基于 MDP/POMDP 的智能体行为规划研究

Research On MDP/POMDP Based Agent Planning

培 养 单 位 ： 计 算 机 科 学 与 技 术 系

专　　业 ： 计 算 机 科 学 与 技 术 专 业

学　　生 ： 柏 爱 俊

学　　号 ： PB05210411

指 导 教 师 ： 陈 小 平

二〇〇九年六月

致　　谢

本研究及学位论文是在我的导师陈小平教授的亲切关怀和悉心指导下完成的。他严肃的科学态度，严谨的治学精神，精益求精的工作作风，深深地感染和激励着我。从课题的选择到项目的最终完成，陈老师都始终给予我细心的指导和不懈的支持。在实验室这两年多来，陈教授不仅在学业上给我以精心指导，同时还在思想、生活上给我以无微不至的关怀，在此谨向陈老师致以诚挚的谢意和崇高的敬意。

在此，我还要感谢在一起愉快参加实验室工作的各位学长，正是由于你们的帮助和支持，我才能克服一个一个的困难和疑惑，直至本文的顺利完成。

最后我还要感谢培养我长大含辛茹苦的父母，谢谢你们!

目　　录

摘　　要

近年来，智能体及多智能体规划问题成为人工智能领域新的研究热点，且有着广泛的应用前景。MDP/POMDP 及其相关理论成为研究智能体及多智能体不确定性规划问题的标准模型，本文对此展开研究。

首先扼要介绍了智能体、多值能体系统和决策的基本概念和一般模型，然后重点介绍了 MDP 和 POMDP 的基本概念和求解方法，最后结合 RoboCup 仿真 2D 这个标准测试平台，分析了在这种接近现实世界应用的问题中，进行整体规划所需要处理的一些子问题的设计方法，并通过结合现有马尔可夫决策过程相关理论对这些问题进行建模及分析，给出该平台更一般的研究意义。

关键词：多智能体、马尔可夫决策过程、强化学习、机器人足球

Abstract

In recent years, agent and multi-agent planning problem have been new research hotspots in the field of Artificial Intelligence with a broad prospect.MDP/POMDP and its related theories become the standard model of solving agent and multi-agent planning problem in large scale uncertain environment which is mainly researched in this paper.

A brief description of the basic concepts and general model of agent and multi-agent planning problem is introduced in chapter 1. In chapter 2, the basic concepts and general model of MDP and POMDP is mainly introduced. And last based on Robocup simulation 2D which is regarded as a standard platform for research on multi-agent planning problem in large scale uncertain environment, by employing existing Markov decision theory to model these issues, more general research significance of this platfor is gived after some necessary explanation about the platform itself, addressing and analyzing the design methods on some sub-problem for the overall planning in such kinds of real world application.

Key Words: multi-agent, MDP, Reinforcement Learning, RoboCup

第 1 章　绪论

1.1　研究背景

1.1.1　智能体

在人工智能领域，智能体（agent），又叫主体，是指可以不断感知环境并作用于环境，最终完成一定目标的自主实体。

1.1.1.1　智能体结构

智能体可以被形式化定义成智能体函数（agent function）：

$$f:\ P^* \rightarrow\ A \tag{1-1}$$

P^* 是智能体可以感知到的一系列知觉对象（percepts），A 是智能体为作用于环境可能执行的动作序列（actions）。

1.1.1.2　智能体分类

根据智能行为的复杂程度，智能体可以被分成5类：

1. 简单反射式智能体
2. 基于模型的反射式智能体
3. 基于目标的智能体
4. 基于效用的智能体
5. 学习型智能体

简单反射式智能体 简单反射式智能体根据当前感知的信息来行动，智能体函数可以表示成若干*条件- 动作* 规则，即判断*条件*是否成立，如果成立，就执行相应的*动作*。

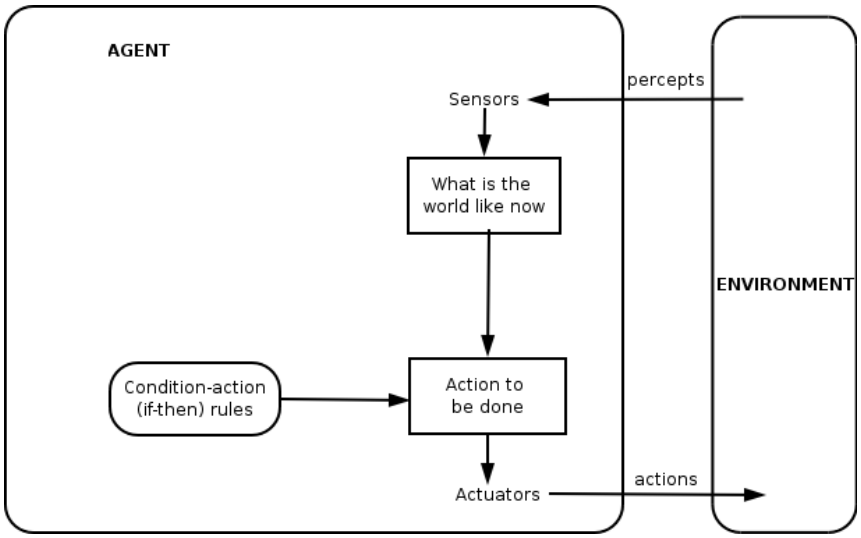


图 1.1　简单反射式智能体

基于模型的反射式智能体 基于模型的反射式智能体为环境维护一个内部状态，称为世界模型（world model），可以描述当前环境中不可感知部分的状态。这类智能体总是跟踪当前状态，并使用*状态- 动作*规则来选择应该执行的动作。

基于目标的智能体 基于目标的智能体可以识别目标状态和非目标状态，所以这允许智能体在多个可选动作中，选择一个能达到目标状态的动作来行动。

基于效用的智能体 基于效用的智能体不仅可以区分目标状态和非目标状态，还可以给每个状态一个评价值，评价值通过*效用函数（utility function）*给出。效用函数是状态到该状态效用值的一个映射。

学习型智能体 学习型智能体智能体可以在未知环境中行动，并且不断提高自己的行动能力，这通常表现为通过不断地学习，得到更好的效用函数。

1.1.1.3　智能体所处的环境

从智能体自身的观察出发，智能体所处的环境有以下一些特点：

完全可观察的 vs. 部分可观察的 如果智能体可以感知到所处环境完整、没有噪

音的信息，就称这个环境是完全可观察的，否则就是部分可观察的。完全可观察的环境一般只出现在一些软件系统中，多数研究的环境都是部分可观察的。

确定的 vs. 非确定的 确定的环境是指在当前状态下，智能体已经选定要执行的动作，那么动作执行后，环境将转移到唯一确定的下一个状态。如果用 $\mathcal{S}$ 表示状态集合，$\mathcal{A}$ 表示动作集合，$s \in \mathcal{S}$ 表示当前状态，$a \in \mathcal{A}$ 表示智能体执行的动作，$s' \in \mathcal{S}$ 表示下一个状态，定义状态转移函数（transfer function）$T(s, a, s')$ 表示智能体在状态 s 下，执行动作 a，转移到 s' 的概率，那么确定环境下状态转移函数表示成：

$$T(s, a, s') \rightarrow \{0, 1\} \tag{1-2}$$

并且：

$$\sum_{s' \in \mathcal{S}} T(s, a, s') = 1 \tag{1-3}$$

非确定的环境是指，动作执行后，状态按照一定概率的进行转移，状态转移函数表示成：

$$T(s, a, s') \rightarrow [0, 1] \tag{1-4}$$

并且：

$$\sum_{s' \in \mathcal{S}} T(s, a, s') = 1 \tag{1-5}$$

静态的 vs. 动态的 静态环境是指环境的状态转移仅依赖于智能体自身的动作，即在智能体决策过程中，环境状态是不会发生改变的。动态环境则相反。

离散的 vs. 连续的 环境是离散的还是连续的，取决于环境包含的状态数，离散环境包含有限个状态，而连续环境包含无限个状态。

1.1.1.4 智能体的分层结构

为了实现一些复杂的任务，智能体通常被设计成分层结构，包含很多子智能

体（sub-agent）。不同的子智能体处理不同的子任务，共同组成一个完整的智能系统。

1.1.2 多智能体系统

多智能体系统（multi-agent system, MAS）由若干相互作用的智能体组成，有以下几点基本特征：

自治性 每个智能体都是自主决策和行动的

局部性 每个智能体都只拥有环境的部分视图

分布性 每个智能体都是平等、独立的，不受其他智能体或系统控制

多智能体系统可以呈现出很复杂的行为，即使每个智能体的个体策略（individual strategies）很简单。

1.1.2.1 多智能体系统的主要研究内容

- 面向智能体的软件工程（agent-oriented software engineering）
- 信念、愿望和意图模型（BDI）
- 团体组织（agent-oriented software engineering）
- 通信（communication）
- 谈判（negotiation）
- 多智能体决策（multi-agent planning）
- 多智能体学习（multi-agent learning）

1.1.3 多智能体决策

多智能体决策（multi-agent planning）是指多智能体系统中的一些智能体一起为完成共同的任务时进行的决策行为。多智能体决策主要关注智能体之间的合作（coordination, cooperation）行为。

1.1.3.1 决策问题简介

决策问题又叫规划问题，是人工智能的基本问题，决策的目的是要研究如何从现有状态达到目标状态，在这个过程中采取什么样的手段和方法。决策的结果是给出一个能达到目标状态的行动序列（action sequence）或者给出每步选择行动的策略（policies/strategies）。

经典决策算法基于确定的环境模型，如 A^* 等。这类方法在实际应用时有很大的局限性。现实中的决策问题，环境经常是动态的、不确定的和部分可观察的，面对这类问题，研究者提出了基于马尔可夫模型的决策模型：马尔可夫决策过程（MDP）和部分可观察马尔可夫决策过程（POMDP）。

马尔可夫决策过程 尔可夫决策过程适用的系统有三大特点：一是马尔可夫性，又称无后效性，即系统的 $t+1$ 时刻的状态仅取决于 t 时刻的状态和智能体在 t 时刻选择的动作；二是状态转移可以有不确定性；三是智能体所处的每步状态可以被完全观察。马尔可夫决策过程是马尔可夫链（Markov chain）的扩展，如果智能体只能选择一个动作或每步状态可执行的动作是事先确定的，那么马尔可夫决策过程就退化为马尔可夫链。马尔可夫决策过程的求解方法主要有动态规划和强化学习。动态规划类的方法又主要分值迭代和策略迭代。

部分可观察马尔可夫决策过程 部分可观察马尔可夫决策过程适用于环境是部分可观察的情况。由于环境的部分可观察性，智能体不清楚自己到底处在怎样的状态，但是可以知道自己可能出于的一些状态和它们的分布，这就是信念状态（belief state）。引入信念状态，部分可观察马尔可夫决策过程退化为连续状态的马尔可夫决策过程，由于是连续状态的，所以求解仍然很困难。目前的求解算法主要倾向于直接求解部分可观察马尔可夫决策过程，主要有精确解法和近似解法。

1.1.3.2 多智能体决策的一般模型

MDP 和 POMDP 模型都认为环境中只有一个智能体，并把其他一切因素归因于客观环境。但一个环境中有多个智能体时，如果其他智能体的策略已知，那么同样可以把其他智能体都归于环境的一部分，仍可使用 MDP 和 POMDP 求

解。否则，其他智能体的选取何种策略决策也是需要考虑的，这就不能使用 MDP 和 POMDP 求解了。分布式马尔可夫决策过程（DEC-MDP）和分布式部分可观察马尔可夫决策过程（DEC-POMDP）是专门处理这类对智能体合作问题的。

在现实世界中，多智能体之间除了合作也可能存在对抗（competition），这类问题可以归为博弈。合作类问题中，多智能体有相同的收益评价，或者说共同目标是一致的；而博弈类问题中，多智能体之间的收益评价存在区别，甚至完全对立。部分可观察的随机博弈（POSG）便是进一步扩展的多智能体决策模型，可以处理这类带不确定性的博弈问题。

1.1.3.3 多智能体决策的一般过程

实际求解多智能体决策问题时，常常为了简化问题，把多智能体共同的任务细分成若干可并行处理的子任务，简称任务（task）。那么多智能体决策的一般过程为：

1. 任务分配（task allocation）
2. 决策前合作（coordination before planning）
3. 个体决策（individual planning）
4. 决策后合作（coordination after planning）
5. 决策执行（plan execution）

个体决策阶段要解决的是单智能体的决策问题，方便使用各种已有决策方法。

1.2 研究平台

仿真 2D 机器人足球是本文主要采用的研究平台。机器人足球比赛的最初想法由加拿大大不列颠哥伦比亚大学的 Alan Mackworth 教授于 1992 年正式提出。随后，Minoru Asada、Hiroaki Kitano、Yasuo Kuniyoshi 等学者创立了 RoboCup 机器人足球世界杯比赛。1997 年，在国际最权威的人工智能系列学术大会——第 15 届国际人工智能联合会议（The 15th International Joint Conference on Artificial

Intelligence, 简称 IJCAI-97）上，机器人足球比赛被正式列为人工智能的一项挑战。至此，机器人足球比赛成为人工智能和机器人学的一个标准问题以及公共测试平台。

1.2.1 RoboCup 简介

RoboCup 的长期目标是：到2050 年一支完全类人的机器人足球队能够战胜当时的人类足球世界冠军队伍（Kitano, 1998）。从 1903 年莱特兄弟飞机上天到 1969 年阿波罗登月成功花了整整 66 年；从 1946 年首台通用电子计算机问世到 1997 年深蓝战胜当时的国际象棋世界冠军经历了 51 年，Robocup 组织也将自身长期目标的实现定为半个世纪左右的时间。事实上，机器人足球所涵盖的关键技术的深度及广度，意味着这一目标的实现将基本代表了机器人已可以真正走进人们的生活。回顾人类历史，第一第二次工业革命，人类分别进入蒸汽时代与电气时代，使人类从体力劳动中解放出来；第三次工业革命，计算机及网络技术的蓬勃发展,作为人类智能的延伸，人类进入信息技术时代。展望未来，人工智能技术的发展将使人类逐渐从脑力劳动中解放出来，而机器人则是所有这些技术的一个综合应用所在。做为 RoboCup 的长期目标，也做为一项极大的挑战，人类是否能够再次完成这一新的里程碑式的任务，将具有深远的意义。

就近期而言，RoboCup 通过其不同项目组为人工智能和机器人学提供了多个标准的测试平台，可用来检验信息自动化前沿最新研究成果，包括动态不确定的对抗环境下的多智能体合作、知识表示、实时推理、机器学习和策略获取等当前人工智能的热点问题以及自动控制、传感与感知融合、无线通讯、精密机械和仿生材料等众多学科的前沿研究与综合集成。同时与影响范围最广的足球运动结合，容易受到公众的关注，利于促进了基础研究和实际应用的联系与转化。

1.2.2 仿真 2D 平台的特点

RoboCup 仿真 2D 机器人足球比赛提供了一个典型的研究大规模不确定环境下的多智能体合作及对抗问题的测试平台，研究的核心是多智能体决策理论。仿真 2D 涉及的决策问题有以下特点：

- 问题规模巨大

  - 战胜人类国际象棋冠军的“深蓝”所处的国际象棋问题的规模约为 10^{20}；围棋问题的规模约为 10^{200}
  - 而 RoboCup 2D 具有连续的状态及行动空间，按粗略离散，其决策问题的规模约为 10^{400} 以上
- 多智能体
  - 由于队友的存在，涉及合作
  - 由于对手的存在，涉及对抗
- 大量不确定因素
  - 环境部分可观察且存在噪音
  - 行动结果具有不确定性
  - 受限的通信模型
  - 对手模型未知
- 实时系统
  - 在每个极短的仿真周期内作出决策，考虑网络延迟的影响，每步决策时间只有几十毫秒

1.2.3 仿真 2D 平台简介

RoboCup 仿真 2D 机器人足球比赛是在标准计算机环境内进行的。比赛规则基本与国际足球联合会的比赛规则一致。比赛平台采用 Client/Server 结构，比赛时，参赛双的球队客户端（SoccerClient，以下简称‘client）通过网络连接到 RoboCup 委员会提供的标准比赛模拟器（SoccerServer，以下简称 server）上进行比赛，如图 1.2 所示。

典型 client 的程序流程和 server 的程序流程如图 1.3 所示。

server 发给 client 的信息包括感知信息、视觉信息和听觉信息。由于每个球员都有一定的视野范围，所以视觉信息是局部的，只包括视野内物体的相对位置、速度，并且附加了随机误差；听觉信息包含了某个队友或己方教练上周期说

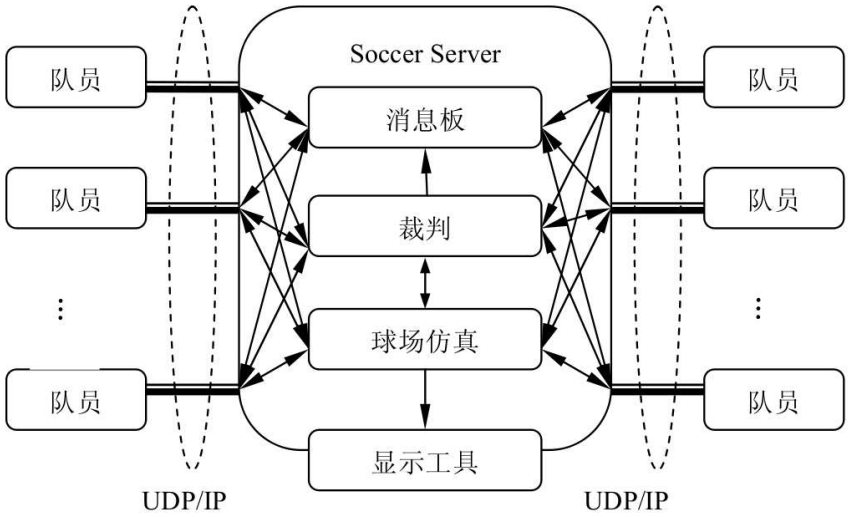
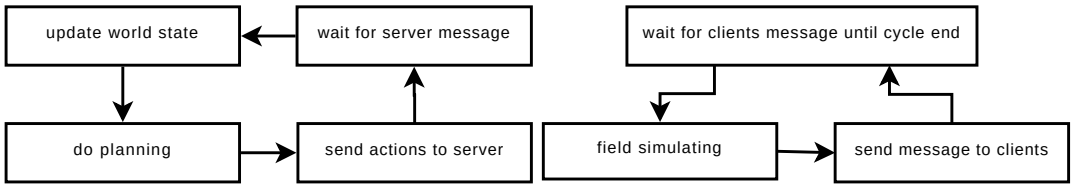


图 1.2　仿真 2D 机器人足球比赛平台



(a) 典型 client 的程序流程　　(b) server 的程序流程

图 1.3　平台流程

的话，server 规定球员间一次通信只能发送长为 10 个字节的信息，并且裁判信息也是通过听觉的形式发送给球员的；感知信息主要包体力、速度等球员自身的信息。

client 发给 server 的信息是决策后选择的动作信息。server 为球员提供的原子动作主要有：

- say(*message*) 喊话给其他队员
- turn(*angle*) 转身 *angle* 大小
- dash(*power, angle*) 加速，沿相对 *angle* 产生大小为 $f(power, angle)$ 的加速度

- kick(*power, angle*) 踢球，沿相对 *angle* 产生大小为 $g(power, angle)$ 的加速度
- turn_neck(*angle*) 调整脖子角度
- change_view(*view_mode*) 调整视角宽度[①]
- catch(*angle*) 守门员可以沿相对 *angle* 方向扑球

其中 turn，kick，dash 是互斥命令，即每周期只能发送这些命令中的一个。

① 脖子宽度越宽，视觉延迟越大

第 2 章　马尔可夫决策基础理论

首先介绍最基础的马尔可夫决策过程理论，然后在此基础上引入部分可观察性，介绍部分可观察马尔可夫决策过程理论。

2.1　MDP 简介

MDP 为大量规划问题提供了形式上的基础。MDP 的优势是可以处理状态转移的不确定性，并且可以把环境中主体以外因素造成的改变都归于环境的不确定性，从而简化了问题。

2.1.1　MDP 的基本框架

形式上，MDP 是一个四元组：$\langle \mathcal{S}, \mathcal{A}, T, R \rangle$，$\mathcal{S}$ 是环境中所有状态的集合，又称为状态空间（state space），$\mathcal{A}$ 是主体可以执行的动作集合，成为行动空间（action space），$\mathcal{T}$ 描述了状态转移模型，$\mathcal{R}$ 是回报模型。

状态空间 $\mathcal{S}$，是环境中所有状态的集合，在某一时刻 t，系统仅处于某一确定的状态 $s_t \in \mathcal{S}$。

在时刻 t，主体选择一个动作 $a_t \in \mathcal{A}$ 并执行这个动作（并不是所有状态下都可以选择所有的动作）。

转移模型 T 给出了状态被动作改变的具体方式。形式上，T 是一个函数，$T\colon \mathcal{S} \times \mathcal{A} \to P(\mathcal{S}; \mathcal{S}, \mathcal{A})$，从状态、动作映射到状态上的概率分布。我们把状态 s 下执行动作 a，转移到状态 s' 的概率记为 $P(s'|s, a)$。

回报模型 R 给出了一次状态转移下的立即回报，这是一个函数$\mathcal{S}\colon \mathcal{S} \times \mathcal{A} \times \mathcal{A} \to \mathbf{R}$。那么状态 s 下选择动作 a 的立即回报为：

$$R(s, a) = \sum_{s' \in \mathcal{S}} P(s'|s, a) \cdot R(s, a, s') \tag{2-1}$$

转移模型和回报模型都符合马尔可夫性，即：

$$P(s_{t+1}|s_0, a_0, s_1, a_1, \ldots, s_t, a_t) = P(s_{t+1}|s_t, a_t) \tag{2-2}$$

和

$$R(s_t, a_t|s_{t-1}, a_{t-1}, s_{t-2}, a_{t-2}, \ldots, s_0, a_0) = R(s_t, a_t) \tag{2-3}$$

MDP 问题的解称为策略（policy），这是一个从状态到动作的映射，给出了当前状态下应该执行的动作。主体完成任务执行的动作数，称为视界（horizon）。如果一个策略里面，每步执行的最优动作跟这步所处的阶段数无关，那么这个策略被称为是稳定的，否则就是非稳定的。有限视界的 MDP 问题一般具有非稳定策略，表示成一系列的动作映射 $\pi_t(s)$，其中 $t = 0, 1, \ldots, h$；而无限视界的 MDP 问题总是具有稳定策略，表示成 $\pi(s)$。

2.1.2 求解 MDP 问题

求解的目的是找到使长期期望回报（expected cumulative reward）最大化的策略。对于有限视界的 MDP 问题，长期期望回报为：

$$E\left[\sum_{t=0}^{h} R_t\right] \tag{2-4}$$

其中 R_t 是阶段 t 时获得的立即回报：

$$R_t = R(s_t, \pi_t(s_t)) \tag{2-5}$$

对于无限视界的 MDP 问题，引入折扣因子 $\lambda \in (0, 1)$ 得到带折扣的长期期望回报（expected cumulative discounted reward）：

$$E\left[\sum_{t=0}^{\infty} \lambda^t R_t\right] \tag{2-6}$$

定义值函数（value function）$V^\pi(s)$: $\mathcal{S} \to \mathbf{R}$ 为采用策略 π 时在状态 s 下的长期期望回报：

$$V^\pi(s) = E\left[\sum_{t=0}^{\infty} \lambda^t R_t\right] \tag{2-7}$$

可以递归表示成：

$$V^\pi(s) = R(s, \pi(s)) + \lambda \sum_{s' \in \mathcal{S}} P(s'|s, \pi(s)) V_\pi(s') \tag{2-8}$$

上述定义给出了计算策略 π 的值函数的方法，同时也可以通过值函数计算得到相应的策略。为此，定义行动值函数 $Q^{\pi}: \mathcal{S} \times \mathcal{A} \rightarrow \mathbf{R}$ 为在状态 s 下执行行动 a 的期望回报：

$$Q^{\pi}(s,a) = R(s,a) + \lambda \sum_{s' \in \mathcal{S}} P(s,a,s') V^{\pi}(s') \tag{2-9}$$

那么，状态 s 下的最优动作应为：

$$\pi(s) = \arg\max_{a \in \mathcal{A}} Q(s,a) \tag{2-10}$$

即：

$$\pi(s) = \arg\max_{a \in \mathcal{A}} \left[R(s,a) + \lambda \sum_{s' \in \mathcal{S}} P(s,a,s') V^{\pi}(s') \right] \tag{2-11}$$

同时有：

$$V^{\pi}(s) = \max_{a \in \mathcal{A}} \left[R(s,a) + \lambda \sum_{s' \in \mathcal{S}} P(s,a,s') V^{\pi}(s') \right] \tag{2-12}$$

公式 2-12 被称为 Bellman 迭代式，这是求解 MDP 问题的理论基础。

若对于所有状态 s，有 $V^{\pi}(s) \geq V^{\pi'}(s)$，则称 $V^{\pi} \geq V^{\pi'}$。使 V^{π} 达到最大的策略 π 为最优策略，用 π^* 表示，π^* 的值函数记为 V^*。

常见的求解算法主要分值迭代和策略迭代两类，值迭代算法的思想是用 V 通过迭代逐步逼进 V^*，记 $V_1, V_2, \ldots, V_k, \ldots$ 为逼进序列。迭代过程按照 Bellman 迭代式进行，容易用动态规划方法实现，见算法 1

从算法 1可以看出，动态规划需要事先知道状态转移模型 T 和回报回报 R，即需要一个环境的完整模型，很多问题事先很难给出环境的完整模型（如机器人足球，博弈问题），这时可以用强化学习方法。强化学习的主要思想是“与环境交互”和“试错”。让主体在环境中不停的决策，选择一个动作用于环境，环境接受该动作后状态发生变化，同时产生一个立即回报反馈给主体，主体根据立即回报序列不断修正状态的值函数，最终逼近最优策略的值函数，完成学习过程。常用算法如下：

蒙特卡罗方法 蒙特卡罗方法把交互过程中得到的即时回报的平均作为值函数的估计。蒙特卡罗方法仅适用于有终任务，当一个任务完成后，计算出此任

```
1  foreach s do
2      V_1[s] ← 0;
3      t ← 0;
4  repeat
5      t = t + 1;
6      foreach s ∈ S do
7          foreach a ∈ A do
8              Q_t[s,a] ← R(s,a) + λ Σ_{s'∈S} P(s,a,s') V_{t-1}[s'];
9          V_t[s] ← max_a Q_t[s,a];
10 until |V_t[s] − V_{t-1}[s]| < ε for all s ∈ S;
```

1 Dynamic Programming

务状态序列的每个状态 s 的长期回报 $R(s,\ a)$，回报函数可以由环境给出也可以编制到程序内部，用多次执行任务的平均值更新值函数。伪码见算法 2。 其中$visits[s]$ 记录了状态 s 在整个学习过程中被访问的次数，利用它实现对评价估计求平均。

λ-时序差分 时序差分（*Temporal Difference*）结合了蒙特卡罗方法和动态规划的思想，和蒙特卡罗方法一样不需要完整的环境模型；同时也像动态规划一样，随时更新评价估计，而不需要等任务完成。一个简单的时序差分学习按下式进行迭代：

$$V(s_t)\ \leftarrow\ V(s_t)\ +\ \alpha\,[r_{t+1}\ +\ \gamma V(s_{t+1})\ -\ V(s_t)] \tag{2-13}$$

在更新评价时，不是让它直接等于即时回报和后续评价之和，而是引入学习率 α 逐步调整，可以设定为一个 0 到 1 之间的常数，也可以如下：

$$\alpha\ =\ \frac{1}{1\ +\ visits[s_t]} \tag{2-14}$$

在学习过程中根据状态 s 的访问次数增多而不断减小。

每步行动产生的影响不仅是当前状态，也可能影响以后的状态，虽然迭代公式考虑了这种影响，把即时回报和后续评价作为评价更新的依据，但这种更新对回报函数大多数为零的问题显得较慢。如果每次行动之后不仅对

```
1  foreach s do
2      V[s] ← 0;
3      visits[s] ← 0;
4  repeat
5      s ← Initial( );
6      repeat
7          (S, A) ← GenerateEpisode(s, π);
8          foreach (s_t, a_t) ∈ (S, A) do
9              R ← Σ_{i≥0} γ^i r(s_{t+i+1}, a_{t+i+1});
10
11             R ← (V[s_t] × visits[s_t] + R) / (visits[s_t] + 1);
12
13             V[s_t] ← R;
14             visits[s_t] ← visits[s_t] + 1;
15     until s = terminal ;
16 until end ;
```

2 Monte Carlo

当前状态更新，也对之前的很多步更新，则可以较快得完成学习，为此引入一个和回溯有关的参数 λ 来调整这种影响，得到λ-时序差分（$TD(\lambda)$）。

记时刻 t 的评价更新差值为：

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{2-15}$$

把迭代公式改写为:

$$V(s) \leftarrow V(s) + \alpha\delta_t e_t(s) \tag{2-16}$$

上式的 $e_t(s)$ 称为回溯权重，离当前状态 s_t 越远的状态 s，$e_t(s)$ 应越小。

$e(s)$ 在每步行动后根据参数 λ 进行衰减，一种衰减方式为：

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) + 1 & s = s_t \\ \gamma\lambda e_{t-1}(s) & s \neq s_t \end{cases} \tag{2-17}$$

算法 3给出了 λ 时序差分算法。

```
1  foreach s do
2      V[s] ← 0;
3      e[s] ← 0;
4  repeat
5      s ← Initial( );
6      S ← ∅;
7      repeat
8          a ← π(s);
9          Execute(a);
10         r ← Reward(s, a);
11         s′ ← Observe( );
12         S ← S ∪ s;
13         δ ← r + γV[s′] − V[s];
14         e[s] ← e[s] + 1;
15         foreach s ∈ S do
16             V[s] ← V[s] + αδe[s];
17             e[s] ← γλe[s];
18         s ← s′;
19     until s = terminal ;
20 until end ;
```

3 λ Temporal Difference

2.2 POMDP 简介

MDP 可以处理状态转移的不确定性，但却要求状态是确定的，所以不能适用于部分可观察环境，为此引入观察和观察模型，得到 POMDP 框架。

2.2.1 POMDP 的基本框架

POMDP 是一个六元组：$\langle \mathcal{S}, \mathcal{O}, \mathcal{A}, T, O, R \rangle$，其中 $\mathcal{S}$，$\mathcal{A}$，T，R 跟 MDP 中的定义一样。$\mathcal{O}$ 是主体所有可能接受到的观察的集合，称为观察空间。O 定义了观察模型，$O : \mathcal{A} \times \mathcal{S} \to P(\mathcal{O}|\mathcal{A}, \mathcal{S})$，是从状态、动作到观察上概率分布的映射。$P(o|a, s')$ 表示执行完动作 $a \in \mathcal{A}$ 转移到状态 $s \in \mathcal{S}$，获得观察 $o \in \mathcal{O}$ 的概率。

注意到回报函数 R 也可能跟观察有关，但仍然可以写成 $R(s, a)$ 的形式：

$$R(s, a) = \sum_{s' \in \mathcal{S}} \sum_{o \in \mathcal{O}} P(s'|s, a) \cdot P(o|a, s') \cdot R(s, a, s', o) \tag{2-18}$$

由于在 POMDP 中，主体不能知道自己所处的真实状态，所以 POMDP 的策略不再是从状态到动作的映射。相反，在时刻 t，主体必需根据此前的观察历史 $\langle (o_0, a_0), (o_1, a_1), \ldots, (o_t, a_t) \rangle$ 选择自己的最佳动作。当视界很大时，维护这样一个历史会占用很大资源，不过可以证明维护信念状态（belief state）是等效的。

信念状态 b 是状态上的概率分布，$b(s)$ 即为当前处于状态 s 下的概率。POMDP 问题具有一个初始信念状态，描述了初始状态的概率分布，以后每个阶段利用观察和行动更新这个信念状态，更新过程如下：

$$b_a^o(s') = \frac{P(o|a, s') \sum_{s \in \mathcal{S}} P(s'|s, a) b(s)}{P(o|a, b)} \tag{2-19}$$

其中

$$P(o|a, b) = \sum_{s' \in \mathcal{S}} P(o|a, s') \sum_{s \in \mathcal{S}} P(s'|s, a) b(s) \tag{2-20}$$

这样，POMDP 里策略就是从信念状态到动作的映射。

2.2.2 求解 POMDP 问题

我们已经看到信念状态可以完全表示观察历史，并且策略就是从信念状态到动作的映射，那么现在的问题是如何求解得到最优策略。类似地，定义策略 π 下，信念状态 b 的值函数为：

$$V^{\pi}(b) = R(b,s) + \lambda \sum_{o \in \mathcal{O}} P(o|a,b) V^{\pi}(b_a^o) \tag{2-21}$$

其中 $R(b,s) = \sum_{s \in \mathcal{S}} R(s,a)b(s)$，$P(o|a,b)$ 如式 2-20 所定义。

所以 POMDP 的 Bellman 迭代式为：

$$V^{*}(b) = \max_{a \in \mathcal{A}} \left[R(b,s) + \lambda \sum_{o \in \mathcal{O}} P(o|a,b) V^{*}(b_a^o) \right] \tag{2-22}$$

这使利用动态规划方法求解 POMDP 问题称为可能，但由于信念状态空间是连续的，实际求解还是有难度的，这里就不介绍了。

第 3 章　仿真 2D 平台中相关子问题的研究

RoboCup 仿真 2D 比赛为研究大规模不确定环境下的多主体决策问题提供了标准测试平台。然而，由于规模上的原因，在该平台上进行整体规划的设计离不开对某些子问题的处理。本章将配合对 2D 平台的一些必要的说明，针对一些典型的子问题进行介绍。

3.1　基本介绍

主体，也就是参与比赛的球员程序的基本结构如图 3.1 所示，主要包括三部分内容：服务器接口、信息状态模块、决策模块。

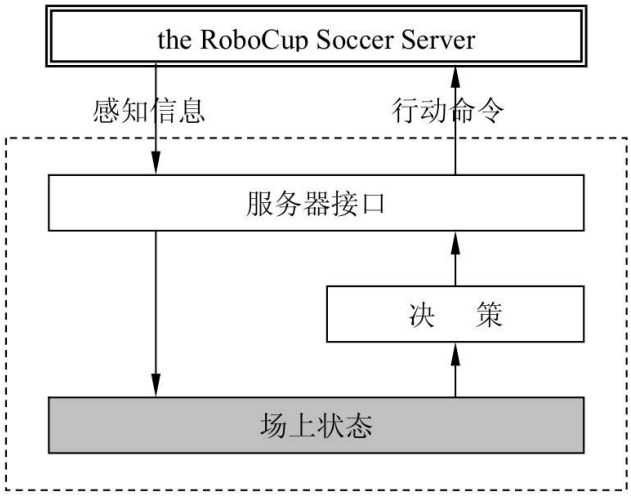


图 3.1　主体总体结构

服务器接口 服务器接口是主体与环境交互的桥梁，一方面它接收来自 server 的感知信息，并加以解释，转化成信息状态模块可以理解的信息结构，另一方面它负责把决策模块产生的动作命令发送给 server

信息状态模块 能供决策模块决策使用的信息统称信息状态，主要包括世界状态、截球信息、策略信息等，其中截球信息、策略信息是由世界状态经过一些计算得到的较高级的信息。信息状态模块主要负责信息状态的维护和更新

决策模块 决策模块是主体的核心，它使用信息状态模块提供的信息进行决策，最终产生 server 接受的动作命令，通过服务器接口发送给 server

下面重点介绍一下信息状态模块和决策模块。

3.1.1 信息状态模块

信息状态模块的详细结构如图 3.2 所示。 Parser 将从 server 直接接收

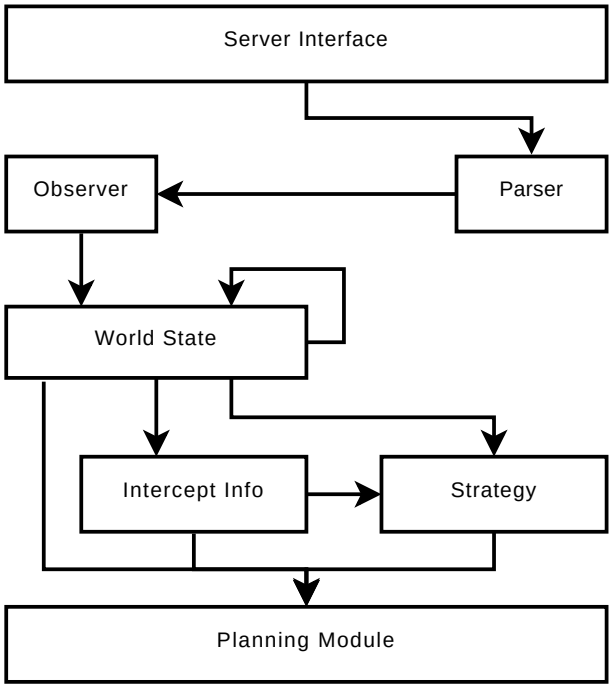


图 3.2 信息状态模块的详细结构

来的字符串信息解析成包括视觉信息、感觉信息等在内的感知对象交给 Observer，WorldState 使用从 Observer 得到的感知对象并结合历史世界状态完成更新，InterceptInfo 是从 WorldState 里面挖掘得到的截球信息，Strategy

存储了最快截到球的队员、球自由运动剩余的周期数等方便使用的策略信息。WorldState、InterceptInfo、Strategy 一并作为信息状态交给决策模块决策使用。

3.1.2 决策模块

采用分层的思想，决策主要分为三个层次：战略决策、战术决策和个人技术决策。

3.1.2.1 战略决策

战略层是在决策系统模块中的最高层。在这层中自主体决定自己是哪一角色，所处的阵形，判断出自己处于哪个模式，是进攻，还是防守，要执行什么任务，或配合完成什么合作协议，明确当前要达到的战略目标。战略决策是三层决策中最高层的决策，也正因为它的决策高度，它所考虑的信息是非常之多的，也是非常复杂的，在目前除了根据人类踢足球的常识应用分析的方法来实现外，还没有其他更好的办法。我们使用的最为成功的一个方法就是一种能基于状态的搜索，这也是要基于一定的足球知识来得到的。

3.1.2.2 战术决策

当战略层确定了所处的行为模式后，战术层来具体贯彻执行。通过一些细节分析后，本层制订出为了完成这一特定的目标，所应该采取的行为策略。包括跑位、造越位、抢球、拦截、带球、传球、射门和盯人等等。在实际中我们都为每个行为量身定做了一套模型，这套模型中普遍的思想是利用试错搜索返回的错误信息结合具体经验的启发式搜索。

3.1.2.3 个人技术决策

个人技术决策是决策系统的最低层，用以实现自主体的个人技能(如踢球、跑动等)。在每个周期，最终都是由这一层发出本周期所要执行的基本动作命令。这一层更多关注的平台本身的特点和高层的良好交互，被抽象成很多优化问题，适合使用 MDP 或 POMDP 求解。

3.2 基于 MDP 模型的 GoToPoint 个人技术决策

GoToPoint 动作属于个人技术决策，提供了使球员快速跑到某点的功能，输入为球员的当前状态和要跑的点，输出为球员应该执行的原子动作，包括转身和加速。下面首先介绍这两个原子动作模型。

3.2.1 转身模型

球员可以用原子动作 turn 改变自己的身体方向，turn 命令格式为：turn (*moment*)，其中 $moment \in [-180°, 180°]$，turn 以后，身体方向按下式更新：

$$\begin{aligned} eff\_moment &= \frac{moment}{1.0+inertia\_moment \cdot speed} \\ body\_dir &= body\_dir + (1.0 + r) * eff\_moment \end{aligned} \tag{3-1}$$

其中，$speed$ 为球员速度的大小，$inertia\_moment$ 是一个常数，r 为 server 附加的随机噪音。这个 turn 模型，使球员在速度较大时转身比较困难，即一次转身的最大角度小于 $moment$，符合实际情况。

3.2.2 加速模型

加速模型提供了球员基本的加速能力，传统的 dash 命令格式为: dash (*power*)，其中$power \in [-100, 100]$，$power > 0$ 时沿身体方向加速，$power < 0$ 时沿身体方向的反方向加速。最新的 server 支持多向 dash，dash 命令的格式为：dash (*power dir*)，考虑到近阶段要兼容老的球队，目前 dash 方向只提供身体方向、身体方向的反方向和两个身体侧方向，所以 $dir \in \{-90°, 0°, 90°, 180°\}$。考虑到人沿不同方向加速时效率不一样，表现为沿身体方向加速最容易、身体反方向加速较困难、身体侧方向加速最困难，server 引入 dir_rate(*dir*) 来刻划这种区别，

$$\mathrm{dir\_rate}(dir) = \begin{cases} 1.0 & dir = 0° \\ 0.5 & dir = -180° \\ 0.25 & dir \in \{-90°, 90°\} \end{cases} \tag{3-2}$$

dash 产生的加速度公式为：

$$acc = power \times \text{dir\_rate}(dir) \times \text{dash\_power\_rate} \times effort \tag{3-3}$$

其中，dash_power_rate 对于球员而言是一个常数，$effort$ 跟球员的体力有关，体力充沛时（即体力数值上大于一个 server 给定的阈值），也可以视为常数。加速度方向沿 $body\_dir + dir$ 方向，其中 $body\_dir$ 为球员的身体方向。dash 以后，球员位置、速度的更新按下式进行：

$$\begin{aligned} \vec{u}_{t+1} &= \vec{v}_t + \vec{a}_t + \vec{r}_t \\ \vec{p}_{t+1} &= \vec{p}_t + \vec{u}_{t+1} \\ \vec{v}_{t+1} &= \vec{u}_{t+1} \cdot \text{decay} \\ \vec{a}_{t+1} &= 0 \end{aligned} \tag{3-4}$$

其中 $\vec{a}_t$、$\vec{v}_t$、$\vec{p}_t$ 分别表示球员在 t 时刻的加速度、速度、位置，decay 是速度衰减因子，对球员而言是为常数，$\vec{r}_t$ 为 server 附加的随机噪音。

3.2.3 使用 MDP 建模求解

为了简化问题，首先通过平移和旋转，把一般 GoToPoint 问题转化成目标点为原点、球员初始位置在负 x 轴上的 GoToOrigin 问题。server 给出的运动模型满足马尔可夫性，很直接想到使用 MDP 建模 GoToOrigin 问题并求解，为此给出这个 MDP 问题的基本定义如下：

状态空间 使用一个五维向量 $(p_x, p_y, v_x, v_y, \alpha)$ 表示状态，其中 $(p_x, p_y) = \vec{p}$，$(v_x, v_y) = \vec{v}$，α 为身体角度。由于球员跑到某一点并不要求精确跑到该点，所以目标状态取为原点附近半径为球员可踢半径的一个圆。这里状态空间是连续的，下面会专门介绍。

动作空间 主体可以选择的动作为 turn(*angle*) 和 dash(*power*, *dir*)，$angle \in [-180, 180] \cap angle \neq 0°$ 按照 2° 进行离散，$power \in [-100, 100] \wedge power \neq 0$ 按照 5 进行离散，$dir \in \{-90°, 0°, 90°, 180°\}$。

转移函数 这里为了简化问题，认为状态的转移是确定，严格按照不加噪音的 server 模型进行转移。

回报函数 在目标状态执行任何动作回报为 0，其他状态回报为 -1，折扣因子取 1.0，则一定策略下，某一状态长期回报的相反数就是从该状态到目标状态所要经历的周期数。

由于状态空间是连续的，不能枚举出所有状态，所以不能采用一般查表的方法表示值函数，考虑到人工神经网络具有较好的范化能力，这里使用一个反向传播网络表示值函数。网络的结构为 $6 \times 6 \times 6 \times 1$，六个输入分别为：$(f(p_x), g(p_y), h(v_x), h(v_y), \cos(\alpha), \sin(\alpha))$。其中 f, g, h 是线性函数，目的是把各自的输入在其定义域内规范化到 $[-1, 1]$ 内；身体方向 α 被表示成两个输入 $(\cos(\alpha), \sin(\alpha))$，是为了使相邻的角度能表示成相近的输入。网络的输出为对应输入状态的值函数。

因为无法遍历所有状态，不能有效的使用基于动态规划的值迭代算法来进行求解，而是采用了强化学习方法。

3.2.3.1 学习过程

学习时，通过一个内建的模拟器，让主体在模拟器环境里学习。学习的目的是保证距离目标点 45 范围内 GoToOrigin 策略最优，45 这个值取自实际比赛的经验值，因为实际比赛中很少有出现要规划一个大于 45 的 GoToOrigin 动作序列，对于大于 45 的问题可以采取分段的 GoToOrigin 来逼近最优策略。模拟器模拟的场地是一个长为 52，宽为 5 的矩形，注意到这个场地比实际比赛场地要小，这是考虑到实际的最优策略不可能会使主体超出这个矩形，凡是超出这个矩形的行动序列肯定不是最优策略，都将被舍弃，不会被用来更新值函数（网络）。

主体在环境里采用 ϵ-贪心算法进行动作选择，即以 ϵ 的概率随机选取一个动作，以 $1 - \epsilon$ 按照 $\pi(s)$ 选取动作，这样做的目的是为了防止值函数陷入局部最优，而无法得到进一步修正，实现时 $\epsilon = 0.0025$。

每个学习周期内，首先把主体随机赋值一个随机状态，主体从随机状态开始进行动作选择，模拟器模拟出主体下一个状态，不断重复此过程，直到主体成功到达目标区域或失败（包括出界和形成环）。对于成功的样例，最后一个成功状态的值函数为 0，从后往前回溯值函数依次递减，如此得到一系列网络的训练样本，使用批处理的训练方式训练网络；对于失败的样例，则不做处理。

学习过程中发现，即使一次批处理训练网络时误差之和很小（小于 0.001），但实际测试仍然会有失败的情况，所以不好使用网络的误差小于某个阈值来判断网络是否已经收敛，而是使用了基于测试判断方法，即每 1000 个学习周期后，中断学习并随机测试 1000 次，如果这 1000 次测试中，都没有出现失败的情况，就认为网络已经收敛了，这个方法虽然也不能保证网络已经收敛了，但由于测试强度很大，所以通过测试的网络实际使用效果还是不错的。

3.2.3.2 实验结果

图 3.3，3.4，3.5 学习得到的值函数的视图。

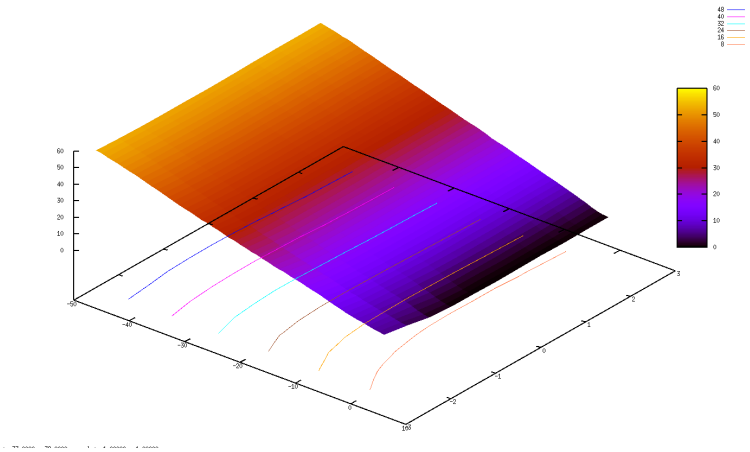


图 3.3 值函数视图

3.3 基于 POMDP 的视觉决策系统

视觉决策的目的是为了能较好的掌握场上物体（包括球和球员）最新的状态。

3.3.1 视觉模型

视觉相关的动作有 change_view 和 turn_neck，下面分别介绍这两个动作。

change_view change_view 命令用来改变视觉宽度，球员一共有三种视觉宽

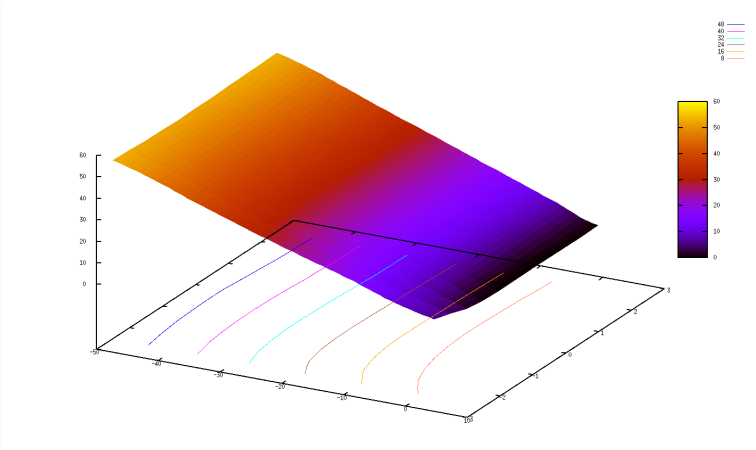


图 3.4　值函数视图

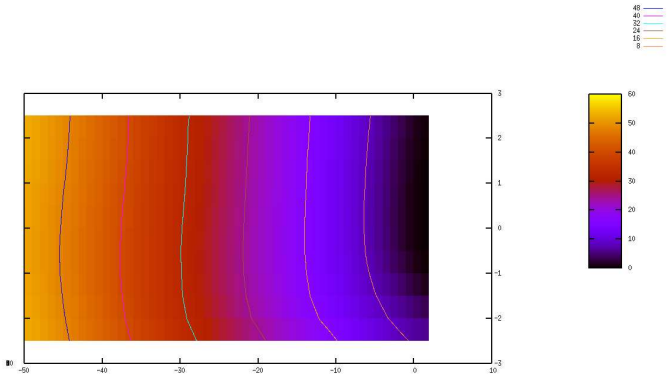


图 3.5　值函数视图

度：narraw，normal 和 wide，三种宽度对应的视觉延迟（就是上次收到视觉到收到新视觉的周期差）不一样，分别为 1，2 和 3。

turn_neck turn_neck 命令改变球员的脖子角度，便于观察不同的方向，球员脖子角度在 $[-90°, 90°]$ 范围内。

3.3.2 视觉信念状态

在视觉决策系统里面，场上的物体（球和球员）被抽象成视觉对象，视觉对象主要包含三部分数据：关注频率 $freq$，$freq$ 越小的物体越应该受到关注，关注频率由各个战术行为根据当前周期下的具体情况提出；视觉延迟 $cycle\_delay$，为这个物体没有被看见的周期数；活动范围，维护了这个物体在场上每点出现的概率，实现时，把场地离散成了一些方格，所以实际维护了在每个方格内出现的概率。视觉信念状态就定义成场上每个视觉对象活动范围的联合分布。

3.3.3 信念状态的更新

视觉决策系统的核心就是维护信念状态。信念状态的更新主要是每个视觉对象活动范围的更新，其更新原则如下：如果周期开始时看到了这个物体，那么这个物体就以概率 1 出现在一个确定的格子内；否则假设其活动范围按照随机游动模型进行扩散，事实上得到一个高斯分布。下面给出典型信念状态的例子。

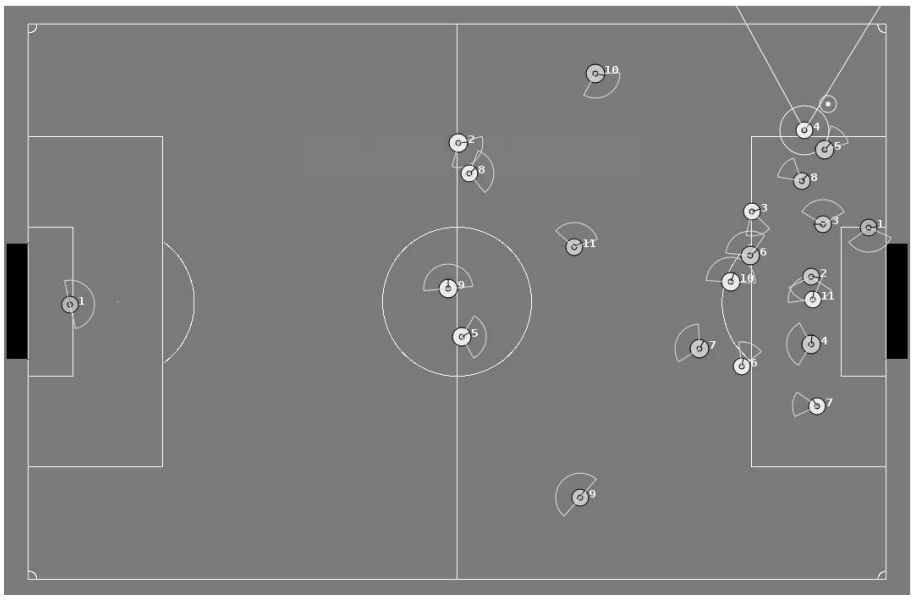


图 3.6　真实世界模型

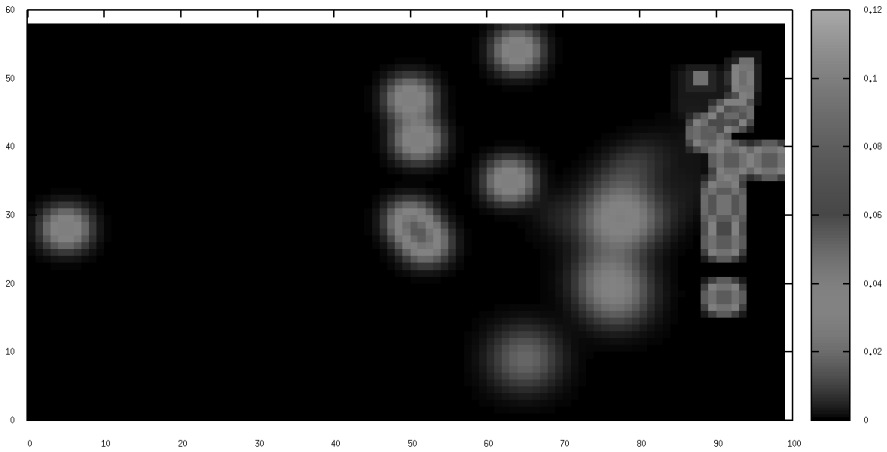


图 3.7　4号球员的信念状态

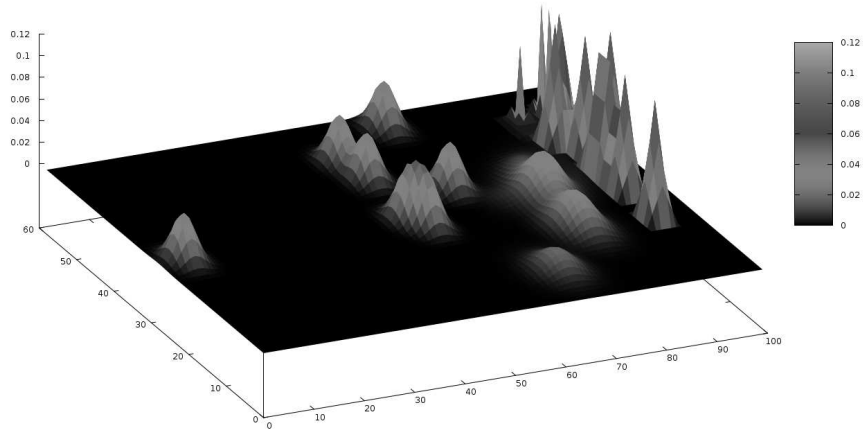


图 3.8　4号球员的信念状态（3D视图）

3.3.4　视觉决策的立即回报

视觉决策的立即回报是一个评价系统，首先定义每个视觉对象 o 完全被看到的回报为：

$$full\_reward(o) = 1 - \max(1 - (cycle\_delay(o)/freq(o)), 0) \qquad (3\text{-}5)$$

视觉动作被抽象成一个二元组：$(view\_width, neck\_dir)$，用 a 表示，则在视觉动作 a 下，视觉对象 o 的实际回报为：

$$reward(o,a) = full\_reward(o) \times \frac{\sum_{g\in\mathcal{G}} visible(g,a)\cdot appear(o,g)}{sight\_in\_delay(a)} \tag{3-6}$$

其中：$\mathcal{G}$ 表示场地上所有格子的集合；函数 $visible(g,a) = 1$，如果动作 a 下可以看到格子 g，否则等于 0；$appear(o,g)$ 为对象 o 在格子 g 出现的概率；$sight\_in\_delay(a)$ 为视觉动作 a 下，新视觉到来前没有视觉的周期数。

这样，信念状态 b 下，执行视觉动作 a 的立即回报即为：

$$R(b,a) = \sum_{o\in\mathcal{O}} reward(o,a) \tag{3-7}$$

其中，$\mathcal{O}$ 为场上所有应考虑的视觉对象的集合。

3.3.5 视觉决策的动作选择

由于视觉对象的关注频率信息只能通过外部调用获得，无法在视觉决策里面做出较好的预测，考虑视界大于 1 的视觉决策困难比较大，所以目前视觉决策系统考虑的视界为 1。这种情况下，$Q(b,a) = R(b,a)$，所以视觉决策的动作选择按下式进行：

$$\pi(b) = \arg\max_{a\in\mathcal{A}} R(b,a) \tag{3-8}$$

由于某些特殊情况下，战术层行为决策需要强制看某一物体，这时就不能只使用关注频率来突出这一物体的重要性了，为此通过强制把需要强制看的物体的得分设置成一个较高的值来解决这一问题，设置公式如下：

$$reward[o] \leftarrow important(o) * (1 + (cycle\_delay(o) + 1)/freq(o)) \tag{3-9}$$

其中，$important(o)$ 是为了反应不同物体的重要性，以便处理存在多个强制要看的物体而产生竞争的问题：

$$important(o) = \begin{cases} 50 & o\ is\ ball \\ 10 & otherwise \end{cases} \tag{3-10}$$

3.3.5.1 实验结果

编程实现时，场地被均匀分成了 100×60 个小方格，每个格子内维护了每个物体在该格出现的概率。如果对物体在某个格子内出现概率的下限不设限制，则一个长时间看不到的物体就会扩散成一个很大的范围，导致每次扩散时花费较多的时间，目前选择 0.005 作为格子内物体出现概率的下限，即对扩散后出现小于 0.005 情况的格子不进行扩散。这个概率下限是可以调整的，使用概率下限为 0.005 的版本的球队跟 为1.000 的版本的球队自动测试 100 场，结果见表 3.1。

$Version$	$Games$	$Goals$	$Points$	$AvgGoals$	$AvgPoints$	Win	$Draw$	$Lost$
0.005	100	338	199	3.38	1.99	60	19	21
1.000	100	231	82	2.31	0.82	21	19	60

表 3.1 自动比赛测试结果

事实上，概率下限为 1.0 的版本的球队没有维护完整的信念状态，只是简单地的直接使用每个物体上次看到的位置进行视觉决策，这跟以前球队老的视觉决策系统是等价的。测试结果标明，基于 POMDP 的视觉决策系统对球队整体性能的改进效果是很明显的。

第 4 章　总结与展望

本文通过对马尔可夫决策过程相关基础理论模型及算法的总结和分析，并结合在 Robocup 仿真 2D 这一测试平台上的工作主要取得以下结论及成果：

1. 对仿真 2D 可以作为一个标准马尔可夫决策理论研究及实验平台的确定；
2. 使用 MDP 模型解决了 GoToPoint（GoToOrigin）智能体行为规划问题；
3. 设计并完成了基于 POMDP 模型的视觉决策系统。

目前还存在的问题和缺陷包括：

1. 求解 GoToPoint（GoToOrigin）智能体行为规划问题时，网络收敛太慢；
2. 目前的视觉决策系统只能处理视界为 1 的简化情形。

下一阶段将主要针对上述问题和缺陷展开研究，试图解决这些问题和缺陷。本文已有的研究工作和取得的实验效果表明 MDP/POMDP 在求解大规模不确定环境下的多值能体行为规划问题上具有较好的前景，未来将继续就这一主题进行更深入的研究。

参考文献

[1] Nils J. Nilsson: *Artifficial Intelligence: A New Synthesis*,
China Machine Press, Beijing, 2000

[2] Tom M. Mitchell: *Machine Learning*,
China Machine Press, Beijing, 2003

[3] Richard S. Sutton and Andrew G. Barto: *Reinforcement Learning: An Introduction*,
MIT Press, Cambridge, MA, 1998

[4] Mance E. Harmon and Stephanie S. Harmon: *Reinforcement Learning: A Tutorial*,
Wright-Patterson AFB, OH, 45433

[5] Gerhard Weiss: *Multiagent System: A Modern Approach To Distributed Artificial Intelligence*,
The MIT Press Cambridge, Massachusetts London, England

[6] Mathijs de Weerdt, Adriaan ter Mors, and Cees Witteveen: *Multi-agent Planning: An introduction to planning and coordination*,
Dept. of Software Technology, Delft University of Technology

[7] Leslie Pack Kaelbling: *Planning and acting in partially observable stochastic domains*,
AJU, 1998

[8] 宋志伟: *基于逻辑马尔可夫决策过程的关系强化学习研究*,
中国科学技术大学博士学位论文

[9] 范长杰: *基于马尔可夫决策理论的规划问题的研究*,
中国科学技术大学博士学位论文

[10] 吴锋：*多主体系统中的对手建模和策略协作*,
中国科学技术大学学士学位论文

[11] 宋志伟，陈小平：*仿真机器人足球中的强化学习*,
机器人，25(7):761-766, S, 2003