國立臺灣大學電機資訊學院電機工程學系

碩士論文

Department of Electrical Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

使移動機器人執行動態多社交任務之最佳化導航系統

Optimal Navigation System for a Mobile Robot to Execute Dynamical Multiple Social Tasks

詹少宏

Shao-Hung Chan

指導教授：傅立成 博士

Advisor: Li-Chen Fu, Ph.D.

中華民國 108 年 8 月

August, 2019

doi:10.6342/NTU201902426

國立臺灣大學碩士學位論文
口試委員會審定書

使移動機器人執行動態多社交任務之最佳化導航系統
Optimal Navigation System for a Mobile Robot to Execute Dynamical Multiple Social Tasks

本論文係 詹少宏 君（學號 R06921017）在國立臺灣大學電機工程學系完成之碩士學位論文，於民國 108 年 07 月 26 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（簽名）
（指導教授）

施吉昇

許永真　簡忠漢

曾士桓

系主任　劉志文（簽名）

doi:10.6342/NTU201902426

誌謝

少宏 August 2nd, 2019

本篇研究得以開花結果，首先我必須先感謝指導教授傅立成博士，在兩年的碩士生涯，傅教授不只提供人工智慧中心的環境、電腦與機器人等硬體設備，更在論文撰寫與研究方向上給予精闢的見解和指導。另外，教授以英文帶領實驗室也讓我在國際場合更加得心應手。多虧教授的幫助，讓我不只有很多機會參加國際知名的機器人研討會，更為我未來的學術研究奠定紮實的基礎。求學期間針對外賓所做的展演也不斷訓練我以系統的觀點切入整個機器人程式，讓本篇研究得以開花結果，進而給予學生機會角逐最佳碩士論文獎。之後所撰寫的推薦信，更讓我得以順利申請到美國南加州大學資工研究所持續深造。對於老師的盡心栽培，學生的感激之情溢於言表，實非三言兩語可以形容。

此外，謝謝吳秉蒼學長啟發我利用 ROS 進行程式開發，並帶領我發表五篇論文，學長不只豎立了研究生應有的楷模，更督促我帶領學弟令實驗室產生更多研究成果。謝謝江明理教授、劉安陞學長、簡欣怡學姊、張偉德學長不斷給予正面的建議，讓我的研究著作可以順利發表、並使我有充分的準備來面對口試。謝謝學弟奇安與瀟越，看著你們勇於承擔實驗室的各種展演、帶專題生、蒐集資料及修訂論文，真的非常感動。期望你們也能持續在學術上有傑出貢獻，視發表論文為一條必經之路，讓實驗室繼續發光發熱。謝謝陳和麟教授、李宏毅教授，以及同窗子芸、昱文、恩宇、逸霖、宇閎、喬宇、瀚宇、凱傑、捷耀在我遇到瓶頸時給予演算法與實驗的實質建議；謝謝天時、立圃、Edwinn、禹齊、宇謙、子翔、恩德、文婷、昇毅、嘉星、羽庭、啟維、雅慧，不管是每天一起奮鬥寫程式、一起討論新想法、還是出國參加研討會，都讓我有非常精采的碩士生活。也謝謝助理們處理各項實驗室事務，讓我可以專心做研究。

最後，我要由衷謝謝我摯愛的家人，你們生活上的支持與精神上的鼓勵永遠是我持續做研究的最大動力。沒有你們各方面身心靈的協助，這篇研究不會這麼順利產出。因此，我在此致上十二萬分的謝意，並帶著你們的祝福持續精進。

doi:10.6342/NTU201902426

摘要

近年來，由於人口老化與少子化等因素，老人長照以及居家陪伴等需求日顯重要，與之相對的社交與陪伴型機器人的相關研究隨之增加。這些機器人更展現了在未來高齡化社會中的潛在應用能力。為了能使機器人輔助家庭成員與年長者的生活起居，基本的功能包括強健的定位能力、導航能力、與感測能力。此外，機器人亦應該具備能基於影像及語音等感測資訊產生對環境的即時認知或推論。換言之，機器人要能評估使用者的狀態與語言指示並進而完成人機互動領域中的社交與服務的任務。因此，一個動態、長時間的決策系統能夠使社交陪伴機器人自動產生合適的動態任務與動作規劃 (Task And Motion Planning, TAMP)。另一方面，為了使社交機器人能夠趨向實際應用的階段甚至更加地普及於未來的居家環境當中，該決策系統必須將有限的運算資源以及有效率的運算列入考量。

在本篇研究當中，基於動態任務與動作規劃，我們透過機器人感知提出了一個以任務導向為主之導航決策系統來令機器人完成複雜的動態多社交任務。為了組織這些社交任務，我們提出了一個具有隨時間遞減獎勵機制的指令架構。此外，我們將室內環境模擬成圖以定位指令，並提出一個相對應之動態任務規劃演算法。該演算法藉由最佳化累積獎勵使得機器人能同時考量指令優先度以及總執行時間。至於感知部分，視覺上除了人物定位及辨識之外，我們提出一個階層式子系統來辨識人類行為，並在聽覺上設計一個結合語音與情緒辨識的子系統。在有限運算資源之下，本系統致力於結合深度學習框架與啟發式演算法以同步處理感知與決策資訊。得力於本系統，社交型機器人有能力滿足每位使用者的需求，並在多人環境中充分展現出有效率的人機互動。

關鍵字：任務導向導航系統、動態任務與動作規劃、機器人感知、人機互動

doi:10.6342/NTU201902426

ABSTRACT

In recent years, researches related to social and companion robots have gradually increased, showing its importance in the field of daily healthcare and human companion. Those robots also demonstrate potential applications especially in the society where elderly people growing year by year. In order for robots to provide assistance toward family members and elders in a household environment, the prerequisite capabilities are to perform robust localization, navigation, and sensing ability. In addition to that, the robots should also be capable of perceiving the environment and human beings based on the visual and audio sensor data. In other words, robots should know how to estimate human status and understand his/her verbal commands so as to complete social and service tasks in the area of intelligent human robot interaction. More practically, a dynamic, any-time decision making system is necessary for social and companion robots to generate adequate task and motion planning (TAMP) over a long period of time. On the other hand, with the purpose of making robots widely deployed in the future, efficient calculation under limited computation resource should be taken into consideration while designing the overall system.

In this thesis, inspired from the Dynamic TAMP framework, we propose a novel task-oriented navigation system for robots to achieve social interaction tasks with the help of perceptions. To organize these social tasks, we propose an instruction structure consisting decaying reward with regard to priorities and time. Moreover, we model the indoor scenario into a graph structure to allocate instructions, and propose a task planning algorithm that considers not only the priorities among multiple tasks but also time efficiency through optimizing the accumulative reward. As for the perceptions

doi:10.6342/NTU201902426

that help assign priorities of instructions, we propose a sub-system for human localization, identification, and framewise hierarchical activity recognition in the visual aspect. As for verbal perception, we design a sub-system to understand human words as well as sentiments. Note that under the limited computational speed and resource, the system aims to simultaneously perform perception and decision making using both deep learning modules and heuristic algorithms. With the help of our system, the social robot is able to not only meet human requirements but also interact with people in a multiple-human environment efficiently, achieving sophisticated human robot interaction (HRI).

Keywords: Task-oriented Navigation System, Dynamic Task and Motion Planning, Robot Perception, Human Robot Interaction

doi:10.6342/NTU201902426

doi:10.6342/NTU201902426

TABLE OF CONTENTS

口試委員會審定書 .... #

誌謝 I

摘要 II

ABSTRACT III

TABLE OF CONTENTS VI

LIST OF FIGURES IX

LIST OF TABLES XII

Chapter 1 Introduction 1

1.1 Motivation 1

1.2 Challenges 3

1.3 Contributions 3

1.4 Thesis Overview 4

Chapter 2 Background and Related Works 6

2.1 Robot Perceptions 6

2.1.1 Introduction to Robot Perceptions 6

2.1.2 Laser Perceptions: Simultaneous Localization and Mapping 7

2.1.3 Visual Perceptions: Object Detection and Object Affordance 8

2.1.4 Summary of Robot Perceptions 9

2.2 Task and Motion Planning (TAMP) 9

2.2.1 Introduction to TAMP 10

2.2.2 TAMP for Mobile Robots 12

2.2.3 TAMP for Human Robot Interaction (HRI) 13

2.2.4 Summary of TAMP 14

doi:10.6342/NTU201902426

Chapter 3 Visual and Verbal Perceptions 15

3.1 Preliminary 16

3.1.1 You Only Look Once (YOLO) 16

3.1.2 OpenPose: Human Anatomic Skeleton Detection 18

3.1.3 Speech to Text and Emotion Recognition 21

3.2 Methodology for Visual Perception 22

3.2.1 Image Stitching 23

3.2.2 Human Localization 24

3.2.3 Human Identification 27

3.2.4 Framewise Hierarchical Human Activity Recognition 30

3.3 Methodology for Verbal Perception 36

3.4 Human ID Database Organization 37

Chapter 4 Dynamic Multi-Task Social Navigation 38

4.1 Preliminary 39

4.1.1 Laser-based SLAM: GMapping, AMCL, and Navigation Stack 39

4.1.2 Fundamentals of Complexity 41

4.1.3 Introduction to Value Iteration 43

4.2 Methodology 45

4.2.1 Transformation from Perceptions into Instructions 45

4.2.2 Problem Formulation for Task Planning 47

4.2.3 Algorithm for Task Planner 49

4.2.4 Correctness of the Proposed Task Planning Algorithm 55

4.2.5 Integration of Proposed TAMP 61

Chapter 5 Experimental Results 63

5.1 Environment Setup 63

doi:10.6342/NTU201902426

5.2 Experiments: Visual Perception 64
5.2.1 Human Localization Evaluation 65
5.2.2 Pepper Image Testing Dataset 66
5.2.3 Human Identification Evaluation 68
5.2.4 Framewise Hierarchical Human Action Detection Evaluation 69
5.3 Experiments: Dynamic Multi-Task Social Navigation 73
5.3.1 Optimality Comparison of Task Planner Algorithm 73
5.3.2 Efficiency Comparison of Task Planner Algorithm 78
5.3.3 Real World Implementation and Analysis 80
5.3.4 Similarity Comparison of Task Planner Algorithm 82
Chapter 6 Conclusion and Future Works 84
REFERENCE 87

doi:10.6342/NTU201902426

LIST OF FIGURES

Figure 2-1 Basic modules of autonomous robots 6

Figure 2-2 The hierarchical architecture for task and motion planning (TAMP) 12

Figure 3-1 YOLO system for real-time visual object detection [20] 17

Figure 3-2 Image processing flow of OpenPose 19

Figure 3-3 Detail network architecture of OpenPose. 19

Figure 3-4 Labels of joints for OpenPose 20

Figure 3-5 OpenPose examples [51] 20

Figure 3-6 The GUI of ROS Voice Message 21

Figure 3-7 Block diagram of visual perception 23

Figure 3-8 An example of image stitching 24

Figure 3-9 The parameters for coordinate transformation 25

Figure 3-10 The RGBD view of robot top camera 26

Figure 3-11 The localization results. Yellow box is the human location on the global map, while the red arrow is the robot current location. 27

Figure 3-12 Human perception for detailed activity detection 31

Figure 3-13 Overall relation for detailed indoor action recognition 32

Figure 3-14 An illustrative example in the view of the robot 35

Figure 3-15 The flow chart for verbal perception 36

Figure 4-1 Overall TAMP system for dynamic multi-task social navigation 38

Figure 4-2 An illustrative example of robot in a grid maze. Gray circle indicated the robot position; black blocks are the obstacles; read blocks are the hell that contains negative rewards, and yellow block is the treasure that contains positive reward. 43

doi:10.6342/NTU201902426

Figure 4-3 Process of value iteration given positive reward $r+=1$, negative reward $r-=1$, discount factor $\gamma=0.9$, and transition probability $P=0.8$. 44
Figure 4-4 Household environment to test our method, gray map is the grid occupancy map and the discrete graph indicates the topological map. 48
Figure 4-5 A special case of our modeling scenario 49
Figure 4-6 The expected reward estimation in the topological map 51
Figure 4-7 Robot state representation in the topological map 52
Figure 4-8 An example to explain the task planner algorithm 53
Figure 4-9 The scenario of proving the optimality 55
Figure 4-10 The expanding rewards of two instructions along AB 56
Figure 4-11 Overall reward distribution of two instructions 57
Figure 4-12 The expanding rewards along AB 58
Figure 4-13 The accumulative reward function along AB 60
Figure 4-14 The integration flow chart of the proposed TAMP 62
Figure 5-1 Environment hardware setup 63
Figure 5-2 The visualization of human localization. (a) shows the depth image and the human bounding box. (b) is the result of localization, where blue points are the human walking path and green square is the ground truth trajectory. 65
Figure 5-3 The number distribution of our testing dataset 66
Figure 5-4 Some example images inside the Pepper Image Testing Dataset. 67
Figure 5-5 Confusion matrix of proposed action recognition algorithm 72
Figure 5-6 The accumulative reward of example Figure 5-13 75
Figure 5-7 The accumulative reward of arbitrary ten instructions 76
Figure 5-8 Processing time of different numbers of instructions 78
Figure 5-9 Time ratio of processing instructions 79

doi:10.6342/NTU201902426

Figure 5-10 The accumulative reward of simulation and real world implementation 80

Figure 5-11 The total processing time in simulation and real world implementation.....81

Figure 5-12 The time ratio of simulation and real world implementation 81

Figure 5-13 An example questionnaire of random instructions for human scheduling ..82

doi:10.6342/NTU201902426

LIST OF TABLES

Table 3-1 *Human* data structure..37

Table 4-1 Instruction structure...46

Table 5-1 An illustrative example of color metric 68

Table 5-2 The approximated color metric of query images and testing images 69

Table 5-3 The visualization of action detection process (1) 70

Table 5-4 The visualization of action detection process (2) 71

Table 5-5 The table of priority quantification..73

Table 5-6 The example task scenario of our simulation 74

Table 5-7 The average reward and completing time of given random instructions 76

Table 5-8 Similarity comparison of instruction order of planners...................................77

Table 5-9 The average similarity cost of given random instructions...............................77

Table 5-10 Similarity comparison between human scheduling and optimal solution.....83

doi:10.6342/NTU201902426

doi:10.6342/NTU201902426

Chapter 1 Introduction

In this chapter, we introduce the overall concept of our proposed system. The content includes motivation, research objective, contribution, and the overview of the thesis.

1.1 Motivation

Robotics has been one of the vigorous research in engineering. Due to numerous potential applications in both industrial environment and field of home care, robots are regarded as a suitable platform for carrying cutting-edge technology so as to serve the human society better. On the other hand, thanks to the advances in computing, increasing speed and storage, artificial intelligence as well as deep learning have successfully demonstrated the power in solving complicated problems. Therefore, how to realize those methods into robots has already been one of the crucial and fascinating researches in the field of robotics.

Among all robotics researches, one of the gradually increasing topics is the social and companion robots. Due to aging of the society and decrease of children, such robots provide aids to household environment including regular family and senior center. The resulting state apparently is an integration among robotics, perception, and human robot interaction (HRI). As a result, an efficient system that consists of multiple applications is definitely required so that robots can be deployed into our human society realistically. In our scenario, the objective is to enable a social and companion robot to accomplish various tasks from different users while the robot is navigating in a household environment. On top of that, the robot tries to generate tasks on its own so as to assist human beings actively.

Nevertheless, in contrast to our approach, the current research often focuses on

doi:10.6342/NTU201902426

optimizing a certain function instead of taking overall system into consideration. For example, deep learning approaches in the field of robot perception concentrate on reaching high accuracy in dataset for specific usage such as action recognition. However, such system may consume a great deal of computational resources merely for a single application. For implementation in, say, senior center, this is somehow inefficient as more computers may occupy too much space and cause too excessive power consumption. What's worse, more human efforts are required for maintaining the functionality of the whole system. Rather, we prefer to allocate resources for deep learning algorithms which deal with more general functions, but then achieve specific goals through heuristic methods on the basis of those functions.

Take activity recognition for instance, deep learning models [1][2] have remarkable performance, whose accuracy exceeds 90%, on open-source datasets like UCF-101[3]. Nonetheless, these methods are often offline, require powerful hardware, and thus become impractical for real-world demonstrations. On the contrary, we utilize deep learning models to find objects and human skeleton which are more general for different applications and design heuristic probability model to achieve framewise real-time activity recognition on top of them. Note that these detection results from deep learning approaches can also be utilized to other applications like human localization and identification. With high accuracy and robustness on fundamental deep learning techniques, our robot system can react to different circumstances properly in time.

In addition, traditional planning algorithms often considers constant reward values and deadlines such as traveling salesman with profits [4][5]. However, such algorithms may lead to unpleasant user experience in HRI. Thus, we here take into account that robot should accomplish every instruction and try its best to serve everyone effectively and efficiently.

doi:10.6342/NTU201902426

1.2 Challenges

First of all, designing a comprehensive system for a robot to interact with human rationally requires numerous techniques, not to mention the limited computational power. Traditionally, most robot systems aim to complete relatively simple tasks that are far from suitable human interaction. Secondly, to make the robot more practical in assisting human daily lives, the system should not only enable the robot to timely perceive the environment but also react to human requests as fast as possible. Therefore, framewise visual perception methods and efficient transformation between perception and decision making are challenges for us to push the robotic research into a new era.

Last but not the least, the robot needs to model the environment such that the problem formulation of completing human tasks can be analyzed in a heuristic way. On top of that, an algorithm for optimally scheduling those tasks as well so as making the system perform smoothly is required. It is worth mentioning that elderly care can be regarded as an activity that requires not only performance but also safety. To address such caring applications, apparently robustness and error handling in the robot programming will definitely be needed. Thus, only when integrating all the components under limited computational resource and timing analysis can the robot become “considerate” in the indoor environment.

1.3 Contributions

In this thesis, we propose a novel system that integrates both sensor perceptions as well as decision making such that the social and companion robot is capable of serving a group of people in a household environment. Through our system, the robot can react to human as soon as it receives requests. Furthermore, while dealing with multiple tasks, the

doi:10.6342/NTU201902426

robot can not only sort them intelligently but also complete them one by one reliably. The major contributions are listed as follows:

- We propose a task-oriented navigation system combining the robot perceptions and decision making in order to achieve complicated human robot interaction. Through the proposed system, the robot is capable of organizing tasks concerning both human needs and its own status.
- To formulate those tasks, we propose an instruction structure composed of decaying reward with regard to priorities and time. Unlike traditional algorithms, our system can accomplish every instruction efficiently without deadline limitation.
- To allocate instructions, we model the indoor scenario into a graph structure. The nodes are semantic locations containing instructions, and edges are Euclidean distances between nodes. As the reward decaying with time, the overall system can be viewed as an optimization process that aims to maximize the reward while minimizing the navigation path. Therefore, we take not only the priority among multiple tasks but also time efficiency of instruction execution into consideration.
- We propose a hierarchical structure for visual perception that takes human-object interaction into account to recognize human activities in real time. Moreover, a verbal perception system is proposed for understanding human requirements as well as sentiments. These perceptions can be applied to assigning priorities of instructions.

1.4 Thesis Overview

The thesis is organized as follows: Chapter 2 introduces the background and related works of the system, including the robot perceptions, task planning, and motion planning, which belong to the field of decision making in robotics and are implemented to become

doi:10.6342/NTU201902426

our system framework. In Chapter 3, we propose the visual and verbal perception methods to achieve human localization, human identification, and action recognition on the basis of techniques using deep-learning based object detection and skeleton detection networks. Chapter 4 reveals the formulation of the perceptions, modeling of the scenario, and the proposed algorithm that solves the task planning problem. Moreover, the complexity of the problem and the accuracy of the proposed algorithm are also discussed. Chapter 5 discusses the simulation results and the real world experiments. For the perception part, we evaluate the human localization, identification, and the action recognition frame by frame. On the other hand, as for the decision making, not only did we discuss the performance of our task planner but also its efficiency. Besides, we compare the human scheduling and our proposed task planner to show if the system is user-friendly or not. That is to say, whether our robot is “considerate” enough. Finally, Chapter 6 is the conclusion and the future works that can be extended based on our work.

doi:10.6342/NTU201902426

Chapter 2 Background and Related Works

In this chapter, we will discuss some background and works that are related to our work. Specifically, there are mainly two foci of our thesis: one is the perception system for the robot to build connection to the indoor environment; another is the task and motion planning, known as TAMP in the field of robotics, that includes decision making process for robot to respond to requests.

2.1 Robot Perceptions

In this section, how robot perceives the world is introduced. The following texts contain introduction and perceptions on the basis of different sensors, including laser-range finder, visual as well as verbal sensors. Finally, a brief discussion about how perceptions affect robot behaviors is given.

2.1.1 Introduction to Robot Perceptions

Figure 2-1 shows some basic modules of autonomous robots, from which one can easily understand that the main objective of perception is to transfer sensor data into semantic meaning. As social robots serve mostly in household environment, it is necessary for them to understand, interpret, and represent the surrounding efficiently and consistently [6]. Thanks to advanced hardware that digitalizes information from the continuous environment, robotics engineers can design further algorithm to link various

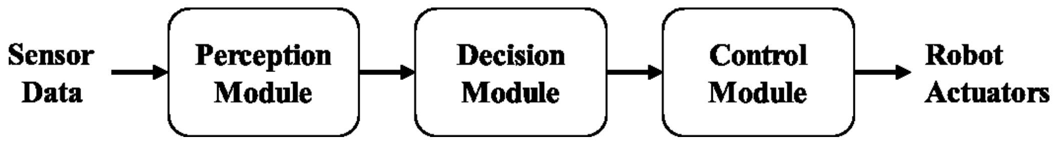


Figure 2-1 Basic modules of autonomous robots

doi:10.6342/NTU201902426

sensory data and come up with semantics, which normally can be categorized as *modeling*, *classification*, and *recognition*. Without these semantics, it is hardly possible for the autonomous robot to make suitable decisions. For instance, without laser perception modules like Simultaneous Localization and Mapping (SLAM) algorithms that model the environment and recognize special relations through creating maps, it is difficult for robots to localize themselves and navigate to the desired destinations [7].

2.1.2 Laser Perceptions: Simultaneous Localization and Mapping

In order to move safe and sound in the real-world environment, it is crucial for a robot to realize *where am I* and *where have I been* [8]. Researches that deal with these problems can be viewed as the Simultaneous Localization and Mapping, also known as SLAM in brief. Typically, SLAM can be decomposed into two portions: *Localizing* and *Mapping*. The purpose of localization is to let the robot be able to estimate its position given the currently built map as well as the on-line sensory data. On the other hand, the mapping procedure is to provide geometry relations among received sensory data so that the robot is able to memorize the structure of the environment. Through processing "localization" and "mapping" simultaneously, the robot can map sensory data to precise location while recognizing its own location at the same time [9].

Among all the available sensors, the two dimensional Laser Range Finder (LRF) provides precise geometry information with relatively lower cost than those of the sensors like Light Detection and Ranging (LiDAR) and is thus widely used for implementing SLAM algorithms. Note that another commonly-used sensor is the camera that performs the so-called visual SLAM [10]. Though visual SLAM may provide more semantic features, its geometry information is usually not as accurate as that of laser-based SLAM and takes more time to come up with the resulting map and the robot location to converge.

doi:10.6342/NTU201902426

For example, the state-of-the-art visual SLAM system, ORB-SLAM [11][12], easily lose track while the social robot is interacting with human. Such imperfection may lower the capability for real-world application of visual SLAM.

On the other hand, laser-based SLAM has shown impressive improvement after three decade development [13]. Through representing the uncertainty of the environment with probability theories, scan matching, occupancy girds, and numerous filters can be applied to SLAM solutions [14]. For instance, in [15], the authors proposed a laser-based SLAM system on the basis of Rao-Blackwellized particle filter, where computing can be speed up using multi-thread of a computer with multi-processor architecture as proposed in [16]. In [17], the authors proposed a laser-based SLAM system through graph optimization [18]. These methods developed robust and precise laser-based SLAM system and remain popular even till now. The SLAM package we implement for the proposed system will be discussed more in *Section 4.1.1*.

2.1.3 Visual Perceptions: Object Detection and Object Affordance

On the contrary to two-dimensional LRF that provides geometry information over a plane, the visual RGBD camera equipped on the robot provide more detailed semantic features for robot to extract useful information for sophisticated decision making modules. In the proposed system, one of the most crucial visual perception is the real-time object detection, meaning that not only classifying certain objects but also locating them on the image coordinate frame-by-frame. With the existence of deep convolutional neural networks, works that aim to solve this problem make significant progress, such as the well-known Fast R-CNN [19] and You Only Look Once (YOLO) [20][21][22]. Those works both extract the bounding boxes of detected objects and reveal the confidence of the results, which can be further utilized for mapping objects on the SLAM systems or

doi:10.6342/NTU201902426

recognize scene according to their semantics.

Among all semantics provided from the objects, we regard the affordance as the most important portion for our system. The original concept of affordance came from psychologist, James J. Gibson, in 1966 [23], which indicates the functionality of objects and how human interacts with the objects. Take a sofa for example, the affordance of the sofa can be *a sitting tool for human to watch television*. On the other hand, it also contains the affordance as *a furniture for people to take a nap*. Thus, the robot can infer more semantic information from the environment with the assistance of affordance, especially for heuristic human action and activity detection. In [24], the robot predicts human actions through probability model on the basis of affordance. In [25], the robot utilizes the anticipatory temporal conditional random field (ATCRF) to infer special-temporal relation of human activities in the environment.

2.1.4 Summary of Robot Perceptions

The robot perception module provides a connection between raw data from the hardware sensor and the decision module. Through modeling, classification and recognition, robot can explore more in the environment and further generate more delicate decisions based on the perception results. Therefore, beside the hardware abilities, the perception module can be viewed as the foundation of an autonomous robotic system.

2.2 Task and Motion Planning (TAMP)

In this section, we will introduce the background and the concept of TAMP, which can be regarded as the major decision module in our social robot system. The content includes how TAMP betters the performance of mobile robot manipulation and human robot interaction (HRI).

doi:10.6342/NTU201902426

2.2.1 Introduction to TAMP

In the field of autonomous robotics, the ultimate goal is to make robots behave intelligently in the real world environment. Among all these subjects that combines Artificial Intelligence (AI) and Robotics, planning is one of the critical components for a robot to complete numerous tasks and instructions robustly and efficiently. Traditional planning problems often emphasize on finding a collision-free motion for the robot to transfer from one state to another given a task. Such planning solvers are generally referred to as the *motion planning* [26]. More precisely, the purpose of motion planning serves as a connection between robot commands and actuators. Robot platforms can be chosen from either robotic arms or mobile robots, and the motion planner will generate a suitable trajectory which avoids obstacles in the continuous space given a specific goal and the configurations of robots. For example, the well-known A* algorithm [27], D* Lite algorithm [28], and Rapidly-Exploring Random Trees [29] all tend to search for a collision-free trajectory given the current environmental state observed from the robot. Thus, the motion planning can be viewed as a command dispatcher for robot manipulation and navigation, as robot actuators complete a series of motions under the constraints of motion planners.

Nevertheless, to deploy robots into real-world environments like industrial factories, rescuing places, or senior center in order to complete a series of tasks as well as to manage the overall situations, it is definitely not enough for a robot to only have the capability of motion planning. To figure out why, we know that motion planning algorithms often concentrate on solving "continuous" problems, which will be extremely time-consuming if the task requires long-term motions [30]. What's more, the more complex a task is, the more computational resources will be need so as to generate a global optimal trajectory due to the accumulative constraints. On top of that, as the number of tasks increases,

doi:10.6342/NTU201902426

motion planning lacks an efficient way to organize different tasks, leading to time-consuming executions or unsatisfied user experience (UX). In short, these factors not only lower the efficiency of robots but also make real-world application more difficult.

As a consequence, designing a mechanism which can connect the given tasks and the motion planner becomes an important topic to deal with various real-world situations systematically. Studies of these task-level robot systems can be dated back to 1961 [31]. Since then, one of the solutions among all is to define a set of discrete motions that are constantly executed under the circumstances of user's performance, and to design an algorithm that is able to complete given tasks by permutations and combinations of these motions [32]. These can be referred to as *task planning*, which takes symbolic tasks as inputs, sorts them, and generates a sequence of discrete motions [33].

Given the description of these two planning algorithms so far, their combination forms a hierarchical architecture, known as *task and motion planning,* or TAMP in brief [34]. That is, provided with a series of tasks, the task planner transforms them into a motion sequence and then pass it to the motion planner, which then generates collision-free paths for actuators to execute. Through this structure, the robot can generate the near-optimal solutions with computational time and resource far lower than those set for the global optimal motion planning. Furthermore, the system can provide a more user-friendly interface since the robot can extract semantic meanings during the task-level. That is to say, human, especially the elders, neither are required to give specific goal position nor to learn programming skills in order to interact with the robot. The overall architecture of TAMP is shown in Figure 2-2.

doi:10.6342/NTU201902426

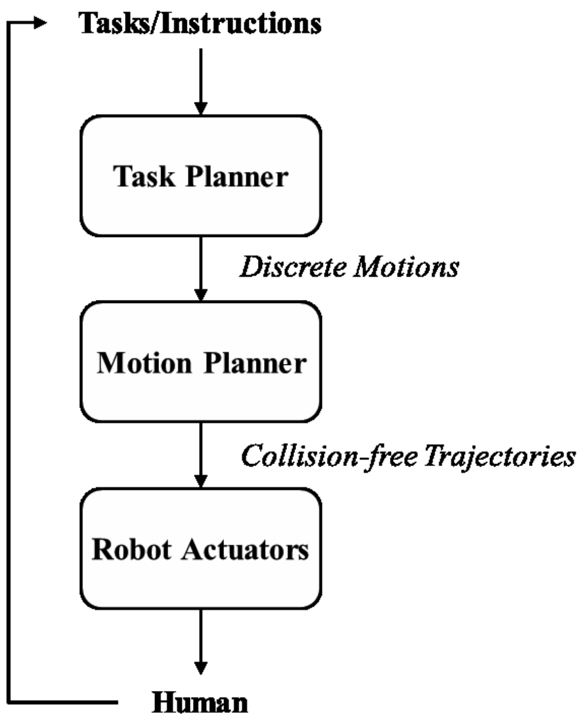


Figure 2-2 The hierarchical architecture for task and motion planning (TAMP)

TAMP is often applied to “pick and place” problem for robotic arm manipulation. It is because that given tasks impose restrictions on the feasible solution space, leading to speed-up of the searching process [35]. In [36], the robot arm tries to search a better grasp position according to the shape of objects. Through this method, the robot can not only eliminate the searching space, but also predict the location of its end effector. In [37], the industrial robot arm operates under task points according to the genetic algorithm. In [38], the work utilize learning algorithm to predict solution constraints instead of solutions and thus speed up the searching time. These related works show that TAMP can enhance the efficiency in robotics in comparison with those subject to traditional motion searching.

2.2.2 TAMP for Mobile Robots

As TAMP being widely used for arm manipulation [32]-[38], it can also be applied to solve navigation tasks for mobile robots to adapt to complex surroundings [39]. The

doi:10.6342/NTU201902426

planning dimension of mobile robots, usually under two-dimensional environment, is less than that of robotic arm manipulation, which equals to the number of joints. However, the processing time for mobile robots to navigate from one position to another is usually longer than that of the arm. Therefore, TAMP can still show its advantage in improving the efficiency when performing low-dimensional planning while the mobile robot is under navigation. In [40], the authors propose a planning system that is able to demonstrate dynamic low-level path corrections and high-level re-planning functions using the hierarchical properties of the TAMP. In [41], the work introduces reinforcement learning (RL) and designs a system with inner and outer loop architectures to speed up the convergence time. The system is evaluated in the simulated scenario with discrete motions: *approach*, *open door*, and *go through*. The results show that the TAMP system can be refined faster with the help of RL. Nevertheless, these works simply conduct experiments either merely simulation results with rather simple motions, which is not sophisticated enough for real world applications like in the senior center.

2.2.3 TAMP for Human Robot Interaction (HRI)

Due to the fact that this thesis is aimed at enabling a robot to provide assistance in a senior center. Thus, human robot interaction (HRI) becomes a must. Through TAMP, robots can make adequate and user-friendly decisions, building a connection between human and robots [42]. In [43], the task planner predicts the human motion and generates safe trajectories in a human-robot shared environment. The authors demonstrate their system under a human robot collaborative scenario and show the anticipatory behavior towards the robot system. In [44], the authors proposed an autonomous assistive robot that is able to serve multiple users by incorporating a finite state machine of different tasks. On top of that, the same research team proposed [45], with the task planning being

doi:10.6342/NTU201902426

solved through Mixed-Integer Programming (MIP) and Constraint Programming (CP) for optimizing the task planning subject to different temporal constraints. The experiments of both [44] and [45] were designed for mobile humanoid robot to remind and to host bingo games for a group of people under the commands from the caregiver.

2.2.4 Summary of TAMP

What have been described previously shows the significant positive influence about how the TAMP can improve the robot operations including robotic arm manipulation, mobile robot navigation, and human robot interaction. Not only can the robot generate suitable, collision-free trajectories, but it also can understand the semantic meaning of requests and complete them through discrete motions in a systematic way. By applying TAMP in robotic applications, robots are then capable of making more intelligent decision to handle various HRI problems in the complex human-involved environment.

doi:10.6342/NTU201902426

Chapter 3 Visual and Verbal Perceptions

In this chapter, we are going to discuss how robot perceives the environment. Basically, the two main sensors are the microphone and cameras embedded on the robot. There may exist other embedded sensors such as Sonar, Infrared (IR), Laser Range Finder (LRF), and even Light Detection and Ranging (LiDAR). Although these sensors may have high precision and are useful for mapping and localization process, they simply focus on extracting geometry features, which is insufficient for the social companion robot to deal with perception task while interacting with human. Therefore, we mainly discuss on the usages of cameras and microphones in the following sections.

A social companion robot may need a microphone to receive verbal requests from human beings. In order to deal with verbal perception, the robot needs to first convert human speech into words, known as Speech to Text (STT). After that, not only do the robot receive the commands from people, but also estimate the human emotions from the words. These words can be formulated into instructions for our decision system, which will be discussed in the sequel of this chapter. Moreover, the robot can generate suitable responses while having a chat with human employing the verbal perception techniques.

Other than verbal perceptions, visual perception is also crucial for a social robot to make adequate response in a household environment. Compared with other commonly-seen embedded sensors, cameras contain useful information for robot to infer the surrounding other than geometry structure with relatively low cost. For example, the RGBD cameras have higher Cost-Performance Ratio than LiDAR when speaking of three-dimensional spatial detections. Through implementing Computer Vision techniques, the robot is able to detect human beings, recognize the indoor scene, and detect human actions as well as activities. This information collected by visual perception can be

doi:10.6342/NTU201902426

utilized for us to design sophisticated methods to make the social robot to generate suitable decisions while interacting with human-beings.

3.1 Preliminary

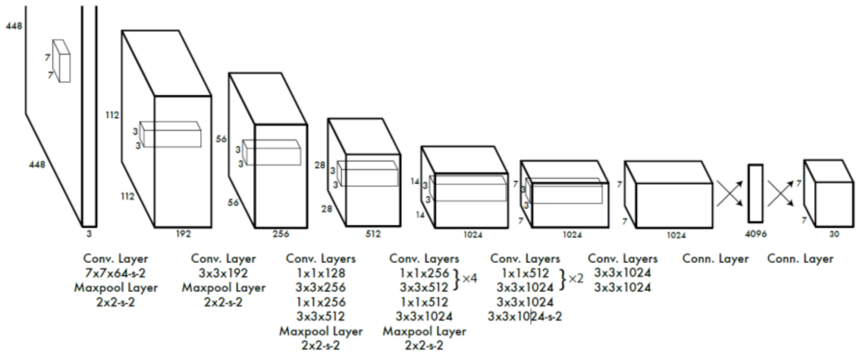
In this section, we first introduce the two open source packages on which this thesis is based. More specifically, in order to obtain accurate and robust results on object detection and human skeleton detection in real time, two open source packages, YOLO[20][21][22] and OpenPose [46][47][48][49], are chosen. These deep learning-based methods provide high performance for our system such that more high-level recognition can be designed. In the following sections, we briefly introduce these two packages respectively

.

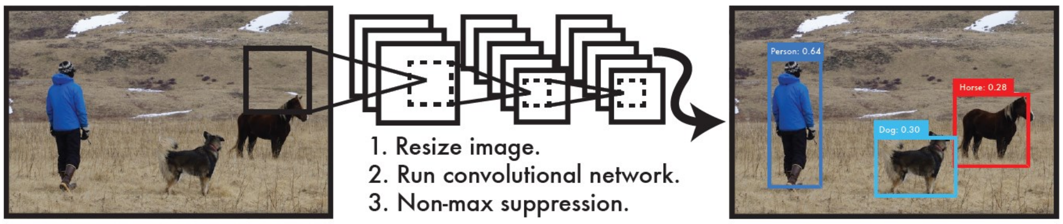
3.1.1 You Only Look Once (YOLO)

YOLO is an open source package for real-time object detection system. Not only does it exist gradually improved versions but also can be viewed as the state-of-the-art deep learning architecture for computer vision detection. YOLO system utilizes a single Convolutional Neural Network (CNN) model so as to perform end-to-end architecture, as shown in Figure 3-1(a). In other words, it only requires a series of images as input and bounding boxes as ground truth to train the overall model, which can be viewed as a regression problem. It also provides an idea that divides the resized images into $S \times S$ grids and then predicts the distribution over labels of classes, the center, and the width and height of objects in each grid, shown in Figure 3-1(b) and (c). On top of that, given a pre-trained YOLO model, one can directly apply it for framewise real-time object detection through providing raw images constantly. The input images will be resized to

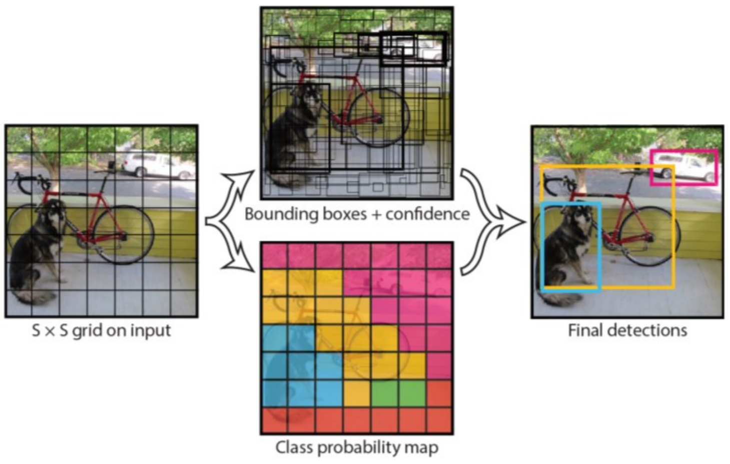
doi:10.6342/NTU201902426



(a) The network architecture of YOLO



(b) The image processing flow of YOLO



(c) Separating images into grids and transferring into regression problem

Figure 3-1 YOLO system for real-time visual object detection [20]

rectangles with length being 448, and pass through the CNN. After that, the output result will be the bounding boxes of each detected objects as well as their class probabilities, also known as the confidence of existence. In the paper, they define the confidence as shown in Eq. (3-1):

$$Confidence = \ P_r(Object) \times IOU_{pred}^{truth} \tag{3-1}$$

doi:10.6342/NTU201902426

Besides, during the testing procedure, the class-specific confidence scores for every bounding box can be calculated through Eq. (3-2):

$$P_r(Class_i|Object) \times P_r(Object) \times IOU_{pred}^{truth} = P_r(Class_i) \times IOU_{pred}^{truth} \quad (3\text{-}2)$$

The main contribution of this network is that not only does it generate results in a short period of time but also is more accurate than other real-time systems comparing the mean average precision (mAP). In brief, [20][21][22] summarize the strength and robustness of YOLO network as follows:

- For real-time detection, the base model is able to process images at 45 frames per second (fps), and 155 fps if one uses a faster version called Fast YOLO with satisfactory mAP outcomes [20]. As for improved version, YOLOv3, a single image can be processed within 22 milliseconds at 28.2 mAP [22].
- The false positive prediction on the background of is lower than those of the other state-of-the-art networks such as Fast R-CNN [19].
- The generalizable representations of object images can be learned by the network.

In our system, we apply YOLOv3 as our fundamental object/human detection system. Based on this deep-learning method, the robot is able to capture human beings as well as objects surround them.

3.1.2 OpenPose: Human Anatomic Skeleton Detection

The OpenPose is another open source package developed by Carnegie Mellon University [46] with the aim to detect multiple human poses with RGB images as inputs in real time. While training and evaluating on the COCO 2016 key-points challenge as well as MPII datasets, this multi-threading system written in C++ language with Open Source Computer Vision (OpenCV) and Caffe [50] is able to perform real-time multiple

doi:10.6342/NTU201902426

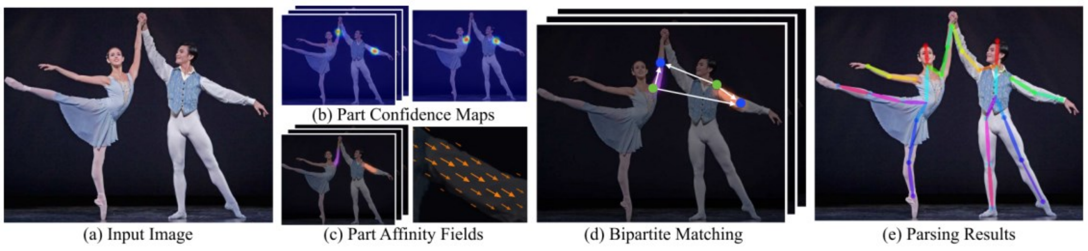


Figure 3-2 Image processing flow of OpenPose

human skeleton detection, which can be further applied for locating people as well as interpreting their body languages. The architecture is able to learn not only the location of body parts, represented as Part Confidence Maps, but also their association among one another through the non-parametric representation, known as Part Affinity Fields (PAFs), in order to increase the detection accuracy. As shown in Figure 3-2, the system takes the raw RGB image as inputs and jointly predicts the confidence location of body parts and the PAFs. Then, the system performs bipartite matching so that the body joints of a person can be linked, while that of different people can be separated. The detailed network architecture is shown in Figure 3-3.

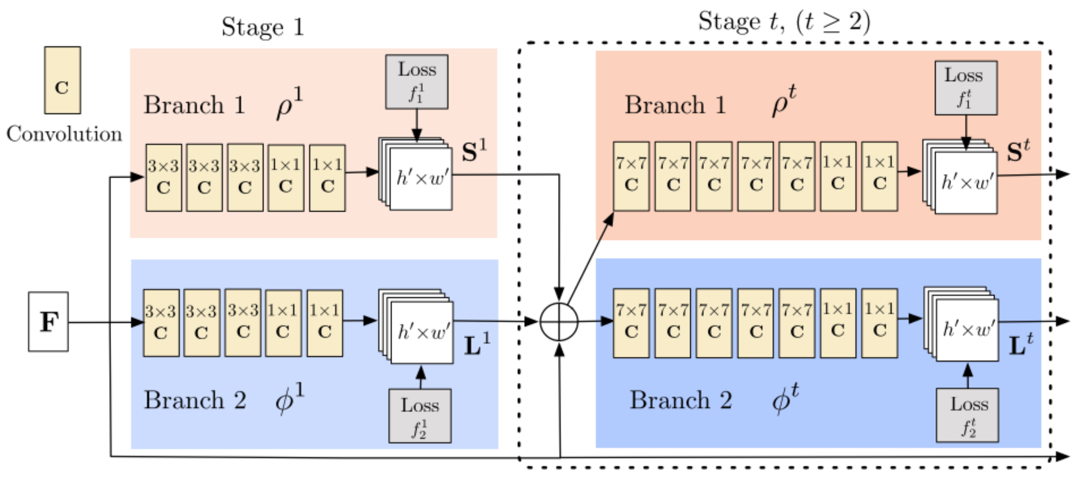


Figure 3-3 Detail network architecture of OpenPose.

doi:10.6342/NTU201902426

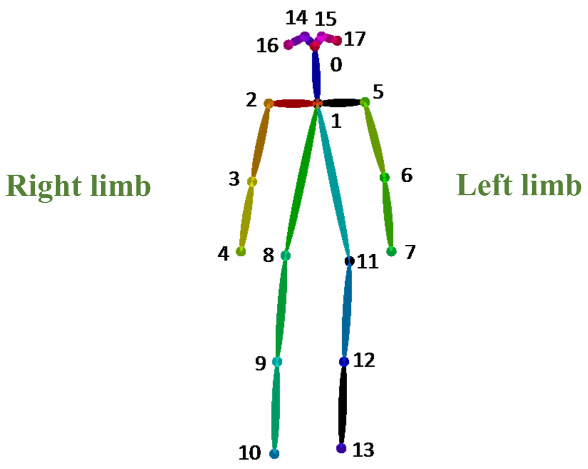
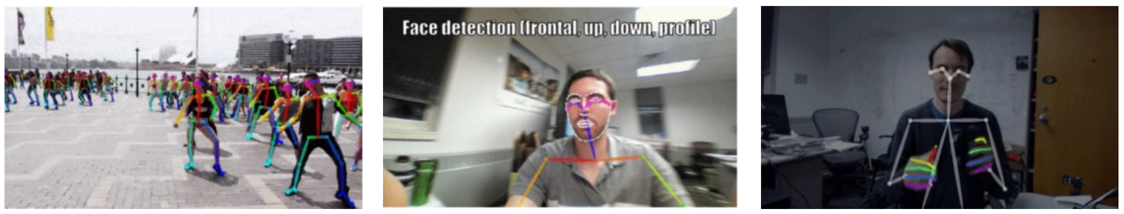


Figure 3-4 Labels of joints for OpenPose



(a) Multiple people pose detection in RGB image (b) Face detection and upper limbs detections (c) A example of upper limbs and hands detection

Figure 3-5 OpenPose examples [51]

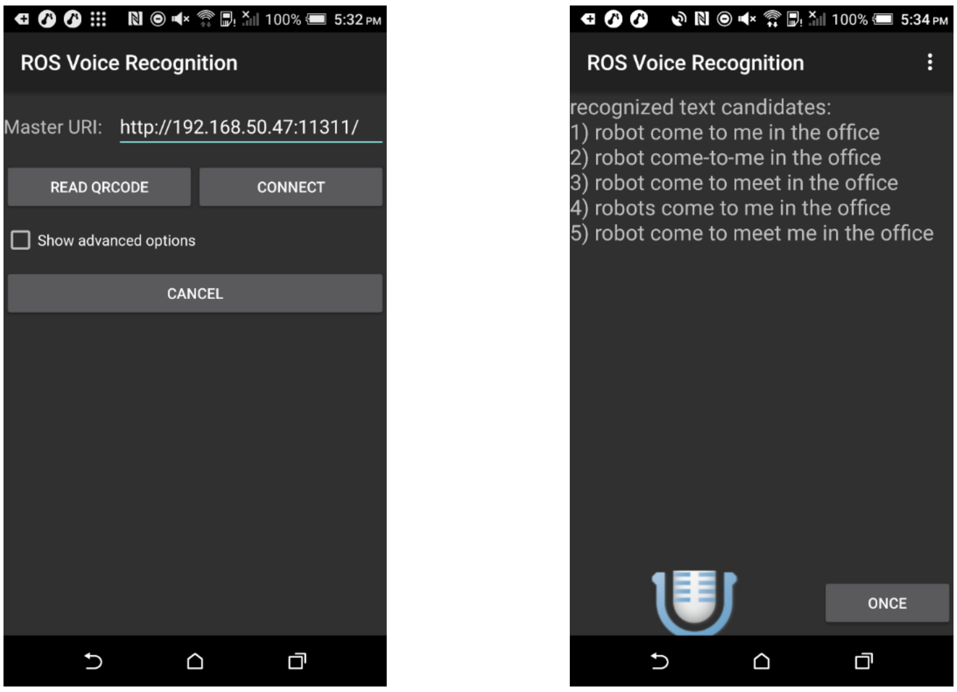
With the assistance of OpenPose, the gap between human verbal messages and body gesture can be filled. On the other hand, robots are able to have better visual perception ability to sense human beings. Furthermore, more heuristic algorithms can be designed based on the skeleton detection results from OpenPose so that the robots have the capability to identify human and recognize his or her actions. Robots may also sense emergency condition actively without people request for help. On the other hand, OpenPose also provides alternative version hand-specific pose detections that catch positions and movements of fingers, indicating that hand gestures can be one of the communication tools among human-robot interaction (HRI). The human anatomic skeleton joints are labeled in Figure 3-4, and some example results from the OpenPose are shown in Figure 3-5.

doi:10.6342/NTU201902426

3.1.3 Speech to Text and Emotion Recognition

As for the speech to text (STT) package, we utilize the application called the *ROS Voice Message* embedded on the Android operation system developed by the Jouhou System Kougaku Laboratory [52]. The reason we perform STT on the smart phones is that the system is targeted at the household environment, and smart phones become adequate tools for family members to communicate to robot. People can send requests in their individual rooms to ask the social robot for help. Figure 3-6 shows the graphic user interface of this package, where the mobile phone first connects to our desktop server and then infers the STT results once receiving human voice.

Another crucial information for our verbal perception system is the emotion hidden in human words, known as sentence sentiment classification. Although there are related works that take advantages from deep leaning techniques to recognize sentiment from sentences, they usually require heavy computational resource, lowering the performance



Master URI: http://192.168.50.47:11311/

recognized text candidates:
1) robot come to me in the office
2) robot come-to-me in the office
3) robot come to meet in the office
4) robots come to me in the office
5) robot come to meet me in the office

(a) Connection to the server (b) STT results

Figure 3-6 The GUI of ROS Voice Message

doi:10.6342/NTU201902426

of the overall system. Therefore, we choose *Valence Aware Dictionary and sEntiment Reasoner (VADER)*, a package that analyzes sentiments through heuristic algorithms [53]. This package is a simple rule-based model with high efficiency and performance in comparison to the state-of-the-art semantic analysis including machine-learning methods. With the usage of the open-source packages, the verbal perception system can be designed in an efficient way as discussed in *Section 3.3*.

3.2 Methodology for Visual Perception

In this section, the detailed methodology of visual perception techniques is introduced. We mainly concentrate on the following problems: where is the human, who is the human, and what is the human doing. These problem are equivalent to *Human Localization*, *Human Identification*, and *Action Detection* respectively. The purpose of our system is to design efficient and robust algorithms on the basis of aforementioned deep-learning packages to give solutions toward the above issues. There may exist other deep-learning methods to deal with the same issues with larger scale such as more human instances in the human identification task or more action categories in action recognition. Nevertheless, these methods often provide off-line detection given a pre-recorded video while consuming a lot of computational resource. In contrast, for indoor scenario like family household or elder house, human identification can be condensed into family members or relatives that often exists in the home environment, and action categories can be eliminated into indoor activities. Thus, the advantage of our algorithms is that the system can generate satisfactory outcomes shortly after the low-level detection results without consuming large computational resources like GPU. On top of that, we design a human database which can store the perceived information as human status so that the

doi:10.6342/NTU201902426

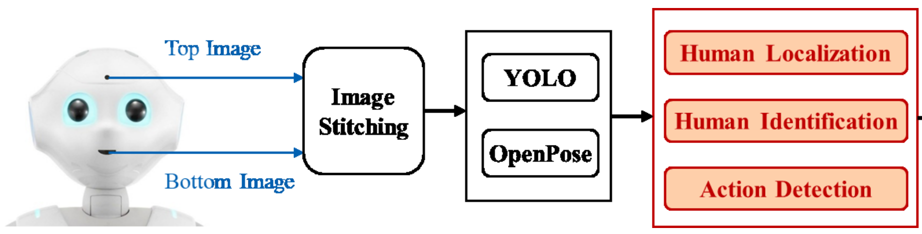


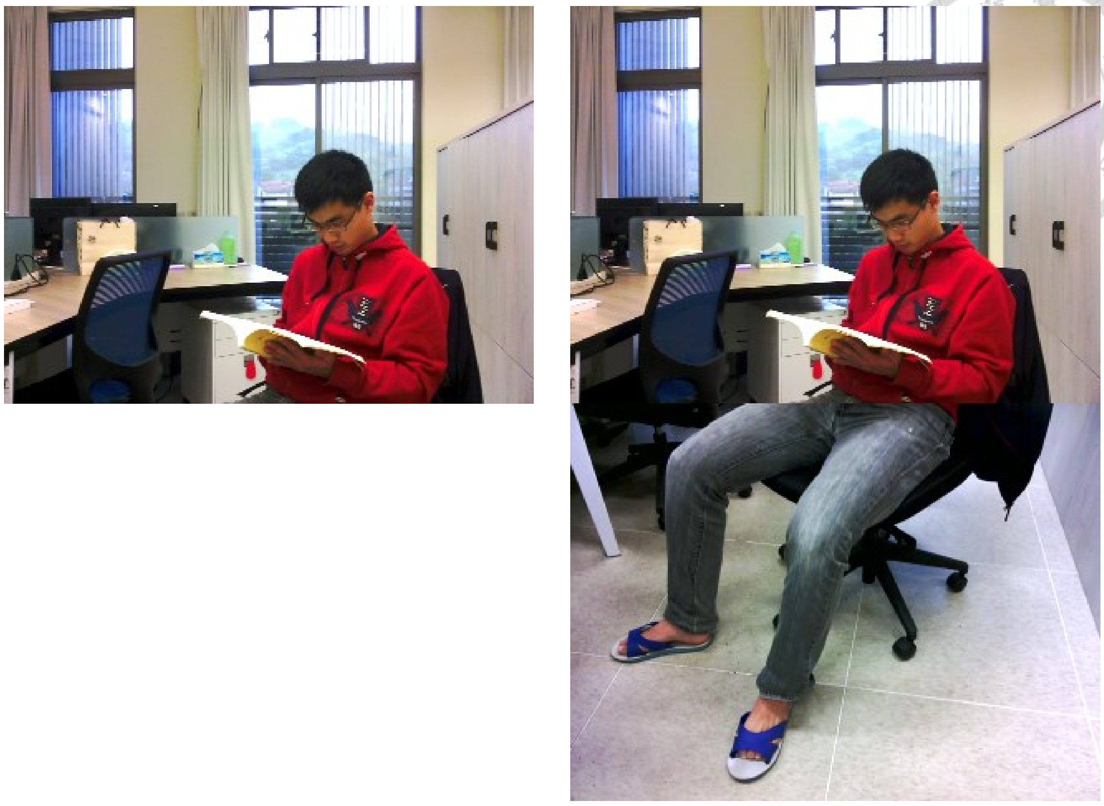
Figure 3-7 Block diagram of visual perception

robot can recall at any time. With the assistance of our methodology, the robot is capable of reacting under social circumstances quickly. The sub-system flow chart of our visual perception module is shown in Figure 3-7.

3.2.1 Image Stitching

For our social robot, the field of a single camera is too small to detect the whole body of human beings. Without the overall body pose, it is difficult for robot to generate suitable social response. Fortunately, there are two identical cameras embedded in our robot system in vertical direction. As a result, we concatenate two images and run the deep-learning models. Through the image stitching technique, not only can more objects be detected, but also the whole human anatomic body skeleton can be recognized. Figure 3-8 shows some detection results between single camera images and stitched images. Although there may exist gap after stitching, the detection of YOLO and OpenPose can still come up with true positive result, especially the human skeleton. An example is shown in Figure 3-14

doi:10.6342/NTU201902426



(a) Single RGB image from top camera (b) Stitched image

Figure 3-8 An example of image stitching

3.2.2 Human Localization

In our system, the robot would like to know where people are in the indoor environment in order to serve them. Practically speaking, the location under semantic map is much more crucial for us than the geometry location. For example, *"Alex is in the living room"* is more meaningful for the robot to approach and complete tasks in comparison to *"Alex is in position (x, y)."* Therefore, we approximate the location of people using the RGBD camera. Note that other methods may utilize RGB camera as well as laser range finder and localize human through sensor fusion. Nevertheless, the laser range of Pepper is simply 1.5 meters, which is relatively shorter in comparison to depth camera. In the following *Section 5.2.1*, we show that our system can localize human within 2.7 meters through RGBD camera.

doi:10.6342/NTU201902426

With advantage of RGBD camera and human/object detection system as YOLO mentioned above, the robot is able to infer the location of objects as well as people. Introduced from literatures related to computer vision [54], the two-dimensional pixel coordinates in RGBD images can be mapped into three-dimensional coordinate through the so-called *Pinhole Model*. The equations of the *Pinhole Model* are shown from Eq. (3-3) to Eq. (3-5):

$$x_r = \frac{(p_x - c_x)}{f_x} \times d \tag{3-3}$$

$$y_r = \frac{(p_y - c_y)}{f_y} \times d \tag{3-4}$$

$$z_r = d \tag{3-5}$$

where the (x_r, y_r, z_r) are the position in the three-dimensional coordinate relative to the robot view frame (X_r, Y_r, Z_r), (p_x, p_y) is the pixel position relative to the RGBD image coordinate (u, v), (c_x, c_y) denotes the center of the camera, (f_x, f_y) is the focal length in the x, y direction. d is the value on the (p_x, p_y) in the depth image. Figure 3-9 shows an example of parameters in an image.

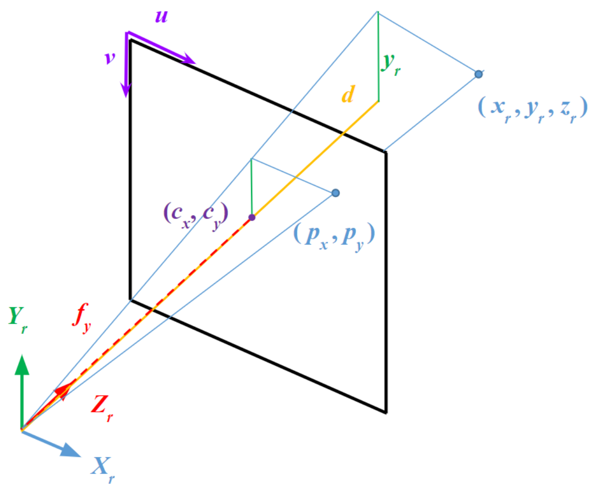
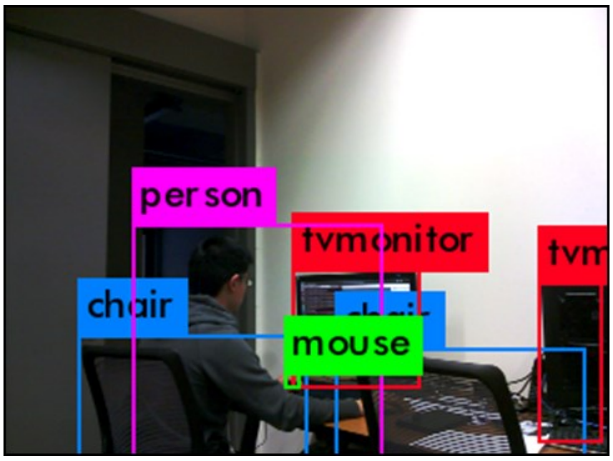
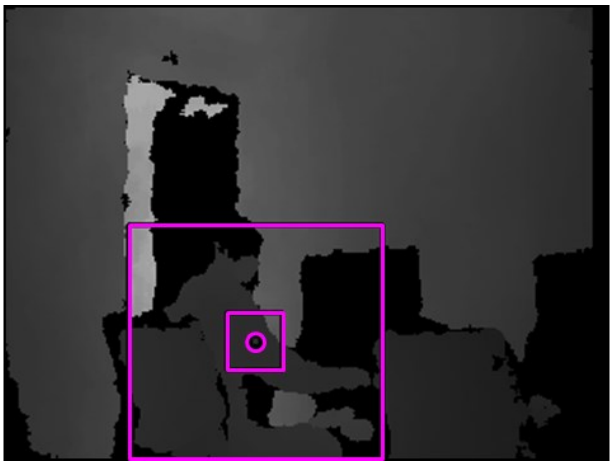


Figure 3-9 The parameters for coordinate transformation

doi:10.6342/NTU201902426



(a) RGB image of the robot top camera with YOLO detection

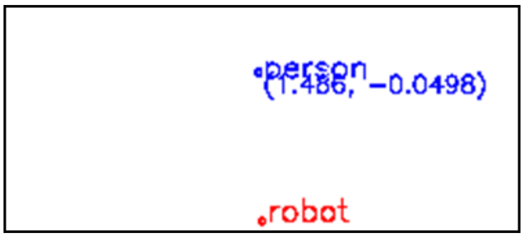


(b) Depth image of robot camera with human bounding box

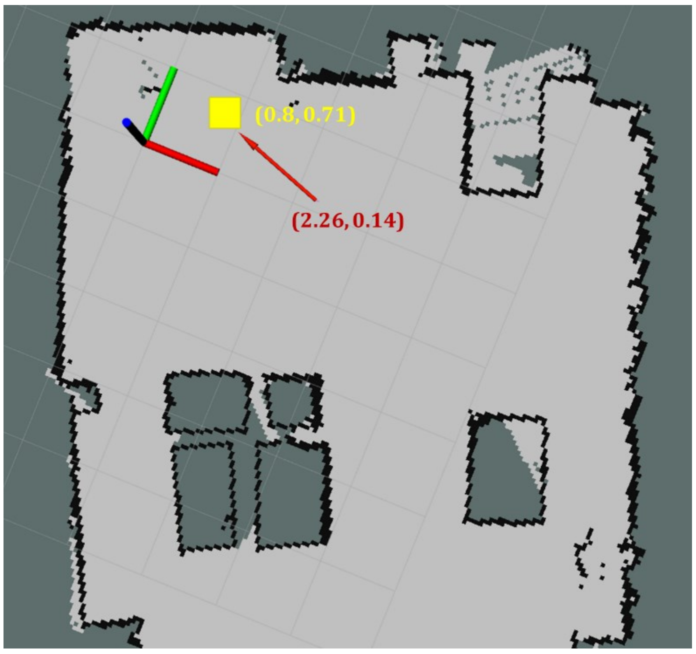
Figure 3-10 The RGBD view of robot top camera

Given the bounding boxes of targets from RGBD camera, the robot can obtain their depth values through depth images. In our approach, we choose the box center and a rectangular boundary that is adaptive to the size of a bounding box from the depth image and calculate the average depth value, as show in the Figure 3-10(b). Note that one can also implement human/object segmentation from the depth image. Nevertheless, we approximate the average depth of objects and people using the value around the center of the depth image so as to lower the computation consumption. After that, the system can project the three-dimensional position into two-dimensional semantic map and perform

doi:10.6342/NTU201902426



(a) Position relative to robot base



(b) Position in the global map

Figure 3-11 The localization results. Yellow box is the human location on the global map, while the red arrow is the robot current location.

the coordinate transformation so as to map the approximate location from the robot frame to the global map frame and extract the semantic location. Figure 3-11 illustrates the relative and global position of our proposed localization result.

3.2.3 Human Identification

While approaching toward human, the robot needs to further understand "who" it is interacting with. This refers to the human identification problem. Though there are works related to human identification using facial features, they may require human to face

doi:10.6342/NTU201902426

toward the robot directly. What's more, robot may need to reach very close to human in order to achieve high enough resolution. This is somehow difficult for household applications as human may feel uncomfortable from robot interruptions. As a consequence, our system tends to identify human-beings through their dressing.

The whole process can be separated into two phases: *Initialization* and *Recall*. During the Initialization phase, the robot will approach an unknown person and greet to him or her, asking the name, gender, and age. The current receiving image of that person is known as the query image. With the advantage of OpenPose, the system can capture the RGB value on the pixel of the body joint. The system then extracts and stores the RGB values on *chest*, *left shoulder*, and *right shoulder*, which are *joint 1*, *joint 2*, and *joint5* in the Figure 3-4 respectively. The Recall phase will be triggered when robot is identifying the person before executing an instruction. Given the person inside the current camera frame, also known as the testing image, the robot first extracts the RGB value of joints 1, 2, and 5 on the basis of OpenPose. Then it will compare the color similarity using a low-cost approximation of color metrics inspired from [55]. This algorithm takes the concept of weighted Euclidean distance, with the weight factors representing how intense the "Red" component in the color is. Given two colors, $\boldsymbol{C_1}$ and$\boldsymbol{C_2}$, with RGB values $(\boldsymbol{C_{1,R}}, \boldsymbol{C_{1,G}}, \boldsymbol{C_{1,B}})$ and $(\boldsymbol{C_{2,R}}, \boldsymbol{C_{2,G}}, \boldsymbol{C_{2,B}})$ respectively, the equations of calculating the approximated distance metrics, $\Delta\boldsymbol{C}$, are shown from Eq. (3-6) to Eq. (3-10):

$$\bar{\boldsymbol{r}} = \frac{\boldsymbol{C_{1,R}} + \boldsymbol{C_{2,R}}}{\boldsymbol{2}} \tag{3-6}$$

$$\Delta\boldsymbol{R} = \boldsymbol{C_{1,R}} - \boldsymbol{C_{2,R}} \tag{3-7}$$

$$\Delta\boldsymbol{G} = \boldsymbol{C_{1,G}} - \boldsymbol{C_{2,G}} \tag{3-8}$$

$$\Delta\boldsymbol{B} = \boldsymbol{C_{1,B}} - \boldsymbol{C_{2,B}} \tag{3-9}$$

doi:10.6342/NTU201902426

$$\Delta \boldsymbol{C} = \sqrt{\left(\mathbf{2} + \frac{\bar{\boldsymbol{r}}}{\mathbf{256}}\right) \times \Delta \boldsymbol{R}^{\mathbf{2}} + \mathbf{4} \times \Delta \boldsymbol{G}^{\mathbf{2}} + \left(\mathbf{2} + \frac{\mathbf{255} - \bar{\boldsymbol{r}}}{\mathbf{256}}\right) \times \Delta \boldsymbol{B}^{\mathbf{2}}} \tag{3-10}$$

According to the approximation, the smaller ΔC is, the more similar C_1 and C_2 are. Therefore, the robot can identify human anytime by comparing the current observation with the database built from the initial phase. Note that common identification methods usually rely on human faces. Nonetheless, such methods are not reliable for cameras with low resolution. What's more, these methods often require human to face toward the robot, which may interrupt people who are reading, chatting, or even sleeping. Thus, although our system needs to update once the human changes his/her clothes, we can still obtain satisfactory results on low resolution cameras without interfering human beings. **Algorithm 3-1** shows the detailed of our identification method works:

Algorithm 3-1 Human Identification

```
1.  Define: Joint positions of a person according to Figure 3-4:
2.          $J = \{(x_{0,}y_0), (x_{1,}y_1), \dots, (x_{17,}y_{17})\}$
3.          Color threshold $C_{th}$
4.  Input: Current stitched image $I = (R_I, G_I, B_I)$
5.          Joint position of N people in current frame $\boldsymbol{J_c} = \{J_{c,1}, J_{c,2}, J_{c,3}, \dots, J_{c,N}\}$
6.          Joint RGB color of M people in database Human:
7.              $\boldsymbol{R_d} = \{R_{d,1}[J], R_{d,2}[J], \dots, R_{d,M}[J]\}$
8.              $\boldsymbol{G_d} = \{G_{d,1}[J], G_{d,2}[J], \dots, G_{d,M}[J]\}$
9.              $\boldsymbol{B_d} = \{B_{d,1}[J], B_{d,2}[J], \dots, B_{d,M}[J]\}$
10. for $n$ in $range(N)$:
11.     for $m$ in $range(M)$:
12.         $\bar{r} = \frac{R_I[J_{c,n}] + R_{d,m}[J]}{2}$
13.         $\Delta R = R_I[J_{c,n}] - R_{d,m}[J]$
14.         $\Delta G = G_I[J_{c,n}] - G_{d,m}[J]$
15.         $\Delta B = B_I[J_{c,n}] - B_{d,m}[J]$
16.         $\Delta C = \sqrt{\left(2 + \frac{\bar{r}}{256}\right) \times \Delta R^2 + 4 \times \Delta G^2 + \left(2 + \frac{255 - \bar{r}}{256}\right) \times \Delta B^2}$
17.         if $\overline{\Delta \boldsymbol{C}} < C_{th}$:
18.             return Human(m)
```

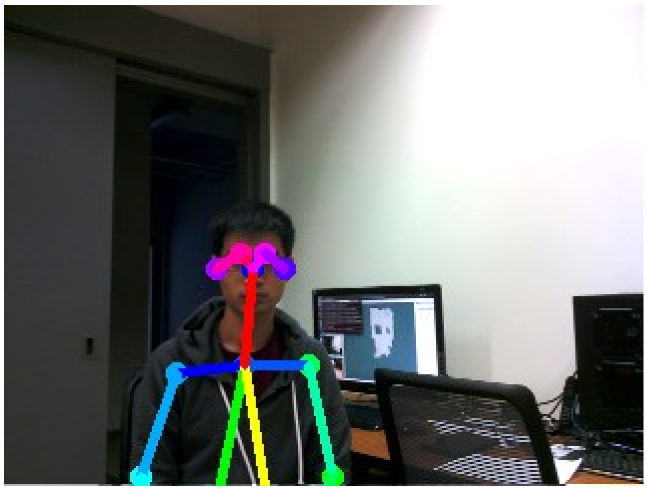
doi:10.6342/NTU201902426

3.2.4 Framewise Hierarchical Human Activity Recognition

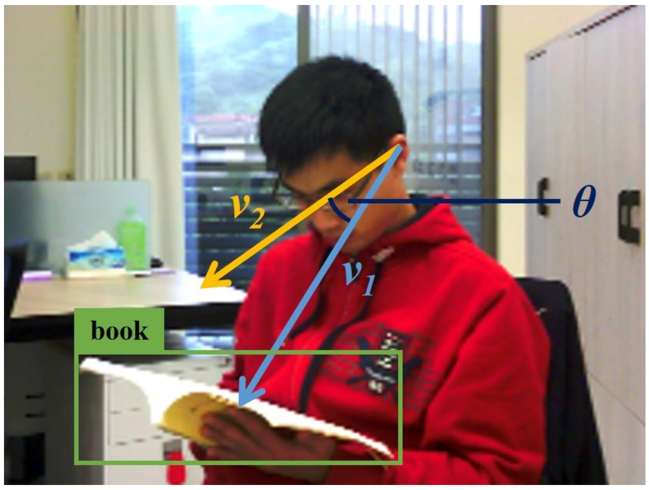
After understanding “where” and “who” the human is, the next step for the system is to detect what he or she is doing. Due to the fact that the system aims to serve in the indoor environment, the number of activity categories are much fewer than that of open source dataset which may require deep learning techniques to make accurate inference. Therefore, in this thesis work, the system utilizes heuristic algorithms to achieve real-time human activity detection. Basically, indoor activities can be separate into two hierarchical types. One is the general activities which can be recognized from anatomic skeleton, such as *sitting*, *standing*, and *lying*. The other type contains more detailed activities that may require other properties from the visual perception like *reading*, *working*, and *watching TV*.

In order to obtain the detailed activities, the system combines the results from both object detection and human pose detection. The indoor activities can be inferred from mainly three sources: *hands*, *eyes*, and *objects*. In other words, the system takes the interactions between these sources and calculate activity scores. Given the detected objects with individual confidences and the skeleton detection results provided by previous perception methods performing on the stitched image of robot camera, the system is capable of fetching the position of hands as well as neighboring objects. In this way, the system is able to recognize what objects the person is holding or may use in the future. Besides, the robot can also find out whether the human is facing toward itself or not through the head orientation and the eye direction. If not, the system forms two vectors, both starts at the ear (v_1 in Figure 3-12 (b)) and ends at the object center and nose (v_2 in Figure 3-12 (b)) respectively. Then, the robot checks whether the object are under human eye sight by calculating the angle in between. Figure 3-12 are some of the examples,

doi:10.6342/NTU201902426



(a) The case when human is facing the robot, robot can detect whether human is facing it by checking the head direction.



(b) The case when human is reading, showing the concept of forming vectors for checking objects in the eye sight.

Figure 3-12 Human perception for detailed activity detection

where v_1 and v_2 are vectors and θ is the angle in (b). Therefore, the system can now know what the person is looking at, which may be highly related to what he or she is doing.

After knowing the set of objects the person is holding and watching. The system can infer the related activities by a pre-defined affordance list. For example, after recognizing

doi:10.6342/NTU201902426

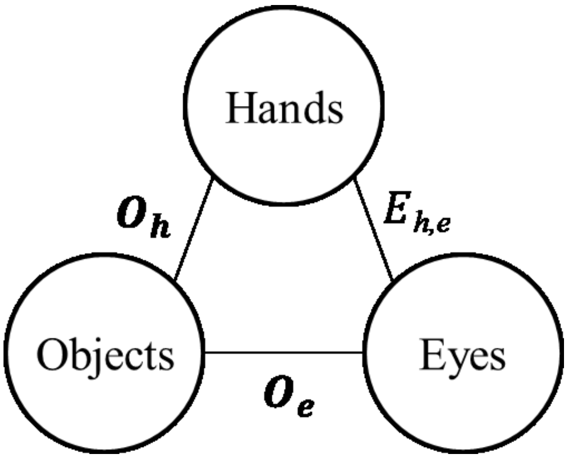


Figure 3-13 Overall relation for detailed indoor action recognition

a bowl near a person's hands, the system may infer that the person is probably *eating*, as a bowl is an *eating tool* according to the affordance list.

In addition to recognizing object affordances for activity inference, we also consider whether the human is watching his/her hands. It is because that some indoor activities may have higher probability given this condition. For example, people tend to look at hands holding a fork while *eating*, and look at hands buckling buttons while *dressing*. The overall relations are shown in Figure 3-13, where $\boldsymbol{O_h}$, $\boldsymbol{O_e}$ are the sets of objects near hands and eye sight respectively, and $E_{h,e}$ is the event whether the target person is looking at his/her hands.

From the above relation, the robot can infer the human activity through calculating scores based on the conditional probability as shown by Eq. (3-11), (3-12) and (3-13):

$$\boldsymbol{Score(a) = w_h P_h(a|O_h)P(O_h) + w_e P_e(a|O_e)P(O_e) + w_E P_E\big(a\big|E_{h,e}\big)P\big(E_{h,e}\big)} \quad (3\text{-}11)$$

$$\boldsymbol{P_h(a|O_h)P(O_h) = \sum_{o \in O_h} P_h(a|o)P(o)} \quad (3\text{-}12)$$

$$\boldsymbol{P_e(a|O_e)P(O_e) = \sum_{o \in O_e} P_e(a|o)P(o)} \quad (3\text{-}13)$$

doi:10.6342/NTU201902426

where a represents the particular activity among total 11 categories, namely, *eat*, *drink*, *watch TV*, *chat*, *call*, *read*, *work*, *store*, *sleep*, *go out*, *wash*, and *others*, $Score(a)$ is the score of all indoor action categories. $P_h(a|\boldsymbol{O_h})P(\boldsymbol{O_h})$ and $P_e(a|\boldsymbol{O_e})P(\boldsymbol{O_e})$ are the action probabilities from given object sets of hands and eye sight, with $P(o)$ being the confidence results from the object detection system. As for every included objects, the term $P_h(a|o)$ and $P_e(a|o)$ mean that how the object o that nears the hands and under view sight respectively can affect the probability of certain action a by providing its affordances. The hyper-parameters w_h, w_e, w_E are weightings that serve as balancing the overall probability and normalizing the final action probability. **Algorithm 3-2** shows how objects are included into sets of $\boldsymbol{O_h}$ and $\boldsymbol{O_e}$.

With the use of detecting sets of objects that may be potentially utilized by the person on the foundation of object detection system and human pose detection system, our proposed action recognition algorithm can capture more sophisticated indoor actions like *eating*, *working*, *reading*, *dressing*, to name a few. In addition, this system is designed under a hierarchical structure. That is, if the probabilities of those detailed actions are too low, the output of the action will become the pose action, namely *sitting*, *standing*, and *lying*. Moreover, since the input of the proposed method is simply an image frame and does not require GPU more heavy computation, the action detection can perform framewise action detection in real time. **Algorithm 3-3** is related to the detailed process of how robot recognize human actions. While there may exist cases that human is peaking at the robot, leading to false positive of $\boldsymbol{O_e}$, these actions are mostly temporary and may be updated as the system is running.

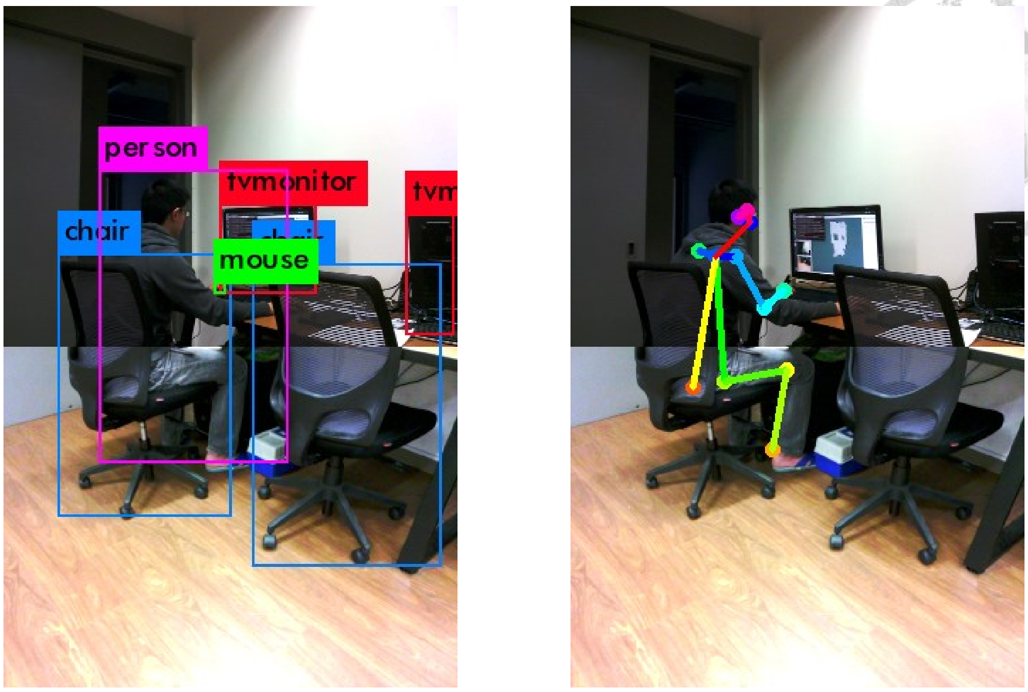
doi:10.6342/NTU201902426

Algorithm 3-2 Get objects around hands and eyesight

```
1. Define: Range factor $\alpha_{hand}$, Eyesight angle range $\delta_{eye}$
2. Input: Current stitched image $I = (R_I, G_I, B_I)$
3.        Joint position of a person $J = \{(x_0, y_0), (x_1, y_1), \dots, (x_{17}, y_{17})\}$
4.        Objects from detection $\boldsymbol{o} = \{o_1, o_2, \dots, o_M\}$
5.        Centers of bounding boxes of $M$ objects $\boldsymbol{c} = \{c_1, c_2, \dots, c_M\}$
6. Initial: $\boldsymbol{O_h} = \boldsymbol{O_e} = \emptyset,\ E_{h,e} \leftarrow \boldsymbol{False}$

7. $\theta_{h,e} = \min_{\substack{(x_0,y_0)\in\{(x_{16},y_{16}),(x_{17},y_{17})\} \\ (x_h,y_h)\in\{(x_4,y_4),(x_7,y_7)\} \\ (x_e,y_e)\in\{(x_{14},y_{14}),(x_{15},y_{15})\}}} \cos^{-1} \frac{(x_e-x_0,y_e-y_0)\cdot(x_h-x_0,y_h-y_0)}{\|(x_e-x_0,y_e-y_0)\|_2 \|(x_h-x_0,y_h-y_0)\|_2}$:

8. if $|\theta_{h,e}| < \delta_{eye}$:
9.     $E_{h,e} \leftarrow \boldsymbol{True}$

10. for $c_i$ in $\boldsymbol{c}$:
11.     $d_{hand} = \min_{(x,y)\in\{(x_4,y_4),(x_7,y_7)\}} \|(x - c_x, y - c_y)\|_2$

12.     $\theta_{eye} = \min_{\substack{(x_0,y_0)\in\{(x_{16},y_{16}),(x_{17},y_{17})\} \\ (x_e,y_e)\in\{(x_{14},y_{14}),(x_{15},y_{15})\}}} \cos^{-1} \frac{(x_e-x_0,y_e-y_0)\cdot(c_x-x_0,c_y-y_0)}{\|(x-x_0,y-y_0)\|_2 \|(c_x-x_0,c_y-y_0)\|_2}$

13.     if $d_{hand} < \alpha_{hand} \cdot \boldsymbol{max}(\|(x_3 - x_4, y_3 - y_4)\|_2, \|(x_6 - x_7, y_6 - y_7)\|_2)$:
14.         $\boldsymbol{O_h} \leftarrow o_i$
15.     if $|\theta_{eye}| < \delta_{eye}$:
16.         $\boldsymbol{O_e} \leftarrow o_i$

17. return $\boldsymbol{O_h}, \boldsymbol{O_e}, E_{h,e}$
```

Algorithm 3-3 Action recognition

```
1. Input: Human-object relations: $\boldsymbol{O_h}, \boldsymbol{O_e}, E_{h,e}$
2.        Probability of each detected objects $P(o)$
3. Define: Probability threshold $P_{th}$, Weighting factors $w_h, w_e, w_E$
4. Output: $P(a)$
5. for $o$ in $\boldsymbol{O_h}$:
6.     $P_h(a|\boldsymbol{O_h})P(\boldsymbol{O_h}) += P_h(a|o)P(o)$
7. for $o$ in $\boldsymbol{O_e}$:
8.     $P_e(a|\boldsymbol{O_e})P(\boldsymbol{O_e}) += P_e(a|o)P(o)$
9. return $P(a)$ according to Eq. (3-11)
```



(a) YOLO object detection (b) OpenPose skeleton detection

Figure 3-14 An illustrative example in the view of the robot

Take Figure 3-14 for instance, the robot observes a person holding a *mouse*, looking at a *monitor*, and not looking at hands. As according to the well-known affordance network, AfNet [56], the mouse contains affordances such as *socio-cultural preference conditioning* and *wrap-ability* that can be inferred as electronics, whereas the monitor contains affordances as *display-ability*. Therefore, given thirteen action categories, $P_h(work|mouse) = 1$, and $P_e(work|monitor) = P_e(watch\ \ TV|monitor) = \frac{1}{2}$. Other action probabilities remain 0 as these objects does not provides affordances that indicates them. If the weighting remains the same, $w_h = w_e$, then:

$$P(work) = \frac{1 + 0.5}{1 + 0.5 + 0.5} = 0.75 \tag{3-14}$$

$$P(watch\ \ TV) = \frac{0.5}{1 + 0.5 + 0.5} = 0.25 \tag{3-15}$$

As a consequence, the robot can infer the person is working as it has the highest probability among all other actions.

doi:10.6342/NTU201902426

3.3 Methodology for Verbal Perception

The proposed verbal perception system is relative straightforward. Leveraging from the open-source packages, the system receives human speech and IP address from the smart phone and transforms speech into texts and analyze his/her emotion through ROS Voice Message on phone as *Speech To Text* and VADER as *Sentiment Analysis* on desktop respectively. The IP address helps the robot to fetch human names and status from the human database. As for the sentiment analysis, according to [57], the threshold of positive, neutral, and negative emotions is ± 0.05. Namely, if the compound score is in the range between 0.05 and -0.05, then the human emotion can be detected as neutral; otherwise the words may contain positive and negative feelings depending on whether the score is larger than 0.05 or lower than -0.05. Finally, the human requests are obtained through predefined key word extraction from the *Extract Requested Robot Functions*. Those key words can be mapped to specific task sequences, which is constructed from pre-defined robot functions. For example, the physical terms in the verbal information will be detected due to a dictionary in the function constructor containing *headache*, *fever*, *sore throat*, to name a few. The overall flow chart is shown in Figure 3-15, in which we present an example of a person Alex sending request as *"Chat with me, I feel bad."* Consequently, the robot may understand the request as *"Alex feels bad and may need to chat."*

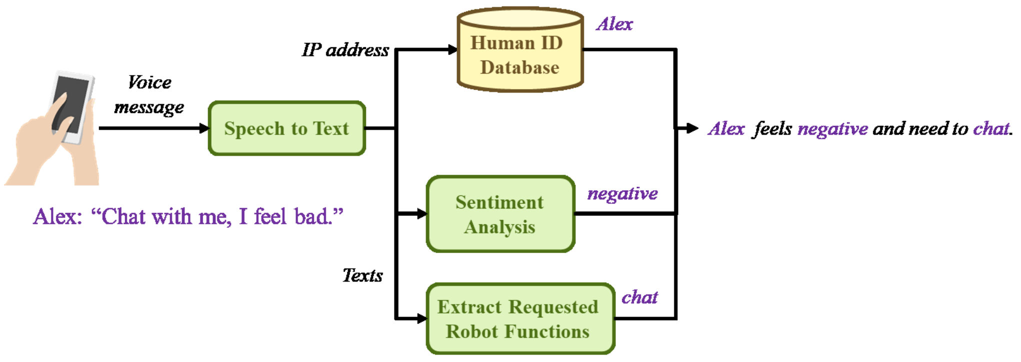

Voice message
Alex: "Chat with me, I feel bad."
Speech to Text
IP address
Texts
Human ID Database
Alex
Sentiment Analysis
negative
Extract Requested Robot Functions
chat
Alex feels negative and need to chat.

Figure 3-15 The flow chart for verbal perception

doi:10.6342/NTU201902426

3.4 Human ID Database Organization

After applying the above methods from the visual perception, the robot can now detect *"where the person is," "who is the person,"* and *"what is the person doing."* Besides, during the *Initialization* phase of human identification, the name, IP address, gender, age, shirt color, and schedules of the person can be obtained while the robot greets to him/her. While the robot is processing tasks inside the environment, visual perception results such as human location, current activity can be memorized. To organize the information, our system forms a *Human* data structure as shown in Table 3-1. With a person mapping to one unique table, the database can be loaded as a dictionary structure and store efficiently in the computer. Therefore, the system can build a large memory database based on individual human beings. Through this organization, the robot can store the visual and verbal perception results in an efficient way.

Table 3-1 *Human* data structure

string **name**	Name of the person
string **IP address**	Mobile phone IP address of the person
bool **gender**	The gender of the person
int8 **age**	The age of the person
int8[] **shirt color**	The shirt color of the person
int8 **location**	The semantic location of the person
int8 **activity**	The current activity the person is doing
string[] **schedules**	The predefined schedules of the person

doi:10.6342/NTU201902426

Chapter 4 Dynamic Multi-Task Social Navigation

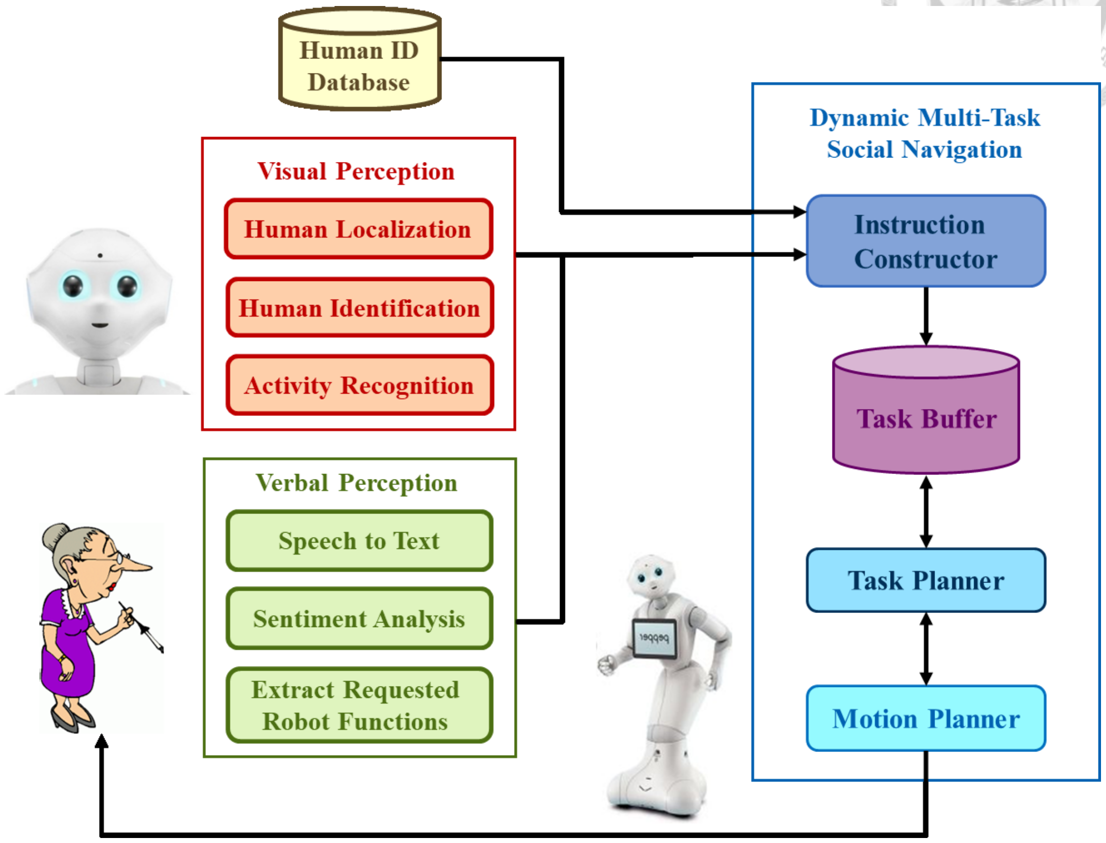


Figure 4-1 Overall TAMP system for dynamic multi-task social navigation

In this chapter, we will explain how the robot completes tasks dynamically while navigating in the indoor environment. The definition of "dynamic" in this thesis is that the robot will react as soon as additional tasks are launched and might therefore change the destination. We utilize the concept of task and motion planning (TAMP) as the framework of our decision making system. The TAMP separates decision making into two parts: task planning and motion planning. Since the motion planning algorithm is mostly related to the preliminary that applying open-source packages to generate a collision-free path, we put our focus on the problem formulation and the algorithm of our task planning sub-system. Figure 4-1 shows the whole structure of our proposed system.

The following sections first reveals the preliminary of this system, including the usage

doi:10.6342/NTU201902426

of laser SLAM as well as the navigation packages, and the complexity analysis of decision problems in the field of algorithm. Secondly, we formulate the perception results into instructions and model the household scenario into discrete graph. Next, we propose our task planning algorithm on the basis of the graph and given instructions. Besides, we also explain the accuracy of the algorithm, meaning that the robot will eventually reaches the instruction destinations instead of getting stuck at the edge. Last but not least, the integration of task and motion planning is introduced, through which we mimic the computer architecture and design the overall TAMP system as well as the data flow.

4.1 Preliminary

In this section, we will discuss the preliminary of the proposed navigation system. First of all, the system takes large advantages from the open-source Robot Operating System (ROS) [58], which contains laser-based SLAM, robust localization, and save navigation packages. On the other hand, the complexity of our proposed algorithm will be discussed in the following sections, we will give a brief introduction to the complexity of algorithm as well as the renowned value iteration method.

4.1.1 Laser-based SLAM: GMapping, AMCL, and Navigation Stack

Since the purpose of this thesis is to design a navigation system for mobile social robot to interact with multiple people, the Simultaneous Localization and Mapping (SLAM) techniques become an important background. In this section, we will discuss the laser-based SLAM which is implemented as the basis for the motion planner of the proposed system, known as GMapping [15][16]. On top of that, given the map built from GMapping, the laser-based localization method, Adaptive Monte Carlo Localization

doi:10.6342/NTU201902426

(AMCL) [59][60][61], can provide the robot position toward the proposed TAMP system to define the robot state.

The GMapping SLAM utilizes the particle filter theory that each particle represents a single map of the environment. Then the critical problem becomes how to decrease the number of particles and generate the final mapping. Based on [15] and [16], the Rao-Blackwellized particle filter utilizes the adaptive techniques to eliminate the particles. Besides, the techniques of taking the robot movements and current observations into account is adopted, and parallel computing speeds up the overall calculation.

After building the map from GMapping package, the robot can answer where it is by applying the AMCL package [62]. The AMCL is originates from the Monte Carlo Localization (MCL), which regards the robot location as particles and try to eliminate them such that the particles will eventually converge to the accurate position. Upgrading from the MCL, the AMCL utilizes the Kullback-Leibler divergence, defined in Eq. (4-1) with P and Q being two probability distributions, to update those particles. In general, the concept of this localization method is to spread particles according to Gaussian distribution centering at the initial position on the built map, with each particle represents a possible location of the robot. After receiving a motion command, the algorithm performs the movement on each particle, calculates the expecting observation, compares with the real world laser measurement, and assigns the corresponding probability of every particle according to the similarity from sensor matching. As the process goes on, the particles will be eliminated while the robot location converges to correct place.

$$D_{KL}(P||Q) = \sum_{i} P(i) \log \frac{P(i)}{Q(i)} \tag{4-1}$$

After knowing the current location as well as the pre-constructed map, the navigation stack package generates a single global path given the target position using either A*

doi:10.6342/NTU201902426

algorithm [27] or Dijkstra algorithm [63]. Furthermore, obstacles in the static map from GMapping package may inflate according to the shape of the robot while performing the global path planning for safe navigation, known as the costmap [64]. For instance, there may exist a shortest path that guides the robot passing a narrow alley. However, as the global path planning is generated on the costmap, the inflated obstacles make the algorithm take robot shape into consideration and thus come up with a less optimal but safer path for robot to avoid collision. While the robot is moving, the Dynamic Window Approach (DWA) [65] generates local path planning for robot to perform save navigation. Through DWA, the robot can avoid collision even observing unknown obstacles that does not exist on the map. Note that there are other local path planning algorithms such as time-elastic band, which is well-known in the field of planning paths for car-like robots and differential wheels [66][67][68][69][70].

4.1.2 Fundamentals of Complexity

Before analyzing the complexity of our scenario, some definition and theorem of algorithm complexity is introduced in this section. According to the lecture [63], complexity analyses aim to classify decision problems with the answer being merely "true" or "false". A common sense is that the decision problems are often denoted with capital letters. Note that all the optimization problem, which complexity analysis cannot be applied directly, can be transformed into decision problems by offering a bound on the value to be optimized. For instance, in [63], the optimization problem SHORTEST-PATH can be interpreted to decision problem, PATH, as: given a graph *G*, two vertices *u*, and *v*, and a number *k*, does there exist a path from *u* to *v* with total cost at most *k*? This relation provides a clue that if the decision problem is "easy" from the complexity analysis, the optimization is not so hard as well.

doi:10.6342/NTU201902426

Typically, the complexity of decision problems can be classified into three types: P, NP, and NP-complete. By definition, if a decision problem is in P if and only if there is a polynomial-time algorithm A such that given an instance s is true if and only if $A(s) = 1$, and $A(s) = 0$ otherwise. On the other hand, a decision problem is NP if and only if there is an algorithm $B(s, t)$ with running time $O(|s|^n)$ such that an instance s is true if and only if there exists a certificate t and $B(s, t) = 1$. In other words, if the answer of the instance is *false*, the algorithm of NP problems will absolutely output 0; however, it will generate 1 with a certain probability if the answer is *true*.

Before introducing the definition of NP-complete, the definition of *reducible* is needed, where there exists a function f between two decision problems X and Y such that all the instances in Y can be mapped to instances in X with the same answer. Based on this definition, a decision problem X is NP-complete if and only if the following two conditions are true:

- X ∈ NP
- ∀ Y ∈ NP, Y is polynomial-time *reducible* to X

That is to say, if X has a solution, then so does Y, but if Y has a solution, then X is not guaranteed to have a solution. Nonetheless, this definition is still abstract. Therefore, to prove that a decision problem is NP-complete, ***Theorem 4-1*** is often applied:

Theorem 4-1

If Problem X ∈ NP, there exists a Problem Y ∈ NP-complete and Problem Y is polynomial-time reducible to Problem X, then Problem X ∈ NP-complete.

In the following sections, we will also show that our household scenario formulation is NP-complete by utilizing ***Theorem 4-1***.

doi:10.6342/NTU201902426

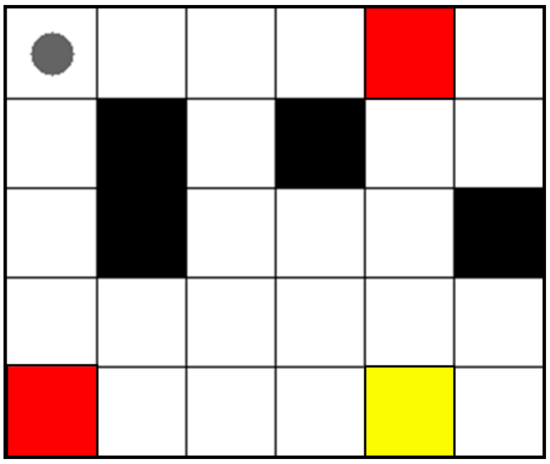


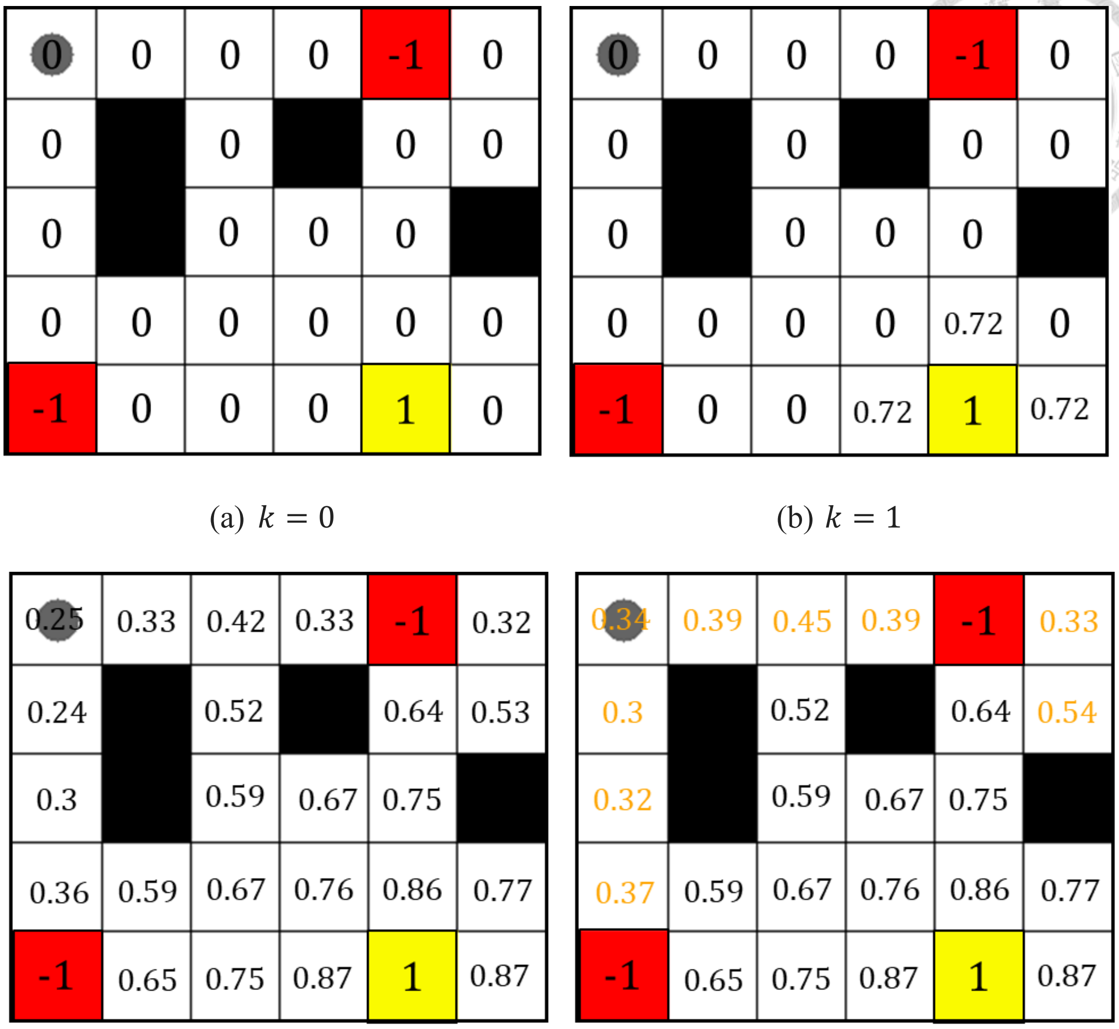
Figure 4-2 An illustrative example of robot in a grid maze. Gray circle indicated the robot position; black blocks are the obstacles; read blocks are the hell that contains negative rewards, and yellow block is the treasure that contains positive reward.

4.1.3 Introduction to Value Iteration

Originated from the Markov Decision Process, the value iteration method [71][72][73] applies dynamic programming to solve the reinforcement learning. Its key concept is to estimate the expected reward on every state taking the probability of individual actions into account. For instance as Figure 4-2, in a grid maze with a single reward, traps, and obstacles, the robot tries to reach the reward by iteratively analyzing the expected reward on every gird.

In this environment, the robot state can be represented as the grid location where the robot stands. For each grid, the robot will calculate the maximum expected reward by the multiplication of action probability as well as the expected reward on the next state. The sum of all possible actions and the reward forms the expected reward of the current state. The iteration will terminate once the expected values on grids are saturated. Later on, the robot will follow the path by choosing the grid with the maximum expected reward and reach the target eventually. **Algorithm 4-1** is the pseudo code showing how value iteration works. In this algorithm, $\pi(s)$ is the policy function that takes state s as input and

doi:10.6342/NTU201902426



(a) $k = 0$ (b) $k = 1$

(c) $k = 10$ (d) $k = 22$ (saturation)

Figure 4-3 Process of value iteration given positive reward $r_+ = 1$, negative reward $r_- = 1$, discount factor $\gamma = 0.9$, and transition probability $P = 0.8$.

returns an action a. $P(s'|s,a)$ indicates the probability of current state s changing to next state s' after applying action a; $R(s,a,s')$ is the reward function of giving sequential states as well as the action; $Q(s,a)$ can be viewed as the expected reward of forcing to perform action a on current state s, and $V_k(s)$ is the maximum expected reward on state s at the kth iteration. The iteration process is shown in Figure 4-3.

We take the concept of estimating the expected reward on every state while assuming the action probability being the same due to the robustness of our motion planner.

doi:10.6342/NTU201902426

Algorithm 4-1 Value iteration

```
1.  Inputs:  $\boldsymbol{S}$: a set of all possible robot states
2.           $\boldsymbol{A}$: a set of all possible robot actions
3.           $P$: state transition function
4.           $R$: reward function
5.           $\gamma$: discount factor
6.           $\theta$: threshold of convergence
7.  Initialize:  $k \leftarrow 0$
8.               $V_0(s)$ to arbitrary values
9.  while true:
10.     $k \leftarrow k + 1$
11.     for all $s \in \boldsymbol{S}$:
12.         for all $a \in \boldsymbol{A}$:
13.             $Q(s,a) \leftarrow E[r|s,a] + \gamma \sum_{s' \in \boldsymbol{S}} P(s'|s,a) \cdot [R(s,a,s') + \gamma V_{k-1}(s')]$
14.         $V_k(s) \leftarrow \max_{a} Q(s,a)$
15.     if $|V_k(s) - V_{k-1}(s)| < \theta, \ \forall s$:
16.         break while
17. for all $s \in \boldsymbol{S}$:
18.     $\pi(s) \leftarrow \underset{a \in \boldsymbol{A}}{argmax} \sum_{s'} P(s'|s,a) \cdot [R(s,a,s') + \gamma V_k(s')]$
19. return $\pi(s)$, $V_k(s)$
```

4.2 Methodology

In this section, the methodology of our task planning is discussed. The whole process of the system can be divided as: transforming perceptions into instructions, discretizing the environment, performing task planning through value iteration, executing tasks with motion planning. We will thus discuss these parts accordingly.

4.2.1 Transformation from Perceptions into Instructions

Inspired from the R-type instruction of Microprocessor without Interlocked Pipeline Stages (MIPS) computer structure, our instruction format is designed as Table 4-1:

doi:10.6342/NTU201902426

Table 4-1 Instruction structure

int32 **id**	The identification number of the instruction.
int32 **previous id**	The identification number of the previous instruction
float64 **γ**	Initial reward value
float64 **β**	Decay factor
bool **type**	Whether the instruction is from the human or robot
float64 **start time**	The time once the instruction is launched
int32 **duration**	The time for executing the function
string **source**	The person who launches the instruction
string **target**	The person that the instruction will affect to
int32 **status**	Human actions or sentiment
int32 **function**	Robot actions
int32 **destination**	The place where the instruction will be executed

Note that the previous id in the instruction format links the current instruction with the previous instruction if the task requires a series of instructions to complete. For example, if the care giver is too busy to check the status of elder Bob, he/she can assign the task *"check Bob status"* toward the robot. Thus, the robot will come up with two instructions, *"Check Bob"* and *"Report to the care giver"* and complete them one by one with the order according to the previous id. Besides, we assign an initial reward value and decay factor on every instruction to simulate the importance. Namely, as human may highly be unpleasant if there is no response after launching a task, we can design a monotonous decreasing function containing initial reward and decay factor such that the system is aimed to purchase the maximum reward. The decaying rule of the reward of instruction i is as Eq. (4-2), where t_i is the duration from launching to completing the instruction.

$$r_i(t) = \gamma_i {\beta_i}^{t_i},\ \ \gamma_i > 0,\ \ 1 > \beta_i > 0,\ \ t_i > 0 \quad (4\text{-}2)$$

For our visual perception, when the robot is unoccupied by human commands, it will wander around and observe household members. In this way, not only can the robot come

doi:10.6342/NTU201902426

up with recommendation such as *"play piano music for person who is reading"*, but also set reminder once it observes human actions different from the pre-set schedule. As for our verbal perception, thanks to the open-source STT system [52], we can extract key words from sentences and map to certain functions. On top of that, the robot can also check human sentiments with the help from VADER. On the other hand, as for our visual perceptions, the human actions and locations can provide clues for the robot to generate instructions to itself. For instance, it may recommend a piano music if the human is reading, or remind his/her schedule if the current action is not identical to the preset schedule. Those key words and visual perception results can generate suitable initial rewards which imply the priority of the instructions. In our case, the priority of the instructions is: *physical help*, *negative mood*, *neutral*, *positive mood*, and *self-generated commands from the robot*. Moreover, we design the decay factor in every instruction for the robot to not only purchase merely rewards, but also take the distance and processing time into consideration. Through the instruction formations, we can transfer the household scenario into an optimization problem with regard to the initial rewards and decay factors.

4.2.2 Problem Formulation for Task Planning

Next, in order to speed up computation time while planning tasks, discretizing the real world environment is needed. Figure 4-4 shows the testing environment for our system and the result of discretization. Note that our system can be generalized into different indoor scenarios, and this is simply one of them. The system first predefines locations served as nodes such as office, bedroom, charging place, alley, living room, dining room, places to welcome guests and interact with the care giver according to given semantic mapping system. Note that the semantic mapping system can be achieved

doi:10.6342/NTU201902426

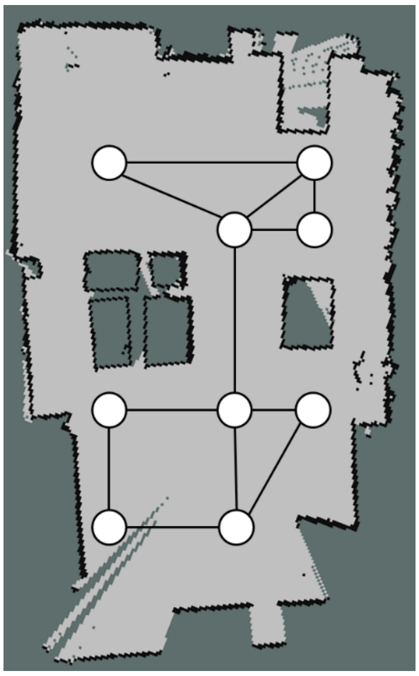


Figure 4-4 Household environment to test our method, gray map is the grid occupancy map and the discrete graph indicates the topological map.

through our previous work [74]. On top of that, it also generates edges in among through applying A* shortest path planning from one node to another, forming into a Euclidean graph. With the aid of instruction formation, and environment discretization, the robot is now able to deal with multiple human requests during task planning procedure.

To discuss the complexity of the planning problem, let's take a special case of our household scenario as shown in Figure 4-5: given a graph $G = (V, E)$ with N vertices $\{v_1, v_2, \dots, v_i, \dots, v_N\}$ and the same edge cost representing the discrete environment, there are instructions on each node containing the same initial reward value γ and decay factor β. Since moving to any node over twice will increase redundant time, the optimal solution will be the path that approaches every node exactly once. In other words, the maximum, or optimal, reward will be:

$$r_{OPT} = \sum_{i=1}^{N} \gamma \cdot \beta^{i \cdot c} \tag{4-3}$$

doi:10.6342/NTU201902426

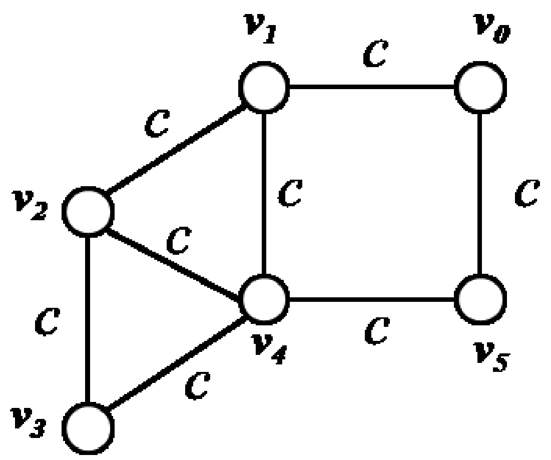


Figure 4-5 A special case of our modeling scenario

As a result, the decision problem of the optimization becomes *"Whether there exists a path with accumulative reward that equals to* r_{OPT}*?"* This statement is identical to the definition of well-known Hamiltonian path. It had been proven that the decision problem HAMILTONIAN PATH, known as "*Given a graph G, does G have a Hamiltonian path?*," is NP-complete [63]. Therefore, if HAMILTONIAN PATH problem is reducible to our scenario in polynomial time, then according to following theorem from, it can also be regarded as a NP-complete problem. It is because that Hamiltonian path is simply a special case in our scenario of assigning same initial reward and decay factor, it takes polynomial time to reduce any instances from HAMILTONIAN PATH to that from our problem. Consequently, according to ***Theorem 4-1*** from *Section 4.1.2*, the complexity of our problem can also be classified into NP-complete.

4.2.3 Algorithm for Task Planner

To come up with an algorithm, assuming the robot is navigating with its minimum velocity, we can divide edges by the multiplication of minimum robot velocity and time

doi:10.6342/NTU201902426

steps. That is to say, even though the instruction is a decay function related to time, we are able to transfer into spatial relation and calculate the expected reward in the indoor environment. Besides, one of the advantages of discretizing the environment is that it lowers the calculation of the system. That is, the task planner only needs to check the expected rewards on individual nodes instead of repeating calculating rewards in the overall continuous environment. Therefore, the hierarchical task motion planning is thus more efficient than simple motion planning with the help of environment discretization. Once launching instructions on a certain node, the expected reward in the special relation can be viewed as expanding ripple-like value centering at the semantic nodes with maximum value, namely the initial reward. Figure 4-6 shows the ripple-like decaying process of the expected reward on our discrete topological map.

After analyzing the accumulative expected rewards on the neighboring nodes, the task planner will choose the one with the highest value as action and pass toward the motion planner, generating a motion approaching to the next state. Nevertheless, it is somehow memory-consuming for the robot to store numerous nodes and edges, not to mention the high variant reward values and the quickly changing robot position. What's worse, the robot moves continuously in the reality. If implementing this discrete TAMP, the robot may move and stop repeatedly until reaching the instruction destination, which is unfortunately both time-consuming and unpleasant user-experience. These drawbacks may cause the robot to fail while processing instructions in practice even though the task planner come up with an adequate decision.

doi:10.6342/NTU201902426

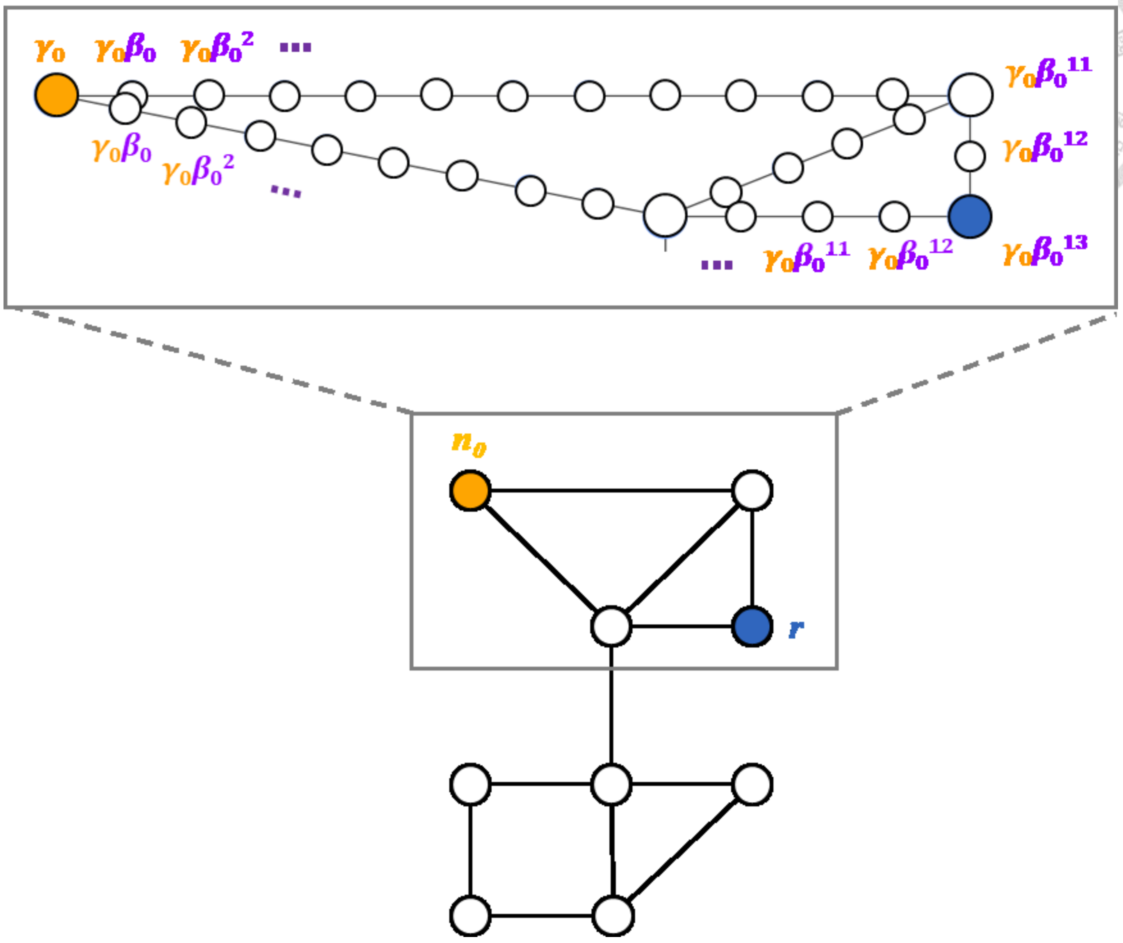
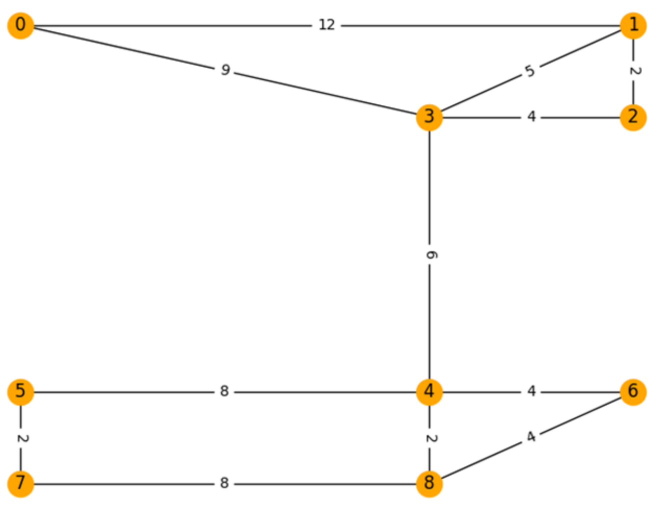


Figure 4-6 The expected reward estimation in the topological map

To overcome the problem, a modified robot state is proposed. Based on the topological map from Figure 4-4, the planner first calculate its adjacency matrix as well as the shortest matrix through Floyd-Warshall algorithm [63]. The robot location, or state, can thus be interpreted as the step distance between the neighboring nodes, as shown an illustrative example in Figure 4-7. Given the topological map $G = (V, E)$, the original method requires $O(|V| + \sum_{e \in E} w_e)$ to memorize the total possible states. With our conversion, the space complexity can be reduced to $O(|V|^2)$ and the robot state can be compressed into a $|V|$ dimensional integer vector. This especially brings an advantage as household space becomes larger and the distance between nodes increases.

doi:10.6342/NTU201902426



(a) Topological map $G = (V, E)$ with euclidean distance as weighted edges

	0	1	2	3	4	5	6	7	8
0	0	12	0	9	0	0	0	0	0
1	12	0	2	5	0	0	0	0	0
2	0	2	0	4	0	0	0	0	0
3	9	5	4	0	6	0	0	0	0
4	0	0	0	6	0	8	4	0	2
5	0	0	0	0	8	0	0	2	0
6	0	0	0	0	4	0	0	0	4
7	0	0	0	0	0	2	0	0	8
8	0	0	0	0	2	0	4	8	0

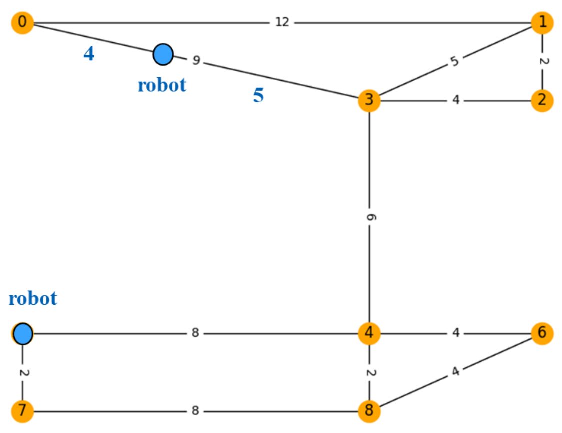
(b) Adjacency matrix of (a)

	0	1	2	3	4	5	6	7	8
0	0	12	13	9	15	23	19	25	17
1	12	0	2	5	11	19	15	21	13
2	13	2	0	4	10	18	14	20	12
3	9	5	4	0	6	14	10	16	8
4	15	11	10	6	0	8	4	10	2
5	23	19	18	14	8	0	12	2	10
6	19	15	14	10	4	12	0	12	4
7	25	21	20	16	10	2	12	0	8
8	17	13	12	8	2	10	4	8	0

(c) Shortest path matrix of (a)

0	1	2	3	4	5	6	7	8
4	0	0	5	0	0	0	0	0

0	1	2	3	4	5	6	7	8
0	0	0	0	8	0	0	2	0



(d) Examples for robot state representation based on (a), (b), and (c)

Figure 4-7 Robot state representation in the topological map

doi:10.6342/NTU201902426

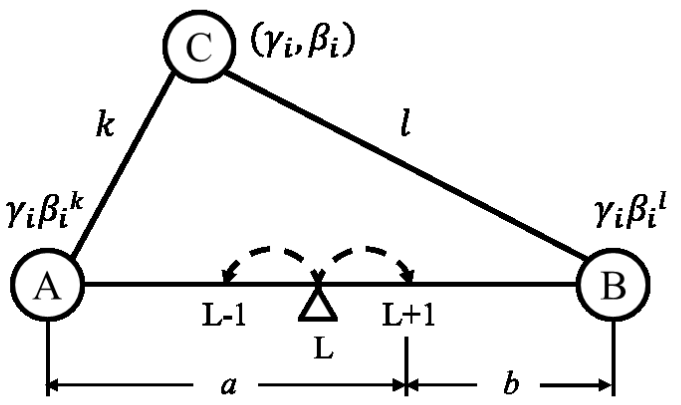


Figure 4-8 An example to explain the task planner algorithm

As for the state transition of the robot, there are three cases. First of all, if the robot is on the edge, given the next neighboring node i, the ith element of the state vector plus one while other non-zero elements minus one. Secondly, if one of the non-zero elements i turns to zero during the state transition, that means that the robot reach node i. Thus, the algorithm will copy the ith raw of the adjacency matrix as the state vector. Lastly, if the robot leaves from node i to nod j, indicating that node i changes to the neighboring node and the ith element of the state vector becoming one, then the jth element minus one while other elements turn to 0.

Through the state transition process, the system can not only record the location of the robot, but also generate reasonable actions. In other words, the task planner simply takes a set of instructions as inputs and returns next neighboring nodes as the direction for the robot to follow. As a consequence, given M instructions $I = \{(\gamma_i, \beta_i) | \forall i < M\}$ at node C and the robot location L with neighboring node A and B, as Figure 4-8 shows, the planner will calculate the accumulative expected rewards on $\mathrm{L}+1$ and $\mathrm{L}-1$, known as candidate steps, and pick one with the higher valve as the next step. For every instruction such as (γ_i, β_i) in node C, the calculation can be divided into two steps. First the algorithm calculates the expected reward $\gamma_i {\beta_i}^t$ on neighboring nodes. Thanks to the

doi:10.6342/NTU201902426

shortest path matrix, t can be obtained by simply extracting the (C, A) element from the matrix as shortest path without redundant shortest path calculation. Furthermore, the accumulative reward values on nodes will be memorized. When new instructions are launched dynamically, the algorithm simply adds those new expected rewards on the existing ones, making the robot handle dynamical multiple social tasks. Next, the expected reward of instruction i on candidate steps is calculated respectively. Take $\mathrm{L}+1$ for example, the algorithm finds the shortest path from the instruction to candidate, making the expected reward $r_{i,L+1} = \gamma_i {\beta_i}^{\min(k+a,l+b)}$. Note that there may exist a case that the expected reward may be $r_{i,L+1} = \gamma_i {\beta_i}^{l-b}$, which is the case that edge l does not exist. Fortunately, this means that $k + a = l - b < l + b$ which still makes the formula $\min(k + a, l + b)$ valid. Through the iteration, the accumulative expected reward on $\mathrm{L}+1$ is as Eq. (4-4), where the accumulative expected reward on $\mathrm{L}-1$ can be calculated in an identical way. Thus, the task planner takes a series of instructions and the current state of the robot as inputs, and returns the next neighbor node for the motion planner by Eq. (4-5). **Algorithm 4-2** is the detailed pseudo code of the proposed task planner. Note that though the time complexity is $O(\|V\|^2 M)$, the following experiments show that the total processing time of the task and motion planning still maintains its high efficiency up to 200 instructions.

$$r_{\mathrm{L}+1} = \sum_{i \in I} \gamma_i {\beta_i}^{t_i}, t_i = min(A[i][A] + a, A[i][B] + b) \tag{4-4}$$

$$\boldsymbol{\hat{n}} = \underset{\boldsymbol{n} \in \boldsymbol{N}}{argmax} \left(\sum_{i=0}^{M-1} \gamma_i {\beta_i}^{\underset{n \in N}{min}(A[m_i][n] + v_l[n])} \right) \tag{4-5}$$

doi:10.6342/NTU201902426

Algorithm 4-2 Value Iteration Task Planning

```
1.  Input: $M$ instructions on $m$ nodes $\boldsymbol{I} = \{(\gamma_i, \beta_i, m_i) | \forall i < M, i \in \mathbb{N}\}$
2.          Robot at current location L with state vector $\boldsymbol{s_t} \in \mathbb{N}^{1\times|V|}$
3.          Current neighboring nodes $\boldsymbol{N} = \{n_0, n_1, \dots, n_{N-1}\}$
4.          Adjacency matrix $A \in \mathbb{R}^{|V|\times|V|}$
5.  Define: Candidate state set $\boldsymbol{C} = \{\boldsymbol{c_0}, \boldsymbol{c_1}, \dots, \boldsymbol{c_{N-1}}\}$
6.  for $n \in \boldsymbol{N}$:
7.      $\boldsymbol{c_n} = state\_transition(\boldsymbol{s_t}, n)$
8.      for $I \in \boldsymbol{I}$:
9.          $path(m_i, I) = \min_{n' \in \boldsymbol{N}}(A[m_i][n'] + \boldsymbol{c_n}[n'])$
10.         $r_i = \gamma_i {\beta_i}^{path(m_i, I)}$
11.     $r_n = \sum_{i=0}^{M-1} r_i$
12. $\hat{\boldsymbol{n}} = \underset{n \in \boldsymbol{N}}{argmax}(r_n)$
13. $\boldsymbol{s_{t+1}} = \boldsymbol{c_{\hat{n}}}$
```

4.2.4 Correctness of the Proposed Task Planning Algorithm

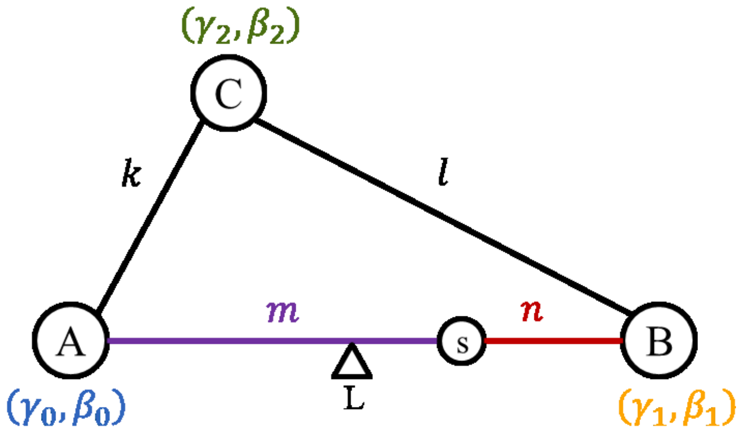


Figure 4-9 The scenario of proving the optimality

To explain the correctness of the proposed method, let's start with the following scenario shown in Figure 4-9: Given three nodes A, B, C and instructions with tuples of initial rewards and decay factors $(\gamma_0, \beta_0), (\gamma_1, \beta_1), (\gamma_2, \beta_2)$ respectively, the goal is to prove that the accumulative reward on the edge will always be lower than that of the

doi:10.6342/NTU201902426

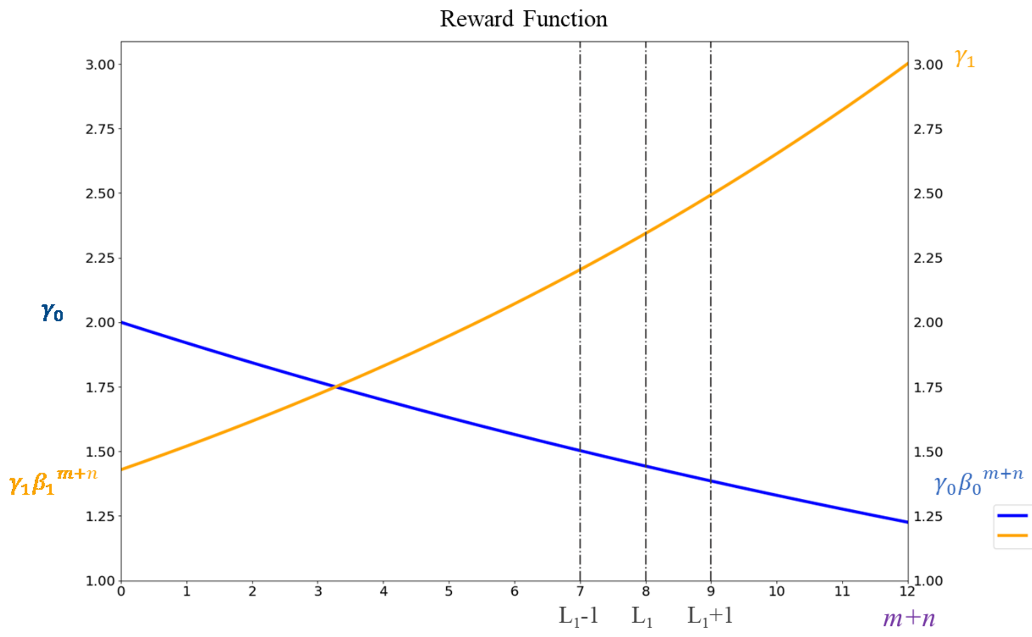


Figure 4-10 The expanding rewards of two instructions along $\overline{AB}$

neighboring nodes. In other words, assuming the robot location L is currently at the edge $\overline{AB}$. k, l, m, n represent the edge length. Node s is the node at the edge such that $k + m = l + n$. The following description will show that L turns out to move toward either node A or node B instead of staying at the edge in between.

To begin with, it is trivial that given a single instruction, the robot will move toward the node with shortest path according the graph. On top of that, let's consider two arbitrary instructions on adjacency node with robot initially staying at the edge in between, which can be viewed as the case when $\gamma_2 = 0$ in Figure 4-9. The expanding reward values are like Figure 4-10, as defining the one-dimensional coordinate starting from node A and ending at node B for better explanation. Before discussing the first case, we define the reward difference function of instruction i between point a and point b as:

$$\Delta r_i(a, b) = r_i(a) - r_i(b) \tag{4-6}$$

doi:10.6342/NTU201902426

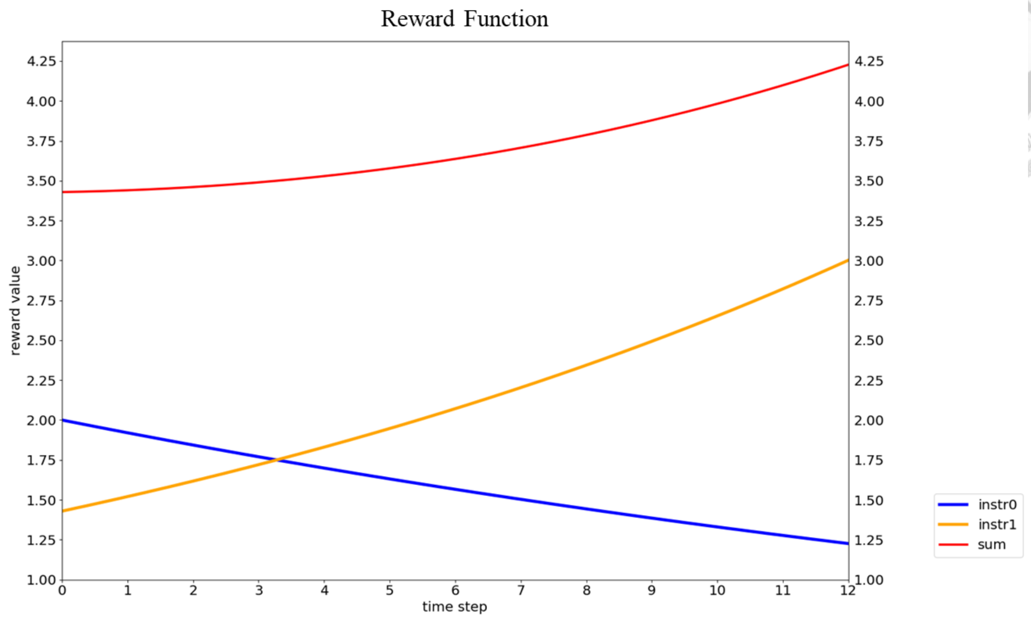


Figure 4-11 Overall reward distribution of two instructions

We will show that the robot will move toward nodes instead of getting stuck on the edge by contradiction. Assuming that the robot starts at $\mathrm{L}_1 - 1$ and stays at L_1 on the $\overline{AB}$ eventually, which means that L_1 has the highest reward, the reward difference has the relationship: $\Delta r_0(\mathrm{L}_1, \mathrm{L}_1 - 1) + \Delta r_1(\mathrm{L}_1, \mathrm{L}_1 - 1) > 0$, with $\Delta r_0(\mathrm{L}_1, \mathrm{L}_1 - 1) < 0$.

However, since the reward functions $r_i(x) = \gamma_i \cdot \beta_i^{|source(i) - x|}$ ($\beta_i < 1$) are monotonous increasing/decreasing and asymptotic to 0 as robot location L approaches to negative/positive infinitive, the total reward difference between $\mathrm{L}_1 + 1$ and L_1 is larger than that between L_1 and $\mathrm{L}_1 - 1$. That is to say:

$$0 > \Delta r_0(\mathrm{L}_1 + 1, \mathrm{L}_1) > \Delta r_0(\mathrm{L}_1, \mathrm{L}_1 - 1) \tag{4-7}$$

$$\Delta r_1(\mathrm{L}_1 + 1, \mathrm{L}_1) > \Delta r_1(\mathrm{L}_1, \mathrm{L}_1 - 1) > 0 \tag{4-8}$$

$$\Delta r_0(\mathrm{L}_1 + 1, \mathrm{L}_1) + \Delta r_1(\mathrm{L}_1 + 1, \mathrm{L}_1) > \Delta r_0(\mathrm{L}_1, \mathrm{L}_1 - 1) + \Delta r_1(\mathrm{L}_1, \mathrm{L}_1 - 1) > 0 \tag{4-9}$$

Therefore, the above equations reveal that the accumulative reward in $\mathrm{L}_1 + 1$ is larger than L_1, which is paradoxical to the assumption. In other words, the robot will at the end

doi:10.6342/NTU201902426

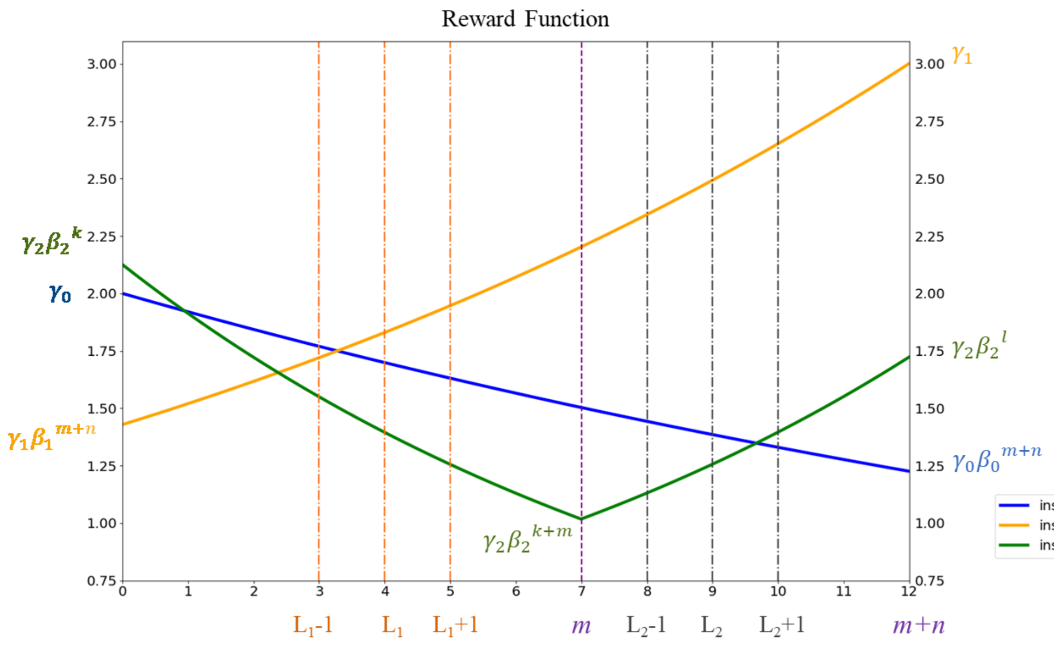


Figure 4-12 The expanding rewards along $\overline{AB}$

reach the node instead of getting stuck at the edge. Figure 4-11 illustrates the overall reward distribution.

Next, the case of three instructions will be discussed. According to the initial rewards and the decay factors, the expanding rewards of individual instructions along $\overline{AB}$ are as Figure 4-12. Here the discussion can be separated into three cases: $0 < \mathrm{L} < m$, $\mathrm{L} = m$, and $m < \mathrm{L} < m + n$.

First of all, given the case $0 < \mathrm{L} < m$, supposed that L will eventually stays at L_1, meaning that "L_1 has the highest accumulative reward along $\overline{AB}$." Therefore, if the robot starts at $\mathrm{L}_1 + 1$, the behavior of moving toward L_1 indicates that the reward differences $\Delta r_0(\mathrm{L}_1, \mathrm{L}_1 + 1) + \Delta r_2(\mathrm{L}_1, \mathrm{L}_1 + 1) + \Delta r_1(\mathrm{L}_1, \mathrm{L}_1 + 1) > 0$, with $\Delta r_1(\mathrm{L}_1, \mathrm{L}_1 + 1) < 0$. Nevertheless, similar to the two-instruction case with reward function $r_i(x)=\gamma_i \cdot \beta_i{}^{|source(i)-x|}(\beta_i < 1)$ being monotonous and asymptotic to 0 at the infinity points, the total reward differences between $\mathrm{L}_1 - 1$ and L_1 is larger than that of L_1 and $\mathrm{L}_1 + 1$. Namely:

doi:10.6342/NTU201902426

$$\Delta r_0(\mathrm{L}_1-1,\mathrm{L}_1) > \Delta r_0(\mathrm{L}_1,\mathrm{L}_1+1) > 0 \tag{4-10}$$

$$0 > \Delta r_1(\mathrm{L}_1-1,\mathrm{L}_1) > \Delta r_1(\mathrm{L}_1,\mathrm{L}_1+1) \tag{4-11}$$

$$\Delta r_2(\mathrm{L}_1-1,\mathrm{L}_1) > \Delta r_2(\mathrm{L}_1,\mathrm{L}_1+1) > 0 \tag{4-12}$$

$$\begin{aligned}&\Delta r_0(\mathrm{L}_1-1,\mathrm{L}_1)+\Delta r_1(\mathrm{L}_1-1,\mathrm{L}_1)+\Delta r_2(\mathrm{L}_1-1,\mathrm{L}_1)\\ &\quad > \Delta r_0(\mathrm{L}_1,\mathrm{L}_1+1)+\Delta r_1(\mathrm{L}_1,\mathrm{L}_1+1)+\Delta r_2(\mathrm{L}_1,\mathrm{L}_1+1) > 0\end{aligned} \tag{4-13}$$

As a result, the larger accumulative reward pulls the robot to move from L_1 to L_1-1, and eventually reaches node A, which contradicts to the assumption "L_1 has the highest accumulative reward along $\overline{AB}$." This also explain the case when $m < \mathrm{L} < m+n$.

As for the $\mathrm{L}=m$ case, the question can be interpreted as "whether the robot will eventually stay at m." If not, then it can be transformed into the above two cases. Supposed L starts at $m-1$, it will move to m according to the scenario, meaning that:

$$\Delta r_0(m-1,m) < 0 \tag{4-14}$$

$$\Delta r_1(m-1,m) > 0 \tag{4-15}$$

$$\Delta r_2(m-1,m) < 0 \tag{4-16}$$

$$\Delta r_0(m-1,m)+\Delta r_1(m-1,m)+\Delta r_2(m-1,m) > 0 \tag{4-17}$$

Nevertheless, as $k+m=l+n$, $\Delta r_2(m,m+1)$ becomes positive and $\Delta r_1(m,m+1)$ increases, the total reward difference, shown from Eq. (4-18) to (4-21), gets higher. Thus, m is not the highest reward and the robot will move to node B.

$$\Delta r_0(m-1,m) < \Delta r_0(m,m+1) < 0 \tag{4-18}$$

$$\Delta r_1(m,m+1) > \Delta r_1(m-1,m) > 0 \tag{4-19}$$

doi:10.6342/NTU201902426

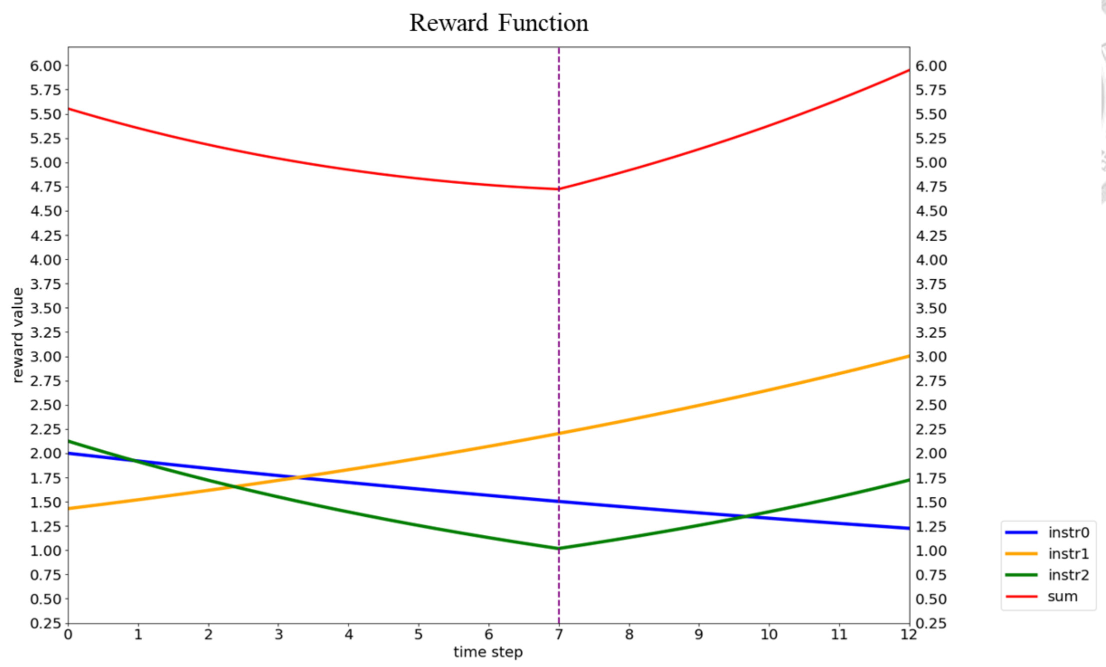


Figure 4-13 The accumulative reward function along $\overline{AB}$

$$\Delta r_2(m-1,m) < 0 < \Delta r_2(m,m+1), |\Delta r_2(m-1,m)| = |\Delta r_2(m,m+1)| \quad (4\text{-}20)$$

$$\begin{aligned}&\Delta r_0(m,m+1) + \Delta r_1(m,m+1) + \Delta r_2(m,m+1)\\ &\qquad > \Delta r_0(m-1,m) + \Delta r_1(m-1,m) + \Delta r_2(m-1,m) > 0\end{aligned} \quad (4\text{-}21)$$

From the description above, one can find out that the highest reward occurs at instruction nodes instead of edges. As a consequence, the correctness of the proposed method can be explained. To give a brief summary, the robot can reach toward human and complete instructions with the proposed value iteration algorithm. The red curve in Figure 4-13 is the accumulative reward function along $\overline{AB}$ from these three instructions given the illustrative example.

Besides, while the expected reward expands in a ripple-like way in the discretized environment, it will not decay with time. The reason is to prevent starving from the users. In other words, if the ripple-like expected reward decays with time, the previous instructions that haven't be done may even be postponed since the newly-add instructions

doi:10.6342/NTU201902426

always contain higher reward values. In this case, those previously launched instructions with medium priorities will not be done until newly-added instructions with lower priorities are completed, leading to unpleasant user experience. Thus, we consider the reward expansion only once such that those previous instructions can still be executed.

4.2.5 Integration of Proposed TAMP

The overall TAMP system can be complete through merging the task planning algorithm and the motion planner. Aside from dynamic window approach (DWA) provided from *Section 4.1.1*, we propose another solution for real-time obstacle avoidance in the motion planner that utilizes long short-term memory (LSTM). The motion planner takes the previous paths as well as surrounding obstacles as inputs and predict a potential collision-free trajectory for the mobile robot. In addition, the proposed LSTM model requires relative small GPU resource that can be implemented on embedded system. This real-time obstacle avoidance method has been proposed in [75].

Recalling the instruction cycle in the computer architecture, an instruction basically goes through four processes: *fetch, decode, execute, write back*. Inspired from this cycle in the central processing unit (CPU), Figure 4-14 shows the flow chart of the proposed integration. The system receives the visual and verbal perception results, transform into instructions, and write into the task buffer. Next, the task planner fetches the instruction, decodes into individual reward, decay factor, and function that affects the target. After the decision, or the next neighboring node to move, is passed to the motion planner, the motion planner will execute the result by generating a collision-free trajectory and navigating toward the destination. Finally, the motion planner will send the message whether the instruction is completed or not and write back to the task buffer. Through this process, the system is able to generate suitable decisions and organize the instruction set.

doi:10.6342/NTU201902426

Therefore, the TAMP serves as a bridge that links the theoretical algorithm into practical applications.

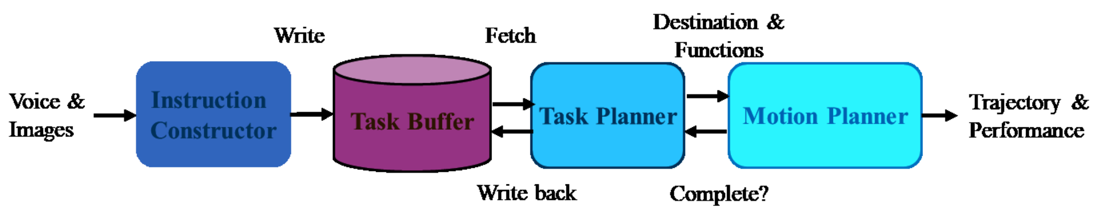


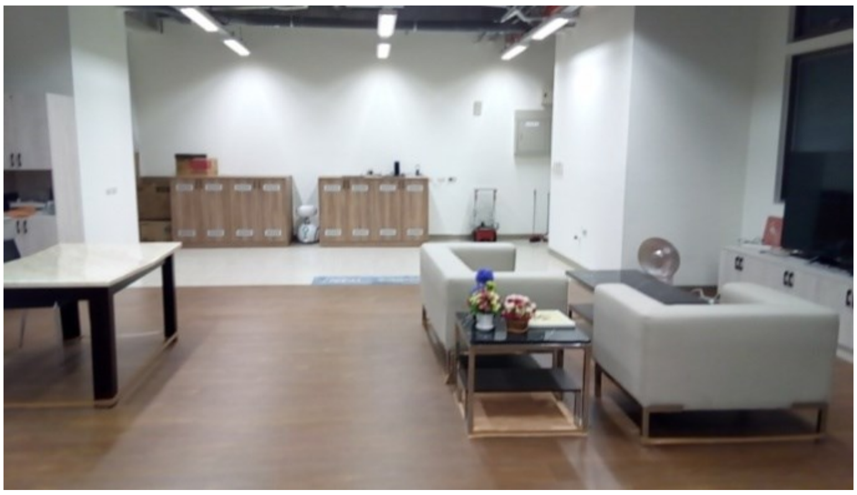
Figure 4-14 The integration flow chart of the proposed TAMP

doi:10.6342/NTU201902426

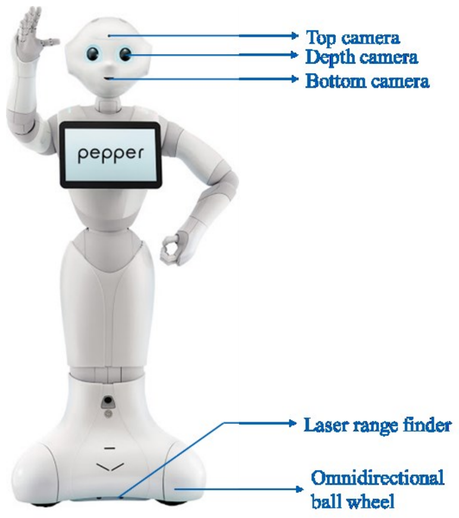
Chapter 5 Experimental Results

In this chapter, we conduct the experiments and discuss the results in both perceptions as well as decision making. The following sections include environment setup, testing dataset based on the Pepper robot, experiments for perceptions, and experiments for dynamic multi-task social navigation.

5.1 Environment Setup



(a) Indoor household environment



(b) Pepper robot and sensors

Figure 5-1 Environment hardware setup

doi:10.6342/NTU201902426

The experiment is conducted in an indoor household environment of with width and depth being 6.8 and 11.8 meters, respectively, as shown in Figure 5-1(a). The built map can be seen in previous Figure 4-4. This environment contains office, bedroom, charging place, living room, and dining room. The overall system is running through the desktop with Intel® Core™ i7-8550U (1.80 GHz x 8) CPU, and NVIDIA-1080Ti GPU. The system is built under ROS (Robot Operating System) with Kinetic attribute and the proposed architecture is programmed by Python 2.7. The robot on which we implement our system is the social robot, Pepper, as shown in Figure 5-1 (b), embedded with Quad Core as CPU. Also, it contains top camera, bottom camera, and depth camera, all with resolution 320×480 and 4 frames per second. As for its mobility, Pepper contains three omnidirectional ball wheels. On top of that, we utilize the laser range finder on Pepper for SLAM which has merely 45 laser beams within 180 degrees.

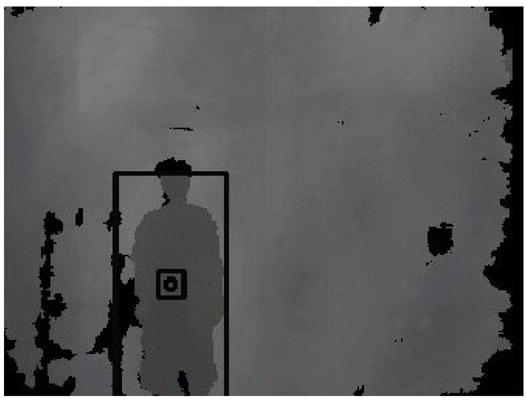
5.2 Experiments: Visual Perception

The most critical part of verbal perceptions in our system lies in the speech to text algorithm. Since we leverage the open-source packages which perform robust STT and sentiment recognition, our experiments mostly focus on evaluations of visual perceptions. With the combination of deep learning techniques as well as heuristic algorithms, our proposed approach produces satisfactory results in real time. Furthermore, we collect a visual dataset with labeled frames so as to evaluate the visual perception abilities. The following sections includes the experiments in human localization, human identification, and action detection.

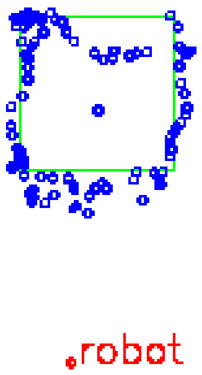
doi:10.6342/NTU201902426

5.2.1 Human Localization Evaluation

To evaluate the human localization ability, we draw a square on the ground with 1.2 m length in each edge in front of Pepper with the front edge being at 1.5 m distance. While the person walks along the edge of the square, the robot localizes him and records the position relative to the robot coordinate. Then, we calculate the Euclidean distance error between the square and the human path. The result shows that the average error is simply 0.28 m, with the maximum error being 0.58 m, which is acceptable for the robot to recognize the semantic location of the human. Figure 5-2 shows the aforementioned visualization of the experimental.



(a) Depth bounding box of human

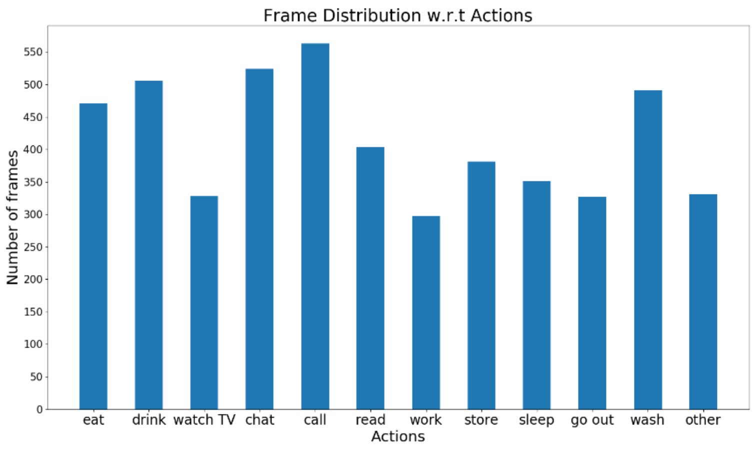


(b) Result of human localization

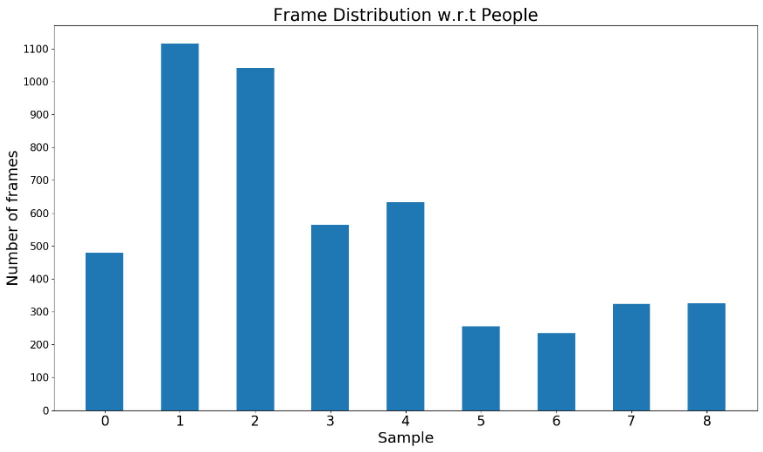
Figure 5-2 The visualization of human localization. (a) shows the depth image and the human bounding box. (b) is the result of localization, where blue points are the human walking path and green square is the ground truth trajectory.

doi:10.6342/NTU201902426

5.2.2 Pepper Image Testing Dataset



(a) Distribution of frames relative to actions



(b) Distribution of frames relative to people

Figure 5-3 The number distribution of our testing dataset

Before discussing the experiments on human identification and action detection, we first introduce the self-collected testing dataset from the stitching image of Pepper's camera. We record 13 actions, including *eating, drinking, watching TV, chatting, calling, reading, working, storing, sleeping, going out,* and *washing*. The dataset consists of 9 people involved in these actions, obtaining 103 video streams. After that, we separate every video into frames and label them one by one through graphic user interface, where each frame is assigned a single action label. Noted that there may exist frames that cannot

doi:10.6342/NTU201902426

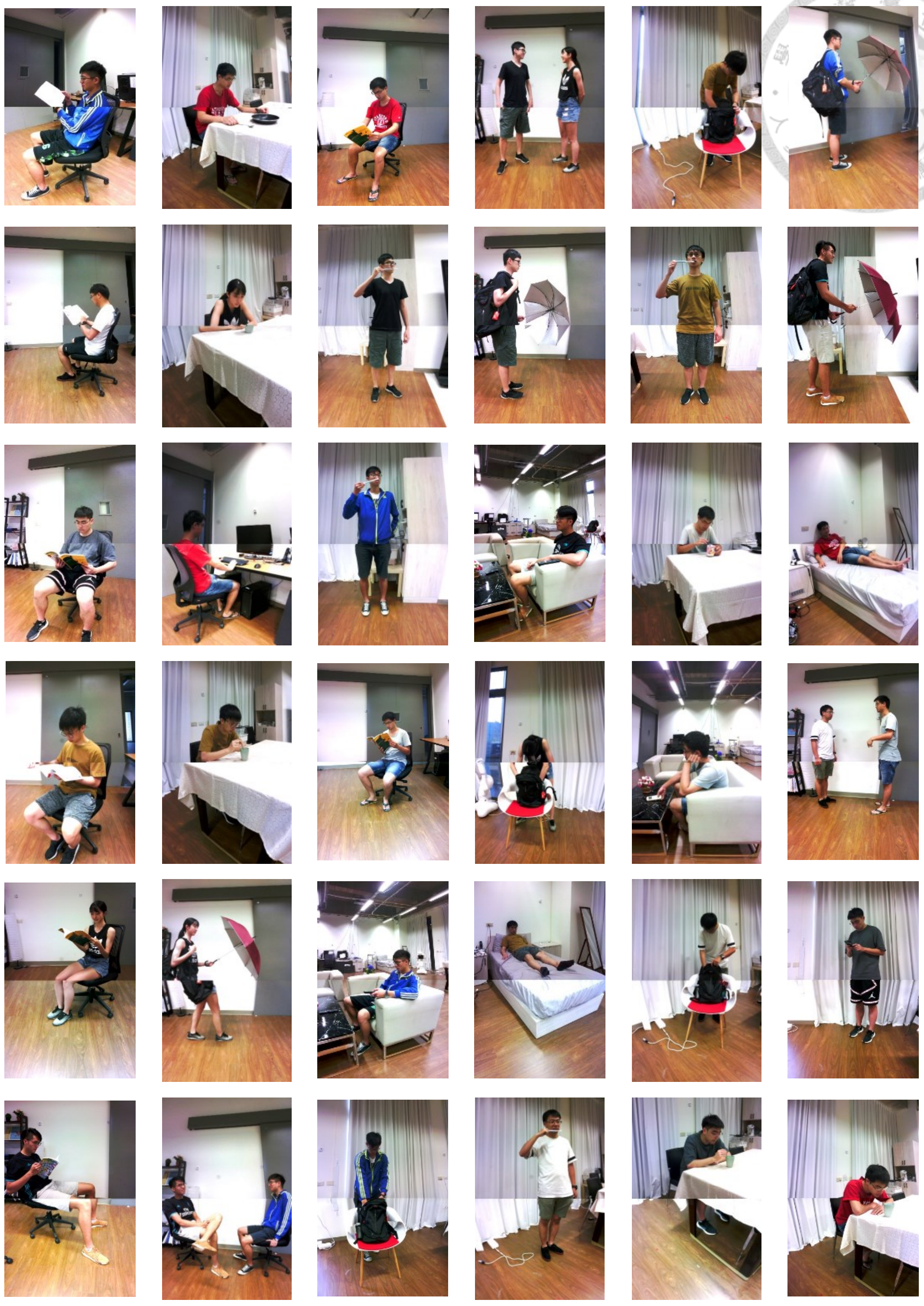


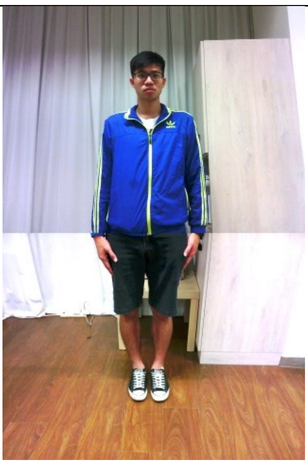
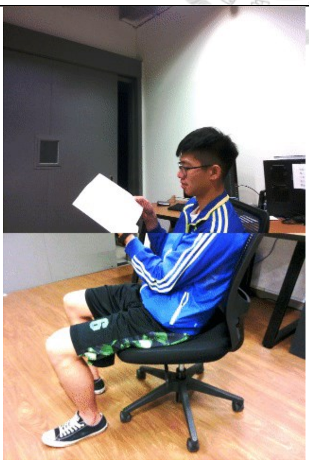
Figure 5-4 Some example images inside the Pepper Image Testing Dataset.

be classified into the above 13 actions, and those frames are labeled as others. The total number of frames after processing is 4974, where the distribution of actions and people are shown in Figure 5-3. Figure 5-4 are some exemplar frames in our testing dataset.

doi:10.6342/NTU201902426

5.2.3 Human Identification Evaluation

Table 5-1 An illustrative example of color metric

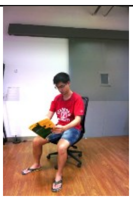
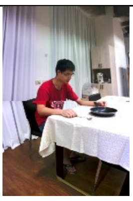
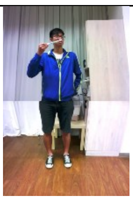
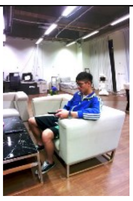
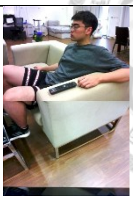
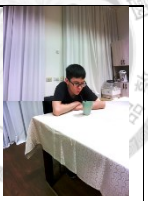
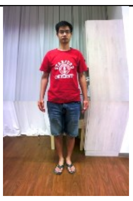
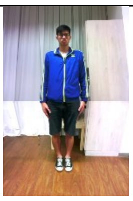
		(a) Query image			(b) Test image		
Image							
Shirt color		🟦	🟩	🟦	🟦	⬜	🟦
Value	R	215	32	29	253	172	87
	G	250	77	74	255	190	100
	B	220	238	219	199	168	142
Color metric		189.43					

Before the experiment, here is an illustrative example of how **Algorithm 3-1** works. Given a query image as shown in Table 5-1(a), the robot captures the shirt color at the joints and stores in the human database with the name of the person. Later on, while the robot navigates in the environment and sees a person as shown in Table 5-1(b), it will again capture the shirt color at the joints and compare the existing colors in the database through the approximated color metrics (Eq. (3-6) to Eq. (3-10)).

Our human identification approach can be evaluated by first giving the query images and then measuring the approximated color metrics between the query images and the testing images. The query images can be obtained while Pepper greets toward the person, and the test images come from the Pepper Image Testing Dataset. Table 5-2 shows the

doi:10.6342/NTU201902426

Table 5-2 The approximated color metric of query images and testing images

Query \ Test							
A		124.4	172.4	422.4	387.2	301.1	300.2
B		387.8	417.5	34.5	186.1	295.1	390.7
Results		A	A	B	B	None	None

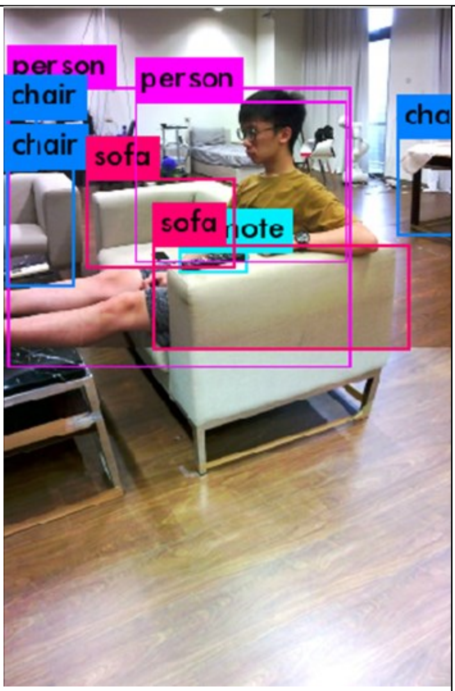
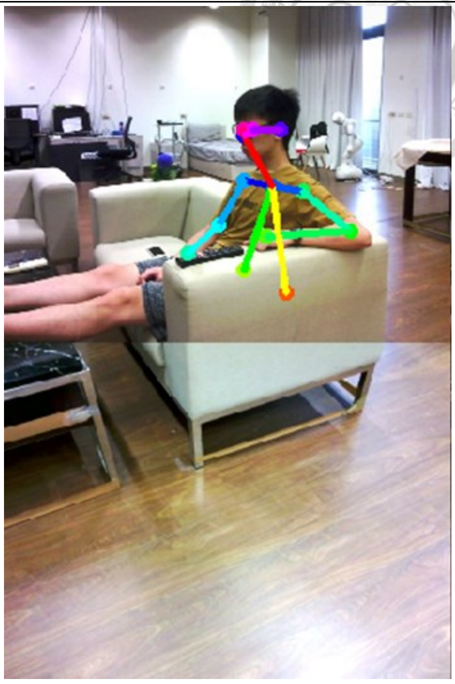
results between query images and test images. As a consequence, we can define the threshold as 200 such that the robot can classify different people from the dataset.

5.2.4 Framewise Hierarchical Human Action Detection Evaluation

The proposed framewise hierarchical human action detection can also be evaluated from the Pepper Image Testing Dataset. Given a series of labeled images separated from the videos, the program outputs predictions on the basis of YOLO [20][21][22] and OpenPose [46]-[49]. Table 5-3 and Table 5-4 show the visualization of the action detection process. In Table 5-3, given the currently observed frame, YOLO and OpenPose detect human and objects as well as and skeleton, respectively. After that, the algorithm captures the objects which near hands are sofa and remote, and which under eye sight are chairs, sofa, and remote. These bring clues that affect the action probabilities, namely, $p(action|\boldsymbol{O_h})$ and $p(action|\boldsymbol{O_e})$, and the algorithm finally classifies the human action with the maximum probability $p(action)$. In this case, the action is watching TV.

doi:10.6342/NTU201902426

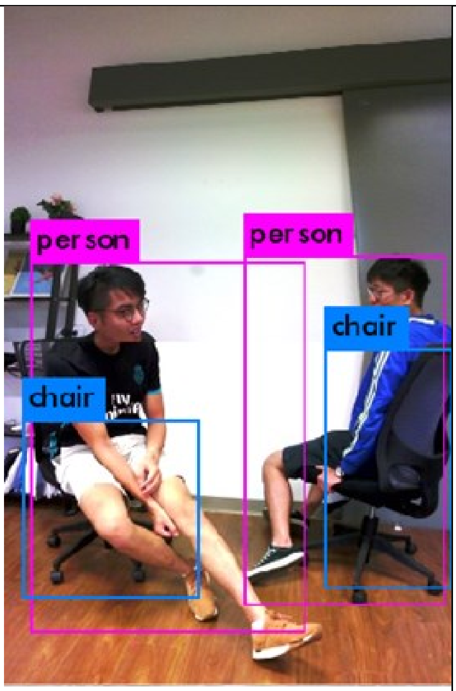
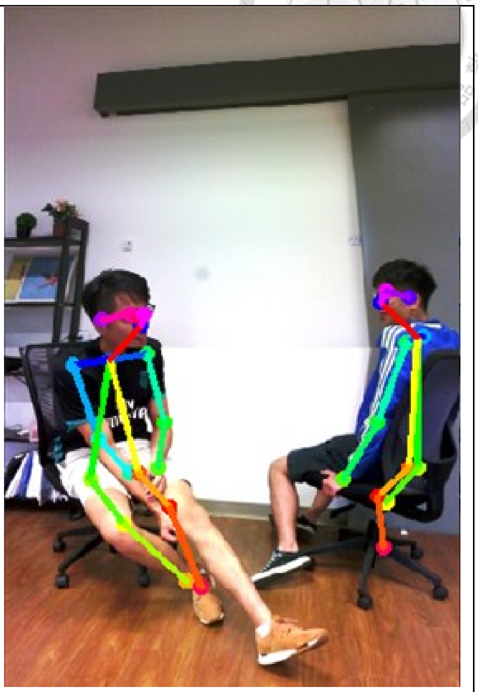
Table 5-3 The visualization of action detection process (1)

Image												
	YOLO object detection						OpenPose skeleton detection					
$\boldsymbol{O_h}$	sofa, remote											
$\boldsymbol{O_e}$	chair, chair, sofa, remote											
	eat	drink	TV	chat	call	read	work	store	sleep	leave	wash	other
$p(action\|\boldsymbol{O_h})$	0	0	0.7	0.06	0.06	0.06	0.06	0	0.06	0	0	0
$p(action\|\boldsymbol{O_e})$	0	0	1	0	0	0	0	0	0	0	0	0
$p(action)$	0	0	0.85	0.03	0.03	0.03	0.03	0	0.03	0	0	0
result	Watch TV											

As shown in the following Table 5-4, after knowing the target through human identification, Pepper is able to perceive that the target is looking at human with the help of YOLO. Thus, through the framewise hierarchical action detection, the robot can easily find out that the target is chatting with another person.

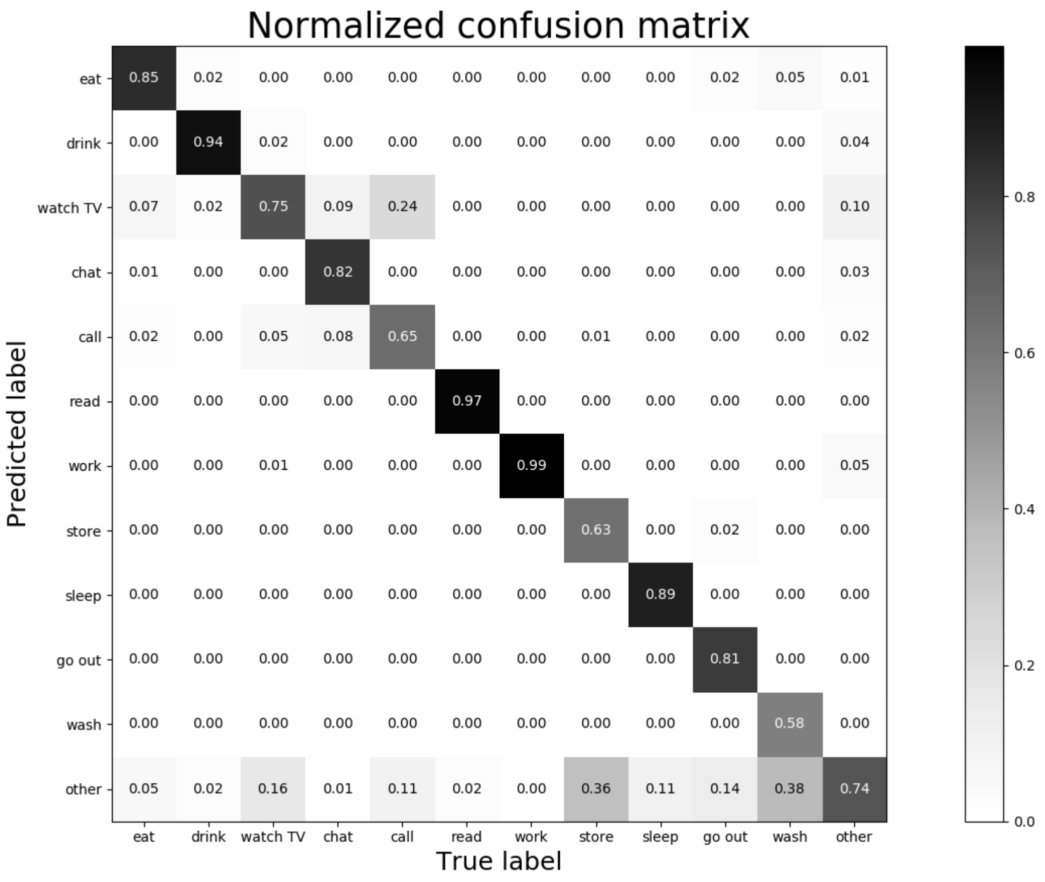
doi:10.6342/NTU201902426

Table 5-4 The visualization of action detection process (2)

Image												
	YOLO object detection						OpenPose skeleton detection					
$\boldsymbol{O_h}$	chair											
$\boldsymbol{O_e}$	person											
	eat	drink	TV	chat	call	read	work	store	sleep	leave	wash	other
$p(action\|\boldsymbol{O_h})$	0	0	0.17	0.17	0.17	0.17	0.17	0	0.17	0	0	0
$p(action\|\boldsymbol{O_e})$	0	0	0	1	0	0	0	0	0	0	0	0
$p(action)$	0	0	0.076	0.62	0.076	0.076	0.076	0	0.076	0	0	0
result	Chat											

The accuracy of the action detection can be evaluated through confusion matrix, as shown in Figure 5-5. Due to the fact that objects such as bowls, cups, books, TV monitor and keyboards are large enough for YOLO to come up with true positive detection robustly, actions like *eat*, *drink*, *read*, and *work* that are related to those objects have higher accuracy over 0.8. On the contrary, since the resolution of the camera on Pepper is too low, objects such as remote and tooth brush are too small to be captured through YOLO. Consequently, the accuracy of action like *wash* is much lower among other actions. Nevertheless, our system can classify those unclear actions to *others*, and thus

doi:10.6342/NTU201902426



Predicted \ True	eat	drink	watch TV	chat	call	read	work	store	sleep	go out	wash	other
eat	0.85	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.05	0.01
drink	0.00	0.94	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04
watch TV	0.07	0.02	0.75	0.09	0.24	0.00	0.00	0.00	0.00	0.00	0.00	0.10
chat	0.01	0.00	0.00	0.82	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03
call	0.02	0.00	0.05	0.08	0.65	0.00	0.00	0.01	0.00	0.00	0.00	0.02
read	0.00	0.00	0.00	0.00	0.00	0.97	0.00	0.00	0.00	0.00	0.00	0.00
work	0.00	0.00	0.01	0.00	0.00	0.00	0.99	0.00	0.00	0.00	0.00	0.05
store	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.63	0.00	0.02	0.00	0.00
sleep	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.89	0.00	0.00	0.00
go out	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.81	0.00	0.00
wash	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.58	0.00
other	0.05	0.02	0.16	0.01	0.11	0.02	0.00	0.36	0.11	0.14	0.38	0.74

Figure 5-5 Confusion matrix of proposed action recognition algorithm

the robot can still detect whether the person is sitting, standing, or lying even under the condition that the object detection is not precise, showing satisfactory robustness of our proposed framewise hierarchical action detection framework.

While taking advantage from both deep learning methods as well as heuristic algorithms, and allocating computational resource, the robot is able to perform real-time visual perception on answering *"where the person is?"*, *"who is the person?"*, and *"what is the person doing?"*. Through the experiments, we demonstrate that our proposed visual perception system can generate adequate results, including human localization, human identification, and framewise action detection, with both efficiency and robustness.

doi:10.6342/NTU201902426

5.3 Experiments: Dynamic Multi-Task Social Navigation

In our decision-making system, we first analyze the proposed system in the aspect of both efficiency and optimality under simulation scenarios, in comparison to other classic planners including *First Come First Serve (FCFS)*, *Randomize (Rand)*, *Priority First (PF)*, and *Shortest Time First (SF)*. *FCFS* simply considers the launching time of instructions, or namely the *id* of the instructions in Table 4-1; *Rand* picks instructions randomly; *PF* always does the instruction with the highest priority, which can be obtained from initial reward γ in Table 4-1; on the other hand, *SF* checks the summation of navigation time and instruction *duration* and processes the one with minimum time. Furthermore, aside from the simulation, the real world applications are also evaluated.

5.3.1 Optimality Comparison of Task Planner Algorithm

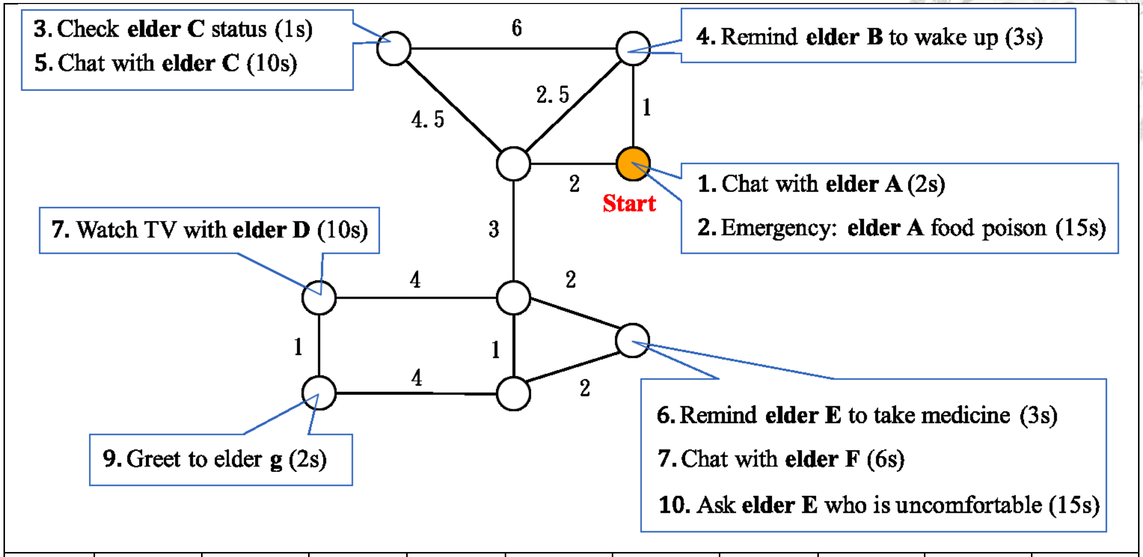
As mentioned in *Section 4.2.1*, the priority sequence of the instructions in a descendent order are: *physical help*, *negative mood*, *neutral mood*, *positive mood*, and *self-generated commands from the robot*. To quantify those priorities, we suggest using descendent Fibonacci sequence (FS) $\boldsymbol{\gamma_{FS}}=\{8, 5, 3, 2, 1\}$ subject to the priority. As for the decay factor β as mentioned before, we simply choose $\boldsymbol{\beta}=\{0.98, 0.96, 0.94, 0.92, 0.9\}$ with regard to the priority. Table 5-5 shows the relations between the priority and the initial reward γ as well as decay factor β. Thus, the accumulative reward of M instructions will be $r = \sum_{i=1}^{M} \gamma_i {\beta_i}^{t_i}$, where t_i indicates the time duration of instruction

Table 5-5 The table of priority quantification

Priority	*physical*	*negative*	*neutral*	*positive*	*self-generated*
Initial reward γ	8	5	3	2	1
Decay factor β	0.98	0.96	0.94	0.92	0.9

doi:10.6342/NTU201902426

Table 5-6 The example task scenario of our simulation


3. Check **elder C** status (1s)
5. Chat with **elder C** (10s)
6
4. Remind **elder B** to wake up (3s)
2. 5
4. 5
1
2
Start
1. Chat with **elder A** (2s)
2. Emergency: **elder A** food poison (15s)
7. Watch TV with **elder D** (10s)
3
4
2
1
1
4
2
9. Greet to elder **g** (2s)
6. Remind **elder E** to take medicine (3s)
7. Chat with **elder F** (6s)
10. Ask **elder E** who is uncomfortable (15s)

id	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
γ	3	8	1	1	2	3	1	5	2	8
β	0.94	0.98	0.9	0.9	0.92	0.96	0.9	0.96	0.92	0.98

i from launching until being completed by the robot. With these definitions, we can obtain the optimal accumulative reward by the proof of exhaustion, which require $O(M!)$ time if there exists M instructions. By analyzing the accumulative reward of given random ten tasks, the optimality of the proposed algorithm with regard to other classic planners can be evaluated.

Given an example scenario as Table 5-6 where there exists ten instructions to be solved, Figure 5-6 presents the accumulative reward curve of different planners with regard to time. First of all, one can easily observes that the optimal planner (*Opt*) undoubted the highest reward in a short period of time, as shown in the pink line. However, since the time complexity is $O(10!)$, this planner becomes too time-consuming to generate the optimal plan, which takes over 200 seconds. As a result, the accumulative reward of the optimal planner with motion turns out to be the lowest when taking the decision time into account, as shown in the grey line (*Opt w/m*), not to mention that it is the most time consuming planner among all. On top of that, our hardware runs out of the

doi:10.6342/NTU201902426

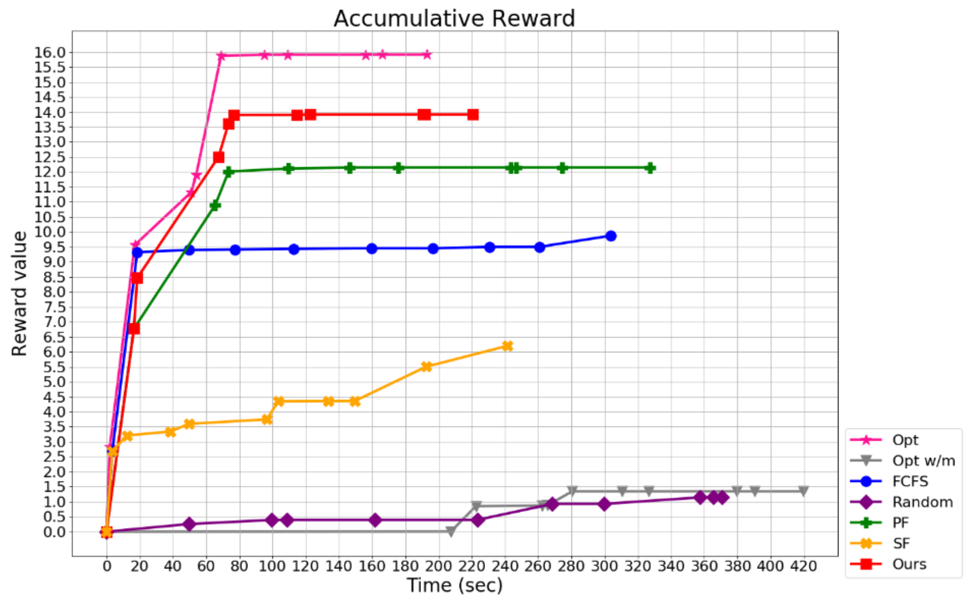


Figure 5-6 The accumulative reward of example Figure 5-13

computational resources and crushes if there exist more than ten remaining instructions for exhaustion method. That is, although the exhaustion method seems to solve the problem wisely, it cannot be implemented in a practical way.

Secondly, since the shortest time first planner (*SF*) always picks the remaining instruction with the minimum time requirement, it completes all the instructions in a relatively short period of time, as the yellow line shown in Figure 5-6. Nonetheless, due to the fact that the SF does not concern the priority of human requests, there is a large gap between the accumulative reward of *SF* and that of the optimal planner. On the other hand, the priority first planner (*PF*) earns more reward than *SF* as it always picks the remaining instruction with the maximum initial reward, as shown in the green line, but it takes more time to finish all the instructions, which is even more inefficient than the first come first serve planner (*FCFS*). As for our planner, the red line, obtains high accumulative reward while completing all the instructions in a short time among all the other planners. That is to say, comparing to the classical planners, our planner shows its optimality in scheduling multiple social instructions. The following Figure 5-7 is another example scenario of

doi:10.6342/NTU201902426

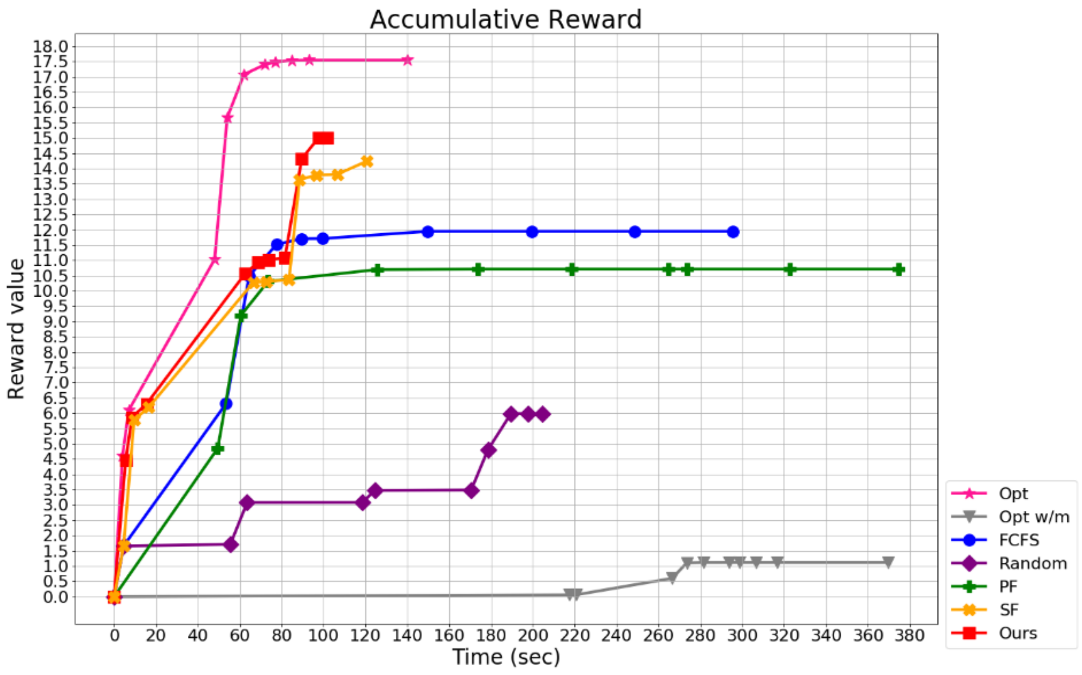


Figure 5-7 The accumulative reward of arbitrary ten instructions

Table 5-7 The average reward and completing time of given random instructions

Planner	*Opt*	*Opt w/m*	*FCFS*	*Rand*	*PF*	*SF*	***Ours***
reward	13.18	0.76	5.75	4.68	8.26	9.79	**11.87**
time	89.44	208.73	345.96	344.3	320.46	158.07	**159.31**

random ten instructions, from which our planner completes all the instructions even faster than the *SF*, while the *PF* shows less optimal and efficient than *FCFS*. Table 5-7 shows the average accumulative reward and completing time of different planners given 10 instructions randomly.

Furthermore, not only can we estimate the accumulative reward with respect to the optimal sequence but also analyze the similarity. Before that, a metric to measure the similarity is required. Given an instruction sequence A, we can define the position of a certain instruction with index id as $pos_A(id)$, starting from 1. For example, given the sequence $A = \{2,\ 4,\ 3,\ 1\}$, the position of instruction index 4 will be $pos_A(4) = 2$. Concerning the similarity degree, we define the cost by comparing the position distance

doi:10.6342/NTU201902426

Table 5-8 Similarity comparison of instruction order of planners

	Instruction **id** sequence										cost
Opt	1	2	8	6	10	9	7	3	5	4	-
FCFS	1	2	3	4	5	6	7	8	9	10	30
Rand	9	1	4	7	5	2	3	10	8	6	40
PF	2	10	8	1	6	9	5	3	4	7	14
SF	1	4	3	5	6	8	9	7	10	2	36
Ours	**2**	**1**	**10**	**8**	**6**	**7**	**9**	**5**	**3**	**4**	**10**

Table 5-9 The average similarity cost of given random instructions

Planner	*Opt*	*FCFS*	*Rand*	*PF*	*SF*	***Ours***
cost	-	31.78	36.22	19.33	28.66	**15.78**

of two instruction sequence. Given M instructions $\{1,\ 2,\ \ldots,\ M\}$ and two different sorted sequences A and B, Eq. (5-1) shows the calculation of similarity cost.

$$similarity\ \ cost = \sum_{i=1}^{M} |pos_A(i) - pos_B(i)| \tag{5-1}$$

For instance, given sequences $A = \{2,\ 4,\ 3,\ 1\}$ and $B = \{1,\ 2,\ 3,\ 4\}$, the similarity cost of single element 1 is $|4-1| = 3$, and the total similarity cost will be $|4-1| + |1-2| + |3-3| + |2-4| = 6$, referring to the above Eq. (5-1).

Table 5-8 is the similarity comparison of scenario in Table 5-6, which shows that our planner has the least cost and highest similarity among other planners with regard to the optimal solution. Moreover, the sequence also shows that both our planner and the optimal one tends to complete all the instruction that place at the same location. The average similarity cost of given random ten instructions is shown in Table 5-9, which also tells that our planner is the most similar toward the optimal planner, inferring its high optimality.

doi:10.6342/NTU201902426

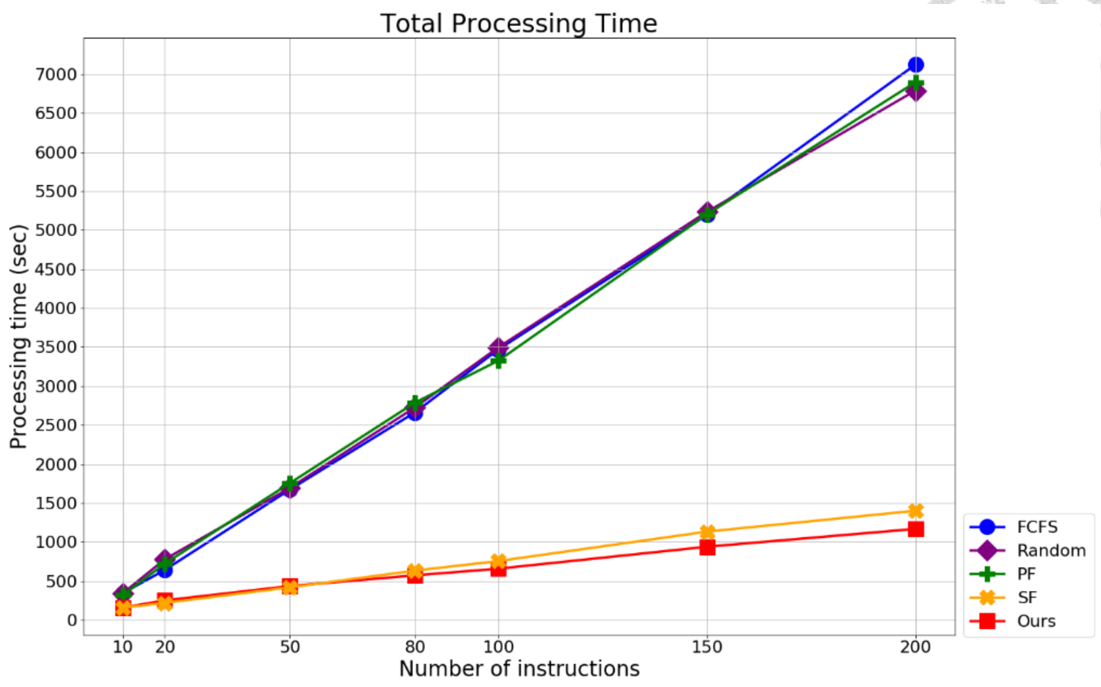


Figure 5-8 Processing time of different numbers of instructions

5.3.2 Efficiency Comparison of Task Planner Algorithm

The efficiencies of different task planners are compared in this section. Given a series of instructions randomly, the total processing time of different planners can be evaluated. Figure 5-8 is the average total processing time with regard to different numbers of instructions. Note that since the optimal planner only accommodates up to ten instructions, this experiment does not take it into comparison.

While increasing the instruction number, the complete time of *FCFS* undoubted rises dramatically since it takes neither priority nor task time into consideration. As for *Rand*, though there may exist cases with lower processing time, most of the possible planning sequences are not efficient enough and thus the average total time sours as the number of instructions increases. The *PF* planner, while merely taking the priority of instructions into consideration, can be regarded as sorting the priority of instructions at first and then performs the *FCFS*. Therefore, the processing time curve is similar to *FCFS* and also quite inefficient. In contrast, the *SF* greedily picks the instruction that requires the

doi:10.6342/NTU201902426

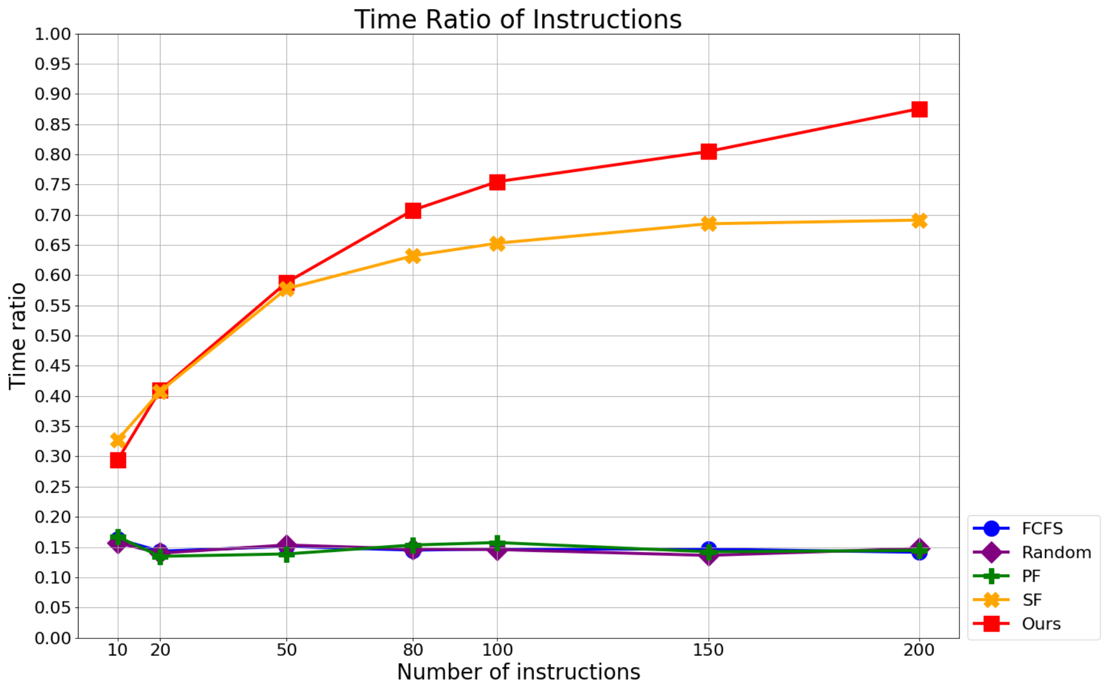


Figure 5-9 Time ratio of processing instructions

minimal navigation time and instruction performing time. Therefore, it decreases the total processing time. Nevertheless, *SF* does not take the priority into concern and may thus not meet the human needs, making the robot less optimal as the previous section presents. In our algorithm, not only does the time but also the priority be concerned when planning. As a consequence, the proposed task planner can complete instructions faster than most of the algorithms, especially in a large scale as 200 instructions.

To further analyze the efficiency, the ratio of performing instructions and total processing time is calculated as Eq. (5-2), where the $total\ \ time$ equals the sum of $task\ \ processing\ \ time$, $decision\ \ time$, and $navigation\ \ time$.

$$time\ \ ratio = \frac{task\ \ processing\ \ time}{total\ \ time} \tag{5-2}$$

That is to say, the higher the ratio is, the more time the planner spends on completing human requests rather than wandering for a long time. Figure 5-9 shows the time ratio with regard to the number of randomized instructions, through which the proposed

doi:10.6342/NTU201902426

algorithm shows high efficiency among other planners. On the contrary, the *FCFS*, *PF*, and *Random* spend more time on navigation, leading to less efficiency than ours. To give a brief summary, our planner shows both optimality and efficiency in the given scenario. Not only does it earn the highest accumulative reward, but also complete all the instructions in a relative short period of time.

5.3.3 Real World Implementation and Analysis

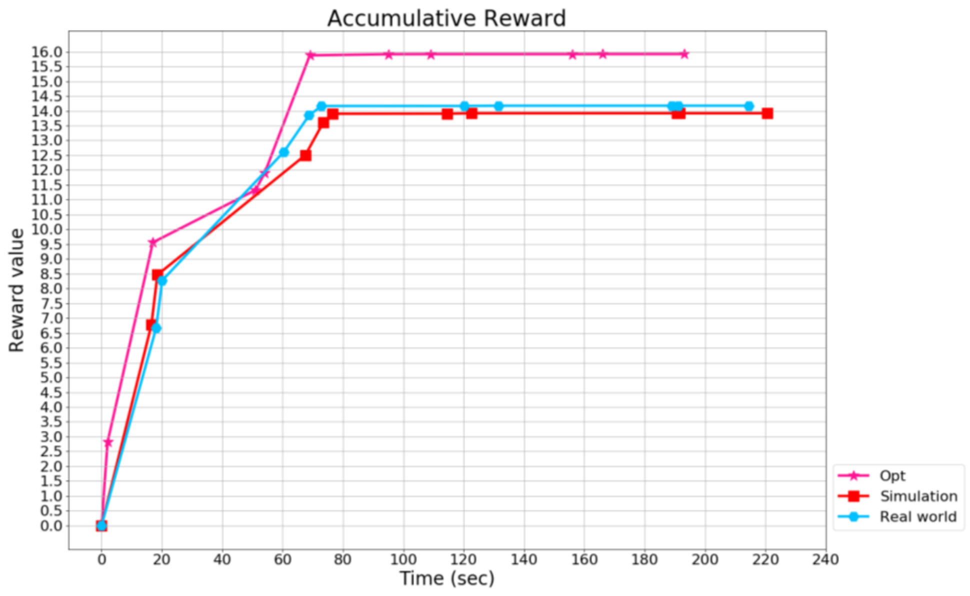


Figure 5-10 The accumulative reward of simulation and real world implementation

After comparing with different planners in the simulation environment, we show that our planner is the most optimal and efficient among all other planners. On top of that, we will show the experimental results of our real world implementation. The accumulative reward curves of our planner in both simulation and real world implementation are shown in Figure 5-10, where the pink curve is still the theoretical optimal reward from exhaustion, red curve is the simulation result, and the light blue curve is the implementation result in the real world.

doi:10.6342/NTU201902426

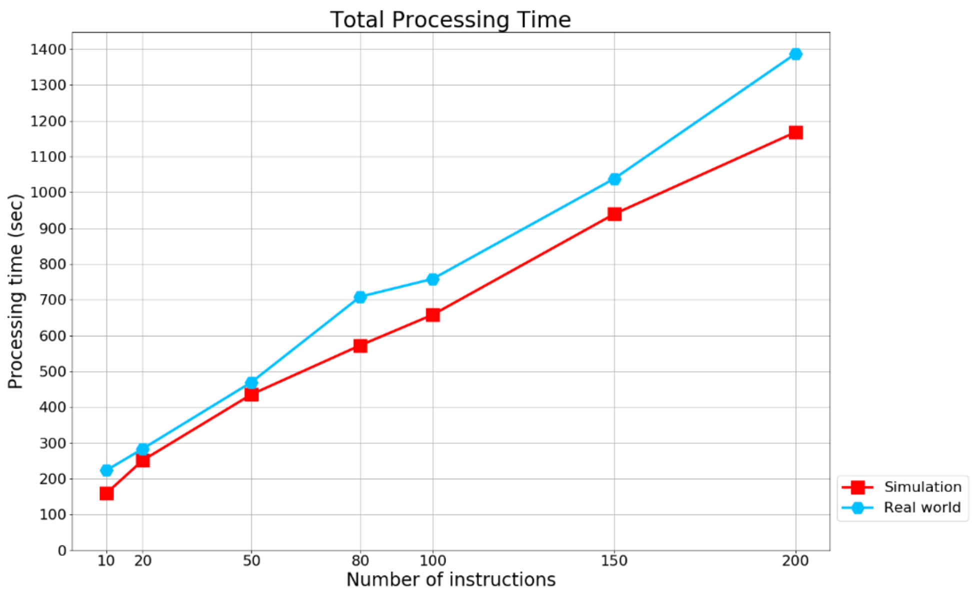


Figure 5-11 The total processing time in simulation and real world implementation

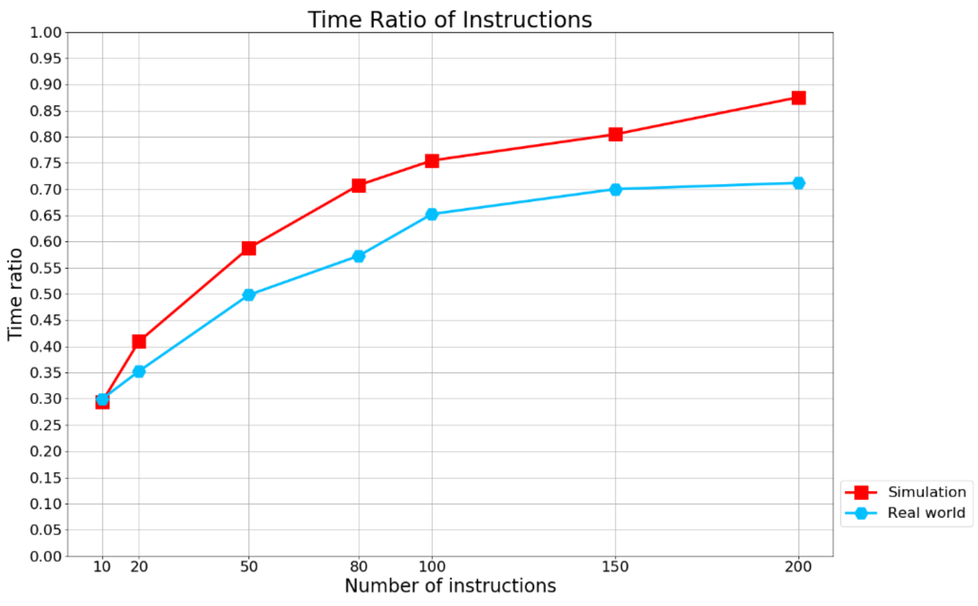
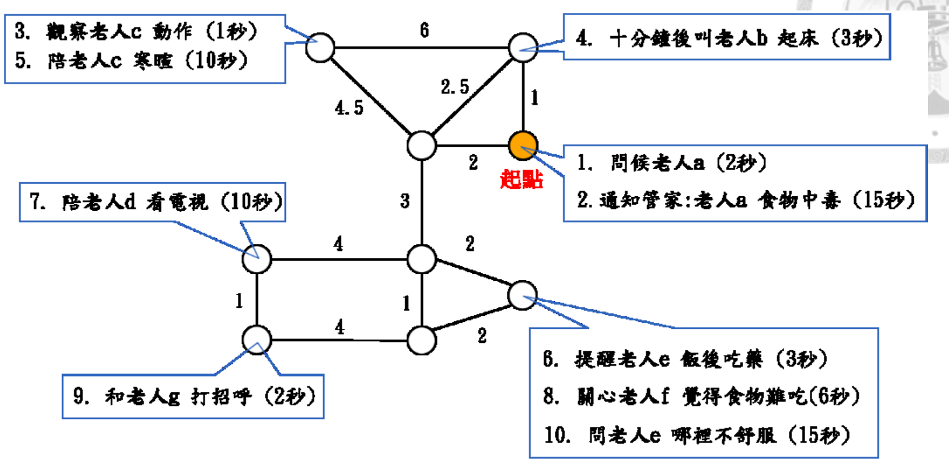


Figure 5-12 The time ratio of simulation and real world implementation

On the other hand, as for the efficiency, we run our system in the real world and analyze the total processing time and time ratio given arbitrary ten instructions. The results are presented in both Figure 5-11 and Figure 5-12, which again show that our implementation results meet the simulation and thus have high efficiency in completing up to 200 tasks. These results show that our real world application truly fits the simulation result with respective to not only the optimality analysis but also the efficiency evaluation.

doi:10.6342/NTU201902426

5.3.4 Similarity Comparison of Task Planner Algorithm



3. 觀察老人c 動作 (1秒)
5. 陪老人c 寒暄 (10秒)
6
4. 十分鐘後叫老人b 起床 (3秒)
2.5
4.5
1
2
起點
1. 問候老人a (2秒)
2. 通知管家:老人a 食物中毒 (15秒)
7. 陪老人d 看電視 (10秒)
3
4
2
1
1
4
2
9. 和老人g 打招呼 (2秒)
6. 提醒老人e 飯後吃藥 (3秒)
8. 關心老人f 覺得食物難吃(6秒)
10. 問老人e 哪裡不舒服 (15秒)

Figure 5-13 An example questionnaire of random instructions for human scheduling

Last but not the least, it is worth mentioned that the optimal solution of our scenario is similar to human scheduling in the small scale within 10 instructions. In other words, given random 10 instructions, the completion of the entire instruction sequence which earns the optimal reward should have low similarity cost with that of human scheduling. To evaluate this, a questionnaire is designed by randomly generating instructions and mapping them on the discrete graph. One of the examples is shown in Figure 5-13. Although the tasks in the questionnaire are written in Chinese for reading convenience, our instructions include functions like *"chatting"*, *"encourage"*, *"check status", "ask physical request"*, which can be successfully mapped to the our proposed instruction structure as listed in Table 4-1.

While the volunteers fill the questionnaire and generate sequences of instructions, we can also select initial reward value to represent the priority. During this experiment, we first calculate the average index of each instruction, where the formula of average index $\overline{x_I}$ of an arbitrary instruction I given M samples is as the following Eq. (5-3).

$$\overline{x_I} = \frac{1}{M}\sum_{k=0}^{M-1} id_k(I) \tag{5-3}$$

doi:10.6342/NTU201902426

Let's take an instruction set $\{1,\ 2,\ 3,\ 4\}$ as example. Supposed there are three samples from the questionnaire: $A = \{2,\ 4,\ 3,\ 1\}$, $B = \{1,\ 2,\ 3,\ 4\}$, and $C = \{3,\ 2,\ 1,\ 4\}$, the average index of instruction 1 equals to $\frac{4+1+3}{1+1+1} = 2.67$, and the overall average index of the instruction set $\{1,\ 2,\ 3,\ 4\}$ will be as $\{2.67, 1.67, 2.33, 3.33\}$ respectively. The instruction sequence sorted by this average index will be $\{2,\ 3,\ 1,\ 4\}$. This analysis highlights the trend of the order among every elements in the sequence even under few samples, which leads to more adequate results than picking the mode from the questionnaire.

As for the user study, we will take Figure 5-13 for instance again. Table 5-10 shows the comparison between human scheduling and the optimal scheduling. The sequence of *Human* is sorted through average index analysis, and the *Optimal* comes from the exhaustion method. With the similarity cost being 2.0 merely, the optimal scheduling shows high similarity relative to human scheduling. Since the optimal scheduling has high similarity with regard to human scheduling, it can be concluded that under the small scale within 10 instructions, the more optimal a planner is, the more human-like task scheduling it generates. In other words, our planner, while aims to complete instructions as fast as possible, sorts the instructions in a human-like way, making the robot become the most "considerate" among all the other planners.

Table 5-10 Similarity comparison between human scheduling and optimal solution

		Instruction **id** sequence										cost
Human	$\bar{x}$	1.2	1.6	4.3	4.7	4.9	6.4	7.1	7.2	8.2	8.5	
	Sequence	1	2	**6**	**8**	10	9	7	3	5	4	
Optimal	Sequence	1	2	**8**	**6**	10	9	7	3	5	4	**2.0**

doi:10.6342/NTU201902426

Chapter 6 Conclusion and Future Works

In this thesis, we propose a system integrating perception and decision making to achieve human robot interaction for social robots. While implementing the perception system, the robot is able to perceive human status both visually and verbally. With the assist of deep learning frameworks such as YOLO and OpenPose, the robot can detect objects and skeletons precisely, and the heuristic algorithms can thus simultaneously generate accurate real-time functionalities like human localization, human identification, and framewise hierarchical action detection. With the combination of deep learning and heuristic algorithms, the system is capable of performing accurate perceptions in real time. Moreover, we design a data structure to store these results in an efficient way so that the robot can memorize human information for future usage. In addition, we also design an instruction structure for the robot to digest perceptions into executable instructions.

As for the decision making system, we propose a model to not only transform the household environment into discrete graph but also formulate instructions into time-decaying reward functions. This model further formulates our goal of completing all the instructions into an optimization problem. Leveraging from the Task and Motion Planning (TAMP) architecture, we design a task planning algorithm with the purpose of maximizing the accumulative reward. Furthermore, this algorithm can also deal with dynamic instruction scenarios in an efficient way such that the robot can react toward newly-added requests from human-beings in time. On top of that, we take the computational resource into consideration such that the whole system can be processed with a single desktop.

The experiments demonstrate that both the perception and decision make effective and efficient outputs. In the visual perception part, the system can localize human in the

doi:10.6342/NTU201902426

semantic location with relative low error in real time. The system can as well identify human with the stored clothing colors given the current observation frame. On top of that, the action detection generates satisfactory outputs. These evaluations can be established from the self-collected Pepper Image Testing Dataset.

As for the decision making validations, we compare our proposed task planning algorithm with classic planners like First Come First Serve, Randomize, Priority First, Shortest Time First. First of all, from the optimality analysis, the proposed algorithm obtains the highest accumulative reward in relatively short time. Secondly, for the efficiency evaluation, the proposed algorithm can complete 200 instructions a lot faster than any other planners. Besides, it also has higher time ratio of instruction processing among all the other planners. Last but not least, from the similarity comparison, we show that the more optimal the planner is, the more human-like decision it makes. Thus, we show that our planner can achieve every tasks considering both human robot interactions and time efficiency.

The future work of this thesis is to expand the system with regard to multiple robots. As there may exist multiple robots in an elder house in the near future, the scenario of distributing instructions for them to accomplish while concerning optimality, efficiency and user experience shall be a critical issue. Besides, the algorithms for error handling with the view to robust applications can also be developed. On top of that, the functionality of robots can also be enlarged on the basis of our instruction structure to achieve even more complex tasks. As for the human identification, the system can apply both faces for static features and cloth colors as dynamic keys to avoid human interruptions. To sum up, our system provides a fundamental framework that mixes perceptions and decision making such that the robot can improve our society in a more practical way.

doi:10.6342/NTU201902426

doi:10.6342/NTU201902426

REFERENCE

[1] C. Feichtenhofer, A. Pinz, and R. P. Wildes, “Spatiotemporal Multiplier Networks for Video Action Recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4768–4777.

[2] Y. Zhu, Z. Lan, S. Newsam, and A. Hauptmann, “Hidden Two-Stream Convolutional Networks for Action Recognition,” in *Computer Vision - ACCV 2018, Lecture Notes in Computer Science, vol 11363*, 2018, pp. 363–378.

[3] K. Soomro, A. R. Zamir, and M. Shah, “UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild,” *CoRR*, vol. abs/1212.0, no. November, 2012.

[4] D. Feillet, P. Dejax, and M. Gendreau, “Taveling Salesman Problems with Profits: An Overview,” *Transp. Sci.*, vol. 39, no. 2, pp. 188–205, 2001.

[5] E. Angelelli, C. Bazgan, M. G. Speranza, and Z. Tuza, “Complexity and approximation for Traveling Salesman Problems with profits,” *Theor. Comput. Sci.*, vol. 531, pp. 54–65, 2014.

[6] S. Vasudevan, S. Gächter, V. Nguyen, and R. Siegwart, “Cognitive maps for mobile robots-an object based approach,” *Rob. Auton. Syst.*, vol. 55, no. 5, pp. 359–371, 2007.

[7] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*, 1st ed., vol. 73. Springer Publishing Company, Incorporated, 2013.

[8] R. R. Murphy, *Introduction to AI Robotics*, 1st ed. Cambridge, MA, USA: MIT Press, 2000.

[9] W. Burgard, C. Stachniss, G. Grisetti, B. Steder, R. Kummerle, C. Dornhege, M. Ruhnke, A. Kleiner, and J. D. Tard´os, “A Comparison of SLAM Algorithms Based on a Graph of Relations,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009, pp. 2089–2095.

[10] J. Engel, T. Sch, and D. Cremers, “LSD-SLAM: Large-Scale Direct Monocular SLAM,” in *Computer Vision – ECCV 2014. Lecture Notes in Computer Science, vol 8690*, 2014, pp. 834–849.

[11] R. Mur-Artal, J. M. M. Montiel, and J. D. Tard´os, “ORB-SLAM: A Versatile and Accurate Monocular SLAM System,” *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, 2015.

[12] R. Mur-Artal and J. D. Tardos, “ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras,” *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, 2017.

[13] J. Tang, Y. Chen, A. Jaakkola, J. Liu, J. Hyyppä, and H. Hyyppä, “NAVIS-An UGV Indoor Positioning System Using Laser Scan Matching for Large-Area Real-Time Applications,” *Sensors (Basel).*, vol. 14, no. 7, pp. 11805–11824, 2014.

[14] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 1999.

[15] G. Grisettiyz, C. Stachniss, and W. Burgard, “Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2005, vol. 17, no. 1, pp. 2432–2437.

[16] G. Grisetti, C. Stachniss, and W. Burgard, “Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters,” *IEEE Trans. Robot.*, vol. 23, no. 1, pp. 34–46, 2007.

[17] H. Kretzschmar and C. Stachniss, “Information-theoretic compression of pose graphs for laser-based SLAM,” *Int. J. Rob. Res.*, vol. 31, no. 11, pp. 1219–1230,

doi:10.6342/NTU201902426

2012.
[18] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, “A Tutorial on Graph-Based SLAM,” *IEEE Intell. Transp. Syst. Mag.*, 2010.
[19] R. Girshick, “Fast R-CNN,” in *IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.
[20] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.
[21] J. Redmon and A. Farhadi, “YOLO9000: Better, Faster, Stronger,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6517–6525.
[22] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” *CoRR*, vol. abs/1804.0, 2018.
[23] J. J. Gibson, *The Senses Considered as Perceptual Systems*, no. c. 2012.
[24] V. Dutta and T. Zielinska, “Action Prediction Based on Physically Grounded Object Affordances in Human-Object Interactions,” in *International Workshop on Robot Motion and Control (RoMoCo)*, 2017, pp. 47–52.
[25] H. S. Koppula and A. Saxena, “Anticipating Human Activities Using Object Affordances for Reactive Robotic Response,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 1, pp. 14–29, 2016.
[26] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. .
[27] P. E. Hart, N. J. Nilsson, and B. Raphael, “A Formal Basis for the Heuristic Determination of Minimum Cost Paths,” *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, 1968.
[28] S. Koenig and M. Likhachev, “D* Lite,” in *Proceedings of the National Conference on Artificial Intelligence*, 2002, pp. 476–483.
[29] S. M. LaValle, “Rapidly-Exploring Random Trees: A New Tool for Path Planning,” *TR 98-11*, 1998.
[30] M. Ghallab, D. Nau, and P. Traverso, *Automated Planning: Theory and Practice*, 3rd ed. Morgan Kaufmann, 2004.
[31] T. Lozano-Pérez, J. L. Jones, E. Mazer, and P. A. O’Donnell, “Task-Level Planning of Pick-and-Place Robot Motions,” *Computer (Long. Beach. Calif).*, vol. 22, no. 3, pp. 21–29, 1989.
[32] N. T. Dantam, S. Chaudhuri, and L. E. Kavraki, “The Task-Motion Kit: An Open Source, General-Purpose Task and Motion-Planning Framework,” *IEEE Robot. Autom. Mag.*, vol. 25, no. 3, pp. 61–70, 2018.
[33] W. Jacak, “Robot Task And Movement Planning,” in *AI, Simulation and Planning in High Autonomy Systems*, 1990, pp. 168–173.
[34] L. P. Kaelbling and T. Lozano-Pérez, “Hierarchical Task and Motion Planning in the Now,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 1470–1477.
[35] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, “Incremental Task and Motion Planning: A Constraint-Based Approach,” *Robot. Sci. Syst.*, 2016.
[36] R. Chitnis, D. Hadfield-Menel, A. Gupta, S. Srivastava, E. Groshev, C. Lin, and P. Abbeel, “Guided Search for Task and Motion Plans Using Learned Heuristics,” in *Proceeding of IEEE International Conference on Robotics and Automation (ICRA)*, 2016, vol. 2016-June, pp. 447–454.

doi:10.6342/NTU201902426

[37] K. Baizid, A. Yousnadj, A. Meddahi, R. Chellali, and J. Iqbal, "Time scheduling and optimization of industrial robotized tasks based on genetic algorithms," *Robot. Comput. Integr. Manuf.*, vol. 34, pp. 140–150, 2015.

[38] B. Kim, L. P. Kaelbling, and T. Lozano-Perez, "Learning to Guide Task and Motion Planning using Score-Space Representation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 2810–2817.

[39] P. Muñoz, M. D. R-Moreno, and D. F. Barrero, "Unified framework for path-planning and task-planning for autonomous robots," *Rob. Auton. Syst.*, vol. 82, pp. 1–14, 2016.

[40] C. Wong, E. Yang, X. Yan, and D. Gu, "Dynamic Anytime Task and Path Planning for Mobile Robots," in *UKRAS19 Conference on Embedded Intelligence*, 2019.

[41] Y. Jiang, F. Yang, S. Zhang, and P. Stone, "Integrating Task-Motion Planning with Reinforcement Learning for Robust Decision Making in Mobile Robots," in *Proceedings of the International Conference on Autonomous Agents and MultiagentSystems (AAMAS)*, 2019.

[42] R. Alami, A. Clodic, V. Montreuil, E. A. Sisbot, and R. Chatila, "Toward Human-Aware Robot Task Planning," in *Proceedings of the AAAI Spring Symposium To Boldly Go Where No Human-Robot Team Has Gone Before.*, 2006.

[43] V. V. Unhelkar , P. A. Lasota , Q. Tyroller , R. Buhai , L. Marceau , B. Deml, and J. A. Shah, "Human-Aware Robotic Assistant for Collaborative Assembly: Integrating Human Motion Prediction With Planning in Time," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 2394–2401, 2018.

[44] W. Y. G. Louie, T. Vaquero, G. Nejat, and J. C. Beck, "An autonomous assistive robot for planning, scheduling and facilitating multi-user activities," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 5292–5298.

[45] K. E. C. Booth, T. T. Tran, G. Nejat, and J. C. Beck, "Mixed-Integer and Constraint Programming Techniques for Mobile Robot Task Planning," *IEEE Robot. Autom. Lett.*, vol. 1, no. 1, pp. 500–507, 2016.

[46] Z. Cao, G. Hidalgo, T. Simon, S. E. Wei, and Y. Sheikh, "OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields," in *arXiv preprint arXiv:1812.08008*, 2018.

[47] Z. Cao, T. Simon, S. -E. Wei, and Y. Sheikh, "Realtime Multi-person 2D Pose Estimation Using Part Affinity Fields," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1302–1310.

[48] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, "Hand Keypoint Detection in Single Images Using Multiview Bootstrapping," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1145–1153.

[49] S. -E.Wei, V. Ramakrishna, T. Kanada, and Y. Sheikh, "Convolutional Pose Machines," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4724–4732.

[50] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional Architecture for Fast Feature Embedding," *arXiv Prepr. arXiv1408.5093*, 2014.

[51] G. Hidalgo, Z. Cao, T. Simon, S. -E. Wei, H. Joo, and Y. Sheikh, "CMU-Perceptual-Computing-Lab / openpose," 2019. [Online]. Available: https://github.com/CMU-Perceptual-Computing-Lab/openpose.

[52] "JSK Ros Packages for Smartphones," 2019. [Online]. Available:

doi:10.6342/NTU201902426

https://github.com/jsk-ros-pkg/jsk_smart_apps.
[53] C. J. Hutto and E. E. Gilbert, "VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text," in *Proceedings of the International AAAI Conference on Web and Social Media (ICWSM)*, 2014, pp. 216–225.
[54] A. Fusiello, *Elements of Geometric Computer Vision*. 2006.
[55] T. Riemersma, "Colour metric," 2010. [Online]. Available: http://www.compuphase.com/cmetric.htm.
[56] K. M. Varadarajan and M. Vincze, "AfNet: The Affordance Network," in *Computer Vision – ACCV 2012. Lecture Notes in Computer Science*, 2012, vol. 7724, pp. 512–523.
[57] "vaderSentiment: VADER Sentiment Analysis," 2019. [Online]. Available: https://github.com/cjhutto/vaderSentiment.
[58] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System Morgan," in *ICRA Workshop on Open Source Software*, 2009.
[59] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots," in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1999.
[60] D. Fox, "KLD-Sampling: Adaptive Particle Filters," in *Proceedings of the International Conference on Neural Information Processing Systems: Natural and Synthetic (NIPS)*, 2001, vol. 14, no. 1, pp. 713–720.
[61] D. Fox, "Adapting the Sample Size in Particle Filters Through KLD-Sampling," *Int. J. Robot. Res.*, 2003.
[62] B. P. Gerkey, "amcl - ROS Wiki," 2018. [Online]. Available: http://wiki.ros.org/amcl.
[63] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*, 3rd ed., no. 2. The MIT Press, 2009.
[64] D. V. Lu, D. Hershberger, and W. D. Smart, "Layered costmaps for context-sensitive navigation," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014, pp. 709–715.
[65] D. Fox, W. Burgard, and S. Thrun, "The Dynamic Window Approach to Collision Avoidance," *IEEE Robot. Autom. Mag.*, vol. 4, pp. 23–33, 1997.
[66] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Trajectory Modification Considering Dynamic Constraints of Autonomous Robots," in *Proceedings of the 7th German Conference on Robotics*, 2012, pp. 74–79.
[67] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Efficient Trajectory Optimization using a Sparse Model," in *Proceedings of the European Conference on Mobile Robots (ECMR)*, 2013, pp. 138–143.
[68] C. Rösmann, F. Hoffmann, and T. Bertram, "Planning of Multiple Robot Trajectories in Distinctive Topologies," in *Proceedings of the European Conference on Mobile Robots (ECMR)*, 2015, pp. 1–6.
[69] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Rob. Auton. Syst.*, vol. 88, pp. 142–153, 2017.
[70] C. Rösmann, F. Hoffmann, and T. Bertram, "Kinodynamic Trajectory Optimization and Control for Car-Like Robots," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 5681–5686.
[71] K. Chatterjee and T. A. Henzinger, *25 Years of Model Checking*. Berlin,

doi:10.6342/NTU201902426

Heidelberg: Springer-Verlag, 2008.

[72] S. J. Russell and P. Norvig, *Artificial Intelligence A Modern Approach*, 3rd ed. Upper Saddle River, N.J. :Prentice Hall, 2010.

[73] D. L. Poole and A. K. Mackworth, *Artificial Intelligence: Foundations of Computational Agents*, 2nd ed. Cambridge University Press, 2017.

[74] P. -T. Wu, C. -A. Yu, S. -H. Chan, M. -L. Chiang, and L. -C. Fu, "Multi-Layer Environmental Affordance Map for Robust Indoor Localization, Event Detection and Social Friendly Navigation," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[75] S. -H. Chan, X. Xu, P. -T. Wu, M. -L. Chiang, and L. -C. Fu, "Real-time Obstacle Avoidance using Supervised Recurrent Neural Network with Automatic Data Collection and Labeling," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2019.

doi:10.6342/NTU201902426