

國立臺灣大學電機資訊學院資訊工程學系

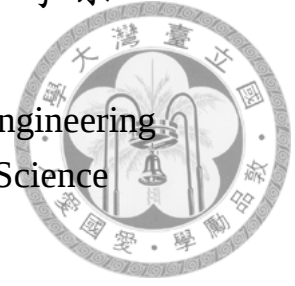
碩士論文

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis



具達成內部平衡之決策模型的使用者感知自主服務型
機器人

A Homeostasis Based Decision Making System on
Human-Aware Autonomous Service Robot

林敬

Ching Lin

指導教授：傅立成博士

Advisor: Li-Chen Fu, Ph.D.

中華民國 105 年 7 月

July, 2016



國立臺灣大學碩士學位論文
口試委員會審定書

具達成內部平衡之決策模型的使用者感知自主服務型
機器人

A Homeostasis Based Decision Making System on
Human-Aware Autonomous Service Robot

本論文係林敬君(學號 R03922121)在國立臺灣大學資訊工程學系完成之碩士學位論文，於民國 105 年 7 月 29 日承下列考試委員審查通過及口試及格，特此證明

口試委員：

傅 立 成

(指導教授)

李 沐 山

蘇 木 春

陳 平 君

徐 國 銳

系主任

趙 坤 茂





誌謝

兩年的碩士生活要結束了，在台大待了七年終於也要離開了。我之所以能夠畢業，首先要感謝傅立成老師給我們足夠的空間尋找自己的論文題目。老師並沒有根據自己的喜好直接指派題目，而是讓研究生們在學習中自行探索，找尋自己研究的方向。在學生迷失的時候，老師偶爾也會出來擔任指路燈，讓學生們在無際的黑暗中屏著微薄的光芒前進。如此一來論文才有一種是「自己的」實感。

感謝士桓學長，在我們遇到問題時請教他都能憑著經驗為我們解惑。實驗室的大小雜事也常常是他處理，給我們更多時間進行研究。士桓學長畢業後想必實驗室的學弟妹會很辛苦吧。感謝趙妍、時安和江元，有一群一起努力的伙伴才能夠分擔痛苦，互相砥礪，若不是他們我可能中途就放棄了。感謝瑞紘在我指派工作給他時便毫無怨言的接下，且隨時接受傳喚，拍攝影片時不但擔任演員還熬夜負責剪輯，我想我還欠他一頓飯。感謝宜倫在我最忙的時候協助我整合計劃的內容，若不是妳幫我分擔整合工作我想我論文到現在還是寫不完。感謝所有學弟妹們任我們呼來喚去，跑腿打雜等等，我能畢業都是靠大家的功勞。

經過這次的磨練，我想我們在各方面，尤其是抗壓性上有了不少的成長，想必可以成為我們未來人生的助力吧。希望畢業後能夠一帆風順，也預祝學弟妹們畢業順利。





摘要

服務型機器人需要能夠選擇行爲，在缺乏使用者命令下自行進行決策，甚至主動提供服務，才能被稱爲「自主」。對於掃地機器人、取物機器人等單用途機器人而言，由於它們有個明確的目標，因此可以人工建構一個完整的決策模型作爲它們的行爲準則。但是對於多用途機器人而言，他們的目標較爲曖昧、模糊，甚至沒有明確目標，此時便較難以爲他們建立完整的行爲模型，使得它們的自主性較低。

「內部平衡理論 (homeostatic drive theory)」是一個在社交機器人中常見的決策理論，它使機器人試著維持其內部狀態的恆定，並根據自身的需求選擇行爲。雖然此方法可以提高機器人的自主性，由於此方法忽略了使用者的需求，讓「使用者感知」的能力降低，因此需要調整才能應用於服務型機器人身上。本篇論文將「使用者意圖」以及「使用者回饋」結合至內部平衡理論中，讓決策模型更以使用者爲中心，同時保有機器人的高自主性。機器人的內部需求 (drives) 將轉化爲動機 (motivations)，且機器人將同時考慮自身的動機以及使用者的意圖來決定自身的行爲。機器人每個行爲的效果並非事先定義好的，而是在互動中利用增強式學習 (reinforcement learning) 所得，使得機器人對於環境以及使用者的先前知識的需求都能降到最低。此決策模型於模擬環境中進行測試及訓練，並將機器人在模擬環境中所學知識轉移至真實的機器人進行實地測試。結果顯示機器人在滿足使用者需求的同時也能夠維持自己體內的恆定，提升自主運作時間，同時達成高自主性以及使用者感知能力。





Abstract

For a service robot to reach high autonomy, it should choose what to do, make its own decisions without user command, and even provide service to the user proactively. For single purpose robots, such as object fetching robots or cleaning robots, since a specific goal is given to each of them, the well-structured decision processes could easily proceed, and decision about that task could be made. However, for robots with vague goals or no specific goal at all, such as caring robots or personal service robots, it is harder to construct a general purpose decision process for them, lowering their autonomy. Homeostasis drive theory is a dominating psychological approach in decision making for social robots. A robot adopting this theory would try to maintain its internal status, and act according to its own need. While achieve better autonomy, this approach ignores the needs of its human user, resulting in low degree of human awareness. This work integrated human intention and human feedback into a homeostasis based system, making the decision process more user-centric, while maintaining high autonomy. The robot's internal needs (drives) generate motivations, and the robot will choose its actions considering both the need of the user and its own motivation. The effects of its actions are not predefined and are learned during interactions by reinforcement learning, making the system require little prior knowledge about the user. The proposed system has been tested in simulations and on a real robot. The results show that the robot can not only satisfy its own needs but also serve the user proactively.





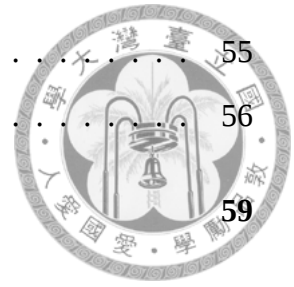
Contents

口試委員會審定書	iii
誌謝	v
摘要	vii
Abstract	ix
1 Introduction	1
1.1 Background and Motivation	1
1.1.1 Service Robots	1
1.1.2 Autonomy	2
1.1.3 Human Awareness	3
1.2 Objectives	3
1.3 Related Work	5
1.4 System Overview	7
1.5 Thesis Organization	8
2 Preliminaries	9
2.1 Markov Models	9
2.1.1 Markov Decision Processes	10
2.2 Reinforcement Learning	12
2.2.1 Standard Modeling	12
2.2.2 Value Functions and Action-Value Functions	13

2.2.3	Q-Learning	15
2.3	Intention Recognition	16
3	Methodology	17
3.1	Terminologies	17
3.1.1	Drive	17
3.1.2	Stimulus	18
3.1.3	Motivation	19
3.1.4	Environment	20
3.1.5	Action	21
3.2	System Model	22
3.2.1	Internal and External States	24
3.2.2	State Reduction	25
3.2.3	Object-Q Learning	27
3.2.4	Selecting Action	27
3.2.5	Reward and Feedback	28
3.2.6	Proposal and Pseudo Update	30
3.3	System Design	31
3.3.1	Drives and Motivations	31
3.3.2	Objects, Stimuli and Actions	35
3.3.3	Degree of Dedication	36
4	Evaluation	39
4.1	Evaluation Metrics	39
4.2	Simulation	42
4.2.1	Setup	42
4.2.2	Result	46
4.2.3	Effects of Motivation Factor	49
4.2.4	Effects of pseudo update	51
4.2.5	Effects of Degree of Dedication	52



4.3	Field Test	55
4.3.1	Robot Platform	55
4.3.2	Result	56
5	Conclusion	59
	Reference	61







List of Figures

1.1	Overview of the decision making system	7
2.1	The graphical presentation of the Markov Decision Process.	11
2.2	General frameworks of reinforcement learning techniques	13
2.3	Bayesian network used to recognize user's intention.	16
3.1	Relation between system parameters.	17
3.2	Flowchart of the decision making system	23
3.3	Weights of drive under different <i>Degree of Dedication</i>	38
4.1	Value of drives in the simulation process	48
4.2	The change in HSR and RSR in the simulation process	48
4.3	Effect of motivation factor on different metrics.	50
4.4	Effect of pseudo update on different metrics.	53
4.5	Effect of <i>Degree of Dedication</i> on different metrics.	54
4.6	The Pepper robot.	55





List of Tables

2.1	Different Types of Markov Models	10
3.1	Selected Internal Variables	34
3.2	Selected Stimuli	36
3.3	Selected Actions	37
4.1	The Effects of Action in Simulation	43
4.2	Increase Rate of Drives in Simulation	43
4.3	Predefined Intentions in the Simulation	44
4.4	Additional parameters used in simulation	46
4.5	Various metrics in the end of simulation	49
4.6	Mapping from temperature status to <i>NRes</i>	56
4.7	Outcome of $DoD = 0.25$	58
4.8	Outcome of $DoD = 0$	58





Chapter 1

Introduction

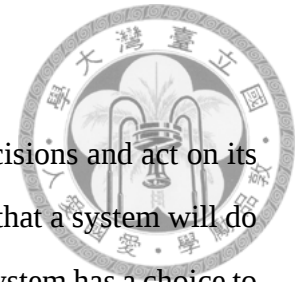
1.1 Background and Motivation

1.1.1 Service Robots

Throughout the past decades, the research and applications of robots have shifted from industrial robots to service robots [1]. Domestic service robots have always been a vision of ours, and the presence of them could often be seen in our imagination of the future, such as science fiction books and movies. The attempts on domestic service robots in early years are usually rather simple and single purpose, with possibly the most famous example being Roomba [2]. These single purpose robots have their own special goal, and they plan their actions around that goal (or not, since some are purely reactive agents such as early Roombas.)

In recent years, many try to develop service robots of more general purposes, such as healthcare robots [3, 4], office robots [5, 6], home service robots [7], personal robots [8, 9], etc. These service robots have designated working environments, however, unlike a task-specific service robot, lack specific goals. Their purpose is to “sense, think, and act to benefit or extend human capabilities and to increase human productivity” [10], yet lacking a specific goal within makes them hard to act on their own and often have to wait for human orders, lowering their autonomy.

1.1.2 Autonomy



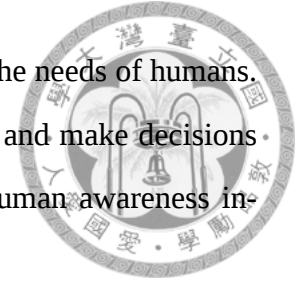
Autonomy, different from automaticity, is the ability to make decisions and act on its own. According to the definition by Clough [11], “automatic means that a system will do exactly as programmed, it has no choice. Autonomous means that a system has a choice to make free of outside influence, *i.e.*, an autonomous system has free will”, which indicates that an autonomous robot should be able to sense the environment and choose its actions according to it. Bekey [12] and Cañamero [13] also gave similar definitions. Thus, in order to achieve a higher level of autonomy, the robot should be able to choose what to do at each moment according to the status of environment at the time, instead of waiting for user’s orders or input. In other words, the robot should have its own decision making ability. This interpretation also matches the statements in [14].

To achieve autonomy, one popular approach is to find inspiration from animals or human [15]. The behavior of living creatures are studied *neurobiologically* and *psychologically*. The goal is to have human/animal behaviors studied, modeled, and implemented on agents, giving them full autonomy. This kind of *biological* approach is widely used in the control of robots, social robotics, and cognitive robotics.

In social robots, a dominant psychological approach is the *homeostatic drive theory* [16]. In this theory, an agent is modeled to have internal *needs*, and the goal of the agent is to satisfy those needs in order to maintain a stable internal state. While this approach does give social robots a goal based on which to choose their actions, and keep the robot content; the needs of its human user wasn’t taken into consideration. Since the main feature of social robots is the ability to communicate and interact with humans/agents socially, the ability to *serve* isn’t necessarily a high priority. However, if we wish to retain the characteristic of “sense, think, and act to benefit or extend human capabilities and to increase human productivity” from service robots, the system should make decisions based on not only its own internal needs, but also the need of its human user. This leads to a higher degree of *human awareness*.

1.1.3 Human Awareness

Human awareness is the ability to be aware of the presence and the needs of humans. The robot should be able to understand the user to a certain degree and make decisions taking human into consideration. The skills required to achieve human awareness include [17]:



1. *human-oriented perception*: human detection and tracking, gesture and speech recognition, etc.
2. *user modeling*: understanding human behaviors and making appropriate decisions.
3. *user sensitivity*: adapting behavior to user, measuring user feedback, and recognizing human state.

Accordingly, the abilities to understand the need of the user, make decisions upon it, measure user feedback and adapt to the user's preference are all important to achieve human awareness. We also believe these are also important traits to build a successful autonomous service robot.

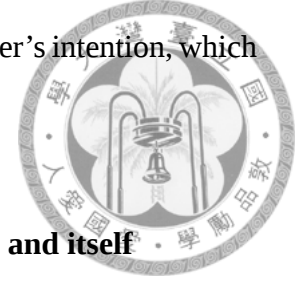
1.2 Objectives

Combining the statements in Section 1.1, the conceived ideal form of service robot is the one that have both high level of *autonomy* and *human awareness*. This work aims to propose a decision making system for service robots that chooses actions autonomously, and serves the user proactively. To achieve this, there are three important aspects of this system:

- **To understand the user's need**

In order to serve the user proactively, first the user's need must be understood. Other than directly commanding the robot, the user might express his/her intention through body language, context of dialog, movements and/or other subtle features in an interaction. These features, expressed intentionally or unintentionally, could be gathered, and the underlying intention could be extracted through a recognition process.

A Bayesian network based intention recognition model have been proposed by our lab previously [18], and it has been adopted to understand the user's intention, which is taken into consideration when making decisions.



- **To make decisions autonomously considering both the user and itself**

To achieve high level of autonomy, biologically inspired approaches are often used. Although the psychology based *homeostasis drive theory* is a dominating approach in social robot genre and indeed endows high autonomy, the underlying *self-centricity* makes it hard to be adopted on service robots at first glance. The research in this thesis, inspired by this theory and works such as [19, 20, 16], aim to propose a system that's both homeostatic and human-aware. That is, the decision making system should consider the human intention while trying to maintain its internal needs. In this way, the robot could benefit from the autonomy introduced by homeostasis, while retaining the role as a service robot.

- **To learn and adapt to user's preference through interactions**

Without preference of the user predefined in the robot's prior knowledge, it is very likely that the decisions made by the robot are not the best choice, or even acceptable to the user at first. This could be solved by injecting a large amount of user profile before the robot's execution. However, this solution is not used in this work, as it requires extensive manual parameter tuning, and is generally not adaptable to changes. Moreover, the user profiles are not usually available beforehand. The robot should learn the correct actions corresponding to user's intention through interactions, and receive user feedback as indications.

With these abilities, the system should be able to make decisions autonomously without predefined goals. It requires little prior knowledge about its user, yet is able to serve the user proactively according to his/her current need through interactive learning.

1.3 Related Work

The methods in decision making are various, with popular ones being Bayesian models, Markov models and their variations. Elinas *et al.* [5] used a factored Partially Observable Markov Decision Process (POMDP) to model the behaviors and the decision theoretic planner of a visually guided interactive mobile robot. The difference in time scales between execution layer and deliberative layer was considered using the factored POMDP, and a heuristic was also proposed to speed up the solving process. Feyzabadi *et al.* [21] proposed a hierarchical solution to the constrained Markov decision process (CMDP) problem. They partitioned the state space of the original CMDP into multiple clusters, solved the smaller abstract CMDP, and projected the abstract policy back to the original space. This method tackles sequential decision problems with multiple objectives, and is implemented in a path planning scenario. Omidshafiei *et al.* [22] used Decentralized POMDP (Dec-POMDP) to solve multi-robot planning problems in continuous spaces. Actions were abstracted into macro-actions, simplifying the original Dec-POMDP problem, making it solvable using discrete methods. Liu *et al.* [23] introduced Episodic Memory-driving MDP (EM-MDP) to solve planning problems. They used state neurons and episodic memories to store learned experience, reducing the high-dimensionality of observation, and is easier to solve than the traditional POMDP model. Zhang *et al.* [24] proposed a combination of logical reasoning and POMDP planning in a dialog system. The prior knowledge of commonsense was used to filter the possible words. Then, a method called P-log was used to extend the logical reasoning results to probabilities, and the probabilities was transferred to POMDP as the current belief of states. The proposed method was shown to be more efficient and accurate than POMDP without commonsense reasoning.

Although these models are popular in robotics, they are more of a planning method of a given task. Without a given task or a well crafted model, these method become somewhat unsuitable. Other than theses models, there are still different approaches. Ko *et al.* [25] adopted *confabulation theory*, using symbols to represent the environment contexts, and wills and behaviors of an virtual agent. Behaviors were first filtered by the dominant wills

and the perceived contexts, then evaluated using Choquet fuzzy integral. Smith *et al.* [26] introduced a fuzzy multi-objective decision making system for the motion control of a mobile robot.

As mentioned before, the *biologically inspired* approaches, where researchers tried to mimic the mind of human *neurobiologically* or *psychologically*, are also popular recently. Maes *et al.* [27] proposed ANA architecture, where the behavior selection and motivation competitions were done using a neural-network based system. Bellas *et al.* [28] introduced Multilevel Darwinist Brain (MDB) architecture, where artificial neural networks (ANNs) were used to represent the world model, internal model, and satisfactory model of the robot. The three models worked together to evaluate the behaviors, and the selected behavior was determined by the output of the satisfactory model. The system of Ando *et al.* [29] sensed the environment and generated urges. Urges competed through predefined priorities, and the dominant urge drove the compensating action. Hoefinghoff *et al.* [30] used somatic markers on each stimuli-action pair. The meaning of a somatic marker is the possible emotion of choosing that action given the stimuli. The somatic markers are used to filter out inappropriate actions, and the remaining actions are chosen randomly. Wilson [31, 32] tried to consider the morality of the actions of the robot, and proposed to choose actions based on utilities that is modified to reflect the moral effect.

In psychological approaches, *homeostasis drive theory* based systems are one of the dominating methods. The term *homeostasis* was first introduced by Cannon [33], and was described as a regulatory system to maintain the body in a stable physiological state. After being adopted by robot decision making systems, it usually means that the agent has several internal needs that need to be checked and maintained. When a need become unsatisfied, a *drive* will be created, triggering appropriate correcting actions. Many famous robots adopted this method, such as AIBO [34] or Kismet [35]. Cao *et al.* [16] introduced ROBEE architecture, where internal needs generate drives. A predefined satiator, a list of preconditions-action pairs, was used to satisfy the highest drive. Cañamero *et al.* [36, 13] introduced motivational states, the intensity of which was affected by both the internal drives and external stimuli. After the intensity of motivations were obtained, the inten-

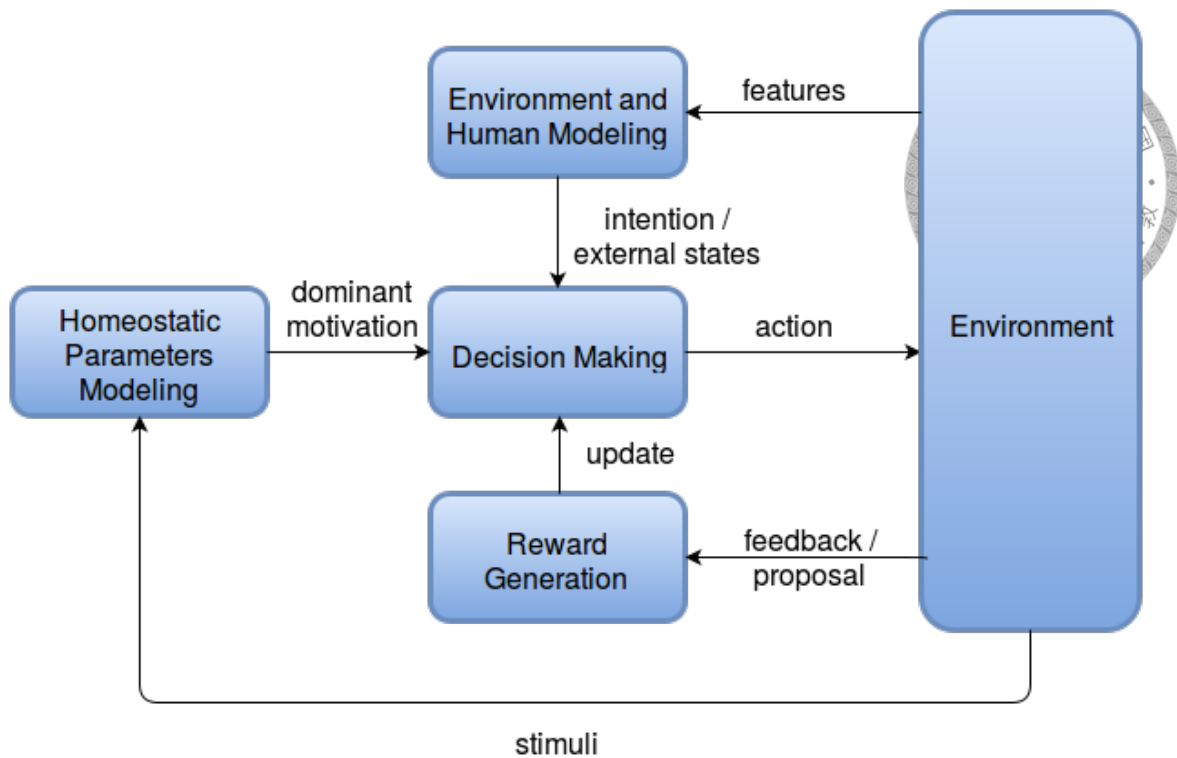


Figure 1.1: Overview of the decision making system

sity of behaviors were calculated accordingly. The behavior with the highest intensity will be the selected one. Gandaho *et al.* [37, 38] associated robot’s goal with homeostatic variables, and used reinforcement learning on the robot’s adaptive controller. Inspired by both Cañamero and Gandoho, Castro-González [19] used drives to model the robot’s internal needs, which, along with stimuli, trigger the robot’s motivations when unsatisfied. The motivation were competed through intensities, and an action was selected trying to match the dominant motivation. The correlations between actions and motivations were not predefined, and must be learned in the interactions. This work, largely influenced by [19], used similar formulation of the robot’s internal variables, while introducing the human user’s intention and feedbacks into the decision system, making the robot more human-aware and able to serve human proactively.

1.4 System Overview

This work, inspired by [20] and such, models the robot’s internal needs as *drives*, which generate *motivations* when needs are unsatisfied, since homeostasis requires the drives to

be kept in range. The intensity of each motivations are calculated, which would then be used to determine the dominant motivation. The dominant motivation would become the internal state of the robot. On the other hand, the robot will extract the status of the environment and recognize the user's intention, passing them to the action selection module as external state. The action selection module, combining both internal and external state, will choose an action accordingly. After the execution of the chosen action, the robot will calculate the effects of the action, receive feedback from the user, and update the model to adapt to the user's preference. Figure 1.1 shows the overview of the proposed system.

The main difference of this work and related works such as [20, 16] is that the user is explicitly modeled into the decision making process, rather than treated as a general object. In other works, the robot behaves solely to achieve homeostasis, and the user has little means to affect the robot's decisions. In this work, the user can directly affect the robot's decisions through expressing intention and giving feedback. The robot will choose its action considering both homeostasis and the user's need.

1.5 Thesis Organization

The rest of this thesis is organized as follows. In Chapter 2, the mathematical tools used in this work, such as Bayesian network, reinforcement learning and its based formulation Markov decision process, are described. The adopted intention recognition method is also described here. In Chapter 3 the terminologies, model formulation of the system, and system design details are described. In Chapter 4, the author proposed several evaluation metrics to measure the performance of the system. Both the simulation results and field test outcomes are shown here. Finally, Chapter 5 concludes the whole thesis.



Chapter 2

Preliminaries

2.1 Markov Models

The Markov model is a statistical tool used to represent the system or the stochastic process's behavior with temporal information. It is a model that assumes the Markov property, which especially assumes future states of the system only depended upon the present state; that is, the present states characterize all the necessary information of the past events and thus enable the reasoning with the model to be tractable. Markov Models can be divided into four different types depending on whether those system are observable, controllable or not, as shown in Table 2.1. Firstly, if the system's states are fully observable and changing spontaneously, then the system is modeled as *Markov chains*. Secondly, the system can be modeled as *Dynamic Bayesian Networks (DBN)*, a generalized hidden Markov model, if the states of this system cannot be fully observed but still be changing spontaneously. On the other hand, if the transition of system's states is held in the system's hand and is fully observable, we call this model as *Markov Decision Processes (MDPs)*. Finally, if states of system are not fully observable and is controlled by the system itself, we define this model as *Partially Observable Markov Decision Processes (POMDP)*. The POMDP model is the most complex architecture due to consideration of the uncertainty and system's decision simultaneously.

Table 2.1: Different Types of Markov Models

	<i>Fully observable</i>	<i>Partially observable</i>
<i>System is autonomous</i>	Markov Chains	Dynamic Bayesian Networks (DBNs)
<i>System is controllable</i>	Markov Decision Processes (MDPs)	Partially Observable Markov Decision Processes

2.1.1 Markov Decision Processes

In Markov Decision Processes (MDPs), the agent fully observes the current state and decides an action to perform. The next state to which the process transfers depends on the current state and the system's action. Hence, a Markov decision process is a quadruple:

$$MDP = \langle S, A, T, R \rangle \quad (2.1)$$

where

- S is a finite set of system's states, describing information of the environment that agent concerns,
- A is a finite set of system's actions,
- T is the transition probability of system's states. As a result of executing action $a \in A$ in state $s \in S$, the environment transitions to state $s' \in S$ with probability $T(s, a, s')$. It is worthy noting that each transition in MDPs is defined as a non-deterministic one.
- R is the reward function. After the system state changes to the next state, the environment responds with an expected reward r , where $r \in R$, $R : S \times A \times S \rightarrow \mathbb{R}$ is a bounded function.

A comprehensive illustration of those relationship is shown in Fig. 2.1. To deal with noisy and incomplete state information, the basic MDP framework can be extended to Partially Observable Markov Decision Processes (POMDPs), where states of the system

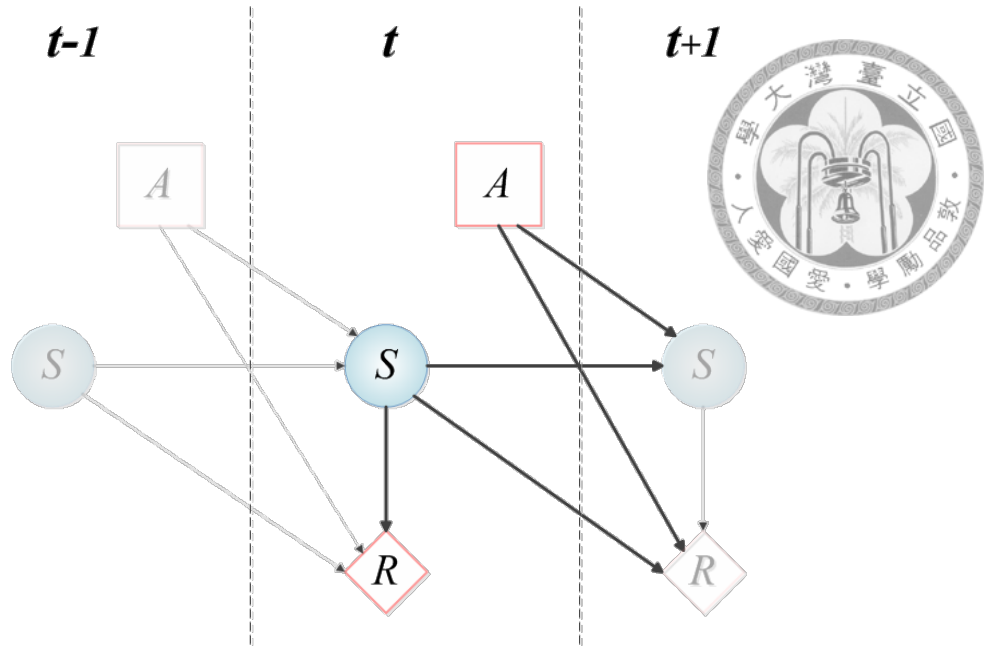


Figure 2.1: The graphical presentation of the Markov Decision Process. The transition of the system is depended on the action A , and the selection of the action is determined by the reward function R .

are represented as hidden states and must be inferred from system's observations and actions, causing the state space become too large to solve the optimal policy efficiently. For a complete introduction of POMDPs, please refer to [39].

Given an MDP, the objective is to construct a policy $\pi : S \rightarrow A$ that maximizes the expected future accumulated reward from each state s . The agent then chooses its appropriate actions according to the policy. This policy, comparing to the immediate received reward, is determined based on the desirability to the goal, which is shaped by the reward function R , in the long run. When decisions are made or evaluated following the policy, the values of action choices are concerned. The expected return of following a policy π from a state s is defined by the value function as shown below:

$$\begin{aligned}
 V_{\pi}(s) &= E_{\pi}[r(t) + \gamma r(t+1) + \gamma^2 r(t+2) + \dots | s_t = s] \\
 &= E_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k \cdot r(t+k) | s_t = s\right] \\
 &= E_{\pi}[r(t) + \gamma V_{\pi}(s_{t+1}) | s_t = s]
 \end{aligned} \tag{2.2}$$

where t is the current time step and r_t is the reward received at the time step t . This quantity $V^\pi(s)$ is called the value of the state s under the policy π . The future rewards are discounted by a factor γ so that the recent returns are emphasized more. Those can be computed using dynamic programming methods, such as value iteration or policy iteration [40, 41]. However, though the optimal policy of MDPs can be solved by dynamic programming methods, this requires the specification of all the parameters of MDPs. In real world applications, those parameters, especially the transition probability T , are too vague to be defined clearly due to complexity and uncertainty of our world. Therefore, the reinforcement learning is proposed to deal with this problem; specifically, instead of solving the optimal policy directly, we learn it through the interaction between agents and the environment.

2.2 Reinforcement Learning

Reinforcement Learning (RL) [42] is usually known as goal-directed learning methods to deal with the problem that can be modeled as a Markov Decision Process. The agent is not instructed what to do but should discover the actions which lead to the most profits. To be more specific, reinforcement learning is the learning that maps situations to actions so that by following the learned policy, the agent collects the maximum rewards. Chain effect is a tricky part of reinforcement learning problem where actions not only affect the current reward but also influence the subsequent situations and future rewards. The agent generally learns under exploration and exploitation and figures out how to take actions in the environment to maximize the long-term returns.

2.2.1 Standard Modeling

The standard modeling of a reinforcement learning problem is shown in Figure 2.2. There are an agent and an environment. The learning process proceeds as follows: at each time t , the environment is in a state s_t , and then the agent observes the state of environment s_t and deliberately selects an action a_t for execution. The action results in

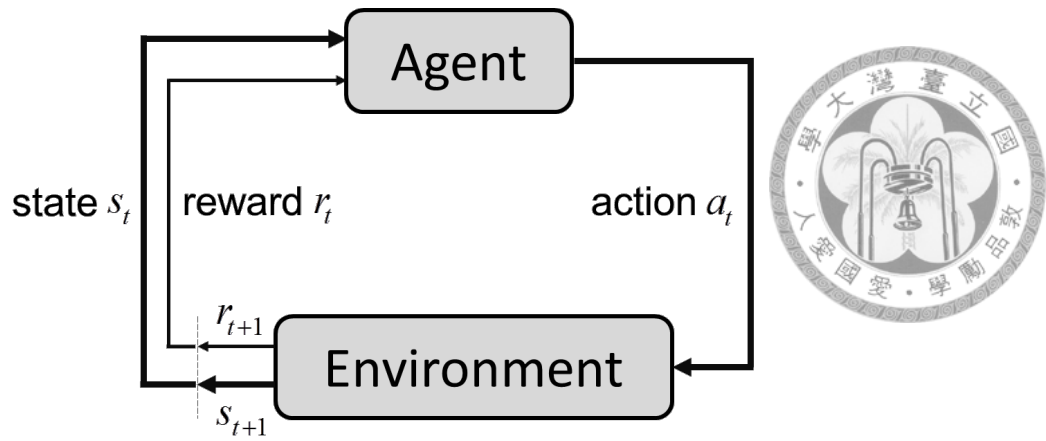


Figure 2.2: General frameworks of reinforcement learning techniques

the state transition of the environment from s_t to s_{t+1} at time $t+1$. The agent then receives the reward r_{t+1} .

Under the global view of the learning agent and the environment, basic reinforcement learning models consist of the two elements of MDPs: a set of system's states S and a set of actions A . The major difference between dynamic programming methods and reinforcement learning techniques is that RL requires the agent to observe the transition of the system and to study its action's influence on the system's state transition, while the dynamic programming method must specify the detail of the transition probability in prior and solve the problem offline. RL is referred to a kind of interactive learning method.

2.2.2 Value Functions and Action-Value Functions

Since the objective of reinforcement learning techniques (or generally, the MDP problem) is to learn the optimal policy that maximizes the expected future accumulated reward by mapping each state to the agent's action, and the value function introduced in Section 2.1.1 serves this purpose. It memorizes the experience that the learning agent had and indicates how likely the states can reach the agent's goal. In this section, we introduce two basic ways to represent the experience of the learning agent in reinforcement learning techniques.

The first one is the *value function*, described in Eq. (2.2). With the knowledge of the transition T and reward functions R , this function can be re-written in the form of Bellman

equation:

$$V_{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s'} T(s, \pi(s), s') V_{\pi}(s') \quad (2.3)$$

The maximum value that can be obtained by a policy is usually denoted as V^* , which is defined as:

$$V^*(s) = \max_a (R(s, \pi(s)) + \gamma \sum_{s'} T(s, \pi(s), s') V^*(s')) \quad (2.4)$$

Similarly, another representation called *action-value functions*, often simplified as Q function, describes the expected long-term return of taking an action a in a state s under the policy π , and the optimal action-value function denoted as Q^* is described below:

$$Q_{\pi}(s, a) = R(s, a) + \gamma \sum_{s'} T(s, \pi(s), s') V_{\pi}(s') \quad (2.5)$$

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s'} [T(s, a, s') \max_{a'} Q^*(s', a')] \quad (2.6)$$

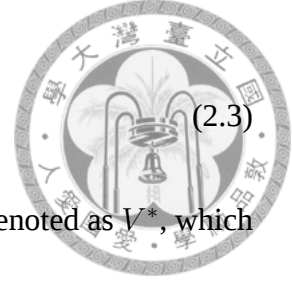
Furthermore, due to the fact that the optimal policy always chooses the action with the maximum action value, the relationship between value and action-value are expressed as:

$$V^*(s) = \max_a Q^*(s, a) \quad (2.7)$$

Once the learning agent obtains the optimal action-value function, it is obvious that the agent can easily take the greedy strategy to choose the best action based on the highest action-value they will receive in the next step, as shown by Eq. (2.7); that is, the optimal policy that follows Q^* is derived as the following form:

$$\pi^*(s) = \arg \max_a Q^*(s, a) \quad (2.8)$$

It is worth to note that either the value function V or action-value function Q require the transition probability T to compute their values. In the next section, the Q-learning al-



gorithm which does not require the specification of transition T will be introduced briefly.

2.2.3 Q-Learning

Q-learning [43] is often used to find an optimal action-selection policy for a given Markov decision process (MDP). It is a model-free reinforcement learning technique since the model of the environment is not required by the algorithm. This algorithm works by learning the action-value function Q (see Eq. (2.5)) through the interaction between the learning agent and the environment. The procedure of Q-learning in iteration n is presented as below:

- observes the environment's current state s_n ,
- selects and performs an action a_n ,
- observes the subsequent state s'_n ,
- receives an immediate reward r_n , and
- adjusts its Q_{n-1} values using a learning factor α_n , according to the update function:

$$Q_n(s, a) = \begin{cases} (1 - \alpha_n)Q_{n-1}(s, a) + \alpha_n[r_n + \gamma V_{n-1}(y_n)] & \text{if } s = s_n, \\ Q_{n-1}(s, a) & \text{otherwise} \end{cases} \quad (2.9)$$

where the initial action-value $Q_0(s, a)$ is assumed given for all states and actions, and the function did not require the specification of the transition probability T .

This algorithm has been shown that it will converge correctly if the action-value function is represented via a look-up table representation [43]. More generally, Q-learning can be combined with function approximation. This may speed up the learning process and let the algorithm be able to deal with problems in continuous space.



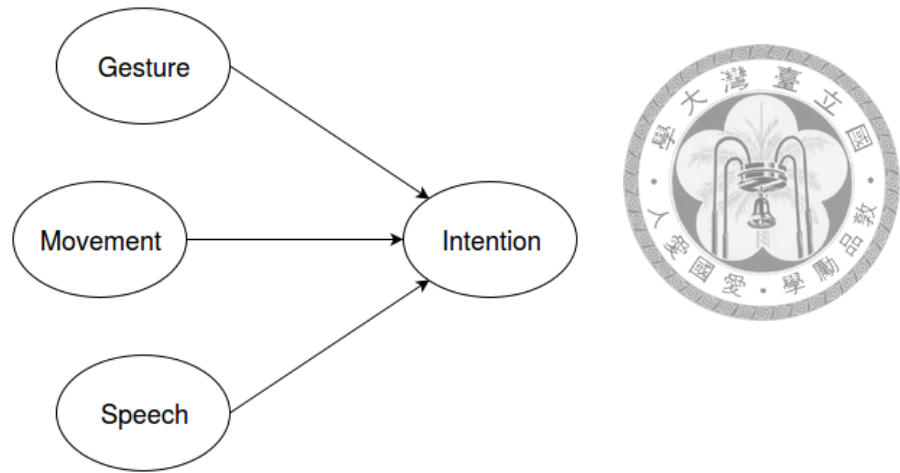


Figure 2.3: Bayesian network used to recognize user's intention. The parameters are updated using EM algorithm if user feedback is received.

2.3 Intention Recognition

In order for the robot to achieve human-awareness, first it have to understand human behaviors and make appropriate actions accordingly [44]. One way to accomplish this is to recognize user intention and take this intention into consideration when making decisions.

This work adopts the intention recognition method from [18], using human user's gesture, movement, and spoken sentence as features and construct a Bayesian network accordingly. Figure 2.3 shows the Bayesian network used in this work, and the recognized intention is calculated as:

$$hi_{recog} = \arg \max_{hi \in HI} P(hi | z) \quad (2.10)$$

where hi is an *human intention*, and z is the observations about human.

To adopt to the user's preference, user feedback about his/her intention could be made, and the model will be updated on-line. If the robot fails to recognize the user's intention, or the recognition result is incorrect, the user could make a feedback, clarifying his/her true intention. When the robot receives this feedback, it will update the parameters of the Bayesian network using expectation-maximization (EM) algorithm [45]. In this way, little or no prior knowledge about the user's behavior is needed, and the relation between user's actions and his/her intentions could be learned through interactions.



Chapter 3

Methodology

3.1 Terminologies

According to homeostatic drive theory [46], “homeostasis means maintaining a stable internal state.” This work, inspired by both [20] and [16], uses *drive*, *motivation* and *stimulus* to determine a robot’s internal status. The relation between these parameters are illustrated in Figure 3.1. The meaning and definition of these and other terms used in this system will be described below.

3.1.1 Drive

In this work, the internal *needs* of the robot are modeled as *drives*. Drives are internal parameters of the robot in the form of real numbers. A drive represents a certain need of the robot, such as its battery level or a sense of loneliness, and the value of the drive indicates

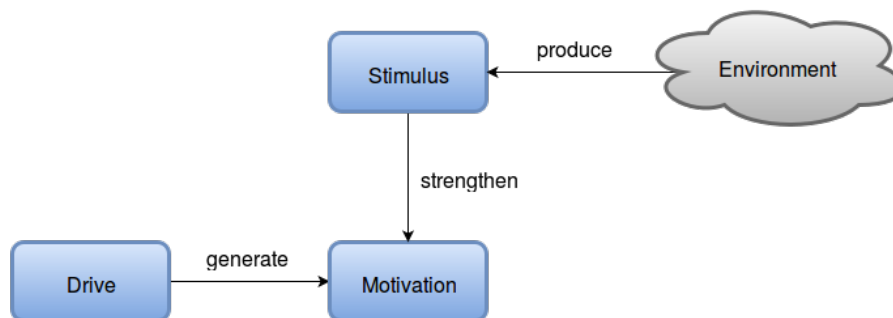


Figure 3.1: Relation between system parameters.

the degree of dissatisfaction of that certain need. The set of drives used in the system is denoted by D . Each drive $d \in D$ is normalized to the range of $[0, 100]$, with 0 meaning the need is fully satisfied (thus no compensating drive) and 100 meaning the need is totally unsatisfied (thus high compensating drive). The value of a drive could be affected by time, the robot's internal status, and the outcome of the selected action. Each drive d is also accompanied with an *activation threshold*. When a drive exceeds its activation threshold, the degree of dissatisfaction is considered high enough, and the corresponding *motivation* will emerge. In this thesis, the term *need* and *drive* are sometimes used interchangeably.

Definition 1. Drives D is a set of internal needs of the robot, the values of which indicate the current degrees of dissatisfaction in different aspects. $\forall d \in D : d \in [0, 100]$.

Definition 2. Activation thresholds AT is a set of thresholds representing robot's tolerances to its needs. There is a bijection mapping from D to AT . That is, for each $d \in D$, there is one and only one corresponding threshold $at \in AT$. In this work, the subscript index i is used to indicate this mapping (*i.e.*, $at_i \in AT$ is the corresponding threshold of $d_i \in D$). $\forall at \in AT : at \in [0, 100]$.

Definition 3. When $d_i > at_i$, d_i is considered *unsatisfied*.

3.1.2 Stimulus

The factors that affect the robot's motivation are twofold, the internal needs — the *drives* — and the external conditions — the *stimuli*. Stimuli are defined as external condition that could affect the intensity of one or more motivations. The set of possible stimuli is denoted by ST . It could be the presence of a certain object, the recognized intention of a human, *etc.* For example, our motivation to eat could be affected by our hunger, an internal drive, and the presence of food, an external stimulus. Similarly, we expect the robot's motivation to serve would be enhanced if the robot sensed the human user's intention. A stimulus could have different effects on different motivations.

Definition 4. Stimuli ST is a set of external conditions that could affect the intensity of motivations of the robot. Given an observed environment, a stimulus $st \in ST$ could be

either present or absent. The set of presenting stimuli is denoted by ST_{exist} , $ST_{exist} \subseteq ST$.

3.1.3 Motivation

Motivations (M) can be considered as the goal of the robot. Motivations are triggered by unsatisfied needs (*i.e.*, drives that exceed activation thresholds) and external stimuli. The relation between drives and motivations are one-to-one, which means that for each drive in the system, a correlated motivation exists. Given the current drive values and the existing external stimuli, the *intensity* of each motivation can be calculated. The intensity of a motivation represents its strength. The higher the intensity is, the more likely that motivation will prevail. For the motivation with the highest intensity, it is considered *dominant*, and the robot should try to act according to it.

Definition 5. Motivations M is a set of goals for the robot to compensate for a unsatisfied drive. There is a bijection mapping from D to M , that is, for each $d \in D$, there is one and only one corresponding $m \in M$. In this work, the subscript index i is used to indicate this mapping (*i.e.*, $m_i \in M$ is the corresponding motivation of $d_i \in D$).

Definition 6. In every iteration, the *intensity* of the motivations could be calculated by the drive values and the stimuli of that time. A motivation is considered *activated* if its intensity is greater than 0.

Definition 7. In every iteration, the dominant motivation m_{dom} is chosen according to the intensities of all motivations. m_{dom} determines the internal state of the robot.

According to Lorenz's hydraulic model [47], internal drive strength interacts with external stimulus strength. If the drive is low, then a strong stimulus is needed to trigger motivation; if the drive is high, then a mild stimulus is sufficient [46]. To illustrate this model, Malfaz [20] used the following equation to calculate the intensity of motivation:

$$intensity(m_i) = \begin{cases} 0 & \text{if } d_i < at_i \\ d_i + effect(st_i, m_i) & \text{otherwise} \end{cases} \quad (3.1)$$

Where $intensity(m_i)$ is the intensity of motivation $m_i \in M$, d_i is the corresponding drive, and $effect(st_i, m_i)$ is the effect of related external stimulus st_i on motivation m_i . The physical meaning of equation Equation (3.1) is that the motivation will be activated only if the corresponding need is unsatisfied. Since in the definition of this work, a stimulus could affect several motivations, and a motivation could be affected by several stimuli, the above equation is adjusted to Equation (3.2).

$$intensity(m_i) = \begin{cases} 0 & \text{if } d_i < at_i \\ d_i + \sum_{st_k \in ST_{exist}} effect(st_k, m_i) & \text{otherwise} \end{cases} \quad (3.2)$$

If there are multiple activated motivation, all activated motivations will compete with one another through comparing intensities, and the motivation with the highest intensity will become the *dominant motivation* m_{dom} of that iteration.

$$m_{dom} = \arg \max_{m \in M} (intensity(m)) \quad (3.3)$$

3.1.4 Environment

There is no common solution to the representation of environment in the research of robotics. The design is often up to the purpose of the robot system. While a navigation system focuses on the location of itself and the obstacles [48], a service robot might uses a higher level of representation of objects [49].

In this work, we regard the key aspect for a personal service robot is conceived to interact with different kinds of objects and/or the human user to satisfy to needs of the robot itself or to serve the user. For example, if the robot is low on energy level, it should operate the *charger* to meet its need; when the user's intention is to watch television, the robot should turn on television to provide service. Following this requirement, the environment in this work is composed of one or more *objects* and a *human user*. Each object is assumed to be recognizable to the robot for simplicity in this work, and is described by a set of *object variables*. The object variables describe the status of the object, and the relation between

it and the robot. Similarly, the human user is described by his/her recognized intention, and also the relation with robot.

Definition 8. *Obj* denotes a set of objects recognized and considered by the robot. Each $o \in Obj$ is represented by a set of *object variables*.

Definition 9. There is one *human user* of the robot. The robot will try to recognize the intention, infer the spatial relation, and determine the interaction status of its user.

Definition 10. For each iteration, when the human user is present, the robot will try to recognize the intention of the human user, and choose the most possible one out of all the possible human intentions *HI*. The chosen intention is denoted by hi_{recog} .

3.1.5 Action

The goal of the proposed system is to choose an *action* out of a predefined action set to cope with the current situation. The purpose of an action is to satisfy the robot's needs and/or to serve the human user through its execution. The target of each action could be an object, the human user, or the robot itself. For example, *plug* and *unplug* could only be operated on the charger, while *chat* can only be used on the human user. To group the actions, we can get several subsets according to the target of an action. However, two of the subsets are distinct from the others, namely, the subset whose target is the human, and that whose target is the robot itself. Besides these two subsets, the targets of all the other subsets are objects. To make the notation uniform, we introduce a *null* object, which serves as the target object for all non-object-related actions.

Definition 11. *A* denotes a set of actions that the robot can perform. Each action has an execution target, and A_o denotes the subset of actions of which the target is object o . For actions without target objects — the target of action is the human user or robot itself — a *null* object is defined to serve as their target. As indicated by Equation (3.4), *A* could be seen as the union of A_o s for all $o \in Obj$ plus the *null* object.

$$A = A_{null} \cup \bigcup_{o \in Obj} A_o \quad (3.4)$$

3.2 System Model

After defining the internal parameters of the robot in the previous section, the robot should decide its own actions to keep its drives in check, and achieve homeostasis. In the mean time, the robot should also consider the user's intention, serve the user when applicable, and learn the user's preference through feedback. These are necessary features for a robot to achieve higher-level human-awareness ability [18]. In the work, a Q-learning based decision making system is proposed. The design of this system is highly influenced by [20], and the feedback method described in [18] is integrated into the system.

Following Figure 1.1, a flowchart is shown in Figure 3.2 to illustrate the decision making process. First the drives are calculated, generating motivations. The motivations compete with each other by comparing intensities, and a dominant motivation will be chosen. The dominant motivation will be passed to the action selection module as internal state. On the other hand, the robot will extract the status of the environment and recognize the user's intention, passing them to the action selection module as external state. The action selection module, combining both internal and external state, will choose an action accordingly. After the execution of the chosen action, the robot will calculate the effects of the action, receive feedback from the user, and update the decision making model.

The main difference of this work and the related works such as [20] or [16] is that the user is explicitly modeled into the decision making process, rather than is treated as a general object. In other works, the robot behaves solely to achieve homeostasis, and the user has little ways to affect the robot's decisions. This characteristic is applicable to pure social robots, but is unsuitable if we wish the robot to also have serviceability. In this work, the user can directly affect the robot's behavior through expressing intention and giving feedback. The robot will choose its action considering both homeostasis and the user's need.

In the following section, first the *state* used by the system will be defined in Section 3.2.1, and the state reduction method will be described in Section 3.2.2. To cope with the *collateral effect* in state reduction, a modified version of Q-learning proposed in [19] will be described in Section 3.2.3. We will describe the stochastic action-selection

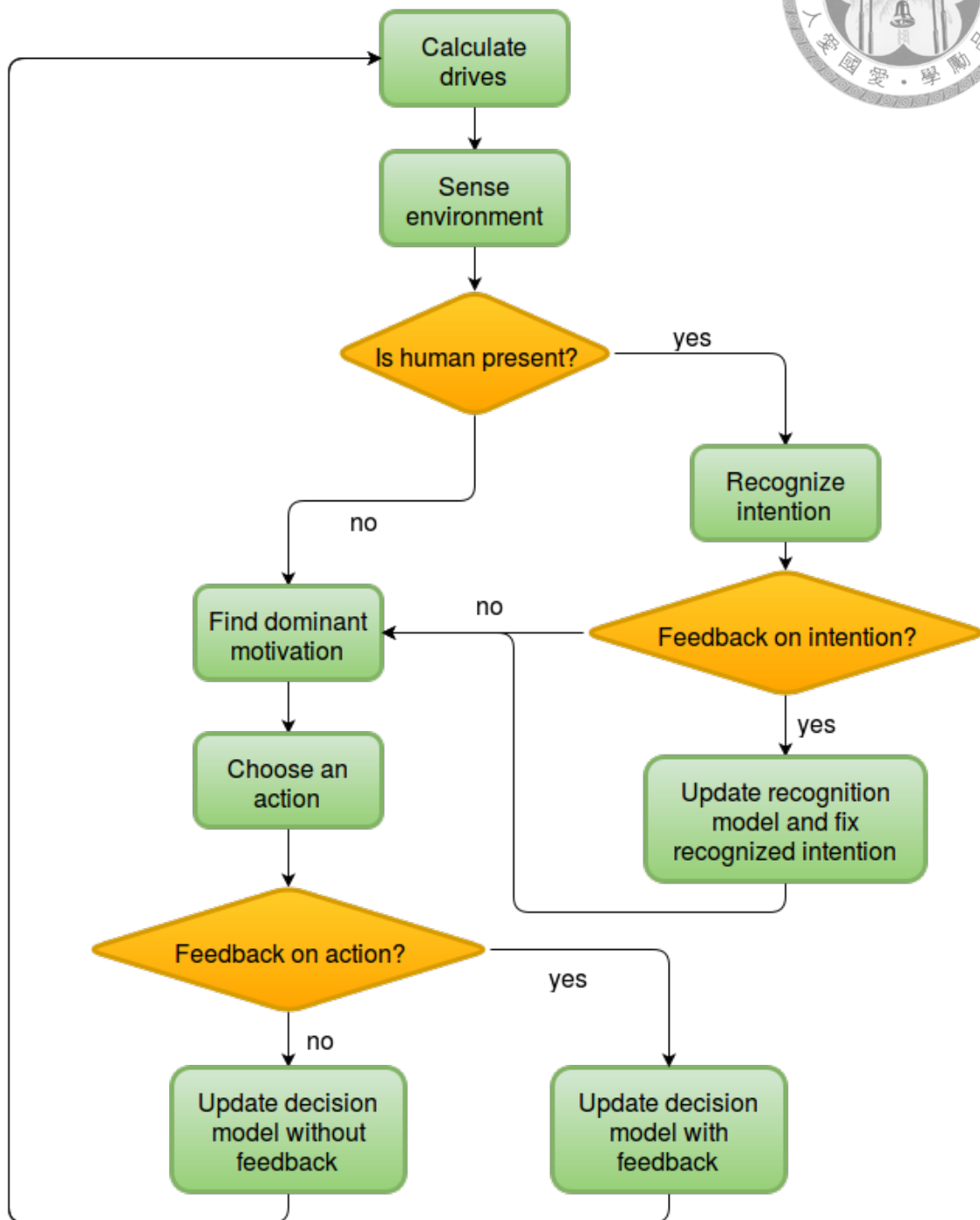


Figure 3.2: Flowchart of the decision making system

method in Section 3.2.4, and the reward function in Section 3.2.5. Finally, a special kind of user feedback called *proposal* and user-induced *pseudo update* will be described in Section 3.2.6.



3.2.1 Internal and External States

In general, the state to be used in the decision making system is composed of the internal state of the robot and the external state of the environmental context, as in Equation (3.5).

$$S = S_{internal} \times S_{external} \quad (3.5)$$

In this work, the internal state is the selected dominant motivation of the robot. The dominant motivation is selected as stated by Equation (3.3). Thus:

$$S_{internal} = m_{dom} = \arg \max_{m \in M} (intensity(m)) \quad (3.6)$$

In [19], the external state is the combination of the states of all the objects in the environment in relation to the robot:

$$S_{external} = \prod_{o \in Obj} S_o \quad (3.7)$$

An object state could contain the spatial relation between the object and the robot, the on/off status of the object, *etc.* Different kinds of objects could have their own definitions of state variables. In Equation (3.7), the state of the human was also treated as an object state. To achieve higher-level of human awareness, we try to integrate the state of the human user into the external state, and thus:

$$S_{external} = S_{hum} \times \prod_{o \in Obj} S_o \quad (3.8)$$

To explicitly model the state of the human (assuming the robot has only one human user), his/her intention and the spatial relation between the user and the robot are consid-

ered. The human state is modeled as follows:

$$S_{hum} = hi_{recog} \times \{near, far, absent\} \quad (3.9)$$

where $hi_{recog} \in HI$ is the recognized intention in Equation (2.10), and $\{near, far, absent\}$ indicate the spatial relation between the robot and the user.



3.2.2 State Reduction

However, as the number of features and objects increase, the state space will grow exponentially. With larger state space, more training data will be required to acquire a usable model. In the case of reinforcement learning such as this work, it means the exploration phase (learning phase) will be exponentially longer to reach a stable decision model, because the Q-value of each state-action pair $Q(s, a)$ would need to be evaluated several times to converge.

Many works had addressed and worked on this problem, such as the ones which use factored Markov Decision Processes (FMDPs) [50, 51]. They represent the complex state space by a finite set of random variables, using a set of dynamic Bayesian networks (DBNs) [52] to express the transition model. Another approach is to perform state aggregation or state abstraction. Li et al. [53] showed several methods for state abstraction. Some require given model structures [54, 55], while some performs aggregation through interaction and construct the abstract states hierarchically [56] or through statistical evaluations [57].

Castro-González et al. [19] proposed a simple yet effective state reduction method for this scenario, and adopted it in their later works [20, 58, 59]. They assumed that the states related to each objects have little effect on each other, and thus could be considered independent of one another. This assumption is based on the observation of human behavior, since “when we interact with different objects in our daily life, one, for example, takes a glass without considering the rest of objects surround.” [19]. As a result, each object states could be considered separately rather than as Cartesian products. Then, Equation (3.7) could be simplified into Equation (3.10).

$$S_{external} = \{S_o \mid o \in Obj\} \quad (3.10)$$

Following this reduction method, Equation (3.8) could be reduced into the Cartesian product of S_{hum} and one of the object states S_o . This product is denoted by $S_{hum,o}$.

$$\forall o \in Obj : S_{hum,o} = S_{hum} \times S_o \quad (3.11)$$

For each $S_{hum,o}$, only the actions that perform on object o (i.e., A_o) are considered in the decision process. In other words, for the Q-value of a state-action pair to exist, the state and the action must related to the same target object. As mentioned in Section 3.1.5, the target of an action could be an object, human, or the robot itself, and a *null* object was introduced to represent the non-object targets. Thus, other than using product of all $S_{hum,o}$ as the external state space, the *null object space* should also be considered to make the notation uniform. The null object space is defined to be an empty space, simply serving as a placeholder. Equation (3.12) shows the definition of the whole external state space, and Equation (3.13) shows the condition for a state-action pair to be considered in the decision making process.

$$S_{external} = S_{hum,null} \cup \bigcup_{o \in Obj} S_{hum,o} \quad (3.12)$$

$$\forall o \in (Obj \cup \{null\}) : \exists Q(s, a) \iff (s \in S_{internal} \times S_{hum,o}) \wedge (a \in A_o) \quad (3.13)$$

The advantage of this reduction method is that the human-object relation could still be preserved, as the intention of the human might be wishing the robot to operate on an object. Viewing objects separately also means that the additional and removal of an object won't compromise the structure of the state space. One could simply add the new object state to the set of external state. If state abstraction method such as [57] is used, since the abstract states are built upon the ground states that contain every object features, changing

state space means that the old abstraction is no longer usable.



3.2.3 Object-Q Learning

Although the object states are assumed independent of each other, the action of the robot *break* this independence. When a robot performs an action, the effect of the action might affect multiple object states. These *collateral effects* are identified and dealt with in [19]. In order to take these effects into consideration, they proposed a modified Q-learning algorithm call *Object Q-learning*. In Object Q-learning, the way of updating Q-value is modified into the following equation:

$$Q^{o_i}(s, a) = (1 - \alpha) \cdot Q^{o_i}(s, a) + \alpha \cdot (r + \gamma \cdot V^{o_i}(s, s')) \quad (3.14)$$

$$V^{o_i}(s, s') = \max_{a \in A_{o_i}} (Q^{o_i}(s', a)) + \sum_{m \neq i} \Delta Q_{\max}^{o_m}(s, s') \quad (3.15)$$

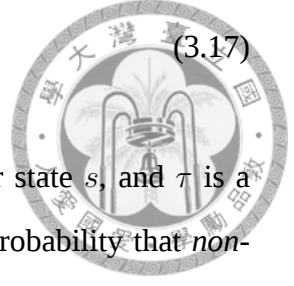
where Q^{o_i} indicate that the Q-value is in relation to the object o_i , and $s \in S_{internal} \times S_{hum, o_i}$ is the reduced state considering object o_i . Action $a \in A_{o_i}$ is the chosen action to be performed on object o_i , s' is the new state in relation to object o_i , r is the reward received, γ is the discount factor and α is the learning rate. $V^{o_i}(s')$ is the value of s' considering the collateral effects on other object states, including the missed and acquired opportunities on other objects after an action.

$$\Delta Q_{\max}^{o_m} = \max_{a \in A_{o_m}} (Q^{o_m}(s', a)) - \max_{a \in A_{o_m}} (Q^{o_m}(s, a)) \quad (3.16)$$

3.2.4 Selecting Action

To choose the best action in object Q-learning, the action with the highest Q-value in the given state, should be chosen. However, always choosing action with the highest value often leads to local optimum, weakening the learning process. To escape local optimum, a certain degree of randomness is often introduced. In reinforcement learning, a common stochastic approach is the *softmax probability function* [60]:

$$P(s, a) = \frac{e^{Q^o(s,a)/\tau}}{\sum_{o \in Obj} \sum_{a \in A_o} e^{Q^o(s,a)/\tau}} \quad (3.17)$$



where $P(s, a)$ represents the probability of choosing action a under state s , and τ is a positive value called *temperature*. The higher τ is, the higher the probability that *non-optimal* actions are chosen. With $\tau \rightarrow \infty$, the probabilities of choosing each action will become uniform, regardless of their Q-values; with $\tau \rightarrow 0^+$, the probability to choose the optimal action will become 1, making the decision process deterministic. To achieve better result, usually a higher rate of exploration — higher τ — should be used in the beginning of the training, and the rate should be lowered as the learned values are somewhat stabilized.

3.2.5 Reward and Feedback

The reward function indicates the quantified evaluation of the robot action. In this work, chosen actions would be evaluated in two aspects:

1. the reduction in drives (*internal reward*), and
2. the feedback of human user (*external reward*).

$$r = r_{drives} + r_{feedback} \quad (3.18)$$

After the execution of each action, each drive will be affected in various degrees. We could examine the difference in these drives before and after the execution of an action, and calculate the reward according to the difference.

$$r_{drives} = - \sum_i \omega_i \cdot \Delta d_i = \sum_i \omega_i \cdot (d_{i,before} - d_{i,after}) \quad (3.19)$$

The drives represent the needs of the robot, so the difference in drives indicates the difference in degree of satisfaction of the robot. A good action is expected to minimize the internal needs, making the robot more content. Since drives are the lower the better, so the negative of difference in drives is taken as the internal reward. The variable ω_i is the

weight related to each drive, indicating the implicit importance of each drive. The weights could be designed so that the robot would value each drive differently, creating different *personalities* for robots.

However, in Equation (3.19), the importance of dominant motivation wasn't emphasized. When the robot has a dominant motivation, we expect it to select an action accommodating to its motivation. In (3.19), the dominant motivation wasn't taken into consideration, so an action may be given the highest reward even it has nothing to do with the dominant motivation. To make the dominant motivation more influential, we have added a *motivation factor* to the internal reward function, resulting in (3.20). Assuming that $m_{dom} = m_k$, and d_k is the corresponding drive:

$$r_{drives} = \sum_i (1 + \delta_{ik} \cdot \xi) \cdot \omega_i \cdot (d_{i,before} - d_{i,after}) \quad (3.20)$$

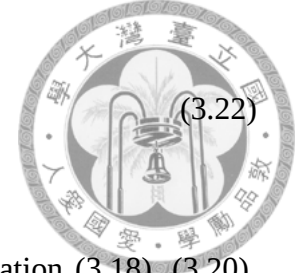
where $\xi > 0$ is the motivation factor, and δ_{im} is Kronecker delta:

$$\delta_{ik} = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{if } i \neq k \end{cases} \quad (3.21)$$

Another aspect of the reward function is the human feedback. At the beginning, the robot doesn't know how to serve the human user, since no prior knowledge about the user was injected. To endow the robot with the ability to serve, the user feedback is modeled into the reinforcement learning process. The resulting *reinforcement learning with human guidance* let the user enter the decision making loop of the robot. The human could then teach robot how to serve, or even how to satisfy the needs of the robot itself.

After execution of an action, the human user could give the robot a positive or a negative feedback. A positive feedback means that the action of the robot met the user's intention, while a negative feedback means otherwise. So the feedback part of the reward function could be represented as:

$$r_{feedback} = \begin{cases} f & \text{if feedback is positive} \\ -f & \text{if feedback is negative} \\ 0 & \text{if no feedback is given} \end{cases} \quad (3.22)$$



where f is a predefined feedback value, $f > 0$. Combining Equation (3.18), (3.20), and (3.22), we could get the final reward function.

3.2.6 Proposal and Pseudo Update

To endow the user with more influence on the robot's decisions, we define another form of feedback called *proposal*. When the user gives a negative feedback, he/she could also propose a *correct* action for the robot to learn. Providing the robot with this additional information could accelerate the learning process, making the robot to correlate the user intention with the correct action in fewer iterations. Thus, when a *proposal* happens, other than giving negative reward to the current action, the robot should also *update the Q-value of the proposed action as if it received positive feedback*. This proposal-induced update is called a *pseudo update* in this work. However, the main problem in updating in such fashion is that Q-learning is a model-free learning method. The transition result of performing a certain action is unknown unless the action is actually performed. Without the newer state s' , $V^{oi}(s, s')$ in (3.14) couldn't be calculated precisely. In this work, several approximations are made to reach a *pseudo new state*:

1. Since the proposed action is treated as the solution to the user's intention, the drive related to serving the user will be updated in the pseudo new state, while other drives are assumed to remain static.
2. The stimuli are assumed to be unchanged.
3. The state of the human and the object states remain the same.

With the above assumptions, a pseudo new state s^\dagger is generated. One could observe that the pseudo new state s^\dagger and the original state s differ only in the internal state, which

is the dominant motivation m_{dom} . Since we have the pseudo new drives, and the feedback of the proposed action is assumed to be positive, we could calculate the pseudo reward r^\dagger accordingly. Thus, when a negative feedback is given to the executed action a and a proposed action a^\dagger is further specified, other than updating the value of $Q(s, a)$ with negative feedback, we should also update the value of $Q(s, a^\dagger)$ with positive feedback, a pseudo new state s^\dagger , and a pseudo reward r^\dagger . Assuming the target of executed action a is object o_i , and the target of proposed action a^\dagger is object o_j , Equation (3.23) shows the formula for pseudo update.

$$Q^{o_j}(s, a^\dagger) = (1 - \alpha) \cdot Q^{o_j}(s, a^\dagger) + \alpha \cdot (r_{drives}^\dagger + f + \gamma \cdot V^{o_j}(s, s^\dagger)) \quad (3.23)$$

The *proposal*, along with user feedback, are the two main methods for the user to enter the learning loop of the robot. Pure homeostatic systems, such as those adopted by social robots in related work, often exclude the user from the decision process. The *learning with user guidance* feature in this work is essential for service robots, especially if one wish to make the robot user-sensitive.

Combining all the elements described in Section 3.2, the decision making process could be written as Algorithm 1.

3.3 System Design

3.3.1 Drives and Motivations

The general model of the system is described in the previous section, and this section gives the design detail of the system. As stated previously, drives are the internal variables of the robot, each representing a certain need. Drives are in the form of real numbers, normalized to the range of $[0, 100]$, with 0 being most satisfied and 100 being least satisfied. Each drive has its own *activation threshold*, above which will the drive be possible to generate a motivation.



Algorithm 1 The decision making process of the system

Require: D := a set of drives

Require: M := a set of motivations

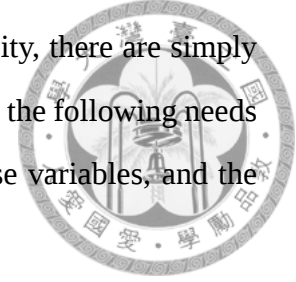
Require: AT := a set of activation thresholds

Require: O := a set of objects

Require: A := a set of actions

```
1: while ture do
2:    $ST_{exist} \leftarrow$  observed stimuli
3:    $intensities \leftarrow calc\_intensities(M, D, AT, ST_{exist})$ 
4:    $m_{dom} \leftarrow \arg \max_{m \in M} intensities(m)$ 
5:    $s_{hum} \leftarrow$  human state
6:   Initialize empty list  $S$ 
7:   for  $o \in O$  do
8:      $s_o \leftarrow$  object state of  $o$ 
9:     Add  $m_{dom} \times s_{hum} \times s_o$  to  $S$ 
10:  end for
11:  Initialize empty list  $Q$ 
12:  for  $a \in A$  do
13:     $s \leftarrow$  reduced state related to  $a$  in  $S$ 
14:    Add q-value of pair  $(s, a)$  to  $Q$ 
15:  end for
16:   $P \leftarrow softmax(Q)$ 
17:   $a^* \leftarrow$  randomly choose an  $a \in A$  according to  $P$ 
18:  Perform  $a^*$ 
19:   $f \leftarrow$  human feedback
20:   $a^\dagger \leftarrow$  human proposal
21:  Perform model update according to  $f$  and difference in  $D$ 
22:  if  $a^\dagger \neq null$  then
23:    Perform pseudo update according to  $a^\dagger$ 
24:  end if
25: end while
```

The requirements of the robot — how we wish it to behave — determines the design of needs of the system. For a personal service robot with social ability, there are simply too many metrics to be all considered at once [17]. In this evaluation, the following needs are addressed. Note that this system is by no means limited to these variables, and the needs could be extended or even replaced to fit different scenarios.



1. Since the goal of this work is to use the system on a personal service robot, providing service is obviously of major importance. A human-aware robot should be able to sense user's needs and behave accordingly [17]. The user's needs are sensed through intention recognition. However, since our robot bases its actions on homeostasis, we must give the robot *a need to serve human* so that the human-awareness aspect could be integrated into the system. This need is modeled as a drive called *Need of Achievement (NAch)* [61]. The robot's need of achievement increases as time passes, and successfully serving the human (receiving positive feedback after action execution) lowers this drive, while getting scolded by the user (negative feedback) increases it.
2. Although the main objective of this system is to endow service robots with high autonomy and human awareness ability, it would be nice if the robot could socialize with the user proactively. To achieve this, a drive called *Need of Socialization (NSoc)* is used. The *NSoc* is robot's need to interact with human while not serving him/her. This need increases when the robot is not interacting with human, and is lowered when the robot chats with human.
3. In our experience with the robot platform we use — Pepper [62] — we found out that the joints of Pepper gets overheated easily. The temperatures of joint increase when in motion or maintaining certain postures. If the temperature rises too high, the robot becomes uncontrollable. To deal with this problem, *Need of Rest (NRes)* is introduced. This value is bound to the joint temperatures of the real robot, and is assumed to increase as robot moves in the simulation. Resting — setting the stiffness of all joints to zero and entering an idle state — could lower this need, as

Table 3.1: Selected Internal Variables

drive d_i	motivation m_i	activation threshold at_i
<i>NAch</i>	<i>Serving</i>	0
<i>NSoc</i>	<i>Social</i>	70
<i>NRes</i>	<i>Relaxative</i>	50
<i>NEng</i>	<i>Survival</i>	50



it lowers the temperature on the real robot.

4. As the robots run on electricity and are not able to generate energy on their own (yet), our robot should be aware of its own energy level, and go to the charging station when needed. *Need of Energy (NEng)* could be seen as a fundamental need of robots, just like hunger to human. In the simulated environment, this need is assumed to increase as time passes, and is determined by the battery level on the real robot.

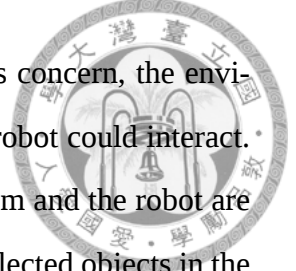
After the selection of drives, correlated motivations should be determined. Note that the correlations between drives and motivations are one-to-one, so there will be one motivation for each drive used. The selected motivations are as follows:

1. *Serving*: related to drive *NAch*,
2. *Social*: related to drive *NSoc*,
3. *Relaxative*: related to drive *NRes*, and
4. *Survival*: related to drive *NEng*.

Table 3.1 shows the correlation between drives and motivations, and their activation thresholds. The activation threshold of *NAch* being zero means that the *Serving* motivation would become active as soon as there's *NAch*, implicating that the main purpose of a service robot is to serve.

3.3.2 Objects, Stimuli and Actions

As stated in Section 3.1.4, as far as the action selection module's concern, the environment contains a *human user*, and a set of *objects* with which the robot could interact. The status of each object/human, and the spatial relation between them and the robot are represented using object states S_o and human state S_{hum} . Here, the selected objects in the environment and their state variables would be described.



- *Charger*: as the robot's battery level runs low, it must find a way to replenish its energy, and charger is the object for that. The charger could be *near* or *far* to the robot, and the status of the charger could be *charging* or *idle*. The charger is assumed to be stationary in the environment, so the robot could only change its spatial relation with the charger through moving itself. Equation (3.24) shows the object state of the charger.

$$S_{charger} = \{near, far\} \times \{charging, idle\} \quad (3.24)$$

- *CD player*: as an object the robot could interact with, the CD player could be turned on or off by the robot. Controlling the player is one of the services the robot could provide. Unlike the charger, the robot could control the player remotely, so modeling its spatial relation with the robot would be of little meaning. Equation (3.25) shows the state of the player.

$$S_{player} = \{playing, idle\} \quad (3.25)$$

- *Light*: similar to the CD player, controlling the light is also one of the services of the robot. Since the light is also remotely controlled, spatial relation is not modeled here, either.

$$S_{light} = \{on, off\} \quad (3.26)$$

Table 3.2: Selected Stimuli

rule	effect
$hi_{recog} \neq none$	<i>Serving</i> + 20
human is not <i>absent</i>	<i>Social</i> + 10
robot is at <i>resting place</i>	<i>Relaxative</i> + 10
charger is <i>near</i>	<i>Survival</i> + 10



The state variables of human is already described in Equation (3.9). The human state is composed of the recognized intention and the spatial relation between it and the robot.

The external stimuli are selected as Table 3.2. When there's a recognized intention of human, the motivation to serve will be enhanced. When the charger is near the robot, the survival motivation will increase, just like the motivation to eat will be increased when food is near for human. When the human is present (either *near* or *far*), the urge to socialize will be enhanced.

For each object, there are some predefined actions for the robot to perform on the object. For example, the robot could attach to, detach from, or move to the charger. And since charging is a time consuming process, the robot could also choose to stay near the charger, allowing it to keep charging for one more iteration if attached. Other than the actions targeting objects, some human-related or self-related actions are also available, such as chatting with human or resting to reduce joint temperature. The full list of available actions is in Table 3.3, grouped by the target of each action. Note that the human- or self-related actions are grouped under the null object, as stated in Chapter 3.

Besides that target objects, there are also some constraints related to each action. These constraints are generally intuitive, designed to prevent awkward situations, such as robot chatting with human in a long distance, or trying to attach to the charger which is far away. The constraints are also described in Table 3.3.

3.3.3 Degree of Dedication

Equation (3.20) shows that when calculating the internal reward (r_{drives}) for model update, a weight (ω_i) is given for each drive (d_i). The weight represent the importance the

Table 3.3: Selected Actions

target	action	description	constraint
charger	<i>MoveToC</i>	Move to the charger	
	<i>Attach</i>	Attach to the charger	charger is <i>near</i>
	<i>Detach</i>	Detach from the charger	
	<i>Stay</i>	Stay near the charger	charger is <i>near</i>
player	<i>Play</i>	Turn on the music	
	<i>Stop</i>	Turn off the music	
light	<i>LightOn</i>	Turn on the light	
	<i>LightOff</i>	Turn off the light	
null	<i>Chat</i>	Chat with human	human is <i>near</i>
	<i>MoveToH</i>	Move to human	
	<i>MoveToR</i>	Move to the resting place	
	<i>Rest</i>	Soften all joints and rest	human is not <i>near</i>



drive is to the robot when selecting actions. The higher the weight is, the larger the reward will be if the corresponding drive decreases in the interaction. In the previous simulation, the weight for all drives are equally 1.0. In this section, a new parameter called *Degree of Dedication* is defined based on the weights of drive, and the effect of it is discussed.

Degree of Dedication (DoD) is the ratio of the importance of *servicing drives* to all drives. More specifically, the ratio of weight of *NAch* to the summation of all weights.

$$DoD = \frac{\omega_{NAch}}{\sum_i \omega_i} \quad (3.27)$$

To avoid the unlimited growth of rewards, the weights are normalized such that the summation of them equals the number of drives.

$$\sum_i \omega_i = |D| \quad (3.28)$$

In this work, 4 drives are used in the system, so the summation of all weights equals 4. For the drives other than *NAch*, their weights are assumed to be equal. Figure 3.3 shows the correlation between weights and *DoD*. The value of *DoD* is in the range of $[0, 1]$.

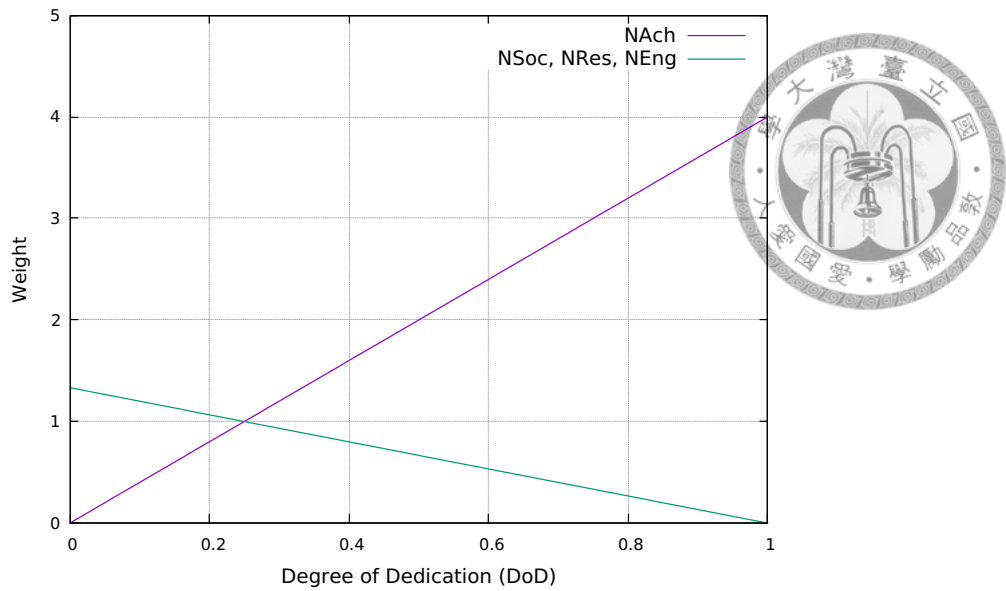


Figure 3.3: Weights of drive under different *Degree of Dedication*.

The higher *DoD* is, the more important *NAch*, thus serving human, will be. If $DoD = 0$, the *NAch* is simply ignored by the robot, and the robot will only serve the human out of user's feedback. If the $DoD = 1$, the robot will only act to please and satisfy the user, ignoring all other needs. A good *DoD* should be chosen to balance the robot's serving and self-sustaining ability.



Chapter 4

Evaluation

To evaluate the usefulness of the proposed system, the system is tested on two different environments. First the system was implemented on a virtual agent working in a simulated environment and with a virtual human agent. After the simulation have been completed, the system was ported to a real robot working in a environment similar to the simulated one, and interacting with real human user. To shorten the exploration (learning) phase, the knowledge learned in the simulation was transferred to the real robot.

We have proposed several evaluation metrics to inspect the performance of the model. The metrics are defined and described in Section 4.1. The details of simulated environment and the result of simulation will be given in 4.2. Several features proposed to enhance the system's performance, including user-induced pseudo update, the motivation factor, and the *Degree of Dedication* will also be evaluated and discussed in the section. Finally, the system was ported to a real robot, and the outcome of the interaction with human user will be shown in Section 4.3.

4.1 Evaluation Metrics

To evaluate if the proposed system indeed satisfies both the user and the robot itself, four metrics are defined and used in this work.

1. Human Satisfaction Rate (*HSR*)

The most intuitive and user-related one is to see if the needs of the user are indeed satisfied by the robot. To the user, *satisfying* is defined as *robot doing action that matches the user's intention*. For every iteration, the human user can express one of the intentions, and the robot's action could be either satisfying or unsatisfying. The rate of robot performing satisfying actions could be seen as an indicator for the total degree of satisfaction of the user. The human satisfaction rate (*HSR*) is calculated as equation 4.1. The denominator is the number of expressed intentions rather than the number of iterations, because the user might not express any intention in an iteration, and no satisfying action could be possibly chosen. *HSR* is a value between 0 and 1, the closer to 1 the better.

$$HSR = \frac{\# \text{ of satisfying actions}}{\# \text{ of user intentions}} \quad (4.1)$$

2. Cumulative Drive Average (*CDA*)

To check the degree of satisfaction of the robot, the values of drive could serve as indicators. The meaning of drives in this work is the internal needs of the robot, so the lower the drives are, the more satisfied the robot is. Since the values of drive vary constantly, the average of them over a period of time could be used. The average value of each drive can be considered separately (CDA_d), or can be considered altogether (*CDA*).

$$CDA_d = \frac{1}{N} \sum_{t=1}^N d(t) \quad (4.2)$$

where $d(t)$ means the value of drive d in time t , and

$$CDA = \left(\frac{\sum_{d \in D} CDA_d}{|D|} \right) \quad (4.3)$$

3. Cumulative Exceeding Drive Average (*CEDA*)

As mentioned in Section 3.1, there's an activation threshold for each drive, and a drive is considered *unsatisfied* if it exceeds the threshold, generating motivation. In

other words, if the drive is lower than the threshold, it is considered acceptable to the robot, and no measurement will be taken to lower that drive. Thus, the values chosen for activation thresholds AT will effect the outcome of drive average directly. For example, if the threshold for $NEng$ is 50, the robot will be likely to only try to charge after $NEng$ exceeds 50. According to this, one can expect the value of $NEng$ to roughly oscillate around 50. To negate the effects of selected AT in evaluation, cumulative exceeding drive average ($CEDA$) is defined. The meaning of $CEDA$ is similar to that of CDA , but rather than taking the average of drives directly, the average value of drives *that exceeds the threshold* is used to represent the degree of dissatisfaction.

$$CEDA_{d_i} = \frac{1}{N} \sum_{t=1}^N \max(d_i(t) - at_i, 0) \quad (4.4)$$

where the subscript i indicates the correlation between d_i and at_i , $d_i(t)$ is means the value of d_i at time t , and

$$CEDA = \left(\frac{\sum_{d \in D} CEDA_d}{|D|} \right) \quad (4.5)$$

4. Robot Secure Rate (RSR)

When a drive exceeds 90, it is considered *insecure*, and having insecure drives is to be avoided. Therefore, another metric representing the time ratio of robot being secure is defined. Secure rate (RSR) is calculated by:

$$RSR = \frac{1}{N} \sum_{t=1}^N se(t) \quad (4.6)$$

where N is the total time, and $se(t)$ is an indicator showing if the robot is secure at time t :

$$se(t) = \begin{cases} 1 & \text{if } \forall d \in D : d(t) < 90 \\ 0 & \text{otherwise} \end{cases} \quad (4.7)$$



RSR is in the range of $[0, 100]$. The higher *RSR* is, the higher ratio of time have the robot spent feeling secure.

4.2 Simulation


4.2.1 Setup

The content of the virtual environment is the same as Section 3.3.2 describes — a virtual human, a virtual robot, a charger, a CD player, and light. To construct the spatial relation between the robot and human/objects, there are three named locations in the virtual environment.

- **H**: the location of human.
- **C**: the location of charger.
- **R**: the resting place for robot.

The location of CD player and light are unimportant here, since these are controlled remotely, and no spatial relation needs to be modeled for them. From the three named locations and the robot's current location in the virtual environment, the spatial relation between robot and the human/objects could be extracted. If the robot is at the same location with another entity, the spatial relation between them is considered as *near*, otherwise *far*. For example, when the robot is at location **R**, then its spatial relation with human is *far*, and that with charger is also *far*; if the robot is at **H**, then the human is considered *near*, while the charger is still *far*. For human, another option is to become *absent*. In this case, the human agent removes itself from the environment, and is of no effect on the robot in that iteration.

Table 4.1: The Effects of Action in Simulation



target	action	effect on env.	effect on drives
charger	<i>MoveToC</i>	change robot's location to C	
	<i>Attach</i>	change charger to <i>charging</i>	$NEng - 20$
	<i>Detach</i>	change charger to <i>idle</i>	
	<i>Stay</i>	none	$NEng - 20$ if <i>charging</i>
player	<i>Play</i>	change player to <i>playing</i>	
	<i>Stop</i>	change player to <i>idle</i>	
light	<i>LightOn</i>	change player to <i>on</i>	
	<i>LightOff</i>	change player to <i>off</i>	
null	<i>Chat</i>		$NSoc - 30$
	<i>MoveToH</i>	change robot's location to H	
	<i>MoveToR</i>	change robot's location to R	
	<i>Rest</i>	none	$NRes - 20$

Table 4.2: Increase Rate of Drives in Simulation

drive	increase rate (per iteration)
$NAch$	3
$NSoc$	3
$NRes$	2
$NEng$	2

The actions, other than changing the environment, also has some effect on the robot itself. More specifically, on the drives of the robot. Some of the drives are bound to the real internal parameters of the robot (such as $NRes$ and $NEng$), while some are artificial ($NAch$ and $NSoc$). However, in virtual robot agent, all parameters are in fact artificial, and the effects of action on them should be predefined. Table 4.1 shows the effects of action on drives and on the environment in the simulation. Besides, the decrease in $NAch$ when human gives positive feedback is 10.

Since all drives are artificial, the increment rate for each drive should be defined as well. Here, all drives are assumed to have a static increase rate, and increases with every iteration. The increase rate of each drive is shown in Table 4.2.

Table 4.3: Predefined Intentions in the Simulation

intention	desired action
<i>light</i>	<i>LightOn</i>
<i>no_light</i>	<i>LightOff</i>
<i>music</i>	<i>Play</i>
<i>no_music</i>	<i>Stop</i>
<i>come</i>	<i>MoveToH</i>
<i>leave</i>	<i>MoveToR</i> or <i>MoveToC</i>



The virtual human is designed as a reactive agent, simple yet enough to express intentions and give feedback. The human has a set of predefined intentions to be expressed, which, along with their actual desired actions, could be found in Table 4.3. Note that in simulation, the intention recognition part is omitted, and the intentions are assumed to be observable directly. For each iteration, the intention is decided according to two rules:

1. If the intention in the previous iteration was not satisfied, it remains the same;
2. Otherwise, list the intentions that are reasonable in the current circumstance, and randomly choose one from the list as the new intention. For example, if the *player* is *playing* at the time, expressing *music* intention is meaningless.

After the virtual robot observed its intention and perform an action, the human agent would see if its intention was satisfied, and give feedback to the robot. If the intention was satisfied, the feedback would be positive; negative if unsatisfied, and neutral if no intention was expressed. The value of feedback used to calculate the reward for actions is 10. That is, if the feedback is positive, $r_{feedback} = 10$, and if the feedback is negative, $r_{feedback} = -10$. The pseudo code of the human AI is shown in Algorithm 2.

As stated in Section 2.2.3 and 3.2.4, there are various parameters that could affect the outcome of Q-learning. The most important ones are the *learning factor* α , and the *temperature* τ for softmax. The higher α is, the easier Q-value would be affected in every update. So a higher α at the beginning of the learning process could help Q-values to converge faster. τ affects the rate of selected action with the highest Q-value. The higher



Algorithm 2 The AI of human agent

Require: I := a set of intentions

Require: A := a set of robot actions

Require: f := value of feedback

```
1: intention = null
2: while ture do
3:   if intention satisfied or intention == null then
4:      $I_{valid} \leftarrow$  reasonable intentions in  $I$ 
5:     intention  $\leftarrow$  random_choice( $I_{valid}$ )
6:   end if
7:   express intention
8:   wait for robot to perform an action
9:   if intention  $\neq$  null then
10:     $feedback \leftarrow 0$ 
11:    proposal  $\leftarrow$  null
12:  else if intention satisfied then
13:     $feedback \leftarrow f$ 
14:    proposal  $\leftarrow$  null
15:  else
16:     $feedback \leftarrow -f$ 
17:    proposal  $\leftarrow$  corresponding action of intention in  $A$ 
18:  end if
19:  express feedback and proposal to the robot
20: end while
```

Table 4.4: Additional parameters used in simulation

parameter	value	description
γ	0.5	The discount factor in Q-learning
ξ	1.0	Motivation factor
all ω_i	1.0	Weight for calculating r_{drives}



τ is, the easier the robot would choose a non-optimal action. This *exploring* behavior should be encouraged in the early stage of learning, as the Q-values are still unstable; while choosing a non-optimal action after numerous interactions might be a bad idea, as the action with the highest Q-value is more likely to be the real optimal solution. Therefore, in the early stage of simulation, higher α (0.3) and τ (4.0) are used. After 1000 iterations, as the Q-values are more stable, α and τ are lowered to 0.1 and 2.0.

4.2.2 Result

Other than the setup described above, some additional parameters shown in Table 4.4 are used in the simulation. The usage of motivation factor ξ and weights ω_i is described in Section 3.2.5. The simulation is run for 5000 iterations. That is, the robot performs 5000 actions and updates its model for the same amount of times. The result is rather satisfying.

The value of each drive, its cumulative drive averages (*CDAs*) and cumulative excess drive averages (*CEDAs*) in the simulation process are illustrated in Figure 4.1, the *Human Satisfaction Rate (HSR)* and the *Robot Secure Rate (RSR)* are illustrated in Figure 4.2. As one can see, since there's little prior knowledge provided to the virtual robot before simulation, the robot doesn't know what to do in the early stage of simulation. The value of drives, *CDAs*, *CEDAs* are all very high, because the robot haven't learned how to lower them yet. After a few hundreds of iterations, all the internal needs start to be lowered and kept within range thanks to the online learning process. The maintenance of internal needs can also be discovered through observing *RSR*. As Section 4.1 described, *RSR* is the ratio of time robot spent felling secure, without extreme internal needs. Figure 4.2b shows that the *RSR* is 1 at the very beginning of the simulation, because the drives of the robot haven't

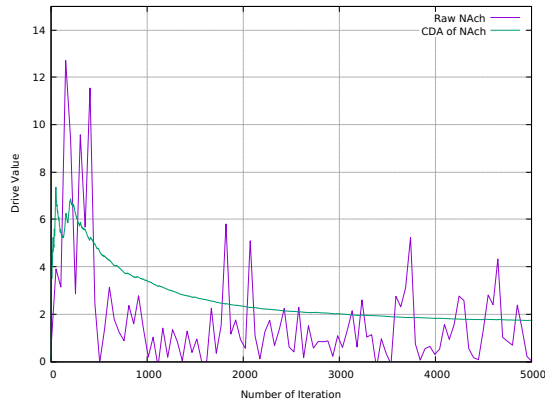
developed enough to make the robot *insecure*. Shortly after, *RSR* drops drastically, as the drives are too high yet the robot haven't learned how to deal with them. *RSR* starts to climb after a few hundreds of iterations and the robot rarely feels insecure in the mid to late stage of simulation.



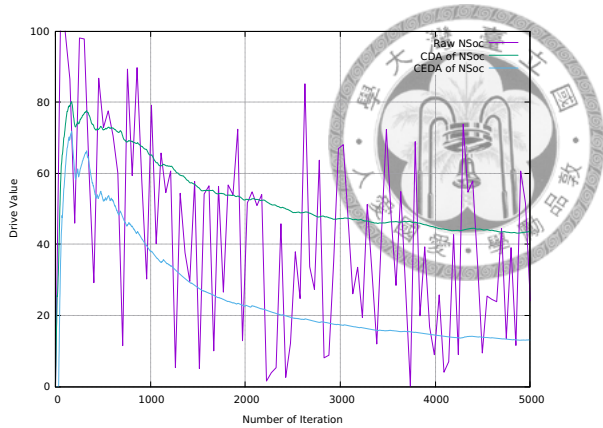
The *Need of Achievement* is found to be particularly low compared to other drives. Other than the lower activation threshold of it, it is also speculated to be the result of user feedback and pseudo update. Since in the early stage of simulation, the robot could do nothing but explore, trying to satisfy the needs. However, the user feedback gives the robot indications of how to serve the user; further more, the user could propose *correct actions* to the robot, and the robot will perform pseudo updates on the proposal (Section 3.2.6). These features make the robot easier to learn the lowering methods of *NAch*.

Other than observing the internal needs of the robot, the satisfaction of human user could be calculated as *HSR*, the result of which is illustrated in Figure 4.2a. The value of *HSR* means the proportion of satisfied intentions of the user. The higher *HSR* is, the more likely the user's intention get satisfied upon its expression. As one can see, the value of *HSR*, though low at the beginning, climbs rapidly and reaches above 0.8 in the late stage of simulation. Also, it is observed that *HSR* is roughly negatively correlated with the *CDA* of *NAch*, since the satisfaction of the user (positive feedback) directly affects the value of *NAch*.

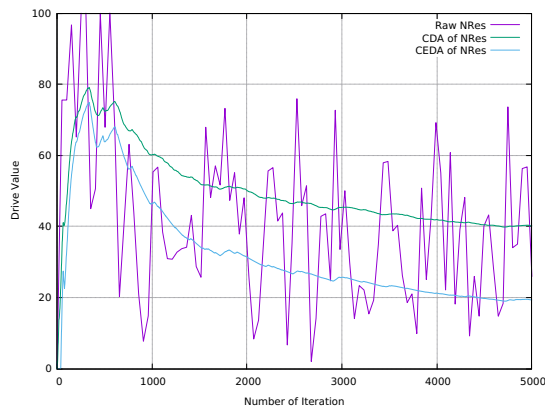
The value of the proposed metrics in the end of the simulation are shown in Table 4.5. The cumulative average of drives other than *NAch* are kept between 40 to 50, due to their higher activation threshold. Observing their *CEDA* respectively, one can see that the average value of *exceeding drive* is very low for all drives. This indicates that the actions of robot successfully eliminate the internal needs, keeping them under the activation thresholds, and keeping the robot itself satisfied. The value of *RSR* shows that the robot spends only 10% of times feeling insecure, most of which comes from the early stage of the simulation. The value of *HSR* shows that when the human user express an intention, 82% of the time the robot will satisfy it immediately. Again, most the dissatisfaction comes from the early stage of simulation, so the values of *RSR* and *HSR* are expected to grow even



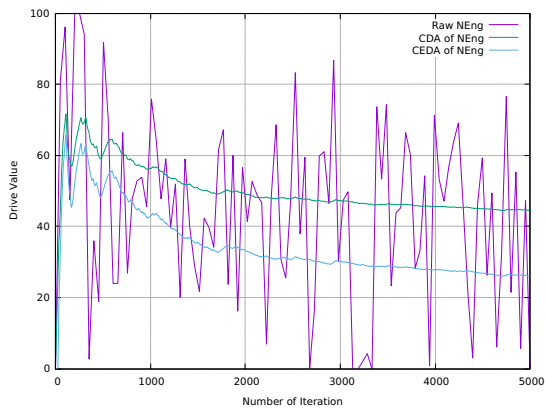
(a) Need of Achievement



(b) Need of Socialization

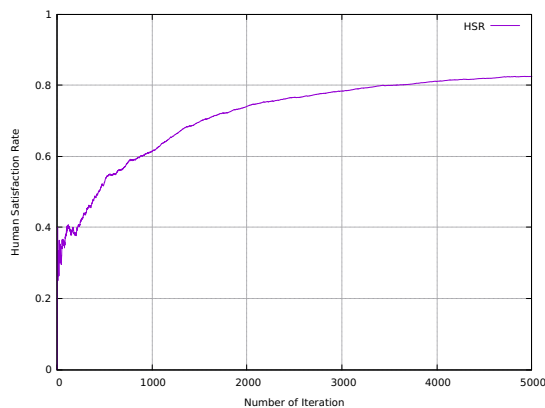


(c) Need of Rest

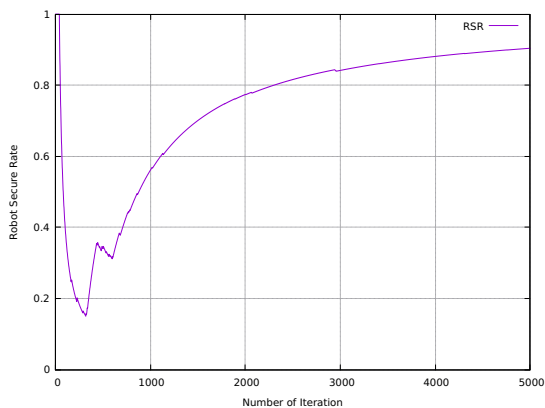


(d) Need of Energy

Figure 4.1: Value of drives in the simulation process. The three lines in each figure are the raw drive value, *CDA*, and *CEDA* of that drive, respectively. Because the activation threshold of *NAch* is zero, so its *CDA* and *CDEA* are the same.



(a) Human Satisfaction Rate



(b) Robot Secure Rate

Figure 4.2: The change in *HSR* and *RSR* (both the higher the better) in the simulation process.

Table 4.5: Various metrics in the end of simulation

	$NAch$	$NSoc$	$NRes$	$NEng$	all
<i>CDA</i>	1.73	43.5	40.1	44.5	32.5
<i>CEDA</i>	1.73	2.05	5.72	6.99	4.12
<i>HSR</i>	—	—	—	—	0.82
<i>RSR</i>	—	—	—	—	0.90

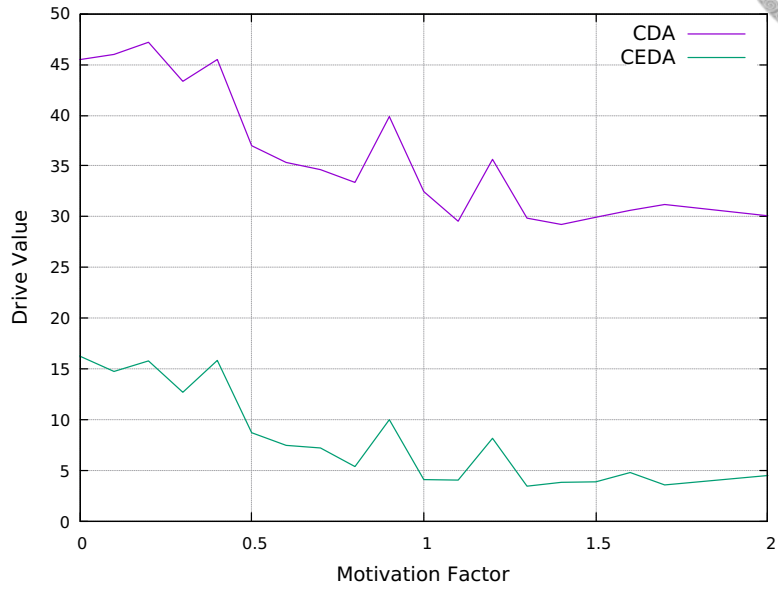
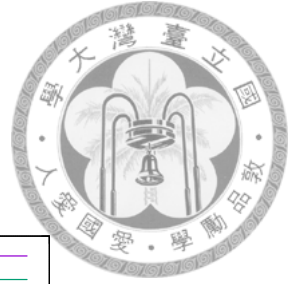


larger as the number of iterations increases.

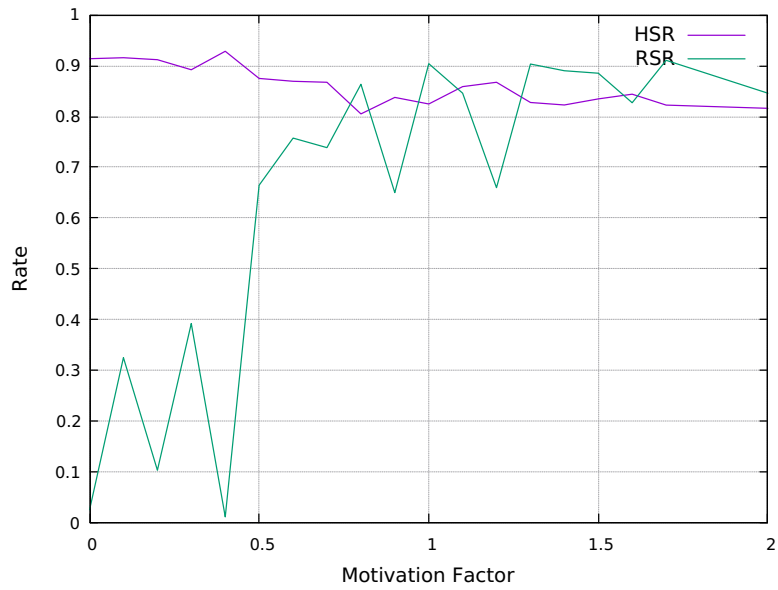
4.2.3 Effects of Motivation Factor

Section 3.2.5 mentioned a parameter in the calculation of internal reward called *motivation factor* (ξ). The usage of ξ can be found in Equation 3.20. As the equation indicates, ξ is used as a bonus factor for the drive related to the dominant motivation. In the previous section, ξ of 1.0 is used, which that means when calculating the internal reward, the change in drive related to the dominant motivation will have an extra 1.0 time of bonus reward, doubling the original reward. For example, suppose that *Need of Energy* is 50 and 30 before and after the robot performs an action respectively, the original reward from this drive is $20\omega_{NEng}$. However, if the dominant motivation of the time is *Survival*, which is related to *NEng*, then the reward from *NEng* becomes $20(1 + \xi)\omega_{NEng} = 40\omega_{NEng}$. This factor encourages the robot to perform actions *according to the current dominant motivation*. Because the dominant motivation is usually related to the highest drive of the time, acting by the dominant motivation should sooth the most dire need. If a good motivation factor is used, the improvement in *SR* and *CEDA* should be discovered as a result. In this section, the results of simulations with different motivation factor are shown, and their meanings are discussed.

To evaluate the usefulness of the motivation factor, several simulations are run with different ξ . Other than ξ , all the other parameters are the same as in the previous section. Figure 4.3 shows the effect of motivation factor ξ on different metrics. As Figure 4.3a shows, using a ξ above 1 is a significant improvement compared to those with lower ξ . This indicates that the robot could satisfy its own needs better if it learns to deal with



(a) *CDA* and *CEDA*



(b) *HSR* and *RSR*

Figure 4.3: Effect of motivation factor on different metrics.

the most dire desire, rather than trying to handle everything at the same time. Similarly, Figure 4.3b shows that *RSR* could be greatly improved with ξ larger than 0.5. Without the direction of dominant motivation, the robot would spend more than 60% of its time being insecure. Interestingly, as a side effect, a higher ξ lowers the *HSR* slightly, as shown in the same figure. This side effect is predictable, since a high motivation factor encourages the robot to choose actions based on the dominant motivation, be it *Serving* or not.

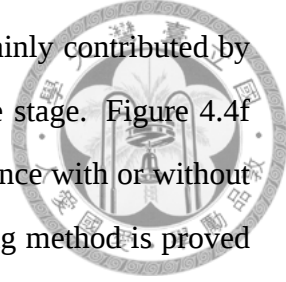
The result shown above also proves the usefulness of dominant motivation itself. A robot with $\xi = 0$ could be considered as a robot without dominant motivation, since the dominant motivation has no actual effect on the selection of actions. Without dominant motivation, the robot could still perform well in serving because of the user feedback, yet unable to satisfy its most dire needs, lowering its autonomy.

4.2.4 Effects of pseudo update

In Section 3.2.6, a kind of user-induced *pseudo update* is introduced. When the user is unsatisfied with the robot's action, other than giving negative feedback, the user could also choose to propose a *correct action* to the robot. The robot would treat the proposed action as if it was performed and a positive feedback was received. This pseudo update is another mean for the user to interfere with the robot's decision model, besides expressing intentions. The effects of the pseudo update are expected to be a higher rate of convergence, and a higher *HSR*.

Figure 4.4 shows the result of simulation with and without pseudo update. Other than pseudo update, parameters in two simulations are the same. As Figure 4.4a shows, pseudo update significantly improve *HSR*. Since the user could propose the correct serving action to match his/her intention, the robot could learn to serve more rapidly, resulting in higher *HSR*, especially in the early stage. This could also be proved by Figure 4.4e. Without pseudo update, the robot needs a few hundreds of iterations to learn the correct service, resulting in high *NAch* in the early stage. With pseudo update, the values of *NAch* in the early stage are drastically reduced, indicating a higher learning rate in serving. Other than the improvement in serving ability, the differences in non-serving related metrics are

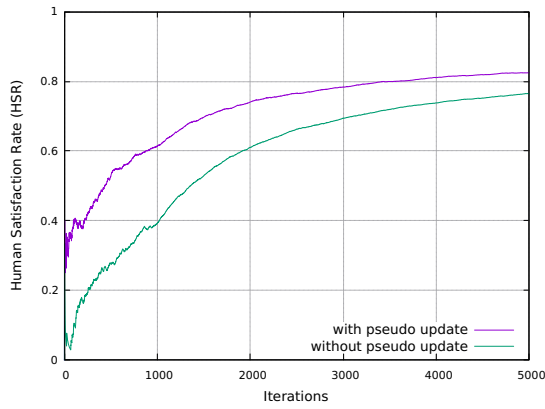
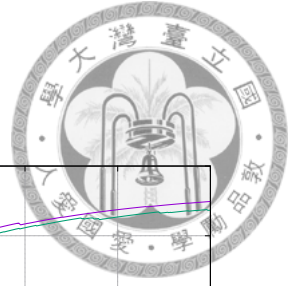
insignificant. The *RSR*, *CDA* are roughly the same with or without pseudo update. *CEDA* is indeed higher without pseudo update in the early stage, but its mainly contributed by *NAch*, and the difference in *CEDA* becomes insignificant in the late stage. Figure 4.4f also shows that *NEng*, a non-serving drive, has no significant difference with or without pseudo update. With the above results, the proposed pseudo updating method is proved useful.



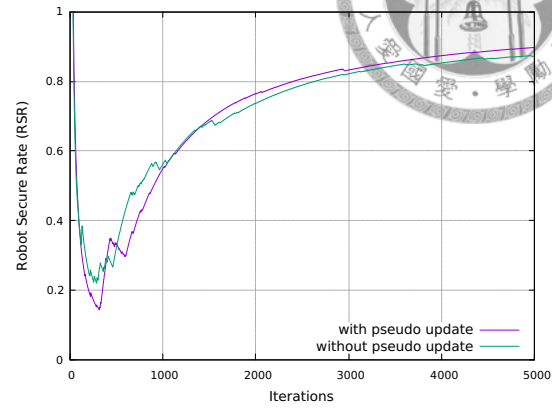
4.2.5 Effects of Degree of Dedication

Degree of Dedication (DoD), introduced in Section 3.3.3, is a parameter of the model to determine how dedicated in serving the robot should be. To see the effects of *DoD*, several simulations are done with different Degree of Dedication, and Figure 4.5 shows the result. As expected, the *HSR* is positively related to *DoD*, while *RSR* is almost negatively linearly related to it. The Figure 4.5c and 4.5d shows that a low *DoD* let the robot maintain *NSoc*, *NRes*, and *NEng* nicely. The well maintained drives indicate the high self-sustainability of the robot, making the robot less likely to run into problems such as insufficient energy or overheating. As a draw back, the robot would be more likely to ignore the user's needs, resulting in lower *HSR*. Interestingly, the value of *HSR* is still around 0.6 even if the *DoD* is 0. This shows that even without the help of *NAch*, the reward from user feedback alone is able to affect the decision of the robot, endowing the robot with serving ability.

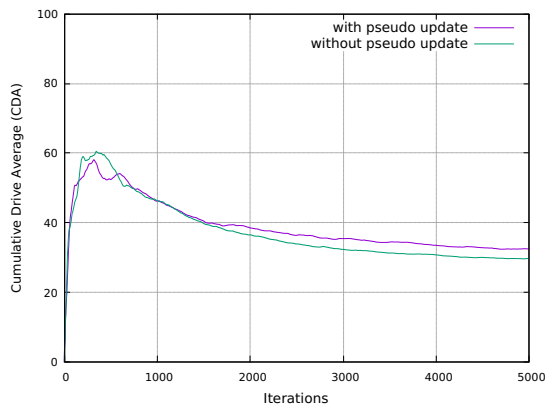
For higher *DoD*, one must be aware of the resulting low *RSR*. Even with a *DoD* higher than 0.5, the robot would spend more than half of the time being insecure. A low *RSR* is dangerous to the robot, especially so if the drives are related to real physical parameters such as battery level and temperature. If the goal of the robot is merely to be self-sustaining, $DoD = 0$ would be the best choice. For a service robot, a carefully chosen *DoD* is required. From the experiment result, a value around 0.2 would be a good choice, since the *HSR* and *RSR* are the most balanced around it.



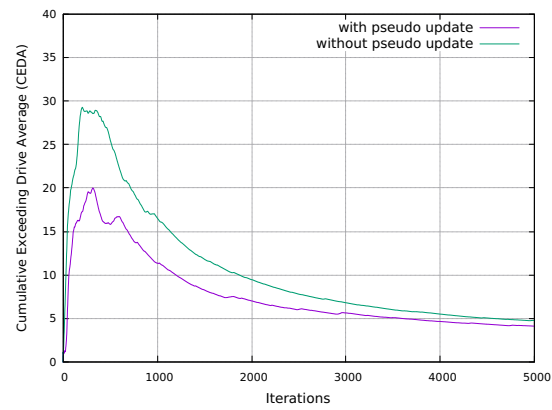
(a) *HSR*



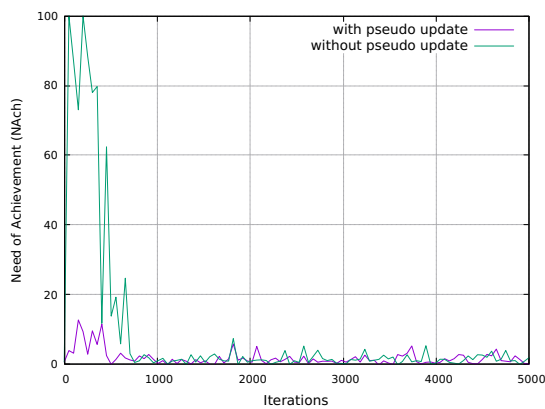
(b) *RSR*



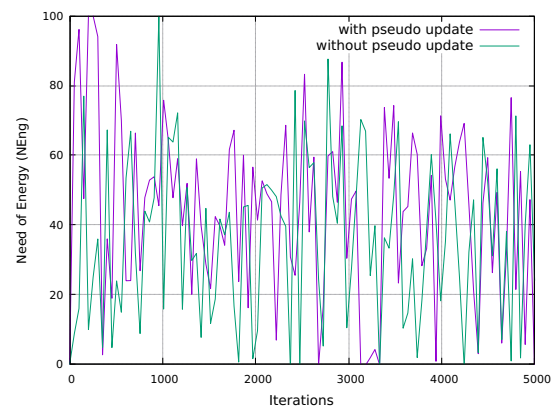
(c) *CDA*



(d) *CEDA*

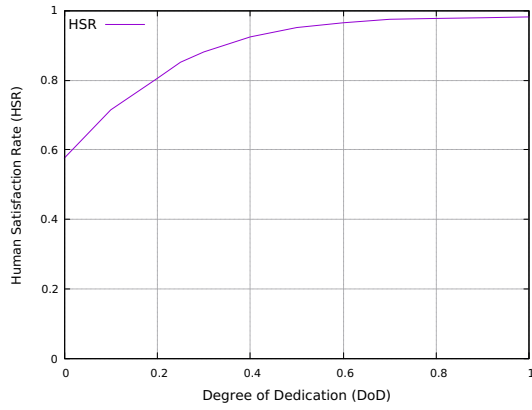


(e) *NAch*

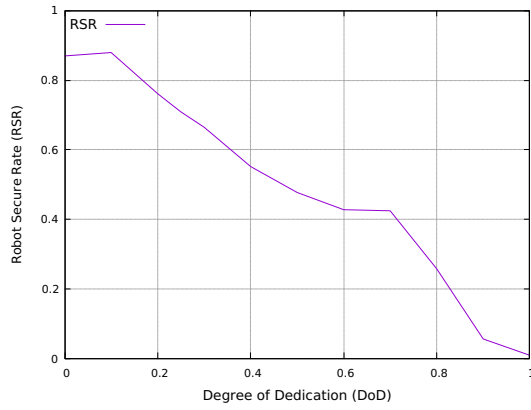


(f) *NEng*

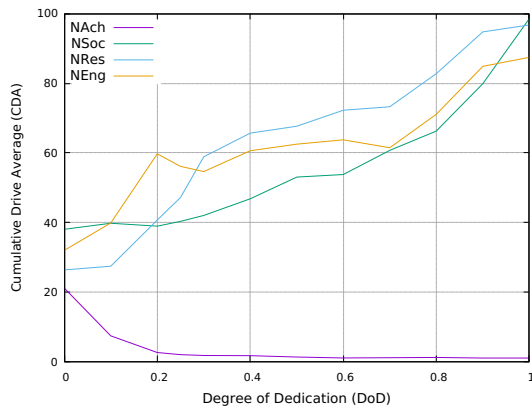
Figure 4.4: Effect of pseudo update on different metrics. In each plot, one metric is shown with and without pseudo update in the simulation process.



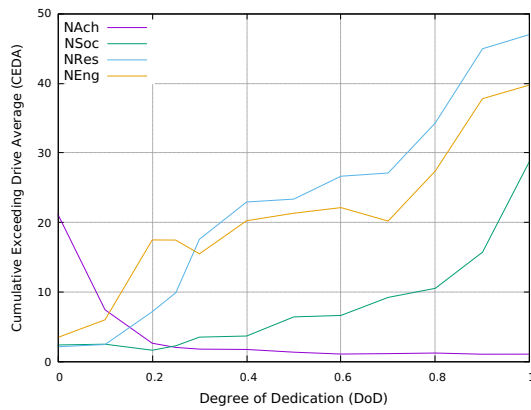
(a) HSR



(b) RSR



(c) CDA



(d) CEDA

Figure 4.5: Effect of *Degree of Dedication* on different metrics.

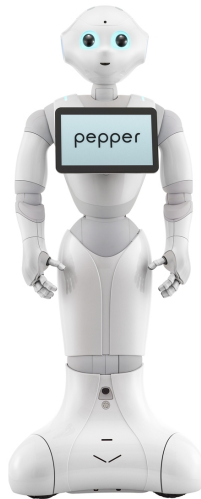


Figure 4.6: The Pepper robot.

4.3 Field Test

4.3.1 Robot Platform

The Pepper robot [62] by SoftBank and Aldebaran is used as the testing platform. The API for Pepper (NAOqi) provides various physical parameters of the robot, some of which are used in the calculation of drives.

- The *temperature status* of joints gives us a basis to calculate $NRes$ on. For each joint, the temperature status has a discrete value of 0, 1, 2, or 3. The higher the value is, the more the robot suffers from overheating. The maximum value of all joints is used as the overall temperature status, since one overheating joint is enough to induce error in the robot. After retrieving the maximum temperature status, the value of $NRes$ is calculated with the mapping described in Table 4.6.
- The battery level returned by the charge sensor is in percentage, and can be transformed into $NEng$ directly.

The robot has touch sensors on the top of its head, which are used in positive feedback. Patting on the robot's head is considered as positive feedback for the robot's selected action. To make negative feedback, scold the robot with an angry voice. Pepper could detect the level of anger in user's voice, and a high level of anger is consider as a negative

feedback. The user could also propose an action when making negative feedback, the proposal is recognized using the built-in speech recognition module.

Intention of the user is recognized the same way as [18], using RGB-D camera to perform gesture recognition, and keywords are extracted with built-in speech recognition. Because the retrieved RGB and depth images are ill-aligned, an external vision source is required for now. An ASUS Xtion PRO LIVE is used as the external vision source, and gesture recognition is performed on the images from it. Please refer to Hua *et al.* [18] for the details of gesture and intention recognition.

The actions of the robot stays the same as in the simulation. However, for *Attach* and *Detach* to the charger, the robot cannot perform these two actions alone due to hardware constraints. In the field test the robot would ask the user for help.

4.3.2 Result

Since reinforcement learning takes time to learn, it is unlikely that the robot learns from scratch in all field tests. To bypass the learning phase and see the result of well-established models, the knowledge learned in the simulation is transferred to the real robot. Because the state space used in simulations is abstract enough to be used here directly, the Q-values at the end of a simulation could be stored and deployed directly to the real robot.

To show the difference in the robot's behavior with different *Degrees of Dedication*, two kind of models were used in the field test. The first with a *DoD* of 0, and the second with *DoD* of 0.25. The robot is nearly depleted of energy at the beginning of each test so that its decisions on serving or self-sustaining could be observed. For convenience, the model with *DoD* of 0.25 will be called a *dedicated robot*, and the one with *DoD* of 0 will

Table 4.6: Mapping from temperature status to *NRes*

temperature status	<i>NRes</i>
0	0
1	50
2	80
3	100

be called a *selfish robot* in the rest of the section. Note that the word *selfish* isn't used derogatorily, but merely to describe the nature of the robot's decisions.

The outcome of a *dedicated robot* is shown in Table 4.7. The robot starts with *Serving* as its dominant motivation. In this stage, the user expresses various intentions, and the robot satisfies them immediately. This indicates that the knowledge learned from the simulation was successfully transferred to the real robot. At the fifth interaction, the dominant motivation becomes *Survival*, because the positive feedback given by the user previously lowered its *Need of Achievement*. For some reason, the robot choose to chat with the user at the fifth interaction. This is predictable since the action selection method is stochastic. From interaction six to eight, the effect of having a higher *Degree of Dedication* is shown here. Although having *Survival* as the dominant motivation, the robot still choose to satisfy the user rather than itself. At the ninth interaction, the user wishes the robot to leave, and the robot chooses to move to the charger. It isn't sure whether the choice to move to the charger is for charging in the future, or it's simply a coincidence. Either way, the robot asks the user to help it attach to the charger at the tenth interaction, since there's no intention expressed by the user. At the end of this experiment, all of the intentions expressed by the user were satisfied, and the *dedicated robot* still managed to take care of itself at its leisure.

Table 4.8 shows the outcome of a *selfish robot*. Similar to the previous experiment, the dominant motivation is *Serving* at the first four interactions, and the robot satisfies the user's needs successfully. As discussed in Section 4.2.5, though having a zero *DoD*, the ability to serve will not be eliminated. The robot would still serve the human in pursuit of positive feedback and to avoid negative feedback. Also, since the dominant motivation is *Serving*, there won't be a bonus from motivation factor in performing other actions, so self-sustaining actions might still be considered unworthy to choose by the robot. At the fifth interaction, the dominant motivation becomes *Social*, which encourages the robot to chat with the user. The robot moves to the user in that interaction, and chat with the user in the next interaction, despite the fact that the user's intention is to have some music. Theses two interactions shows the *selfishness* in the robot. When a need is unsatisfied, the robot

Table 4.7: Outcome of $DoD = 0.25$

interaction	m_{dom}	intention	action	feedback	proposed action
1	<i>Serving</i>	<i>light</i>	<i>LightOn</i>	—	—
2	<i>Serving</i>	<i>no_light</i>	<i>LightOff</i>	—	—
3	<i>Serving</i>	<i>come</i>	<i>MoveToH</i>	positive	—
4	<i>Serving</i>	<i>music</i>	<i>PlayerOn</i>	positive	—
5	<i>Survival</i>	—	<i>Chat</i>	—	—
6	<i>Survival</i>	<i>no_music</i>	<i>PlayerOff</i>	positive	—
7	<i>Survival</i>	<i>light</i>	<i>LightOn</i>	positive	—
8	<i>Survival</i>	<i>music</i>	<i>PlayerOn</i>	positive	—
9	<i>Survival</i>	<i>leave</i>	<i>MoveToC</i>	—	—
10	<i>Survival</i>	—	<i>Attach</i>	—	—

Table 4.8: Outcome of $DoD = 0$

interaction	m_{dom}	intention	action	feedback	proposed action
1	<i>Serving</i>	<i>music</i>	<i>PlayerOn</i>	—	—
2	<i>Serving</i>	<i>come</i>	<i>MoveToH</i>	positive	—
3	<i>Serving</i>	<i>leave</i>	<i>MoveToC</i>	—	—
4	<i>Serving</i>	<i>no_music</i>	<i>PlayerOff</i>	—	—
5	<i>Social</i>	—	<i>MoveToH</i>	—	—
6	<i>Social</i>	<i>music</i>	<i>Chat</i>	negative	<i>PlayerOn</i>
7	<i>Serving</i>	<i>music</i>	<i>PlayerOn</i>	positive	—
8	<i>Survival</i>	<i>light</i>	<i>MoveToC</i>	negative	<i>LightOn</i>
9	<i>Survival</i>	<i>light</i>	<i>Attach</i>	negative	<i>LightOn</i>

would deal with it swiftly with less consideration on the user. At the seventh interaction, the *Need of Socialization* is satisfied, and the robot choose to serve the user again. At the eighth interaction, *Survival* becomes the dominant motivation. The *selfish robot* choose to move to the charger and ask user for help in attaching immediately. This again shows the priority in self-sustaining for a robot with low *DoD*. A *selfish robot* would encounter dangerous situations less often, at the expense of user satisfaction. The *HSR* in this case is 0.625, with 3 out of 8 needs unsatisfied.



Chapter 5

Conclusion

In order to achieve high degrees of both autonomy and human awareness for personal service robots with no predefined goals, a homeostasis based decision-making system is proposed in this work. The *homeostatic drive theory*, a common bio-inspired approach to increase autonomy in social robotics, is adopted by the system and tweaked to fit the serving needs of service robots.

The core of the decision making process is a modified Q-learning algorithm. The intention of human user and the internal needs of the robot will be considered at the same time in the process, and the outcome of each action is learned from scratch with little or no prior knowledge. Using both satisfaction of robot and user feedback in the reward function, the robot learns to serve the user and satisfy its own needs at the same time. The robot could also perform *pseudo updates* on the model according to user's indication. The user-guided leaning process performs better than the one purely using exploration.

The simulation result shows that the performance of the proposed system is admirable. At the end of the simulation, the satisfaction rate of the user reaches 82%, and the robot is able to keep its internal needs in an acceptable range for more than 90% of the time. The effect of motivation factor, a factor proposed in this work to encourage the robot to deal with its most dire need at the time, is proved to keep the robot secure from dangerously high needs. The robot could avoid many dangerous situations such as overheating or insufficient power with a high enough motivation factor. The pseudo update in the learning process is also proved to increase about 10% of user satisfaction, and let the robot learn

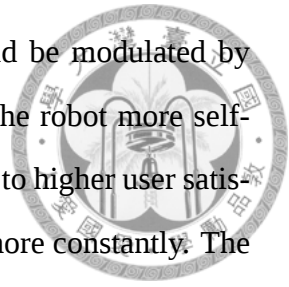
how to serve more rapidly.

The balance between serving and satisfying its own needs could be modulated by the *Degree of Dedication (DoD)*. A lower *DoD* is shown to make the robot more self-sustaining, yet ignoring user's needs more often. A higher *DoD* leads to higher user satisfaction, sacrificing other needs of the robot and put robot in danger more constantly. The *Degree of Dedication* could be seen as one of the personal traits of the robot, and could be adjusted to fit different scenarios.

The proposed homeostatic system gives the robot both self-sustainability and service-ability. The integration of user feedback endows the robot with ability to learn with human guidance, and the introduction of user intention makes the robot less self-centric and more human-aware. For future work, a more sophisticated intention recognition method could be used. With a wider range of expressible intentions, the robot could provide more services proactively. Activity recognition could also be a good addition to the system, since the required services are sometimes related to the current activity of the user.

Generating new actions online could also be a good research direction. Since the proposed system could handle the expansion of actions, if a new action is to be generated, it could be integrated into the system automatically. In this case, we could generate a variation of an existing action to match the user's preference, or generate a whole new kind of action to extend the robot's ability.

The parameters mentioned in this thesis, including but not limited to *DoD*, weights of each drive, activation thresholds, etc., might be decided in a more automated manner. For example, the *DoD* could be adjusted in the interaction according to the user's and robot's current satisfaction. If the user's satisfaction is lower while the robot's satisfaction is still high enough, *DoD* could be increased online. If all parameters could be adjusted automatically, less human intervention would be needed in the construction of the system, and higher autonomy could be achieved.





Reference

- [1] E. Garcia, M. A. Jimenez, P. G. De Santos, and M. Armada, "The evolution of robotics research," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 90–103, 2007.
- [2] "irobot: Your partner for a cleaner home," 2016, accessed: 10-July-2016. [Online]. Available: www.irobot.com
- [3] C. Jayawardena, I. H. Kuo, U. Unger, A. Igic, R. Wong, C. I. Watson, R. Stafford, E. Broadbent, P. Tiwari, J. Warren *et al.*, "Deployment of a service robot to help older people," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 5990–5995.
- [4] C. Jayawardena, I. Kuo, C. Datta, R. Stafford, E. Broadbent, and B. MacDonald, "Design, implementation and field tests of a socially assistive robot for the elderly: Healthbot version 2," in *Proceedings of IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, 2012, pp. 1837–1842.
- [5] P. Elinas and J. J. Little, "Decision theoretic task coordination for a visually-guided interactive mobile robot," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007, pp. 4108–4114.
- [6] K. Sakai, Y. Yasukawa, Y. Murase, S. Kanda, and N. Sawasaki, "Developing a service robot with communication abilities," in *Proceedings of IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN)*, 2005, pp. 91–96.

- [7] T. Breuer, G. R. G. Macedo, R. Hartanto, N. Hochgeschwender, D. Holz, F. Hegger, Z. Jin, C. Müller, J. Paulus, M. Reckhaus *et al.*, “Johnny: An autonomous service robot for domestic environments,” *Journal of intelligent & robotic systems*, vol. 66, no. 1-2, pp. 245–272, 2012.
- [8] K. A. Wyrobek, E. H. Berger, H. M. Van der Loos, and J. K. Salisbury, “Towards a personal robotics development platform: Rationale and design of an intrinsically safe personal robot,” in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2008, pp. 2165–2170.
- [9] S. Lauria, G. Bugmann, T. Kyriacou, J. Bos, and E. Klein, “Personal robot training via natural-language instructions,” *IEEE Intelligent systems*, vol. 16, no. 3, pp. 38–45, 2001.
- [10] J. Pransky, “Service robots—how we should define them?” *Service Robot: An International Journal*, vol. 2, no. 1, pp. 4–5, 1996.
- [11] B. T. Clough, “Metrics, schmetrics! how the heck do you determine a uav’s autonomy anyway,” DTIC Document, Tech. Rep., 2002.
- [12] G. A. Bekey, *Autonomous robots: from biological inspiration to implementation and control*. MIT press, 2005.
- [13] L. Cañamero, “Designing emotions for activity selection in autonomous agents,” *Emotions in humans and artifacts*, vol. 115, p. 148, 2003.
- [14] K. L. Bellman, “5 emotions: Meaningful mappings between the individual and its world,” *Emotions in Humans and Artifacts*, p. 149, 2002.
- [15] T. L. Anderson and M. Donath, “Animal behavior as a paradigm for developing robot autonomy,” *Robotics and Autonomous Systems*, vol. 6, no. 1, pp. 145–168, 1990.
- [16] H.-L. Cao, P. G. Esteban, A. De Beir, R. Simut, G. Van de Perre, D. Lefeber, and B. Vanderborght, “Robee: A homeostatic-based social behavior controller for robots

in human-robot interaction experiments,” in *Proceedings of IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2014, pp. 516–521.

- [17] A. Steinfeld, T. Fong, D. Kaber, M. Lewis, J. Scholtz, A. Schultz, and M. Goodrich, “Common metrics for human-robot interaction,” in *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, 2006, pp. 33–40.
- [18] J.-H. Hua, S.-P. Ma, and L.-C. Fu, “Human awareness based robot performance learning in a social environment,” in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 4291–4296.
- [19] Á. Castro-González, M. Malfaz, and M. A. Salichs, “Selection of actions for an autonomous social robot,” in *Proceedings of International Conference on Social Robotics*, 2010, pp. 110–119.
- [20] M. Malfaz, Á. Castro-González, R. Barber, and M. A. Salichs, “A biologically inspired architecture for an autonomous and social robot,” *IEEE Transactions on Autonomous Mental Development*, vol. 3, no. 3, pp. 232–246, 2011.
- [21] S. Feyzabadi and S. Carpin, “Hcmdp: a hierarchical solution to constrained markov decision processes,” in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 3971–3978.
- [22] S. Omidshafiei, A.-a. Agha-mohammadi, C. Amato, and J. P. How, “Decentralized control of partially observable markov decision processes using belief space macro-actions,” in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 5962–5969.
- [23] D. Liu, M. Cong, Y. Du, and X. Han, “Robotic cognitive behavior control based on biology-inspired episodic memory,” in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 5054–5060.
- [24] S. Zhang and P. Stone, “Corpp: Commonsense reasoning and probabilistic planning, as applied to dialog with a mobile robot.” in *Proceedings of AAAI Conference on Artificial Intelligence*, 2015, pp. 1394–1400.

- [25] W.-R. Ko, H.-S. Hyun, H.-J. Kim, S.-H. Choi, and J.-H. Kim, "Behavior selection method for intelligent artificial creatures using the degree of consideration-based mechanism of thought," in *Proceeding of IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2011, pp. 2461–2467.
- [26] E. B. Smith, "The motion control of a mobile robot using multiobjective decision making," in *Proceedings of the 47th Annual Southeast Regional Conference*. ACM, 2009, p. 32.
- [27] P. Maes, "The agent network architecture (ana)," *Acm sigart bulletin*, vol. 2, no. 4, pp. 115–120, 1991.
- [28] F. Bellas, R. J. Duro, A. Faiña, and D. Souto, "Multilevel darwinist brain (mdb): Artificial evolution in a cognitive architecture for real robots," *IEEE Transactions on Autonomous Mental Development*, vol. 2, no. 4, pp. 340–354, 2010.
- [29] T. Ando and M. Kanoh, "A self-sufficiency model using urge system," in *Proceedings of IEEE International Conference on Fuzzy Systems (FUZZ)*, 2010, pp. 1–6.
- [30] J. Hoefinghoff, A. Rosenthal-von der Pütten, J. Pauli, and N. Krämer, "You and your robot companion—a framework for creating robotic applications usable by non-experts," in *Proceedings of IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2015, pp. 615–620.
- [31] J. R. Wilson, "Towards an affective robot capable of being a long-term companion," in *Proceedings of IEEE International Conference on Affective Computing and Intelligent Interaction (ACII)*, 2015, pp. 754–759.
- [32] —, "Robot assistance in medication management tasks," in *Proceedings of ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2016, pp. 643–644.
- [33] W. B. Cannon, *The wisdom of the body*. WW Norton & Co, 1932.



- [34] R. C. Arkin, M. Fujita, T. Takagi, and R. Hasegawa, “An ethological and emotional basis for human–robot interaction,” *Robotics and Autonomous Systems*, vol. 42, no. 3, pp. 191–201, 2003.
- [35] C. L. Breazeal, *Designing sociable robots*. MIT press, 2004.
- [36] D. Cañamero, “Modeling motivations and emotions as a basis for intelligent behavior,” in *Proceedings of the first International Conference on Autonomous Agents*. ACM, 1997, pp. 148–155.
- [37] S. P. Gadanho, “Reinforcement learning in autonomous robots: an empirical investigation of the role of emotions,” 1999.
- [38] S. C. Gadanho and L. Custódio, “Asynchronous learning by emotions and cognition,” in *Proceedings of the seventh International Conference on Simulation of Adaptive Behavior on From Animals to Animats*. MIT Press, 2002, pp. 224–225.
- [39] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, “Planning and acting in partially observable stochastic domains,” *Artificial intelligence*, vol. 101, no. 1, pp. 99–134, 1998.
- [40] R. Bellman, *Dynamic Programming*, 1st ed. Princeton, NJ, USA: Princeton University Press, 1957.
- [41] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.
- [42] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.
- [43] C. J. Watkins and P. Dayan, “Q-learning,” *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.
- [44] T. Fong, I. Nourbakhsh, and K. Dautenhahn, “A survey of socially interactive robots,” *Robotics and Autonomous Systems*, vol. 42, no. 3, pp. 143–166, 2003.



- [45] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the em algorithm,” *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.
- [46] K. C. Berridge, “Motivation concepts in behavioral neuroscience,” *Physiology & behavior*, vol. 81, no. 2, pp. 179–209, 2004.
- [47] K. Lorenz, P. Leyhausen *et al.*, *Motivation of human and animal behavior; an ethological view*. Van Nostrand Reinhold Company, 1973.
- [48] A. Elfes, “Occupancy grids: A stochastic spatial representation for active robot perception,” *arXiv preprint arXiv:1304.1098*, 2013.
- [49] J. Mason and B. Marthi, “An object-based semantic world model for long-term change detection and semantic querying,” in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 3851–3858.
- [50] C. M. Vigorito and A. G. Barto, “Intrinsically motivated hierarchical skill learning in structured environments,” *IEEE Transactions on Autonomous Mental Development*, vol. 2, no. 2, pp. 132–143, 2010.
- [51] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman, “Efficient solution algorithms for factored mdps,” *Journal of Artificial Intelligence Research*, vol. 19, pp. 399–468, 2003.
- [52] K. P. Murphy, “Dynamic bayesian networks: representation, inference and learning,” Ph.D. dissertation, University of California, Berkeley, 2002.
- [53] L. Li, T. J. Walsh, and M. L. Littman, “Towards a unified theory of state abstraction for mdps.” in *Proceedings of International Symposium on Artificial Intelligence and Mathematics (ISAIM)*, 2006.
- [54] R. Givan, T. Dean, and M. Greig, “Equivalence notions and model minimization in markov decision processes,” *Artificial Intelligence*, vol. 147, no. 1, pp. 163–223, 2003.



- [55] T. G. Dietterich, “Hierarchical reinforcement learning with the maxq value function decomposition,” *J. Artif. Intell. Res.(JAIR)*, vol. 13, pp. 227–303, 2000.
- [56] D. Chapman and L. P. Kaelbling, “Input generalization in delayed reinforcement learning: An algorithm and performance comparisons.” in *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 91, 1991, pp. 726–731.
- [57] N. K. Jong and P. Stone, “State abstraction discovery from irrelevant state variables.” in *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 2005, pp. 752–757.
- [58] M. A. Salichs and M. Malfaz, “A new approach to modeling emotions and their use on a decision-making system for artificial agents,” *IEEE Transactions on affective computing*, vol. 3, no. 1, pp. 56–68, 2012.
- [59] Á. Castro-González, M. Malfaz, and M. A. Salichs, “An autonomous social robot in fear,” *IEEE Transactions on Autonomous Mental Development*, vol. 5, no. 2, pp. 135–151, 2013.
- [60] C. B. Holroyd and M. G. Coles, “The neural basis of human error processing: reinforcement learning, dopamine, and the error-related negativity.” *Psychological review*, vol. 109, no. 4, p. 679, 2002.
- [61] H. A. Murray, “Explorations in personality.” 1938.
- [62] “Pepper, the humanoid robot from aldebaran, a genuine companion,” 2016, accessed: 12-July-2016. [Online]. Available: <https://www.ald.softbankrobotics.com/en/cool-robots/pepper>