

在蒙特卡算法中的经验则是利用策略产生多次试验，根据奖励值判断哪边障碍物多，哪个区域会有更高的奖励值，多次试验收集数据，平均则是对多次采集的数据采取平均值，得到这种经验近似的方法理念。

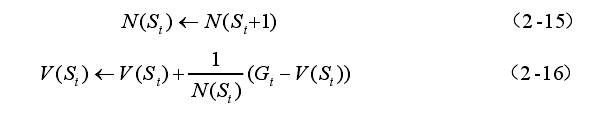


在实际问题中会出现这样的情况，当需要预测的状态出现在该完整序列中的多次出现，也就是在智能体与环境交互时，出现了一次或多次返回该状态的情况，面对这种情况蒙特卡洛算法有两种选择，一是对于整个状态序列中仅把第一次出现该状态的奖励值计算到累计奖励值中进行平均值计算，二是对于每次出的该状态的奖励值都计算到累计奖励值中进行平均值计算。对于这两种蒙特卡洛算法称为首次寻访(First-visit)和每次寻访(Every-visit)蒙特卡洛估计。

蒙特卡洛算法在对经验数据求取平均的时候常常采用累进更近平均值法，该方法的优点在于不需要储存所有的历史数据，公式推导如(2-14)所示：



累进更新平均值算法利用前一次的平均值和数据总个数来计算当前状态的新平均值。当新的数据kx 需要进行平均计算时，先将xk 与前一次的平均值μk-1做差，在将该差值乘上总个数的倒数进行修正。



如果将该算法的平均值看成对于该状态的价值，新数据看成是新状态的获取奖励值，则可以表示首次寻访蒙特克罗算法的更新状态价值的公式为：

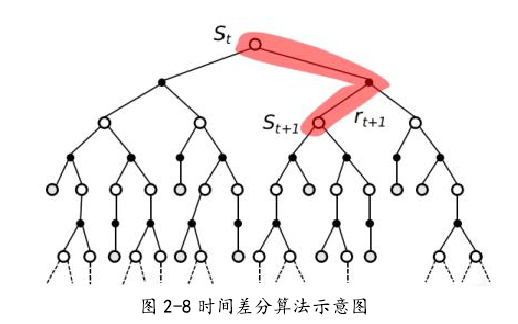
其中， )(tSN 为当前状态的数据总数量， )(tSV 为对当前状态累计奖励值的期望值。即对当前状态的价值，tG 到达当前状态的累计奖励值。以上就是蒙特卡洛算的主要思想描述，蒙特卡洛算法在对状态价值的估计时不借助状态转移概率，而是从完整的状态序列中进行经验学习，使用平均值代替状态的价值，在理论中如果状态序列总够多，则对状态价值的评估就越准确。

### 时间差分算法

蒙特卡洛存在的问题是必须要达到本此回合结束才能进行更新计算，相比如此，时间差分算法在每次动作后都可以进行一次更新运算，更加灵活，应用也更为广泛，时间差分算法的示意图如 2-8。当智能体处于St 时刻时，利用时间差分法对该状态的值函数更新公式如(2-17)所示：



其中，利用历史奖励值rt+1与预测下一状态的值函数V(St+1)来更新当前时刻的值函数V(S)。



实验次数越多，对于得到的状态值函数就越准确，与蒙特卡罗方法相比，由于不需要到达最终的结束时刻，及时更新使得时间差分方法在速度上更快，更容易收敛到真实值。

时间差分算法以更新方式的不同中以分为在线策略(On-policy learning)和离线策略(Off-policy learning)学习。在智能与环境交互时，实际由两个策略的产生，一个用于指执导智能体在环境中执行动作的策略称为行为策略，另一个用于评估状态或行为的价值的策略称为目标策略。智能体在学习过程中，用于评价状态的策略与行为策略为同一策略便是在线策略方法，如果在该过程中，目标策略与行为策略不相同，则为离线策略学习。产生两种学习策略的原因主要是因为 -贪婪策略的存在，在任意时刻的状态 S 下，行为 a 为动作值函数最大的动作，但是行为策略仍存在一定的概率在动作空间去随机选取动作 a'执行，这样增加了更多的探索可能，有利于发现更优的结果。在线策略学习算法的有 Sarsa 算法，离线学习算法则是 Q-learning 学习算法。文章以 DQN 算法为核心，所以本小节中主要对 Q-learning 算法进行介绍。Q 学习算法作为基于值迭代的强化学习算法，利用Q(S,a) 来表示该状态的动态值函数，并将智能体与环境交互时所生成的所有 Q 值储存在一个 Q 表中，进而在每次动作选择是选择 Q 表上对应该状态的最大动作值函数。Q 学习作为典型的离散策略学习算法，行为策略是基于 Q(S,a) 的贪婪策略，目标策略则是基于Q(S,a)的完全贪婪策略。

则 Q(S,a) 的更新公式如(2-18)所示：

其中α后作为目标是基于目标策略执行 A’ 动作得到的 Q 值，依据这种价值评估方式，状态St+1 是根据з-贪婪策略产生的At 动作朝着St+1 状态下确定的最大价值方向更新进行一定比例的更新，这样做能够使得行为策略更加接近贪婪策略，但又同时保证了探索足够的经验数据，并最终收敛到最优策略和最优值函数。

Q学习的值函数更新公式如(2-19)所示：



其中，依据目标策略选择St+1 状态下最大的值函数在乘以衰减系数γ，再加上奖励值 r 构成 Q 现实，从 Q 表中选取该状态下对应的 Q 值为 Q 估计。Q 学习的目标就是得到最优的 Q 值，通过如下计算的最优的 Q 值就决定了该状态的最佳策略。策略Π作为状态与动作空间相互对应，则可以定义为(2-20)：



对于所有的状态动作对，如果一个策略的期望回报大于或等于其它所有策略的期望回报，那么称策略为最优策略Π\*．最优策略的衡量需要状态值函数 V（S）状态动作QΠ（S，a）,共同衡量。智能体按照策略Π（S|a） 执行动作后返回的期望值定义为状态值函数为：



动作值函数反馈的期望值定义为：



则最优状态动作策略则可定义为：



且该最优状态动作策略满足贝尔曼方程，则通过迭代贝尔曼方程求解 Q 值，如(2-24)所示。



当经过无穷此迭代，使 i→∞时， Qi→Q\*使得动作值函数收敛，得到最优策略



### 深度强化学习

强化学习是通过储存在 Q 表中的 Q 值来确定智能体在特定状态下的动作策略，由于 Q 表的储存空间有限，这就导致了强化学习只能处理状态空间和动作空间有限的问题，如果面对连续的动作或者状态空间太大时，由于超出了 Q 表的储存范围，强化学习算法便无法给出相应的状态动作策略。为解决这一问题，DeepMind 公司提出了深度 Q 网络(Deep Q-Learning，DQN)，将强化学习中的Q-learning 与深度神经网络结合在一起。DQN 通过深度卷积神经网络来表征值函数，从而避免了 Q 表储存能力不足的缺陷；通过经验回访机制提高了样本利用效率，强化了学习过程，通过梯度下降的方法来优化损失函数，从而实现了从感知到动作的端对端的学习方法。随后在 2015 年的 Nature 论文中提出了目标网络的概念，进一步降低了数据的关联性。Nature DQN 算法结合神经网络对 Q-learning 算法进行三处优化。（1）卷积神经网络进行行为值函数逼近。Nature DQN 算法中利用神经网络逼近值函数，这里神经网络中的权重便是与值函数对应的参数，用 表示，此时值函数表示为 Q(s,a|) ，当参数 确定时，该状态下的动作也就确定了，当通过不断训练网络，更新网络权重 后，这也表示着智能体在该状态学习到了更优的动作策略。

(2) 经验回放训练过程

上世纪 90 年代也曾有研究者利用神经网络逼近行为值函数，得到的结果却常常出现不收敛、不稳定的情况，这是由于神经网络在训练时存在对训练样本的要求，要求训练的样本必须是独立同分布，但是通过强化学习得到的样本都是智能体与环境的连续交互数据，这样的数据向后相互关联，利用这些样本进行顺序训练时，神经网络便会出现不稳定的情况。而引入的经验回放机制(experiencereplay)则打破了样本的数据相关性。在强化学习的过程中，智能体每次执行动作策略到达新的状态后，通过在线处理得到转移样本，并储存在记忆文件 D 中，当训练轮次大于设定的学习率时，从记忆文件 D 中随机均匀选取一小批样本并使用随机梯度下降的方法更新网络的权值(stochastic gradient descent ,SGD)，通常选取的样本是相互独立的，这样大大降低了样本之间的关联性，从而提升了算法的稳定性。经过处理的样本可得到多次使用，在减少样本相关性的同时又提高了样本的利用率。

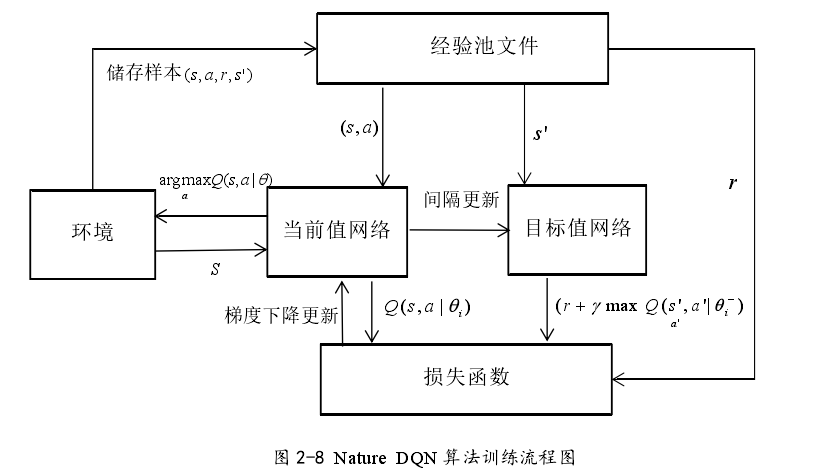
(3) 设立单独的目标网络

DQN 算法在更新网权重时，通过最小化当前 Q 值与目标 Q 值之间的均方误差来更新网络权重的，可如果只有一个网络，即目标 Q 值与当前 Q 值公用了一个网络参数，这样在梯度计算时便加大了数据之间的关联性，降低了训练网络的稳定性。为解决此问题，DeepMind 提出了单独设计 TD 目标网络，TD 目标网络的结构与当前网络的结构相同，但网络参数存在差异，TD 目标网络参数表示为-，当前网络用与智能体实时动作值的估计，网络权重每一步都更新，对于目标网络的更新，需要间隔一定的轮数，由当前网络的权重参数复制给目标网络进行更新。通过该方法降低数据的关联性，加快了网络的收敛速度。当前 Q 值与目标 Q 的误差函数如(2-26)所示：



通过对网络权重参数求偏导，更新参数为



在当前网络根据状态信息估计出动作值函数后，该值函数所确定的动作策略并不是直接由智能体执行的，还需要根据探索与利用两大策略进行选择。利用策略则是执行最大动作值函数所确定的最优动作策略，以获得最高的期望奖励值，但是该策略容易陷入过拟合。而探索策略则是根据探索概率，即便得到该状态的最优动作策略后，仍然有概率执行其他非最优动作策略，这样便可以获得其他动作的期望奖励值，这样有助于探索出更好的结果，避免陷入局部最优。 所以设置一个温和的贪婪策略在探索环境中非常关键。DQN 的完整算法框架如图 2-8 所示。

Nature DQN 算法通过独立目标网络，经验回放机制，和合理的探索策略等改进方式成功的将神经网络引入了强化学习算法中，保证了神经网络输出动作 Q 值的合理性并提高了算法的稳定性与使用范围。尽管主网络与目标网络使用的网络结构与参数相同，但在博弈游戏训练中却取得的出色的表现，证明 Nature DQN算法具备很强的通用性。

本章小结

本章对深度强化学习所用到的基础知识进行相应的介绍，包含深度学习中的神经网络和卷积网络，强化学习的理论框架和基于值函数的强化学习算法，对于深度强化学习中的 DQN 算法进行理论介绍。

本章小结

本章对深度学习所用到的基础知识和算法基础进行相应的介绍，包含深度学习中的各种算法简介，神经网络和卷积网络的概念，强化学习的理论框架和基于值函数的强化学习算法，着重对于深度强化学习中的 DQN 算法和DDPG算法进行理论介绍。

# 机器人导航 3D 仿真系统搭建

## 3D仿真场景搭建

首先，经过比较选择 Gazebo 作为研究导航的仿真环境，并基于fira仿真平台搭建一个典型的室内仿真场景。本章仿真系统的搭建主要借鉴和参考 fira对仿真环境中的场景内容根据研究需要进行调整。

### Gazebo 编辑器简介

Gazebo编辑器主要包含 Palette，2D View 和 3D View 3 个功能区，Palette功能区用于选择仿真场景的建筑材料，颜色等特性，以及添加门、窗户、桌子和沙发等各种基础部件。2D View 功能区是设计、绘制指定尺寸的墙或导入窗和门的编辑区。3D View 预览区用来实时预览编辑区设计的场景，并能够设置场景中墙体的颜色或纹理等特性。

### 基于fira平台导航场景简化建模

将fira平台的场景建模文件导入工作空间的world文件夹中，改写launch文件。

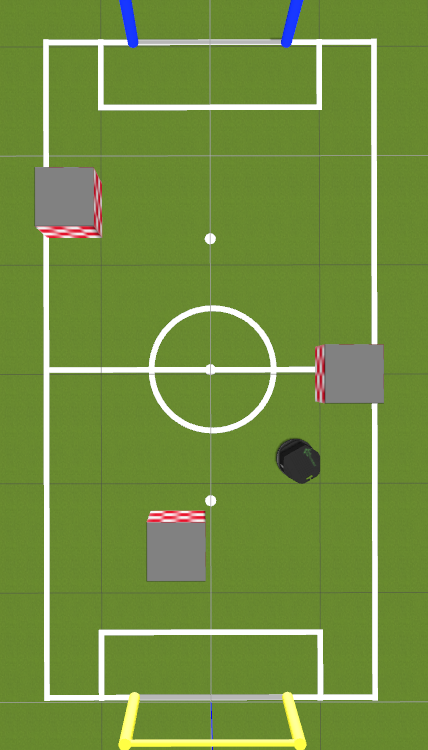


图3.1.2fira仿真场景俯视图

场景为长方形型的仿真空间，类似足球场的布局，地面为绿色，并设有黄色和红色两个球门，其中表面带有红白格子标记的静止方块为障碍物，黄色球门内为机器人起始区，蓝色球门内为目标区域。

可以通过代码实现障碍物位置和机器人起始位置的改变。

本场景用来实现各类机器学习算法训练机器人学习静态避障，即自起始区启动，z

## 仿真机器人搭建

### 仿真 TurtleBot 机器人搭建

实现对 TurtleBot 移动机器人 3D 建模，然后进行初步仿真测试。该机器人平台被用来基于深度强化学习训练以学习导航的能力。

在 Gazebo 中，仿真机器人 3D 模型的建立通过编写 URDF 脚本文件实现。

URDF 是一种描述机器人各个部分结构、关节、自由度等的 XML 脚本文件格式。 和 URDF 文件相比，XACRO 文件是一种更优秀的机器人建模语言，其对 URDF的文件进行了压缩处理，因而 XACRO 文件具有更高的可读性和可维护性。基于 Gazebo 的开源性，本文要使用到的 TurtleBot 移动机器人仿真模型已经提供。

### TurtleBot 机器人仿真效果

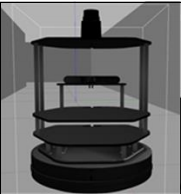


图3.2.2-1

图3.2.2-1为本文仿真用到的 TurtleBot 机器人仿真平台。通过 URDF 可视化工具 urdf\_to\_graphiz 查看整个 TurtleBot的 URDF 模型的关节结构图，部分结构图如图3.2.2-2所示：

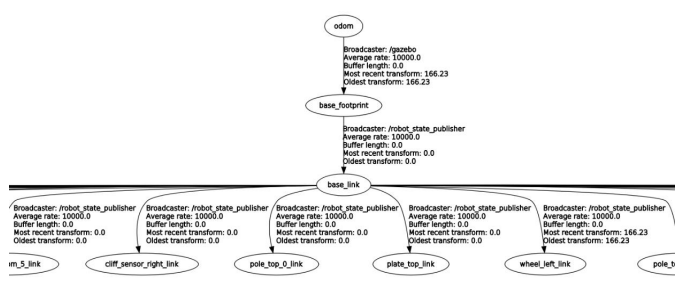


图3.2.2-2

## 场景和机器人的联合仿真

在 Gazebo 中查看搭建好的两个典型的仿真场景和 TurtleBot 机器人，最终联

合仿真效果如图 3.3-1所示。

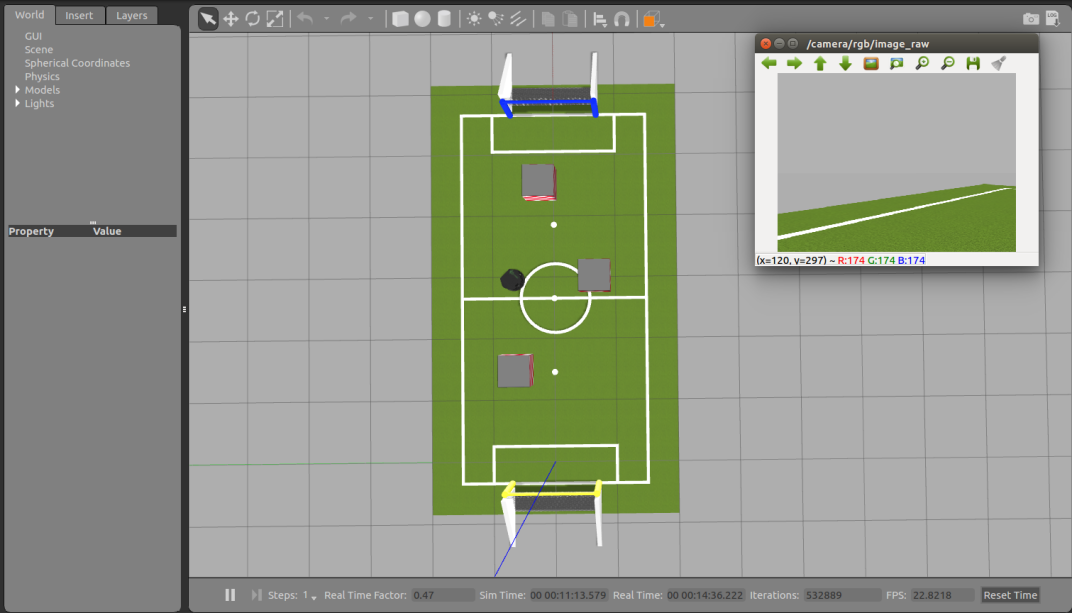


图 3.3-1

# 机器学习模型实现

## 视觉特征提取

### 图像获取

通过代码实现机器人视觉图片的获取，具体代码如图4-1所示：

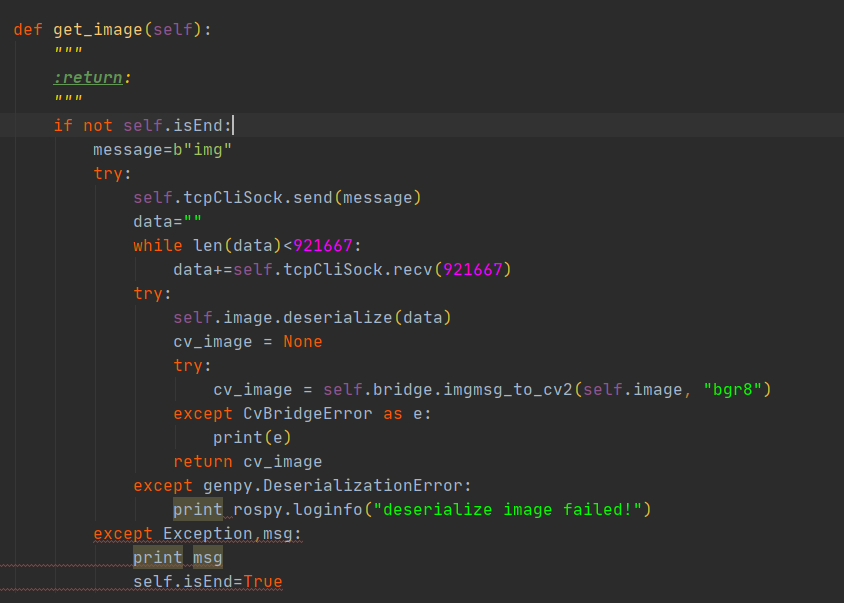
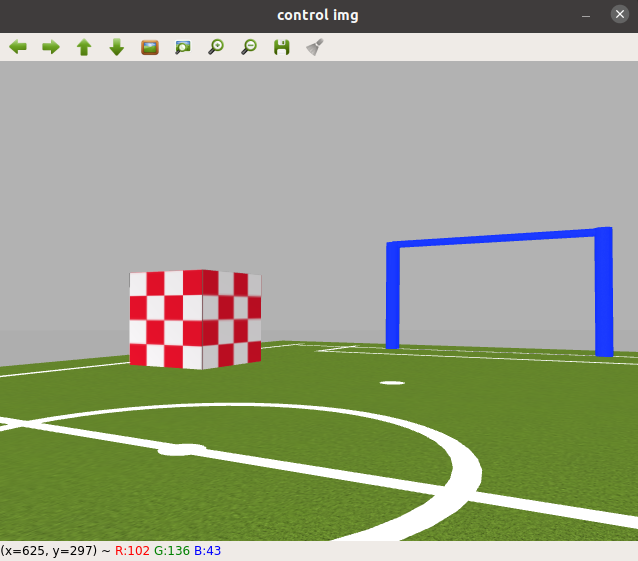
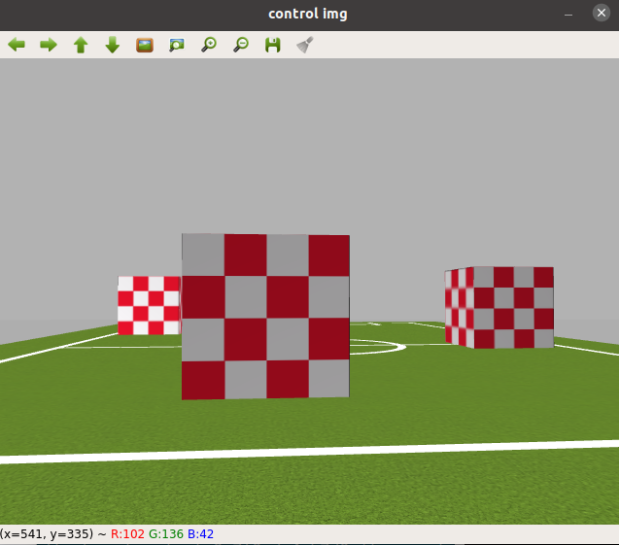


图4-1

图像获取效果：



### 图像处理

通过调用opencv和PIL中的函数编写代码，将获取的机器人视觉图片变形成正方形，具体代码如图4-2所示：

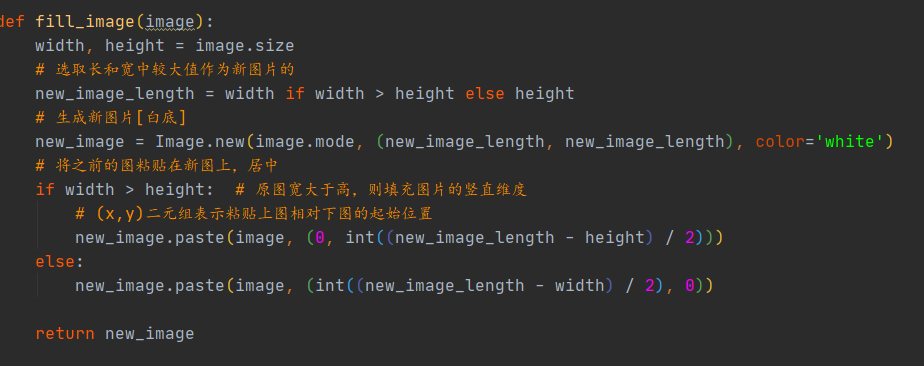


图4-2

将变形后的图片分成九等份，具体代码如图4-3所示：

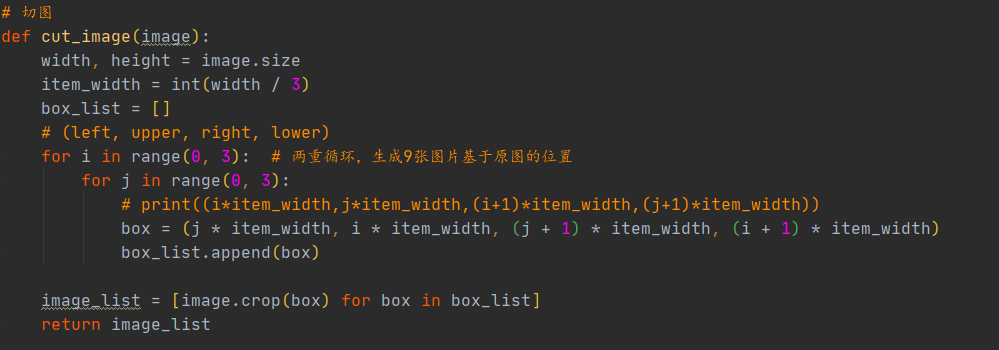
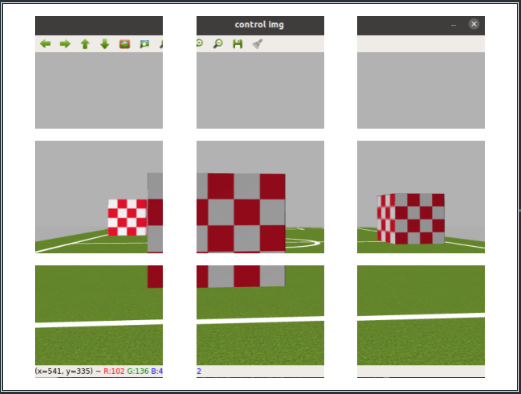


图4-3

图像分割效果：



将图像的颜色空间转换到hsv，设定阈值分别提取红，绿，蓝三色，分别对应障碍物，地面和球门的图像特征信息，调用函数：

mask\_red = cv2.inRange(img, lower\_red, upper\_red)

第一个参数：img指的是原图

第二个参数：lower\_red指的是图像中低于这个lower\_red的值，图像值变为0

第三个参数：upper\_red指的是图像中高于这个upper\_red的值，图像值变为0

得到三张二值化的mask图像，具体代码如图4-4-1、图4-4-2、图4-4-3所示：

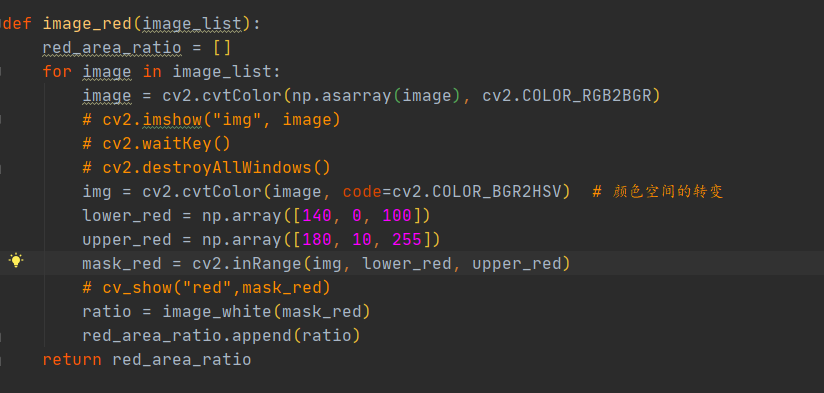


图4-4-1

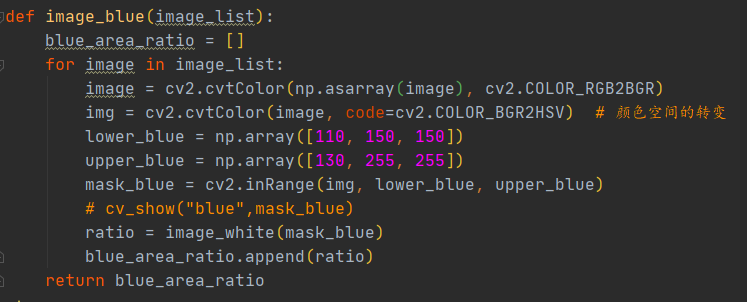


图4-4-2

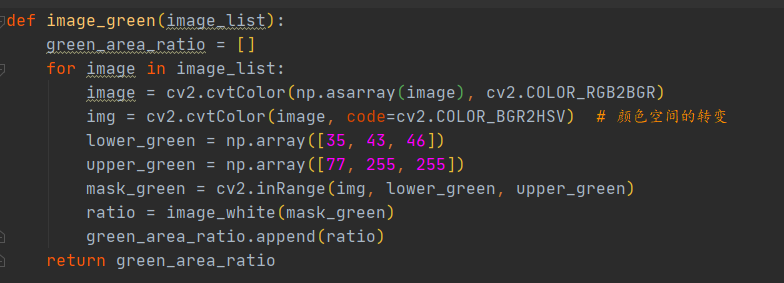
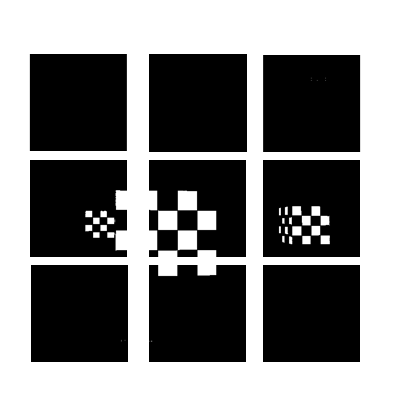


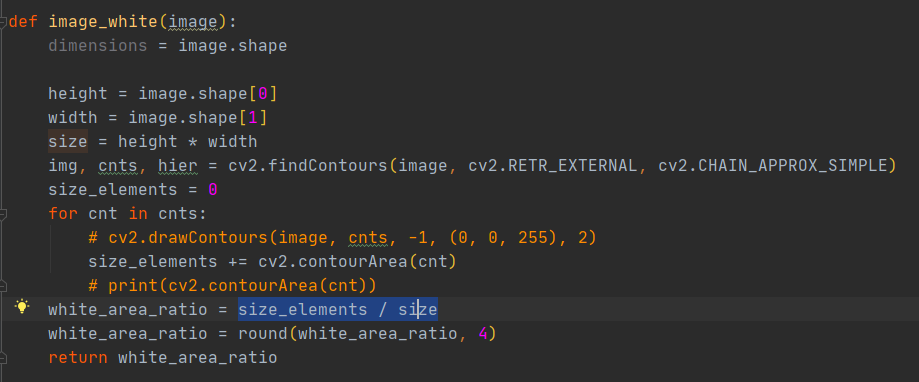
图4-4-3

红色提取效果：

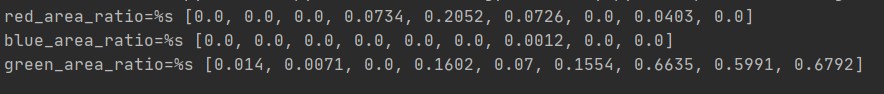


### 获取特征

对图像处理后的mask图像进行遍历查找轮廓，调用 cv2.findContours函数，再对每一个轮廓进行面积计算size\_elements += cv2.contourArea(cnt) 并进行累加。得到每张图片红，绿，蓝三色的面积值。读取图片尺寸并计算总面积，size\_elements / size得到色彩在图像中的面积比例，作为图像的特征值，为一个1\*27的向量，代码如图4-5所示：



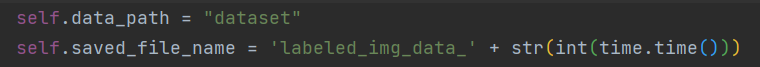
图像处理程序输出：

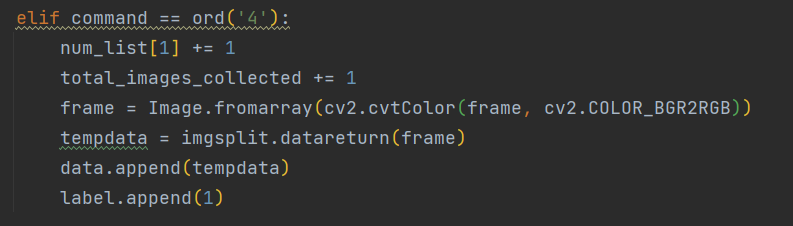


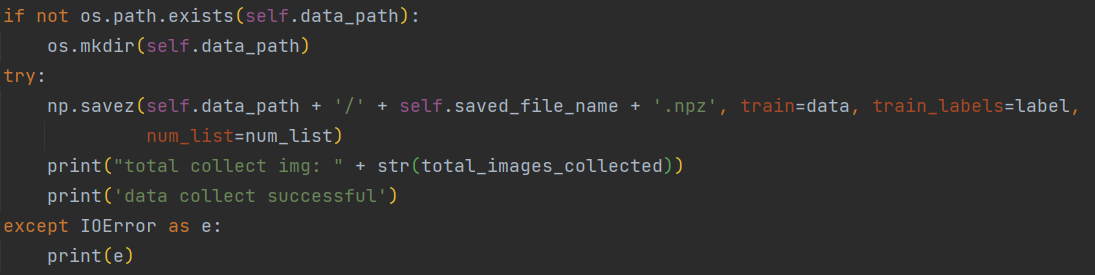
## 样本集制作

运行代码，进入gazebo仿真环境，通过键盘的wad控制机器人进行前、左、右的移动，w对应标签0，a对应标签1，d对应标签2，检测到键盘输入就保存一条数据集。

控制机器人完成一次完整路径后键盘输入“4”保存数据集文件到指定路径。具体代码如下：

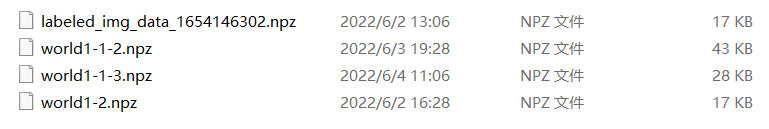


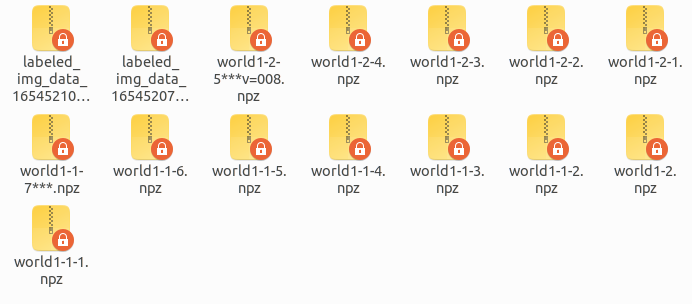




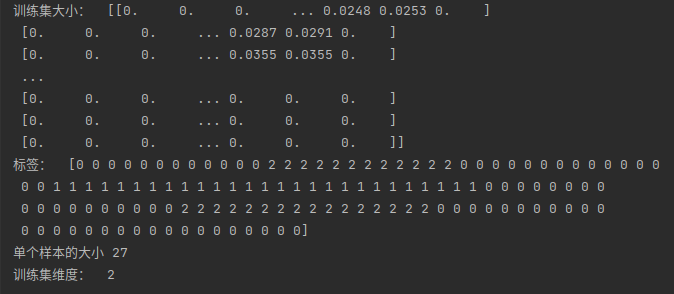


保存的样本集：





样本集分为特征数据和标签两部分。



## 机器学习算法实现与测试

在anaconda中为python下载sklearn函数库，调用其中的深度学习函数。

决策树，随机森林，贝叶斯分类器，投票算法，支持向量机算法的代码实现都是直接调用库中函数，输入特征数据和标签值即可得到训练模型。

另编写了测试函数，将数据集划分为训练集和测试集，通过k折交叉验证测试模型的准确度。

K折交叉验证代码如图所示：

### 决策树

def decisiontree\_decision(current\_data):

clf = tree.DecisionTreeClassifier()

clf.fit(train\_data, train\_labels)

pre = clf.predict(current\_data)

return pre[0]

测试程序代码：

### 随机森林

def randomforest\_decision(current\_data):

rfc = RandomForestClassifier(n\_estimators=25)

rfc.fit(train\_data, train\_labels)

pre = rfc.predict(current\_data)

return pre[0]

测试程序代码：

### KNN算法

def knn\_decision(current\_data):

knn = RandomForestClassifier(n\_estimators=25)

knn.fit(train\_data, train\_labels)

pre = knn.predict(current\_data)

return pre[0]

测试程序代码：

#!/usr/bin/env python

# encoding: utf-8

\_\_author\_\_ = 'Gary\_Zhang'

import numpy as np

from sklearn.model\_selection import train\_test\_split # 划分数据集

from sklearn.model\_selection import cross\_val\_score

from sklearn.neighbors import KNeighborsClassifier

from sklearn.datasets import load\_wine

class KNN:

'''

labeled\_img\_data\_1654146302.npz 准确度81.8

'''

def \_\_init\_\_(self):

datafile = np.load('dataset/world1-1-2.npz') # datefile: train train\_label num\_list

self.train\_data = datafile['train'] # 数据集

self.train\_labels = datafile['train\_labels'] # 标签

print(self.train\_data)

self.KNN\_test()

def KNN\_test(self):

x\_train, x\_test, y\_train, y\_test = train\_test\_split(self.train\_data, self.train\_labels,

test\_size=0.3) # 将数据集划分为训练集和测试集， 比例0.3

knn = KNeighborsClassifier()

knn.fit(x\_train, y\_train)

score = knn.score(x\_test, y\_test)

print("准确度：", score)

knn.fit(self.train\_data, self.train\_labels)

scores = cross\_val\_score(knn, self.train\_data, self.train\_labels, cv=5,

scoring='accuracy') # 采用K折交叉验证的方法来验证算法效果

print('K折准确度:', scores)

if \_\_name\_\_ == '\_\_main\_\_':

KNN()

### 贝叶斯分类器

def naive\_bayes\_decision(current\_data):

GNB = GaussianNB()

GNB.fit(train\_data, train\_labels)

pre = GNB.predict(current\_data) # 用当前位置信息预测运动方向

return pre[0]

测试程序代码：

### 投票算法

random\_state 相当于随机数种子random.seed() ，其作用是相同的。

因为同一算法模型在不同的训练集和测试集的会得到不同的准确率，无法调参。

所以在sklearn 中可以通过添加random\_state，通过固定random\_state的值，每次可以分割得到同样训练集和测试集。

因此random\_state参数主要是为了保证每次都分割一样的训练集和测试机，大小可以是任意一个整数，在调参缓解，只要保证其值一致即可。

def voting\_decision(current\_data):

clf1 = LogisticRegression(random\_state=1)

clf2 = RandomForestClassifier(n\_estimators=50, random\_state=1)

clf3 = GaussianNB()

# 将上面三个基模型集成

eclf = VotingClassifier(

estimators=[('lr', clf1), ('rf', clf2), ('gnb', clf3)],

voting='hard')

eclf.fit(train\_data, train\_labels)

pre = eclf.predict(current\_data) # 用当前位置信息预测运动方向

return pre[0]

测试程序代码：

# encoding: utf-8

\_\_author\_\_ = 'Gary\_Zhang'

import numpy as np

from sklearn.model\_selection import cross\_val\_score

from sklearn.linear\_model import LogisticRegression

from sklearn.naive\_bayes import GaussianNB

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import VotingClassifier

datafile = np.load('dataset/world1-1-3.npz') # datefile: train train\_label num\_list

train\_data = datafile['train'] # 数据集

train\_labels = datafile['train\_labels'] # 标签

# random\_state 相当于随机数种子random.seed() ，其作用是相同的。

# 因为同一算法模型在不同的训练集和测试集的会得到不同的准确率，无法调参。

# 所以在sklearn 中可以通过添加random\_state，通过固定random\_state的值，每次可以分割得到同样训练集和测试集。

# 因此random\_state参数主要是为了保证每次都分割一样的训练集和测试机，大小可以是任意一个整数，在调参缓解，只要保证其值一致即可。

clf1 = LogisticRegression(random\_state=1)

clf2 = RandomForestClassifier(n\_estimators=50, random\_state=1)

clf3 = GaussianNB()

# 将上面三个基模型集成

eclf = VotingClassifier(

estimators=[('lr', clf1), ('rf', clf2), ('gnb', clf3)],

voting='hard')

for clf, label in zip([clf1, clf2, clf3, eclf], ['Logistic Regression', 'Random Forest', 'naive Bayes', 'Ensemble']):

# 参数scoring：accuracy cv：5 将数据集分为大小相同的5份，四份训练，一份测试

# cross\_val\_score训练模型打分函数

scores = cross\_val\_score(clf, train\_data, train\_labels, scoring='accuracy', cv=5)

# scores.mean()分数、scores.std()误差

print("Accuracy: %0.2f (+/- %0.2f) [%s]" % (scores.mean(), scores.std(), label))

### 支持向量机

def svm\_bayes\_decision(current\_data):

svm = SVC(decision\_function\_shape=’ovo’)

svm.fit(train\_data, train\_labels)

pre = SVC.predict(current\_data)

return pre[0]

测试程序代码：

### bp神经网络

#!/usr/bin/env python

# encoding: utf-8

\_\_author\_\_ = 'Gary\_Zhang'

import numpy as np

import tensorflow as tf

from tensorflow import keras

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split # 划分数据集

print(tf.\_\_version\_\_)

print(keras.\_\_version\_\_)

# 加载数据集

datafile = np.load('dataset/world1-1-3.npz') # datefile: train train\_label num\_list

train\_data = datafile['train'] # 数据集

train\_labels = datafile['train\_labels'] # 标签

X\_train\_full, x\_test, y\_train\_full, y\_test = train\_test\_split(train\_data, train\_labels,

test\_size=0.3) # 将数据集划分为训练集和测试集， 比例0.3

# 查看训练集的形状和数据类型

print(X\_train\_full.shape, X\_train\_full.dtype)

# 比例缩放和像素强度降低到0-1,创建一个验证集

X\_valid, X\_train = X\_train\_full[:20], X\_train\_full[20:]

Y\_valid, Y\_train = y\_train\_full[:20], y\_train\_full[20:]

# 搭建网络模型

model = tf.keras.models.Sequential()

model.add(keras.layers.Flatten(input\_shape=[27, 1]))

model.add(keras.layers.Dense(300, activation="relu"))

model.add(keras.layers.Dense(100, activation="relu"))

model.add(keras.layers.Dense(3, activation="softmax")) # 输出十个概率分布，看属于哪一个

model.compile(loss="sparse\_categorical\_crossentropy", optimizer="sgd", metrics=["accuracy"])

history = model.fit(X\_train, Y\_train, epochs=20, validation\_data=(X\_valid, Y\_valid))

# print(history)

# 绘制history曲线

pd.DataFrame(history.history).plot(figsize=(8, 20))

plt.grid(True)

plt.gca().set\_ylim(0, 1.1)

plt.show()

# 在测试集上测试

print(model.evaluate(x\_test, y\_test))

# 仅使用测试集的前三个例子

X\_new = x\_test[:3]

Y\_proba = model.predict(X\_new)

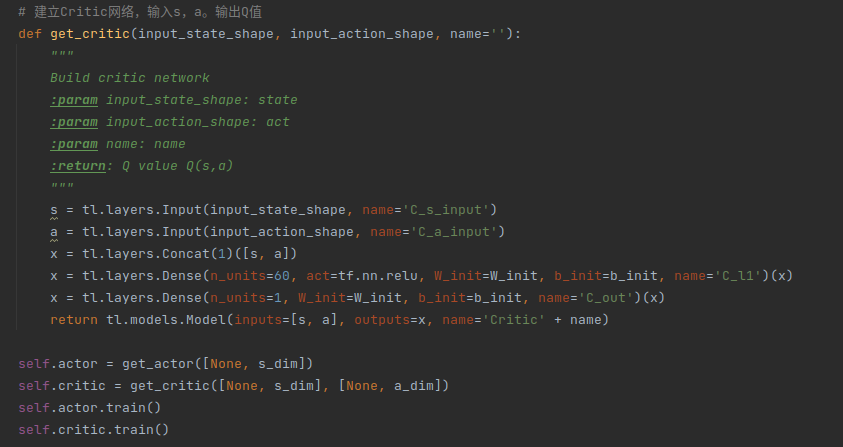
print('Y\_proba: %f', Y\_proba.round(2))

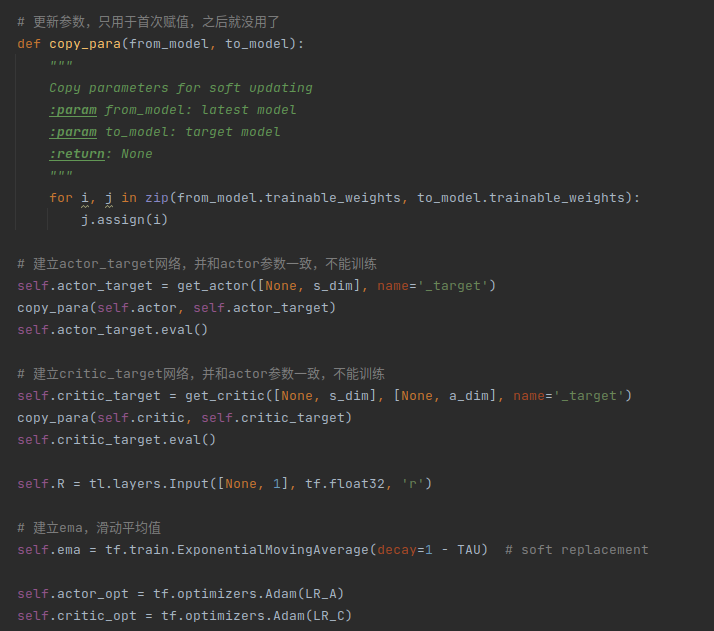
y\_pre = model.predict(X\_new)

print(y\_pre)

### DDPG算法



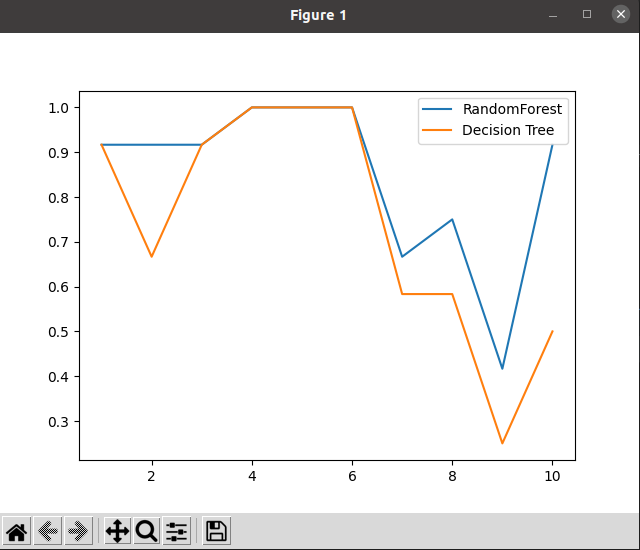
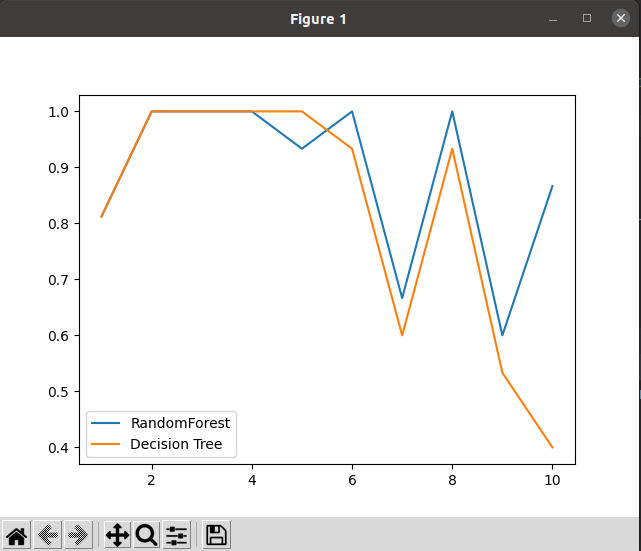




## 算法测试与对比

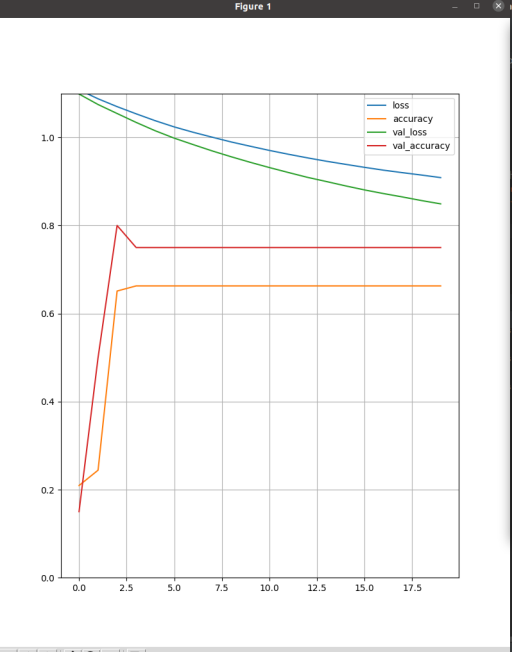
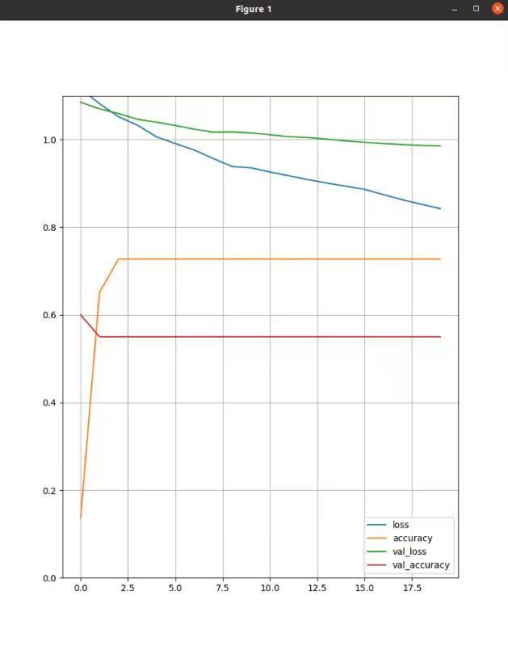
### 决策树与随机森林算法准确度对比

通过测试程序



可以明显观察到随机森林算法的准确率高于决策树算法

### Bp神经网络测试结果



分别对两个不同数据集进行bp神经网络训练，准确率和loss随训练次数的变化如上图所示。

# gazebo仿真测试

## 测试方案

将不同算法训练出来的机器学习模型导入；设定机器人的直线速度与转向速度。运行gazebo。

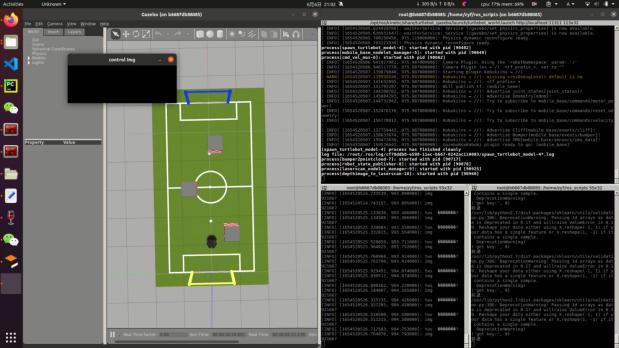
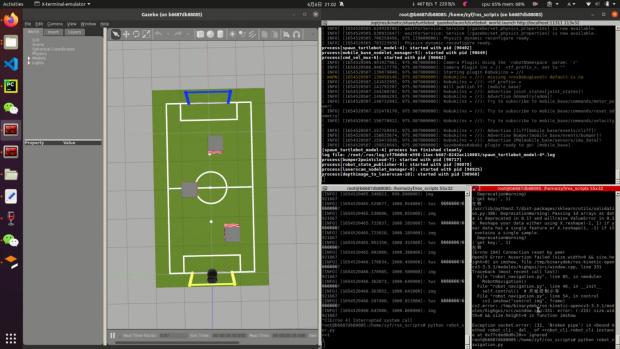
机器人初始化位置在黄框内起始区，手动输入命令行启动机器人；机器人开始在绿色场地上移动并避开障碍物，到达蓝色框内目标区域。

机器人运动全程录像，启动开始计时，机器人碰触到目标区域线停止计时。分别对前文展示的所有机器学习算法所训练出的模型进行仿真测试，观察机器人避障导航情况，并记录时间。

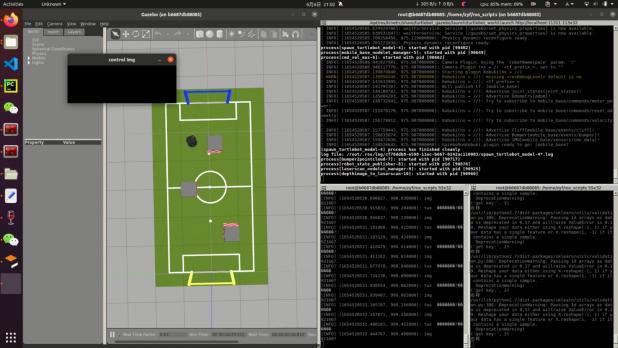
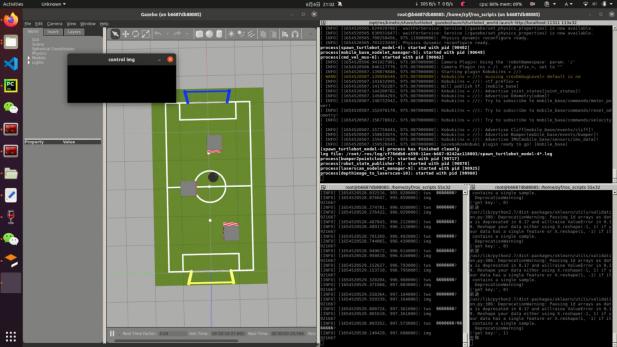
随机生成三张具有三个障碍物的地图，分别进行测试；观察机器人运行状态。

## 测试结果

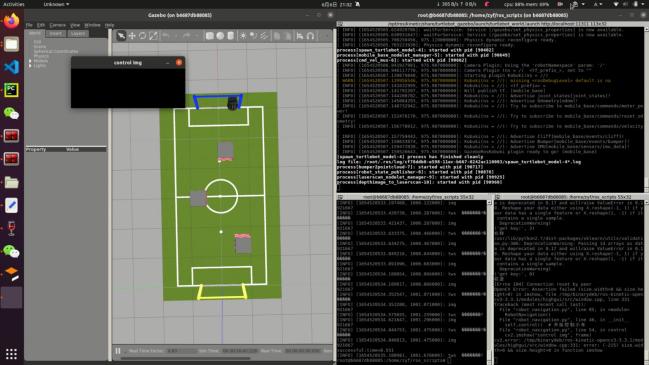
Gezebo运行画面如图5所示：



（a） （b）



（c） （d）



（e）

图5

表5 机器人仿真测试用时（单位：秒）

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | bp神经网络 | 支持向量机 | 朴素贝叶斯 | 决策树 | 投票算法 | 随机森林 | knn | DDPG |
| 图1 | 10.5 | 11.6 | 11.2 | 12.1 | 10.3 | 8.5 | 11.5 | 13.5 |
| 图2 | 10.9 | 11.5 | 13.3 | 12.6 | 11.6 | 8.7 | 12.3 | 14.1 |
| 图3 | 9.9 | 11.1 | 11.5 | 12.5 | 11.2 | 7.9 | 11.9 | 13.9 |
| 均值 | 10.43333333 | 11.4 | 12 | 12.4 | 11.03333333 | 8.366666667 | 11.9 | 13.83333333 |

## 测试结果分析

通过合理的环境构建、机器人建模和软件编程，本组能够完成基于机器学习的机器人导航仿真，实现平稳的行驶和避障过程，仿真测试证明，所有算法模型都可以完成避障；通过比较所用时间，基于随机森林算法所训练出来的模型明显优于其他算法。

参考文献

[1] 王超. 基于深度强化学习的机械臂卷积神经网络控制策略研究[D]. 哈尔滨: 哈尔滨工业

大学, 2018.

[2] 王珂, 卜祥津, 李瑞峰, 等. 景深约束下的深度强化学习机器人路径规划[J]. 华中科技

大学学报(自然科学版), 2018, 46(12): 77-82.

[3] Mirowski P, Grimes M, Malinowski M, et al. Learning to navigate in cities without a map[J].

Advances in Neural Information Processing Systems. 2018: 2419-2430.

[4] Mirowski P, Pascanu R, Viola F, et al. Learning to Navigate in Complex Environments[J].

International Conference on Learning Representations, 2017: 1-16.

[5] 刘全, 翟建伟, 章宗长, 等. 深度强化学习综述[J]. 计算机学报, 2018, 41(01): 1-27.

[6]问泽藤,温淑慧,张迪.未知环境下移动机器人自主避障算法的研究[J].燕山大学学报,2021,45(03):274-282.

[7]陈曦,王熙,李宁.基于分级速度障碍的移动机器人避障算法[J].计算机仿真,2021,38(04):281-285+290.

[8] Tanaka Y, Ji Y, Yamashita A, et al. Fuzzy based Traversability Analysis for a Mobile Robot onRough Terrain[C]. Proceedings of IEEE International Conference on Robotics and Automation,2015: 3965-3970.

[9] 王雷, 石鑫. 改进蚁群算法在移动机器人避障中的应用[J]. 南京航空航天大学学报, 2019,51(5): 728-734.

[10] Lozano-Pérez T, Wesley M. An Algorithm for Planning Collision-free Paths AmongPolyhedral Obstacles[J]. Communications of the ACM, 1979, 22(10): 560-570.

[11] Pandey A, Sonkar R K, Pandey K K, et al. Path Planning Navigation of Mobile Robot withObstacles Avoidance using Fuzzy Logic Controller[C]. 2014 IEEE 8th InternationalConference on Intelligent Systems and Control: Green Challenges and Smart Solutions, 2014:36-41.

[12] 孙钦鹏, 李猛, 王中华. 基于改进快速扩展随机树算法的移动机器人路径规划[J]. 济南大学学报(自然科学版), 2019, 33(5): 431-438.

[13] 钱夔, 宋爱国, 章华涛, 熊鹏文. 基于自适应模糊神经网络的机器人路径规划方法[J]. 东南大学学报(自然科学版), 2012, 42(4): 637-642.

[14]李鹏,阮晓钢,朱晓庆,柴洁,任顶奇,刘鹏飞.基于深度强化学习的区域化视觉导航方法[J].上海交通大学学报,2021,55(05):575-585