Internet-based Teleoperation for Robot Navigation

By

Meng Wang

The Hong Kong Polytechnic University

Department of Computing

A thesis submitted in fulfilment of the requirements for the

Degree of Doctor of Philosophy

May 2006

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ (Signed)

Meng Wang (Name of student)

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my sincere gratitude to the following persons or parties.

First of all, I must thank my supervisor, Dr. James Liu, who has provided constructive, valuable advices and guidance to me throughout the period of my study and research. Without his insight, enthusiasm, encouragement and inspiration, I would not have the courage and determination to complete the thesis.

Second, I would like to thank Dr. Raymond Lee, Dr. John Sum, and Dr. Yangjian, who have given me considerable suggestions and academic comments.

Third, Mr. Martin, who has helped polish my papers and given me a lot of unreserved and remarkable comments, is highly appreciated.

Fourth, I thank Liyan and Jing shuyuan who have discussed with me about academic problems and given me some valuable comments.

Fifth, thanks must be given to my department at PolyU for providing monthly studentship and research finances, and to the technical team including Evan and Amy who have provided excellent support and services.

Sixth, I would like to express my special thanks to the following friends: Yaogang, Fengbo, Raymond Kwong, Steve, Niuben, Zheng yongjie, Molly. Their assistances and the warm relationship are appreciated.

Finally, I thank my parents and my wife for their continuous and tremendous support, encouragement and tolerance as well as their patience during my research.

Without the support and assistance of these people and other persons who have helped me but not appeared here, I would not able to continue my study and research, nor complete the thesis. Thanks again.

ABSTRACT

Internet-based robot teleoperation obviates the need for dedicated networks and devices, reduces costs, extends operating distances, and allows precious resources sharing for public education or academic research. Except for operating in hazardous environments, Internet telerobotics has opened up a new range of real-world applications, involving tele-manufacturing, tele-training, tele-surgery, museum guide, space exploration, disaster rescue, and health care. There are many problems on Internet-based teleoperation that need to be addressed, such as data transmission over uncertain time-delay and unreliable Internet, teleoperation by inexperienced users, short of interactivity, and so on. Moreover, Internet robots require a much higher degree of autonomy than traditional teleoperation so that the robots are able to ensure safe operations and perform some tasks autonomously.

In this thesis, we aim at developing a practical robotic system for the target application: the inexperienced Internet users can remotely control a wheeled robot which is able to perform some complex tasks autonomously (e.g. active map learning, goal-oriented navigation) or to interact with human operator in order to explore unknown and dynamic environments. The experiments are based on a Pioneer robot that is equipped with an onboard camera and eight forward ultrasonic sensors. The control commands transfer through radio Ethernet devices. To help realize such robotic system, the research is conducted on the following aspects:

1) The video transmission via the low-bandwidth Internet is investigated and implemented so that the robot's surroundings can be seen by any remote operators through the images captured from an onboard camera. It is a prerequisite to develop a practical teleoperation system. Traditional approach is via the picture transmission (e.g. JPEG or GIF), which leads to a very poor quality of service (QoS) because of the high latency of the Internet, such as long time delay, data error or restricted bandwidth. The thesis investigates and develops a streaming technology based approach for streaming video transmission. Two video compression algorithms (WMV9 and MPEG4) under different bandwidth, two video encoding methods (CBR and Quality-based VBR) as well as the transmission stability and time delay have been investigated.

2) A framework for autonomous navigation using fuzzy logic is proposed. This work is a base for the subsequent designs of intelligent control programs so that the mobile robot is able to autonomously perform some complex tasks amid various degrees of uncertainties. The proposed framework involves goal determination, preprocessing, behavior design, behavior arbitration, and command fusion. Traditional framework for autonomous navigation is SMPA (Sense-Model-Plan-Act) approach, which is inadequate for dealing with unknown and dynamic real world. The behavior-based approach can act in real-time and has good robustness in such environments. The preprocessing module is used to reduce the complexity of input space by introducing a limited number of intermediate variables. The elementary behavior can be designed using fuzzy logic controller or an analytic algorithm. A behavior arbitration module is used to calculate the crisp weighting factors of each elementary behavior. The final robot motion output is obtained by the command fusion for a weighting combination of all elementary behaviors. A goal-oriented navigation task, combined with obstacle-avoidance (OA) and goal-seeking (GS) behaviors, is implemented as an example of the proposed framework.

3) A new teleoperation approach so called *telecommanding* is proposed to provide an interactive control interface and a complete framework for control management and command processing. The traditional direct control reduces the stability of control loop because the controlled robot has no local intelligence and it needs to maintain continuous connection. The existing supervisory control methods are inadequate mainly in that they fail to provide human-robot interactivity. The proposed approach involves two different but complementary commands: joystick command (e.g. LEFT, RIGHT, UP, and DOWN) and linguistic command (e.g. MOVE, TURN, GOTOEND, WANDER, COORDINATE, and MAPPING). Each command is designed to perform independent task, which is defined with multiple events (non-time action references) and corresponding response functions. Simulated and real world experiments have been conducted to test the use of both joystick commands and linguistic commands for Internet-based robot teleoperation. The advantages and disadvantages as well as stability of telecommanding are analyzed.

4) To model a priori unknown environment (i.e. a MAPPING linguistic command), a new map learning approach called *memory grid mapping* is proposed. The robot builds a map based on robot's sensory information and actively explores the unknown environment. The approach includes a map model, a map update method,

an exploration method, and a map postprocessing method. The map adopts a grid-based representation. A so-called obstacle memory dot (OMD) matrix is designed to save the frequency values which measure the confidence that a cell is occupied by an obstacle. A so-called trajectory memory dot (TMD) matrix is designed to save the trajectory traversed by the robot in order to facilitate the online path planning. Two behaviors, path-exploring behavior and environment-detecting behavior, are coordinated to make the robot exploring a least known environment. The map postprocessing method includes a threshold operation, a template operation, and an insert operation. The efficiency of map learning is investigated. The map accuracy under different cell sizes and different map postprocessing is investigated as well. Experiments are done for the map learning in different simulation environments.

5) For a teleoperated mobile robot that is exploring unknown indoor environments, it is desired that the robot is able to autonomously arrive at a given goal location (i.e. an enhanced COORDINATE linguistic command), even though the environments involve all kinds of complex situations with local minima. The thesis proposes a new navigation method, namely minimum risk method, to realize such function. The method makes use of the proposed memory grid map. When a mobile robot is performing the goal-oriented navigation, it updates a memory grid map in real-time. A novel path-searching (PS) behavior is developed to use the map information and to recommend a safest regional direction that can enable the robot to detect potential local minima and escape from them. The final command outputs are obtained by coordinating the behaviors: PS, OA, and GS. Fuzzy logic controllers are used to implement behavior design and coordination. The method is experimentally demonstrated to give global convergence to a given goal location, even though it is used in the long-wall, large concave, recursive U-shape, unstructured, cluttered, maze-like, or dynamic (i.e. with moving human) environments.

The developed telerobotic system has been demonstrated to be feasible to provide the service of Internet-based teleoperation in university campus and exhibition center. The tests have been performed successfully through the Internet remotely from overseas places (e.g. Canada, Singapore, Chinese Beijing, Shanghai, Xiamen) to Hong Kong.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

PUBLICATIONS ARISING FROM THE THESIS

International Journal papers:

[3] Meng Wang, James N.K. Liu, " Interactive control for Internet-based mobile robot teleoperation", *Robotics and Autonomous Systems*, Vol.52, Iss.2-3, pp.160-179, Aug. 2005

[2] James N.K. Liu, Meng Wang, Feng Bo, "iBotGuard: An Internet-based Intelligent Robot Security System Using Invariant Face Recognition Against Intruder", *IEEE Transactions on Systems, Man, and Cybernetics. Part C*, Vol.3, Iss.1, pp.97-105, Feb. 2005

[1] Meng Wang, James N.K. Liu. "PolyUiBot: Sensibility Improvement Using Streaming Technology for Internet Telerobotics", *WSEAS Transactions on Computers*, Vol.3, Iss.3, pp.592-601, July 2004.

Book chapter:

[1] Meng Wang, James N.K. Liu, "Behavior-based Blind Goal-oriented Robot Navigation by Fuzzy Logic", R.Hkosla et al (Eds.) *Lecture Notes On Artificial Intelligences (LNAI) series by Springer-Verlag*, Vol.3681, pp.686-692, 2005

Under reviewing journal papers:

[1] Meng Wang, James N.K. Liu, "Fuzzy Logic Based Real-Time Robot Navigation in Unknown Environment with Local Minimum", under reviewing in *Robotics and Autonomous Systems*

[2] Meng Wang, James N.K. Liu, "Real-time map building and active exploration for autonomous robot in unknown environment", under reviewing in *Autonomous Robots*

Conference papers:

[8] James N.K. Liu, Meng Wang, "Fuzzy logic based active map learning for autonomous robot", presented in 2006 IEEE International Conference on Fuzzy Systems, Vancouver, Canada, July 16-21, 2006

[7] Meng Wang, James N.K. Liu, "Fuzzy logic based robot path planning in unknown environment", *4th International Conference on Machine Learning and Cybernetics*, Guangzhou, China, pp.813-818, August 18-21, 2005

[6] Meng Wang, James N.K. Liu, "Online Path Searching for Robot Autonomous Navigation", *IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, Singapore, pp.746-751, Dec.1-3, 2004

[5] Meng Wang, James N.K. Liu, "A Novel Teleoperation Paradigm for Human-robot Interaction", *IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, Singapore, pp.13-18, Dec. 1-3, 2004

[4] Meng Wang, James N.K. Liu, "Autonomous Robot Navigation using Fuzzy Logic Controller", *3th International Conference on Machine Learning and Cybernetics*, Shanghai, China, pp.691-696, August 26-29, 2004

[3] Meng Wang, James N.K. Liu. "A Streaming Technology Based Approach for Internet Telerobotics", *5th Proceedings of ACM Postgraduate Research Day*, Hong Kong, pp.194-202, January, 2004.

[2] Meng Wang, James N.K. Liu, "Streaming Technologies based Internet Telerobotics", 2nd *IASTED International Conference on Communications, Internet & Information Technology*, Scottsdale, USA, pp.565-570, Nov. 17-19, 2003

[1] Meng Wang, James N.K. Liu. “Wavelet-based Real-time Video Compression and Transmission System for Wireless Channel and Its Synchronization Technology”, *4th Proceedings of ACM Postgraduate Research Day*, Hong Kong, pp.143-150, January 2003.

CHAPTER 1. INTRODUCTION

1.1 Background

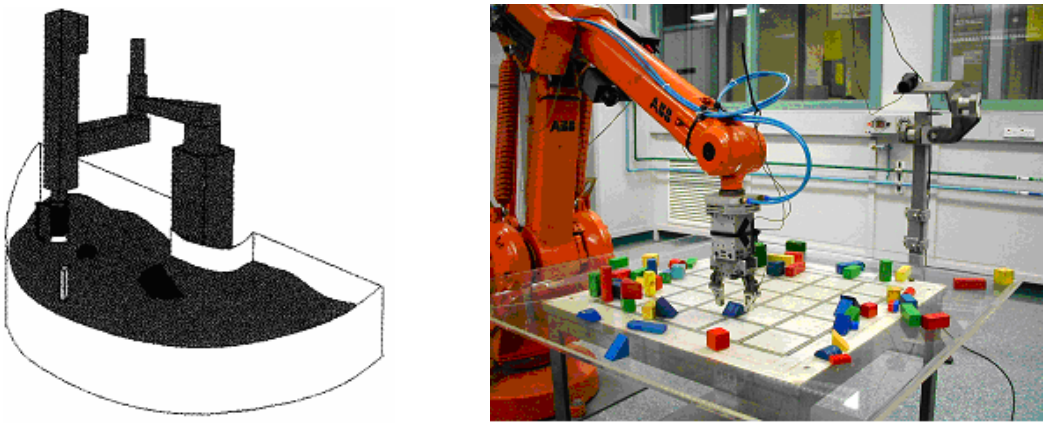
1.1.1 The history and development of Internet telerobotics

A new field, "*Internet telerobotics*" technologies for online robot teleoperation through the Internet, is emerging in the recent decade. *Online Robots* (or *Internet Robots*) are the robots that can be accessible from any computer on the Internet [Goldberg & Siegwart, 2002]. In the late 1950s and early 1960s, engineers began dreaming of remote manipulation where the operator and the manipulation task environment were distance apart and visual feedback was via TV. This kind of operation, to operate a vehicle or manipulator over a distance, is called *teleoperation*. The human is the *operator*, who monitors the operated machine and makes the needed control actions. The first "*remote-manipulators*" were developed for handling radioactive materials during 1950s. Outstanding pioneers were Raymond Goertz and his colleagues at the Argonne National Laboratory outside of Chicago, and Jean Vertut and his engineers at a counterpart nuclear engineering laboratory near Paris [Goertz & Thompson, 1954]. Their first system allowed human operators to stand outside of radioactive "hot cells," peer through leaded glass radiation barriers, and grip "master" arms coupled to "slave" arms and hands inside the cells, which in turn grasped the remote objects. The Internet's key advantage is the flexibility of where the operator can gain access to communication. With the rapid growth of the Internet, more and more intelligent devices or systems have been embedded into it for service, security and entertainment, including distributed computer systems, surveillance cameras, telescopes, manipulators and mobile robots. Moreover, recent advances in computer technology and software engineering and the development of inexpensive sensory equipment have allowed the development of not just local spot robot applications, but of Internet-based, distant-controlled telerobotics.

The Internet has opened the door to a much wider audiences. Some types of remote access technologies on the Internet have broadly used in our daily life. The computer network services, such as FTP, Telnet, the World Wide Web (WWW or the Web), e-mail, etc., provide us convenient tools and devices to transmit remote

information. Most people, however, continue to think of the Internet as a means of sending e-mail and getting information from remote databases. They remain unaware that another huge class of operations lies just ahead, namely the ability to control physical objects remotely over the Internet. What kinds of things? Anything one can imagine what an Internet robot can do. When away from home, for example, one could turn up the heat, start the preparation of a meal, feed the cat, put out the garbage, examine the mail, or check whether someone cuts the grass. An office or factory manager could inspect the work of others or ready a product for shipping. A student could inspect some ancient ruins in a museum, perform an experiment on the ocean floor, shake hands or participate in an experiment or athletic activity with students in another country.

Although the field of *Internet telerobotics* is relatively new and still in its infancy, it has captured the huge interest of many researchers worldwide in the last decade. In 1994 the “Mercury Project” was one of the earliest implementations of telerobotics over the Internet [Goldberg & Gentner et al., 2000], with Australia’s Telerobot [Taylor & Trevelyan, 1995] coming online at almost the same time. Since then, about forty such systems have been put online by research teams around the world. In the Mercury project, a remotely controlled industrial robot arm was used to explore a sandbox filled with buried artifacts (See Figure 1.1(a)). The systems used the HTTP protocol and browser interface. A four-axis IBM robot with camera and air nozzle was set up over a sandbox so that remote viewers could excavate for buried objects by positioning a mouse and clicking from any web browser. Each operation was atomic (self-contained) and the application was designed so that singularities and collisions cannot occur. The system was designed to be operated by nonspecialists and to operate reliably twenty-four hours a day. Telegarden [Goldberg, et al., 1995] replaced the Mercury robot in 1995. The Telegarden system additionally used CAD drawings to animate the state of the manipulator, and allowed the Web users to remotely control an Adept 6 DOF arm to dig and water the plants. Australia's Telerobot on the web [Taylor & Trevelyan, 1995] gives web users the opportunity to build complex structures from toy blocks (See Figure 1.1(b)).

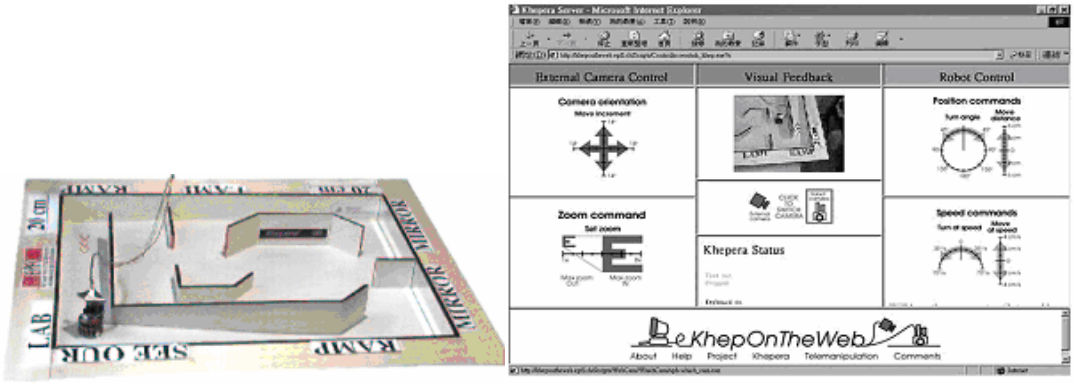


(a) (b)

Figure 1.1: Earliest systems of Internet telerobotics. (a) Mercury robot, camera, and air nozzle above workspace [Goldberg, et al., 2000]. (b) Australia's Telerobot, enables web users to build complex structures from toy blocks [Taylor & Trevelyan, 1995].

The 1st generation of Internet robots is mainly based on robotic arms or simple mobile robots that are directly controlled by human operators. In other words, a human is in the control loop. These online robots operate within a well-structured environment with little uncertainty, and have no local intelligence such as obstacle avoidance. Stein developed an interesting application of an Internet robot: the PumaPaint project [Stein, 2002]. The project is a Web robot that allows any user to control a PUMA 760 robot to paint through the Internet. The robot is equipped with four paintbrushes (red, green, blue, and yellow paint) and two color cameras. Users can select a color and paint on the virtual canvas; the motion will be transformed into sequential commands to the remote robot to apply paint to the real canvas. The Mechanical Gaze system [Paulos & Canny, 1996], developed at Berkeley University, allows remote WWW users to control a robot arm with an attached camera to explore remote objects. Another example is the Bradford Robotic Telescope [Baruch & Cox, 1996]. The WWW users can look at an image taken from an observation with the telescope and compare it with one taken from a star database held at NASA. In The Swiss Federal Institute of Technology, The KhepOnTheWeb [Saucy et al, 2000] system consists of a mobile robot that moves in a wooden maze (see Fig.1.2). The Web users, using clickable images obtained from an onboard camera, can control the robot's movements and orientation. This system was available from May 1997 to May 1998. Although KhepOnTheWeb provides a satisfactory user experience, it has a

major drawback: the direct control of the robot is difficult under important delays without help, so that the system does not scale to real world environments.



(a) (b)

Figure 1.2: The KhepOnTheWeb system. (a) mobile robot with its on-board video camera in a 65×90cm maze; (b) the Web control interface [Saucy et al, 2000].

In contrast, research on the 2nd generation of Internet robots has begun to focus on autonomous mobile robots that navigate in a dynamic and uncertain environment, including the Xavier -- an office exploring robot at CMU [Simmons, et al, 2000], and the museum tour-guide robot RHINO and MINERVA [Thrun, et al, 1999; Schulz et al, 2000]. Xavier (See Figure 1.3) was probably the first mobile robot to operate in a populated office building controlled through the web. Xavier can be advised by web users to move to an office and to tell a "knock-knock" joke after arrival. The robot collects the requests both off-line and on-line and processes them during special working hours. After the successful execution of the mission, Xavier informs the web user via e-mail. Xavier's web interface relies on client-pull and server-push techniques to provide images taken by the robot. Furthermore, it provides a map of the environment and indicates the robot's current position in regular intervals. RHINO and MINERVA (See Figure 1.4) not only can enable Internet users to remote control the robot through the Internet for museum visit, but also can provide a control interface for the local people in the museum. The key features of this generation of Internet robotic projects are their autonomy and reactive behaviours which enable them to navigate and cope with uncertainty in the real world. Supervisory control is the main teleoperation paradigm in building this generation of Internet robots. Unfortunately, this paradigm has a real negative impact on web-based interaction: commanding at a high level is not as interactive as teleoperation using direct control.

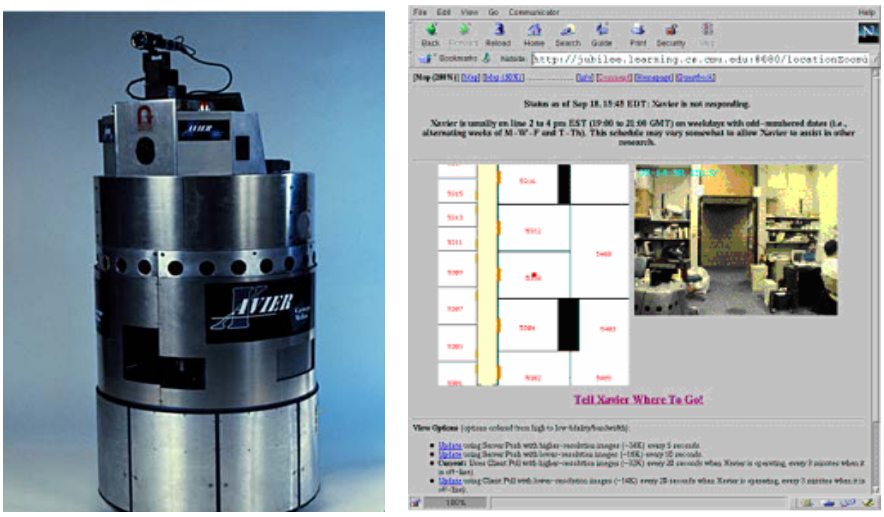
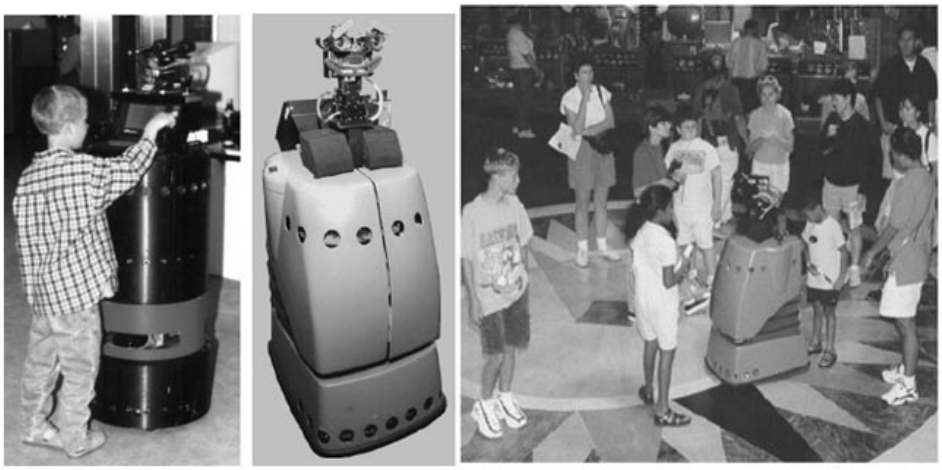


Figure 1.3: Xavier (left) and its web control interface (right) [Simmons, et al, 2000]



(a) (b) (c)

Figure 1.4: Autonomous tour-guide robots (a) RHINO (b) MINERVA (c) MINERVA in the museum [Thrun, et al, 1999; Schulz et al, 2000]

In general, there are four kinds of control architecture for Internet robots: one to one, one to many, many to one, and many to many.

A. *One to One*. This is the common control architecture for most Internet telerobotic systems to provide one user control of one robot (one-one). The examples are Mercury, PumaPaint, KhepOnTheWeb, Xavier, RHINO and MINERVA and so on, which were introduced in the above paragraphs. Another one important example is the NASA's WITS (Web Interface for Telescience), which has been developed to provide Internet-based distributed ground operations for planetary lander and rover missions [Volpe et al., 2000]. The user gets the software through a HTML page and then stops using HTML, as the Java applet is then in charge of accessing the user's

local WITS database and the WITS server. The user generates a sequence of actions locally, using the FIDO simulator to check the results. When finished, the user sends the sequence to the WITS server. It is checked using a sequence integration and verification module and then the full sequence is sent to the rover. The user accesses the data received through the downlink into the remote WITS database. This data includes the robot position and images from the navigation stereo cameras, the panoramic stereo cameras and other sensors.

B. *One to Many*. Some Internet telerobotic systems permit one user control for multiple robots (one-many). As an example, Luo has designed an automatic guided intelligent wheelchair system for hospital automation through the Internet [Luo et al., 1998]. Each mobile robot and the intelligent wheelchairs are individual agents in the hospital automation system. When the human operate orders a command to help one user/wheelchair, the control center starts to broadcast a message to all agents to look for a server agent for completing this task.

C. *Many to One*. Few researchers propose that multiple users control a single robot (many-one). One example is that Goldberg et al. [2000] propose the collaborative teleoperation system. The system allows many users to simultaneously teleoperate an industrial robot arm through the Internet. Their idea is that many people are working together to control a robot, and each user monitors different sensors and submits control inputs based on the different sensor information. Finally, all control inputs must be combined to a single control signal for the robot.

D. *Many to Many*. Several researchers have devoted efforts to the multiple-users-control-multiple-robots system (many-many). For example, Lo and Liu et al [2004] developed a system that enables multiple operators at different sites to cooperatively control multiple robots with real-time force reflecting via the Internet. The operator in China helped the operator in the U.S. to grasp the object by controlling the mobile cameras serve as “mobile eyes” for the operator in the U.S.

Internet-based telerobotics has also attracted interests among researchers in Hong Kong and on the Chinese mainland. The Chinese University of Hong Kong (CUHK), jointly with the universities from the United States, Japan and Chinese mainland, has developed Internet-operated, supermedia-enhanced telerobotic systems that includes the bilateral control of mobile manipulators [Elhajj, et al.,2003; Lo et al, 2004]. CUHK also designed an Internet-based Pulse Palpation system for Chinese medicine

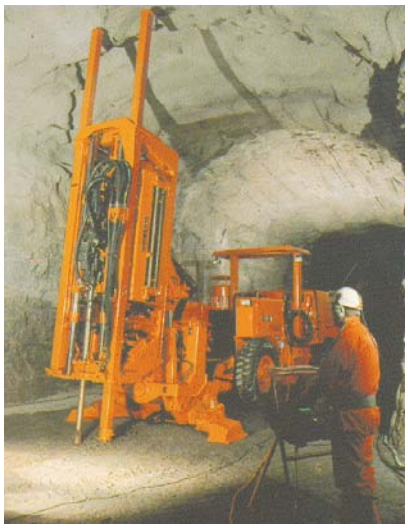
[Xiang, et al., 2002]. The City University of Hong Kong has exploited a human-robot interface that uses agent communication with an XML-based markup language [Makatchev, et al., 2000], as well as investigating dynamic Internet performance and establishing an Internet-based control transmission model. On the mainland, Tsinghua University (TU) has combined an event-based direct control method with a graphic predictive simulation to achieve an Internet-based multi-operator dual-arm teleoperation system [Jinshi Cui, et al., 2002] as well as a robot arm that can be teleoperated through the Internet to write Chinese characters. The Harbin Engineering University (HEU) investigated the round trip delay (RTT) of Internet-based teleoperation [Ye, et al., 2002] and a UDP-based protocol for data transmission [Liu et al., 2002].

Apart from for operating in hazardous environments that are traditional telerobotic areas, Internet telerobotics has opened up a new range of real-world applications, involving tele-manufacturing, tele-training, tele-surgery, museum guide, space exploration, disaster rescue, house cleaning, and health care.

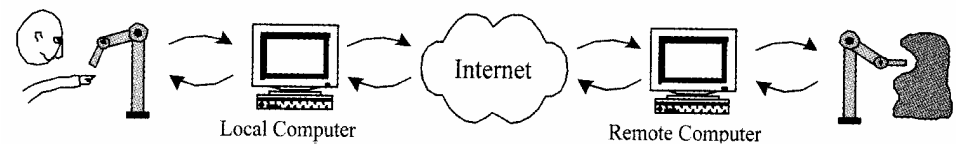
On 17th Sep. 2002, the “Pyramid Rover” robot entered the queen’s tomb in the ancient Egypt pyramid to explore beyond a long-unopened door. This robot was controlled by the traditional direct control of an operator through the reliable cable connection. Worldwide, people watched the event via a live satellite television broadcast. However, if you explore the pyramids by yourself over the Internet, how is your feeling!

1.1.2 Research problems on Internet telerobotics

Internet-based robot teleoperation obviates the need for dedicated networks, devices, and operators, reduces costs, extends operating distances, allows precious resources sharing for public education or academic research, and is accessible from any node on the Internet. Although the Internet provides a cheap and readily available communication channel for teleoperation, there are still many problems that need to be addressed. Figure 1.5(a) shows one example of *traditional teleoperation*. Figure 1.5(b) shows one example of *Internet-based teleoperation*.



(a)



(b)

Figure 1.5: (a) Traditional teleoperation. The human operator has most of the time straight visual contact to the controlled target. Control commands are sent electronically through wire or radio; (b) Internet-based teleoperation. The human operator holds a haptic device attached to the local computer, a robot controlled by the remote computer, and both computers communicating via the Internet.

Internet-based teleoperation differs from traditional teleoperation on several aspects. These differences are also research problems on Internet telerobotics as follows.

- There is much latency on the Internet: restricted bandwidth, uncertain time delay, packet lost, and data error, which is unlike traditional teleoperation where the interfaces have fixed delays and guaranteed services.
- Internet telerobotics must ensure safe operations even if communication breaks down. With communication as unpredictable as it is on the Internet, online robots require a much higher degree of autonomy than traditional teleoperation.
- Internet robots require local intelligence (e.g. obstacle avoidance, path planning, map learning, objective recognition, etc.) to sense the exploring environments and to deal with uncertainties derived from both real world and robots themselves.

Human operators provide such intelligence in traditional teleoperation.

- Internet users require high-quality feedback from remote robots in order to obtain satisfactory experience for virtual tele-presence. Traditional teleoperation provides human operator direct feedback on the spot.
- Internet telerobotics requires a mechanism to provide human operators reliable hands-on control and other high level commands in order to obtain more interactive experience. This is easy for traditional teleoperation without the Internet latency.
- Internet telerobotics requires an intuitive and easy-to-use teleoperation interface because Internet robots are typically remotely controlled by many people with little expertise and few skills. In contrast, traditional teleoperation are handled by trained operators.

In addition, we compare Internet-based teleoperated robots with autonomous and interactive robots. The key difference is the communication between human operators and robots. The latter can real-time communicate with humans while humans are able to easily know robot's current surroundings and working status. The former is more difficult because the Internet leads to uncertain and unreliable information transmission between human operators and teleoperated robots.

1.2 Research objective and outline

The research objective of the thesis is to develop a practical telerobotic system for the target application: inexperienced Internet users can remotely control a mobile robot to perform some complex tasks autonomously (e.g. active map learning, goal-oriented navigation) or to interact with human operators in order to explore unknown and dynamic environments. To help realize such robotic system, we mainly do research on the following aspects:

1) To investigate and implement the video transmission via the low-bandwidth Internet so that the robot's surroundings can be seen by any remote operators through the images captured from an onboard camera.

2) To develop a new teleoperation approach that can provide interactive control interface so that inexperienced operators have better robot teleoperation experiences.

3) To develop and realize some high-level control commands for the use of Internet-based robot teleoperation to navigate the mobile robot that explores

unknown environments.

4) To develop a map learning approach for autonomous robot to actively explore unknown indoor environments and build a map based on robot's sensory information.

5) To develop a new behavior-based navigation method for mobile robot to autonomously search the path and finally arrive at a given goal within unknown and dynamic indoor environments which involve local minima (i.e. dead ends).

There are three phases for the above research. Figure 1.7 shows the design flow diagram of our research.

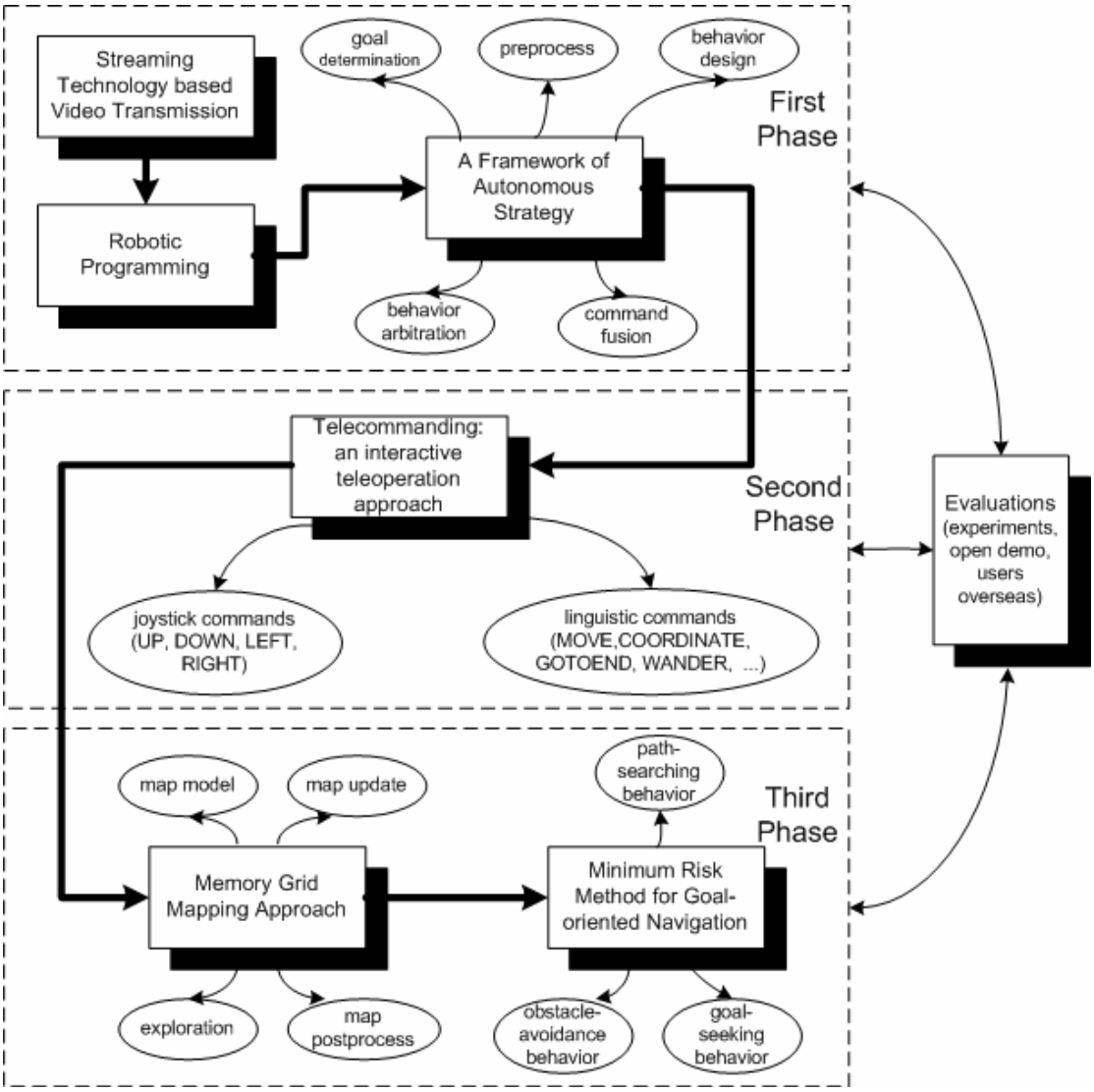


Figure 1.6: The design flow of research outline.

In the first phase, we do the investigations of image transmission over the Internet at first, and implement it based on the streaming technology. This work is a prerequisite to develop a practical Internet-based teleoperation system so that any

authorized users can see the remote robot's surroundings through the images captured from an onboard camera. The robotic programming (see Appendix A) is not an important research work, but it is indeed a basic programming work for subsequent robotic development. Next a framework for autonomous robot navigation is investigated. The framework includes the steps of goal determination, preprocessing, behavior design, behavior arbitration, and command fusion. This work is a base for the subsequent designs of intelligent control programs so that the mobile robot is able to autonomously perform some complex tasks in spite of the uncertainties derived from real world and robot itself.

In the second phase, we implement an interactive teleoperation interface through the Internet. A novel teleoperation approach so called *telecommanding* is proposed. Telecommanding involves two different but complementary commands: joystick and linguistic commands. Each joystick or linguistic command is defined with multiple events (non-time action references) and the corresponding response functions. In this phase, we define and realize four joystick commands (UP, DOWN, LEFT, and RIHGT) and five linguistic commands (MOVE, COORDINATE, TURN, GOTOEND, and WANDER). Another one linguistic command (MAPPING) is realized in the third phase.

In the third phase, we propose a new map learning approach called *memory grid mapping* (i.e. MAPPING linguistic command) to model a priori unknown indoor environments. The approach includes a map model, a map update method, an exploration (i.e. online path planning) method, and a map postprocessing method. Finally, we propose a new behavior-based navigation method called *minimum risk method* to realize an enhanced COORDINATE linguistic command. The method is an application of the proposed memory grid map. It is developed to give global convergence to a given goal in different indoor environments, including long-wall, large concave, recursive U-shape, unstructured, cluttered, maze-like, and dynamic environments.

For each phase, we perform evaluations by specified experiments. In addition, public demos and real teleoperation of remote users overseas are performed throughout the research period. Another encouraging observation is that we provide a website about our telerobotic research during the second phase. From the statistics of website visitors, we are able to know the impact of our research to related academic

researchers.

The experiments of the thesis are based on a mobile robot (vehicle) shown in Figure 1.6. The robot uses a multifunctional Hitachi H8S-based microcontroller, and has a 44cm x 38cm x 22cm aluminum body and a ring of eight forward sonars. The control commands are transferred through radio Ethernet devices, and the video/audio data is fed back through a set of 2.4GHz frequency A/V transmitter-receivers from a pan-tilt-zoom camera mounted on the robot deck.

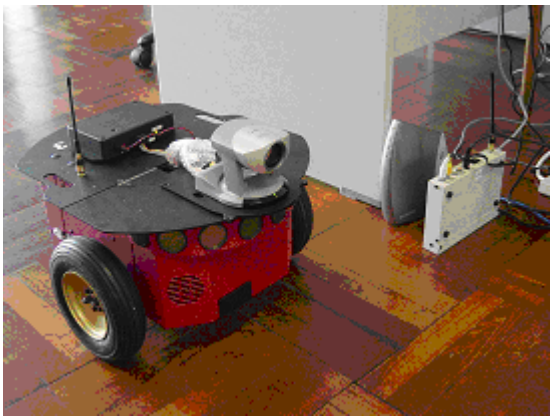


Figure 1.7: The Pioneer robot and its accessories. The robot has eight forward ultrasonic sensors and an onboard pan-tilt-zoom camera.

1.3 Organization of the thesis

This chapter introduces the background and research problems of Internet telerobotics, proposes the research objective and outline, and states the main contributions of the thesis.

Chapter 2 reviews the related literatures. Section 2.1 introduces a time-delay model of Internet-based teleoperation as well as investigations about round-trip time and packet lost rate of data transmission via the Internet. Section 2.2 introduces the existing teleoperation paradigm: direct control and supervisory control. Section 2.3 describes two approaches for autonomous robot navigation: SMPA and behavior-based approach. Section 2.4 introduces the related approaches for real-time map building and exploration, and Section 2.5 the related approaches for goal-oriented navigation in known and unknown environments.

Chapter 3 implements a streaming technology based approach for video transmission. Two video compression algorithms (WMV9 and MPEG4) under

different bandwidth, two video encoding methods (CBR and Quality-based VBR) as well as the transmission stability and time delay have been investigated. A test of real robot teleoperation using direct control via a 33.6Kbps (modem) Internet connection has been done successfully. Finally we compare the performances of different approaches for image transmission.

Chapter 4 proposes a framework of autonomous navigation using fuzzy logic. A goal-oriented navigation task, combining with obstacle-avoidance and goal-seeking behaviors, is implemented and tested as an example of the proposed framework. Finally we discuss pros and cons of the use of fuzzy logic controller as well as machine learning technique.

Chapter 5 proposes a new teleoperation approach so called telecommanding. Experiments have been done to test the use of both joystick commands and linguistic commands for Internet-based simulated and real robot teleoperation. The advantages and disadvantages as well as stability of telecommanding are analyzed. The comparisons with direct control and supervisory controls are made as well.

Chapter 6 proposes a new map learning approach to model a priori unknown indoor environment. The efficiency of map learning is investigated. The map accuracy under different cell sizes and different map postprocessing is investigated as well. Experiments are done for the map learning in different simulation environments.

Chapter 7 proposes a new navigation method to navigate the robot to a given goal within an unknown environment with local minima. Performances of the proposed method in long-wall, large concave, recursive U-shape, unstructured, cluttered, maze-like, and dynamic indoor environments are experimented. A detailed comparison with both boundary-following and virtual-subgoal approaches is made.

Chapter 8 evaluates the research results of the thesis. Public demos and teleoperation of authorized users overseas verify the developed telerobotic system. The advantages and limitations of the research are discussed. In addition, we provide an interesting statistics of our website which has been built to introduce our telerobotic system. The results of this statistics are analyzed to show the impact of our research.

Chapter 9 concludes the thesis and suggests possible future researches.

Appendix A describes the related robotic programming. Appendix B shows one newspaper and two magazines, which reported our telerobotic system to the public in

Hong Kong. Appendix C gives the snapshots of our website. Appendix D introduces the emerging streaming technologies for media transmission through the Internet. Appendix E gives a brief introduce of related fuzzy system theory that would be used in this thesis.

1.4 List of contributions

This section states the contributions of the work in the thesis.

- Chapter 5 proposes a new teleoperation approach so called *telecommanding* in order to provide an interactive control interface and a complete framework for control management and command processing. This work is one of our major contributions. Telecommanding involves two different but complementary commands: joystick and linguistic commands. It gives more experience of interactivity and functionality compared with the existing direct control and supervisory control methods. Under the framework of telecommanding, we extend our system by realizing more linguistic commands.
- Chapter 6 proposes a new map learning approach called *memory grid mapping* in order to model a priori unknown indoor environment. This work is one of our major contributions. The approach includes a map model, a map update method, an exploration method, and a map postprocessing method. The work has addressed an important topic in robotics, and has contributed some useful ideas such as simple map model, exploration method and map postprocessing method.
- Chapter 7 proposes a new behavior-based navigation method called *minimum risk method* in order to address local minimum problem faced by goal-oriented robot navigating in unknown indoor environments. This work is another major contribution. The method is experimentally demonstrated to give global convergence to a given goal location, even though it is used in the long-wall, large concave, recursive U-shape, unstructured, cluttered, maze-like, or dynamic (i.e. with moving human) environments. Compared with the existing boundary-following or virtual-subgoal approach, the proposed method can deal with more complex environments and is able to find the nearest exit to escape from local minimum.
- In addition, the developed prototype system for Internet-based teleoperation turns

out to be practical and be feasible to provide the service at university campus or exhibition center.

- Chapter 3 presents a streaming technology based approach for streaming video feedback from remote robots. This work is a less important part. But the work is beneficial for the researchers in the field of Internet telerobotics to adopt similar techniques in order to improve image transmission and make the Internet-based teleoperation usable.
- Chapter 4 proposes a framework for autonomous robot navigation using fuzzy logic. This framework involves goal determination, preprocessing, behavior design, behavior arbitration, and command fusion. The work in this chapter focuses on the development of a simple and practical navigation framework that is useful for easy realization of building robust control programs. Although this work is a less important part, it is a base for the subsequent research to implement some complex tasks.

CHAPTER 2. LITERATURE REVIEW

2.1 Time delay and data lost of the Internet

Different from traditional teleoperation systems using private transmission media, Internet telerobotics uses the Internet, which is a public transmission media on which unknown numbers of end users share the bandwidth concurrently. Internet robots encounter the uncertain transmitting time-delay and data-loss problems, which always makes the remote control becoming unstable or failing. A diagram of typical Internet-based teleoperation is drawn in Figure 2.1 [Luo and Chen, 2000]. The total time of performing a teleoperation per cycle is *t1 + t2+ t3 + t4*, where the four types of time delay are:

1) *t1*: time delay of transmitting the remote information (e.g., images, sensory data, robot's status data) from the robot to the operator;

2) *t2*: time delay of making control decision by the operator;

3) *t3*: time delay of transmitting a command from the operator side to the robot;

4) *t4*: execution time of the robot to perform a command.

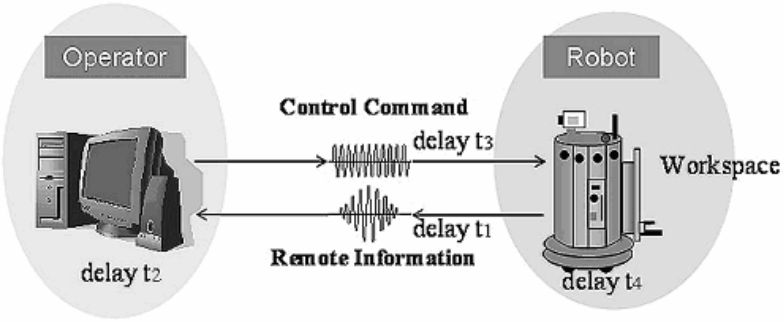


Figure 2.1: The diagram of a typical Internet-based teleoperation. [Luo and Chen, 2000]

Assume "*m*" is the degree of robot's autonomy, the higher the "*m*" degree representing the higher degree of autonomy (i.e., to simplify the problem, operator sends one command and the robot performs "*m*" nonredundant actions to complete it), and *m=1* representing that the robot has no autonomy (i.e., one command and one primitive action). If we assume that each of the four delays is always a constant, and

the desired task requires the robot to perform "*n*" primitive actions (i.e., complexity is *n*) to complete it, the total time spent for completing a task is *(n/m)*(t1+t2+t3)+n*t4*. As a result, the task completion time is inversely proportional to *m*.

Unfortunately, communication through the Internet *t1* and *t3* are usually unpredictable. The latency of the Internet usually contains the uncertain round trip delay and the data loss rate. Luo and Chen [2000] have repeatedly tested the transmitting efficiency of the network by sending 64 bytes data every time from their Web server in laboratory to different remote Web servers. The resulting statistics of round-trip time and data-lost rate are shown in Table 2.1, where *Min.* represents the minimum round trip delay, *Max.* is the maximum, and *Avg.* is the average delay of total tests. It can be seen that the latency of the Internet not only contains the serious and uncertain round-trip delays but also the data-loss rate.

In the TCP/IP protocol, once the data is lost, the remote site will require a retransmission. This leads to a longer delay of total transmission time. Assume the data-lost rate is " *p*" and the average round-trip delay is "*R*" s; the expected time of transmitting a control command with 64 bytes can be roughly estimated by $R/2*(1+p+p^2+p^3+\ldots) = R/(2*(1-p))$ second. In a local area network (LAN) this value (several ms) is small, but for transmission across the Internet, it cannot be guaranteed. Teleoperation of a "puppet like" robot via the high latency Internet is not suitable, but most of the existing systems do this. The long transmission delay may result in the failure of remote controls in a complex task or, more seriously, endanger the robot and its workspace.

TABLE 2.1: Round-trip time and data lost rate of transmitting data between internal Web server and remote others. [Luo and Chen, 2000]

Web Address	**Min. (ms)**	**Avg.(ms)**	**Max.(ms)**	**Loss Rate**
www.ccu.edu.tw (South R.O.C)	1	4	20	< 1%
www.ncku.edu.tw (South R.O.C)	3	4	19	2%
www.ntu.edu.tw (South R.O.C)	11	17	50	20%
www.ncsu.edu (NCSU)	331	375	1616	46%
www.cmu.edu (CMU)	336	358	1461	50%
www.ynu.ac.jp (Japan)	440	493	2576	69%
www.cam.ac.uk (UK)	436	772	4468	51%
www.fu-berlin.de (Germany)	446	860	5505	42%

2.2 Teleoperation paradigm

In general, the teleoperation paradigms of Internet telerobotics can be divided into two types: direct control, supervisory control. Most of Internet-based teleoperation systems are basic extensions of these two paradigms.

A. Direct Control

In the direct control paradigm, the human operator can control the mobile robot directly by sending the primitive commands (e.g. force or velocity commands) and necessary parameters continuously through the Internet. The robot will execute the commands without any intelligence, and it maintains continuous connection with the remote controller. Direct control has obvious drawbacks such as reduced stability of the control loops due to uncertain long delay of the Internet. To alleviate the problems derived from the Internet latency, three main approaches are developed.

1) Predictive aiding approach. With time delay, received remote information may be invalid to represent the current remote situation. The predictive aiding approach is developed to extrapolate forward environmental information and manipulator states in time by stochastic predictors for displaying on the operator's monitor. [Kikuchi, et al., 1999; Schulz, et al., 1998].

2) Simulating and planning display approach. This approach is developed to use local simulated manipulator in order to assist the human operator to control the remote robot more intuitively. The operator can control the simulated manipulator directly, and the computer stores the sample state-command pairs in the memory buffer. When the operator has finished a task by a local simulated device, the queued data will be sent to the actual manipulator to execute. The time and position clutching method [Conway, et al., 1990] is such an example of this approach.

3) Event-based approach. The general idea of non-time based control is to model the system and the trajectory as functions of a non-time based variable, which is called motion reference or action reference. It is also usually denoted as and called the event-based action reference. The stability of teleoperation systems with non-time based motion reference is guaranteed if their local robot controllers are stable and the non-time based motion reference is a non-decreasing function of time. Such non-time based reference is usually related directly to real time sensor measurements or the task. The advantage of this approach, which differentiates it from the other

approaches in the literature, is that stability is proven independently of the specific human operator or the statistics of time delay [Xi & Tarn, 2000].

B. Supervisory Control

In supervisory control paradigm, the remote robot operates in a large autonomous mode and only interacts with the human operator until the robot encounters a situation it cannot handle. The robot requires only specifying its new desired destination or state. Therefore, there is no need for high speed continuous communication. Because of the latency of the Internet and the requirement for safety of a mobile robot, the supervisory control is essential for the Internet application. Many researchers establish the local intelligence of mobile robots, such as collision avoidance, path planning, self-referencing, object recognition, and so on. The RHINO and MINERVA [Thrun, et al., 1999] tour guide robots are operated at this level. Internet users can control the robots to visit an exhibition position via the Web by clicking the marked position on the map. Therefore, the communication content from the user to the robot only consists of the goal command, and the sensory information of the remote environment is not really necessary when the robot is executing the task.

2.3 Autonomous robot navigation

In general, robots can be categorized as two types: mobile manipulators with haptic feedback and mobile vehicles for navigation. This thesis only addresses the Internet-based teleoperation of wheeled robot for navigation. Although human intelligence is important in robot teleoperation systems, it is essential for local robot to have autonomous capabilities to handle unexpected events and dynamic environmental changes.

The goal of autonomous mobile robotics is to build physical systems that can move purposefully and without human intervention in real world. On the one hand, traditional robots lack the ability to provide flexibility and autonomy: typically, perform preprogrammed sequences of operations in highly constrained environments, and are not able to operate in new environments or to face unexpected situations. On the other hand, there is a clear emerging market for truly autonomous robots. Possible applications include intelligent service robots for offices, hospitals, and factory floors; maintenance robots operating in hazardous or inaccessible areas; domestic robots for

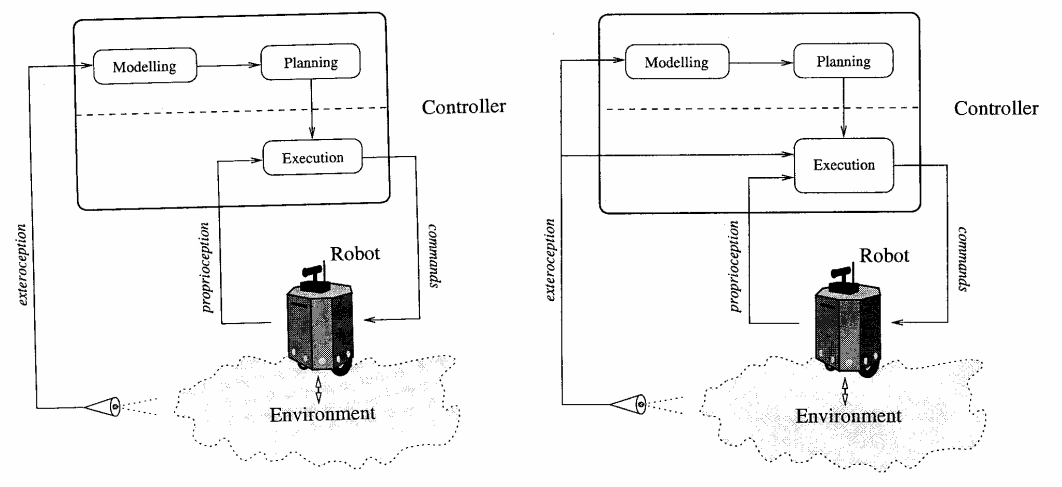
cleaning or entertainment; autonomous and semi-autonomous vehicles for help to the disabled and the elder and so on.

Any approach to control a dynamic system needs to use some knowledge, or model, of the system to be controlled. In the case of a robot, this system consists of the robot itself plus the environment in which it operates. Unfortunately, while a model of the robot on its own can normally be obtained, these environments are characterized by the ubiquitous presence of uncertainty, and we are often not able to precisely model or quantify this uncertainty. First, the uncertainty induced by the presence of people. People move around, and they may change the position of objects. Additionally, results of the robot's movement and sensing actions are influenced by a number of environmental conditions, which are hardly accounted for. For example, the error in the robot's motion may change as a result of a wet floor; and the reliability of distance measured by a sonar sensor is influenced by the geometry and the reflectance properties of the objects in the environment.

A common strategy to cope with this large amount of uncertainty is to abandon the idea of completely modeling the environment at the design phase, and to endow the robot with the capability of building this model by itself on-line. This strategy leads to the so-called SMPA(Sense-Model-Plan-Act) approach [Saffiotti, 2000] (see Figure 2.2(a)). The robot uses exteroceptive sensors, like a camera or a sonar sensor, to observe the state of the environment; it uses proprioceptive sensors, like a compass or shaft encoders on the wheels, to monitor the state of its own body parts. By using the exteroceptive sensors, the robot acquires a model of the workspace as it is during the moment when the task must be performed. From this model, a planning program builds a plan that will perform the given task in the given environment. This plan is then passed to a lower-level control program for execution.

But there are a number of problems using the SMPA approach to deal with real-world environments. The model acquired by the robot is incomplete and inexact, due to the uncertainty in perception. Moreover, this model is likely to rapidly become out of date in a dynamic environment, and the plan built from this model will then turn out to be inadequate for the environment actually encountered during execution. The fact that the modeling and planning processes are usually computationally complex and time consuming exacerbates this problem, because the feedback loop with the environment must pass through all these processes "Sense-Model-Plan-Act" [Cang

Ye, et al., 2000].



(a) (b)

Figure 2.2: (a) SMPA approach architecture; (b) Behavior-based approach architecture. The lower layer uses perception to dynamically adapt plan execution to the environmental contingencies. The execution module must simultaneously consider demands coming from the plan and from the environment.

A modern approach, so-called behavior-based approach [Arkin, 1998], is shown in Figure 2.2(b). The general feeling is that planning should make as few assumptions as possible about the environment actually encountered during execution; and that execution should be sensitive to the environment, and adapt to the contingencies encountered. To achieve this, perceptual data has to be included into the executive layer. This apparently simple extension has two important consequences. First, it makes robot's interaction with the environment much tighter, since the environment is now included in a closed-loop with the (usually fast) execution layer. Second, the complexity of the execution layer has to be greatly increased, since this needs now to consider multiple objectives: pursuing the tactical goals coming from the planner, and reacting to the environmental events detected by perception.

In behavior-based approach, each behavior fully implements a control policy for one specific sub-task, like following a path, avoiding sensed obstacles, or crossing a door way. Simple behaviors are combined in order to produce a complex strategy able to pursue the strategic goals of the agent, while effectively reacting to contingencies.

Fuzzy logic controllers provide a means of transforming linguistic control strategy based on expert knowledge into an automatic control strategy. It appears to be very useful for handling problems that are too complex to be analyzed by conventional quantitative techniques or when the available sources of information provide qualitative, approximate, or uncertain data. Reactive navigation of a mobile robot falls into this class of problems that fuzzy control system copes well.

2.4 Map building and exploration

To efficiently carry out complex missions in indoor environments, autonomous mobile robots must be able to acquire and maintain models of their environments. The problem of acquiring models is difficult and far from being solved. The following impose practical limitations on a robot's ability to learn and use accurate models.

1) *Sensors*. Sensors often are not capable of directly measuring the quantity of interest. For example, ultrasonic sensors measure the distance to obstacles, whereas for navigation one might be interested in assertions such as "there is a door in front of the robot".
2) *Perceptual limitations*. The perceptual range of most sensors (e.g. ultrasonic sonars, cameras) is limited to a small range around the robot. To acquire global information, the robot has to actively explore its environment.
3) *Sensor noise*. Sensor measurements are typically corrupted by noise. Often, the distribution of this noise is not known.
4) *Drift/slippage*. Robot motion is inaccurate since odometric errors accumulate over time. For example, even the smallest rotational errors can have huge effects on subsequent translational errors when estimating the robot's position.
5) *Complexity and dynamics*. Robot environments are complex and dynamic, making it impossible to maintain exact models and to give prediction accurately.
6) *Real-time requirements*. Time requirements often demand that internal models must be simple and easily accessible. For example, accurate fine-grain CAD models of complex indoor environments are often inappropriate if actions have to be generated in real-time.

There are two major representations for mapping indoor environments [Victorino et al, 2003; Meyer & Filliat, 2003]: topological and grid-based. Topology maps permit efficient path planning and have low space complexity, but it is often difficult

to learn and maintain accurate and consistent topology maps in large-scale environments, particularly if sensor information is ambiguous [Thrun, 1998]. Grid-based maps have the disadvantage of being space-consuming, but they can tolerate uncertainties in sensory data and are easier to build and maintain, providing more opportunities to satisfy the requirements of real-time path planning and execution. The grid-based map model represents the robot's work area by a two-dimensional array of square elements denoted as cells. Each cell contains a certainty value to measure the confidence that an obstacle exists within the cell area. Certainty values are updated by a function that takes into account the characteristics of the sensors.

In robotics, there are several different grid-based representations to be used to represent the environment. The main difference among them is the function used to update the cells, for example: probability [Thrun, 1998b; Yamauchi et al., 1998; Wallner & Dillmann, 1994; Dieguez et al., 2003; Song & Chang, 1999; etc.], fuzzy possibility [Oriolo et al., 1998], frequency [Borenstein & Koren, 1991; Edson et al. 2004] and so on.

Probability values are commonly used in grid-based maps. The first grid-based method to use probability values to measure the spatial uncertainty generated by sonar sensors was Occupancy grid [Moravec & Elfes, 1985; Elfes, 1987; Moravec, 1988]. Thrun [1998] used an occupancy-grid framework to implement an incremental mapping scheme. The probability of each cell being occupied is updated using the Bayes rule. This probability is computed using a neural network that has been trained by back-propagation in a known environment. Thrun makes the additional hypothesis that walls are orthogonal. Such a hypothesis limits the estimation error in the robot's direction to values that permit local map-matching and efficient correction of the robot's position estimate. Thrun also resorts to an exploration scheme that allows the robot to drive towards unexplored areas. Yamauchi et al. [1998] provide a similar scheme but without using the orthogonal walls assumption. Their computation of occupancy probabilities is based on the combination of laser-scans and sonar-sensor values. This combination is designed to simultaneously avoid the use of spurious measurements from the sonar-sensors, and to filter too high laser-scan values that arise when the laser ray is targeted above the obstacles. The exploration is directed toward the closest frontier between explored and unexplored areas. Wallner and Dillmann [1994] construct local certainty grids around new detected obstacles. The

methods allow the combination of a parametric description of known obstacles with grid-based mapping. Grid probabilities result from the information obtained from ultrasonic range sensors and an active stereo-vision system. This approach combines cyclic path replanning and grid refinement.

Oriolo et al. [1997; 1998] proposed a grid-based map that was defined as the fuzzy set of unsafe cells whose membership function quantifies the possibility for each cell occupied by some obstacles. Fuzzy set operators are used to process ultrasonic sensor data, producing a grey-level bitmap that provides risk information for each cell. On a fuzzy map, A*-based path planning is performed by searching for optimal paths from the current robot location to the desired goal.

Koren & Borenstein [1991] used frequency values to indicate the measurement of confidence that a cell is occupied by an obstacle. Their histogramic in-motion mapping approach uses a very simple metric sonar model that assumes that a single point in the sonar's direction is detected at the distance measured by the sonar. The frequency value of the cell containing at that point is simply increased, while the frequency values of the cells between the robot and that point are accordingly decreased by a smaller value. Edson et al. [2004] adopted a similar scheme. This approach has the advantage of highly efficient computation.

2.5 Goal-oriented navigation in unknown environment with local minimum

The goal-oriented autonomous navigation is a robot task that is commonly required in Internet-based teleoperation systems such as the office-exploring robot Xavier and museum tour-guide robots RHINO and MINERVA. This task calls for a robot to be given a goal position and for the robot then to arrive at the goal autonomously while to avoid any static or dynamic obstacles in its path. One suggested solution is to use an approach that combines both global path planning and path tracking [Huh et al, 2002; Ryu & Yang, 1999; Meyer & Filliat, 2003]. This approach guarantees global convergence to the goal. We call this scheme "*heuristic goal-oriented navigation*". The key precondition of heuristic goal-oriented navigation is to obtain the requisite environmental knowledge in advance.

Unfortunately, the characteristics of real world applications have created a

number of difficulties in applying an approach that uses heuristic goal-oriented navigation. First, in general, prior knowledge about an environment may be incomplete, uncertain, imprecise, and perhaps even entirely unavailable. Second, the dynamics of real-world environments are typically complex and unpredictable. A third difficulty is created by the fact that robot tasks (e.g. Mars exploration) are often real-time and non-repetitive.

An alternative scheme is one that we call "*myopic goal-oriented navigation*". By "myopic" we mean that the robot is moving in an environment but without prior knowledge about it. The popular control strategy for autonomous navigation, an advance on the early SMPA (Sense-Model-Plan-Act) approach, takes a so-called behavior-based approach [Arkin, 1998]. Local path-planning behaviors use local sensory information in a largely reactive fashion. They are much simpler to implement since they typically map the sensor readings directly to actions. Specific examples include potential-field methods [Tsourveloudis et al., 2001] and neural-fuzzy approaches [Rusu et al., 2003; Godjevac and Steele, 2000]. None of these examples, however, guarantee global convergence to the goal because they are susceptible to get trapped in local minima (or dead ends) of the environments.

In the literatures [Maaref & Barret, 2002; etc.], the local minimum problem, also called the deadlock, dead end or limit-cycle problem, has been addressed using what we categorize as two types of approach: the boundary-following approach, and the virtual subgoal approach. Boundary-following approaches [Huang & Lee, 1992; Kamon & Rivlin, 1997; Lim & Cho, 1998; Krishna & Kalra, 2001; Maaref & Barret, 2002; Chatterjee & Matsuno, 2001] have a common control structure. Initially the robot moves directly toward the goal using a normal navigation module. When the robot judges that the context is satisfying a detection criterion (e.g. an obstacle is hit), it follows the obstacle boundary until an escape criterion is satisfied. In order to detect and escape from the local minimum, boundary-following approaches flexibly change the navigation module by judging the detection and escape criterion. Virtual subgoal approaches [Pin & Bender, 1999; Xu, 2000; Xu & Tso, 1999] have only one navigation module. When a detection criterion is satisfied, a new subgoal is set to guide the robot in escaping from the local minimum. When an escape criterion is satisfied, the original goal is recovered.

CHAPTER 3. VIDEO TRANSMISSION USING A STREAMING TECHNOLOGY BASED APPROACH

This chapter investigates and implements video transmission through the Internet from the robot server to user clients. This work is a prerequisite to develop a practical Internet-based teleoperation system so that any authorized Internet users any where are able to see the remote robot's surroundings through the images captured from an onboard camera. It is desired that the client users under different Internet bandwidth can receive stable and continuous video images with high resolution. Emerging streaming technologies (e.g. MPEG4, RTP, MMS) make it possible to transfer multimedia perception information with good quality of service (QoS) through the Internet.

3.1 Introduction

The most intuitive and informative way to obtain remote robot's surroundings and improve the user experience of virtual tele-presence is via vision feedback. Researchers have approached this problem in a variety of ways. Early researchers used a picture transmission scheme (e.g. JPEG or GIF) or hybrid image and virtual reality [Goldberg et al, 2000; Simmons et al, 2000; Thrun et al, 1999; Schulz et al, 2000; etc.]. The drawback of picture transmission is the very low frame rate and large time delay (over 10-20s). A more serious problem is that Internet performance degrades, such as reductions in bandwidth, may cause service-stop errors. Researchers [Barbera et al, 2001; Safaric et al, 2003] have now begun to use video conferencing systems instead of picture transmissions, but the crucial video coding algorithms of these systems are obsolete (e.g. H.261, H.263). The best current candidates for transferring multimedia perception information with the best quality of service (QoS) through the Internet are emerging streaming technologies such as MPEG4, RTP, and MMS [Mack, 2002].

The rest of this chapter is organized as follows. Section 3.2 proposes and implements an approach that uses the emerging streaming technology for video transmission. Section 3.3 shows the experimental results, involving the real robot

teleoperation over a low-bandwidth Internet connection. Section 3.4 makes a comparison with some techniques used by other telerobotic systems. The final section summarizes the chapter.

3.2 A streaming technology based approach

This section proposes a streaming technology based approach for Internet-based robot teleoperation as shown in Figure 3.1. Internet users (clients) remotely control a robot in response to live streaming video captured by the camera mounted on the robot. The robot server connects the robot and camera over a wireless channel, obviating the problems associated with cables. The streaming server captures and encodes the real-time video from the camera on the robot under the instructions of the robot server. The compressed video images are streamed to transfer to the master client and slave clients. The service of robot server and streaming server can be distributed from the same computer to the Internet.

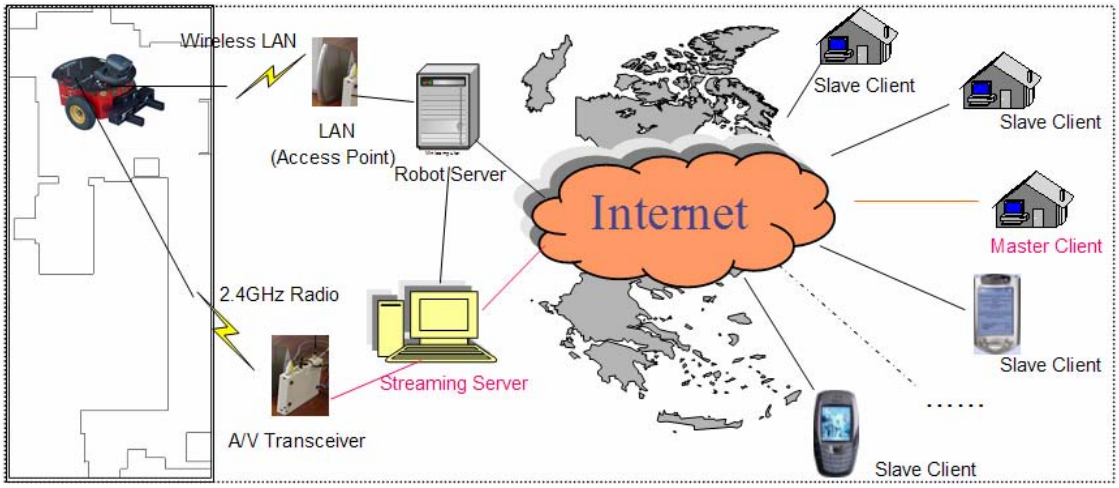


Figure 3.1: Internet-based robot teleoperation using streaming technology for video transmission.

Only one master client dominates the full control privilege to interact with the robot server through the Internet. The robot server interprets and activates the intelligent robot navigation algorithms, as well as the low-level motion control of the robot via the wireless channel. The remote control includes the pan-tilt-zoom commands of the camera on the robot. The other slave clients can simultaneously watch the streaming video using the streaming player, but have no control privilege unless the master client hands over his privilege to another one client.

Two stream transmission schemes can be adopted: "*push*" and "*pull*". In a push scheme, the streaming server actively pushes the encoded stream media to the clients. If the clients do not work, this scheme has the potential to consume lots of network resources. As a result, we have adopted the pull scheme where the streaming server listens to a predefined port, and transfers the stream video after it receives a request from the clients.

There are two video encoding methods that can be applied to a live broadcast, *Constant Bit Rate (CBR) encoding* and *Quality-based Variable Bit Rate (VBR)* encoding. CBR encoding allows us to specify the average bit rate that we want to maintain and to then set the size of the buffer. The bit rate will fluctuate across the stream, but the fluctuations are constrained by the buffer size. Quality-based VBR allows us to specify a desired quality level (from 0 to 100), then during encoding the bit rate fluctuates according to the complexity of the stream. A higher bit rate is used for intense detail or high motion, and a lower bit rate is used for simple content. We compare the two encoding methods in the experiment, and choose the CBR encoding method.

Microsoft provides a complete series of software development kits (SDKs), the Windows Media Series SDK [*http://www.microsoft.com/windows/windowsmedia/*]. The SDK helps researchers to develop their streaming applications based on Windows Media Series 9. Using the SDK, we have implemented a prototype system.

3.3 Experimental results

This section reports the experiments. Section 3.3.1 investigates the compression performance of two video codec under different network bandwidth. Section 3.3.2 investigates the transmission performance of two video encoding methods under different network bandwidth. Section 3.3.3 investigates the transmission stability and time delay. Section 3.3.4 shows a real robot teleoperation through a low-bandwidth Internet connection via a telephone line.

3.3.1 Compression performance of two video codec

We investigate the compression performance of two video codec WMV9 and MPEG4. WMV9 denotes Windows Media Video 9 which is involved in windows media encoder V9 [*http://www.microsoft.com/windows/windowsmedia/*]. MPEG4 implies

ISO MPEG4 video codec [ISO, 2002], which is implemented using QuickTime Player Pro [*http://www.apple.com/mpeg4/*]. We have used these two video codec to compress a 19-second video clip, operating under different network bandwidths. Table 3.1 gives the compression results. In Table 3.1, we categorize the potential audiences into five types with capacities ranging from a 28kbps dial-up modem to a 150kbps LAN or DSL audience. Note that the actual stream media should be lower than the theoretical network bandwidth. For example, in order to ensure stable performance, a 50kbps media stream is provided for a 64kbps Single ISDN audience.

*TABLE 3.1: Network Bandwidth versus Video Codec. Note: fps means frame per second. * means there is no way to produce a stream at 20kbps. The 19-second video clip simulates the rapid movement of the robot in the campus. The source resolution is 320×240, the frame rate is 25 fps, and the data size of uncompressed RGB24 format is 110MB.*

	WMV9	**MPEG4**
20 kbps (28k dial-up modem, 3 fps)	50 KB	*
34 kbps (56k dial-up modem, 12 fps)	92 KB	183 KB
50 kbps (64k Single ISDN, 15 fps)	131 KB	193 KB
100 kbps (128k Dual ISDN, 15 fps)	240 KB	260 KB
150 kbps (150k LAN or DSL, 15 fps)	384 KB	378 KB

The results of Table 3.1 show that at a low bandwidth (< 100kbps) WMV9 is more effective than MPEG4, while at a higher bandwidth, over 100kbps, their performance is similar. More importantly, it is feasible to highly compress the video images for streaming, which is discussed further in the following.

3.3.2 Transmission performance of two video encoding methods

We investigate the transmission performance of two video encoding methods, i.e. CBR and Quality-based VBR. We conducted the experiment, using WMV9 as the video codec and using MMS (TCP) as the streaming protocol at the side of streaming server, with CBR and Quality-based VBR encoding methods at different bandwidth. The source data was a 19-second campus video clip, broadcast ten times for a total of 190 seconds. We measured the actual receiving bit rate every second at the side of the

streaming client. Figure 3.2 shows an early user interface of streaming client. The experimental results are shown in Figure 3.3.

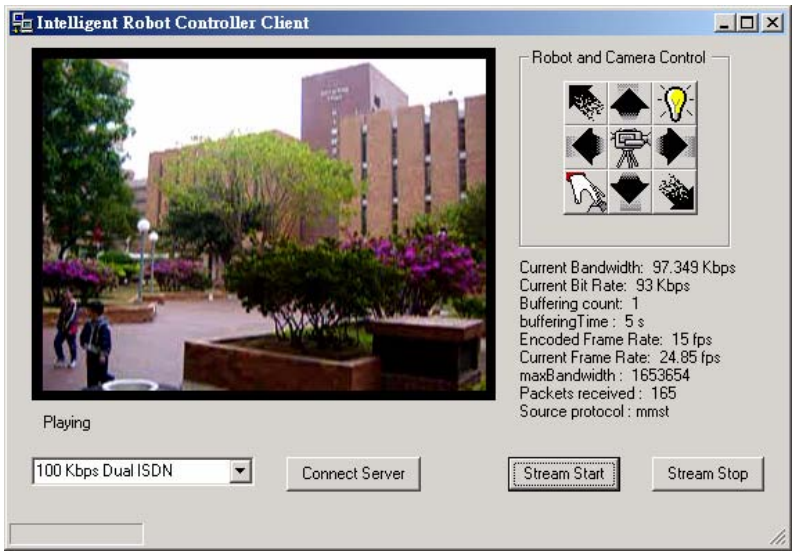
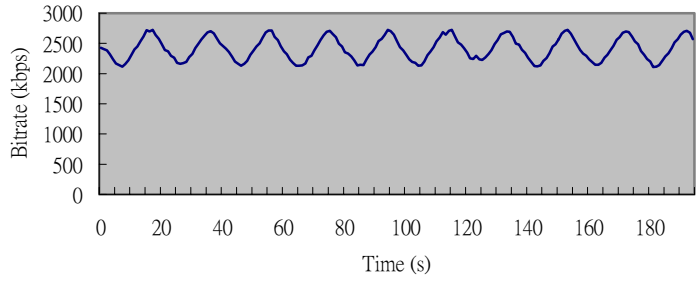
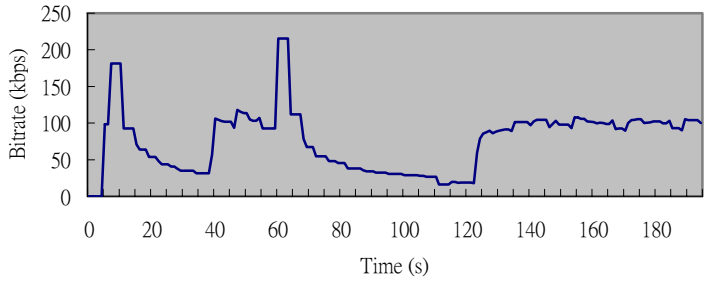


Figure 3.2: An early user interface of streaming client.

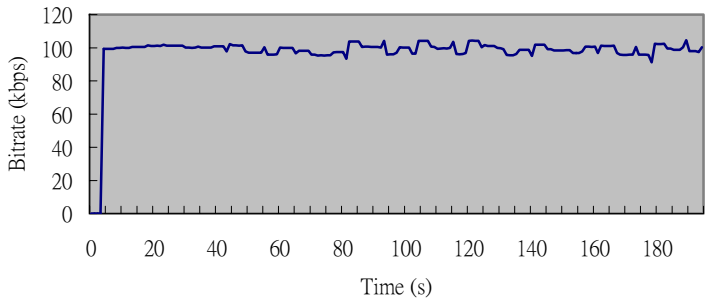
Quality-based VBR performs well at a high bandwidth but poor at a lower bandwidth. The curve as seen in Figure 3.3(a) is well-regulated. This is the reason that Quality-based VBR maintains a consistent quality across all streams at a high bandwidth (over 2.5Mbps). The wave crest and trough represent the repeated scene details. At a lower bandwidth (about 100kbps), however, playback performance is poor as seen in Figure 3.3(b). The advantage of Quality-based VBR encoding is that the quality remains consistent across all streams for which the specified quality setting (i.e. quality level ranging from 0 to 100) is the same. The disadvantage is that we cannot predict the file size or bandwidth requirements of the encoded content. We conclude that Quality-based VBR is not suitable for the live broadcast on Internet-based teleoperation.



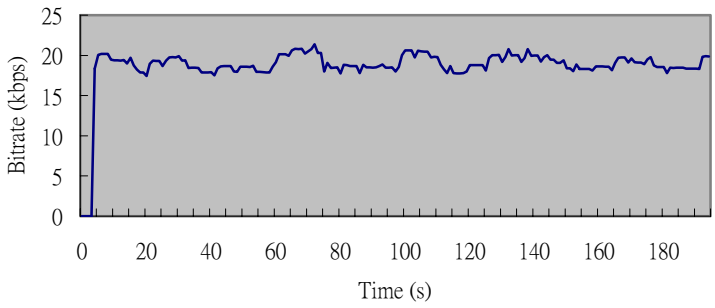
(a) Quality based VBR, quality level 100



(b) Quality based VBR, quality level 50



(c) CBR, 100kbps (campus Internet), 15fps



(d)CBR, 20kbps (33.6kbps modem), 5fps

Figure 3.3: Transmission performance of two video encoding methods (CBR and Quality-based VBR). The source data was a 19-second campus video clip, broadcast ten times for a total of 190 seconds. Note that different frame rate in (c) and (d) is used to ensure the transmission stability.

CBR encoding method performs well under 100kbps (see Figure 3.3(c)) and 20kbps bandwidth (Figure 3.3(d)). The content quality fluctuates to ensure that the buffer does not overflow or underflow. The advantage of CBR encoding is that the bit rate and size of the content are known before encoding, so we can predict the final size and bandwidth requirements of the encoded content. Of course, when content varies in complexity, the encoding quality is not constant. Using CBR encoding on Internet-based teleoperation ensures that the video images are streamed smoothly.

3.3.3 Transmission stability and time delay

We investigate the transmission stability and time delay under different Internet bandwidth by adjusting the video codec parameters. A campus video (320×240 resolution) broadcasts live from the streaming server to the client for playback about 5 minutes per test. Two typical bit rates (100kbps and 20kbps) are the encoded rates for different Internet connection. At the streaming server, WMV9 is used as the video codec, MMS is used as the streaming protocol, and CBR method is used for video encoding. The time delay is estimated for communication between the streaming server and the streaming client. The results are given in Table 3.2.

Whatever the Internet bandwidth is, increasing the codec buffer or decreasing the number of key frames can improve the system performance of transmission stability. The time delay is caused mainly by the buffer time of both encoder and player, which is used to guarantee the quality of service (QoS). Currently, most Internet users in the world use dial-up modem, ISDN, DSL or LAN and so on. The speed varies from 28kbps to 3Mbps or more. The results in Table 3.2 show that it is possible for all kinds of users to remotely monitor the robot surroundings via the video feedback.

Table 3.2: Transmission stability and time delay under different Internet bandwidth. Encoder buffer 5s means that the streaming server needs to cache 5 seconds video data for transmission. Player buffer 5s means that the streaming client needs to cache 5 seconds video data for playback. Key frame 1s means that the interval of two key frames is 1 second. Buffering counts 7 means that the streaming client may buffer 7 times during 5 minutes playback, which represents an unstable transmission. Buffering counts 1 means that the streaming client only buffers once at the beginning, which represents a stable transmission without obvious interruption.

Video codec parameters	Stability & time delay
100Kbps(campus Internet), Encoder buffer 3s, Player buffer 3s, key frame 8s	Buffering Counts: 7 Performance: unstable Time Delay: 10 s
100Kbps(campus Internet), Encoder buffer **5s**, Player buffer **5s**, key frame **8s**	Buffering Counts: 1 Performance: **stable** Time Delay: 12 s
100Kbps(campus Internet), Encoder buffer **3s**, Player buffer **3s**, key frame **1s**	Buffering Counts: 1 Performance: **stable** Time Delay: 10 s
20Kbps(33.6kbps modem), Encoder buffer 3s, Player buffer 3s, key frame 1s	Buffering Count: 15 Performance: unstable Time Delay: 10 s
20Kbps(33.6kbps modem), Encoder buffer 5s, Player buffer 5s, key frame 1s	Buffering Count: 13 Performance: unstable Time Delay: 12 s
20Kbps(33.6kbps modem), Encoder buffer **5s**, Player buffer **5s**, key frame **4s**	Buffering Counts: 1 Performance: **stable** Time Delay: 12 s

3.3.4 Robot teleoperation through low-bandwidth Internet

We set up a real robot teleoperation through a low-bandwidth Internet connection. The robot server and streaming server (see Figure 3.1) are connected to our campus Internet. The master client (human operator) is connected to the Internet via a telephone line using a 33.6Kbps dial-up modem. With the streaming video feedback, the human operator can see the remote robot's surroundings for global information.

The operator is able to remotely control the robot to explore areas of interest, and also able to observe details via the camera pan-tilt-zoom movement. Although there is a large time delay (about 12 seconds), we did succeed in remotely controlling the robot, using direct control to navigate in a complicated hall with many desks and walls (See Figure 3.4). Recently, a latest teleoperation from Canada (a user made connection to the Internet via a 56kbps dial-up modem) to Hong Kong has further demonstrated the feasibility of the use of streaming technology on the Internet telerobotics.

Figure 3.4: The robot to be remotely controlled to navigate in a complicated hall.

3.4 Comparison with other approaches for image feedback

We compare the projects of Internet telerobotics according to their approaches to image feedback. Table 3.3 gives the comparison result.

There are a number of advantages of streaming technology based approach compared with other approaches for image feedback. Some of them are as follows:

- *Better Quality of Service*: Streaming technology, based on WMV9 or MPEG4 compression algorithms, can greatly improve the quality of service over a low-bandwidth and uncertain Internet transmission channel, producing a more stable system, higher image resolution, and smoother image streams.
- *Multicast*: Streaming technology allows many Internet users to monitor the remote robot's surroundings simultaneously, without reducing quality of service or increasing network bandwidth. This function is derived from an attractive feature of streaming technology: Multicast. When using multicast streams, the streaming server generates one single stream that allows multiple player-clients to connect

with it. Users watch the content from the time they join the broadcast. The client is connected to the stream rather than to the server.

- *Extensibility*: Streaming technology can incorporate multiple types of data into a single transmission stream. This function will be an advantage if future applications of Internet telerobotics need more multimedia information feedback, such as audio.

TABLE 3.3: Comparison using different approaches for image feedback

Project name	Test environment	Technology	Image size	Efficiency
Mercury (1994)	14.4K Modem Internet	GIF client pull	192 × 165	1 frame every 60 seconds
Xavier (1995)	Lowest bandwidth Internet	GIF server push	Low resolution	1 frame every 20 seconds
EPFL (1998)	LAN	GIF or JPEG server push	200 × 150	10-15fps
BGen (2001)	Internet	Video conference using H.261	176 × 144	7.5 frames per second (fps)
Essex (2001)	Internet	JPEG server push	200 × 150	7-8 fps/ total 50 frames
VLAB (2003)	Internet	Video conferencing	unknown	3-4 fps
Our system (2003)	33.6K Modem Internet	Streaming technology using WMV9 or MPEG4	320 × 240	5 fps /total 25 frames

The disadvantage of the use of streaming technology is that the buffer time causes a large time delay (over 10 seconds). That's really difficult for the human operator to have enough experience of interactivity with the robot. Therefore, it is desirable for the robot server to feedback more timely information about robot to human operator. Chapter 5 will describe a compensation means to visualize the robot's local information, such as sonar readings or trajectory data.

3.5 Summary

This chapter presents a streaming technology based approach for Internet-based robot teleoperation. The streaming video is used to transmit the images captured by the robot's onboard camera so that remote Internet users can see the robot's surroundings to obtain global information. In Internet telerobotics, few literatures have discussed

the techniques or performances of image transmission over the Internet in details. That's why we investigate the video streaming in this chapter. We indeed do not go to the depth of streaming technology itself to improve its video compression and transmission. It is more of an investigation and implementation of the existing streaming technology, to see that which techniques about video codec or video encoding etc are feasible for Internet telerobotics and how their performance are. The work in this chapter is beneficial for the researchers of Internet telerobotics to adopt similar techniques in order to improve image transmission and make the Internet-based teleoperation usable.

It is experimentally shown that the streaming technology, using WMV9 or MPEG4 algorithm as well as CBR video encoding method, can produce a more stable system, higher image resolution, and smoother image streams. It is also demonstrated to be feasible for real robot teleoperation through a low-bandwidth Internet connection.

CHAPTER 4. A FRAMEWORK OF AUTONOMOUS NAVIGATION USING FUZZY LOGIC

Chapter 3 presents a real robot teleoperation by using direct control. The robot is like a puppet without autonomous capability for sensing the environment and dealing with unexpected events such as moving objects. It is dangerous for such robot to be remotely control through the Internet since the time delay of transmitting images is large. A robot, which is able to sense the environment and perform some tasks autonomously, is highly desired. This chapter proposes a framework of autonomous navigation using fuzzy logic. This work is a base for the subsequent chapters to implement some complex tasks, such as active map learning, goal-oriented navigation.

4.1 Introduction

The main challenge of today's autonomous robotics is to build robust control programs that reliably perform complex tasks in spite of the uncertainties derived from environments and robots themselves. Despite the recent advances in the field of autonomous robotics, there are some problems that have to be addressed in order to exhibit truly autonomous navigation [Saffiotti, 2000]. First, prior knowledge about the environment is, in general, incomplete, uncertain, and approximate. For example, maps typically omit some details and temporary features, spatial relations between objects may have changed since the map was built, and the metric information may be imprecise and inaccurate. Second, perceptually acquired information is usually unreliable. The limited range, combined with the effect of environmental features (e.g., occlusion) and of adverse observation conditions (e.g., poor lighting), leads to noisy and imprecise data; and errors in the measurement interpretation process may lead to incorrect beliefs. Third, real-world environments typically have complex and unpredictable dynamics: objects can move, other agents can modify the environment, and relatively stable features may change with time (e.g., seasonal variations). Finally, the effect of control actions is not completely reliable: wheels may slip, and a gripper may lose its grasp on an object.

Traditional work in robotics has tried to overcome these difficulties by carefully

designing the robot mechanics and sensors, or engineering the environment, or both. Engineering the robot or the environment, however, increases costs, reduces robot's autonomy, and cannot be applied to all domains. If we want to build easily available robots that inhabit our homes, offices, or factory floors, the platform cannot be overly sophisticated, and the environment should not be modified.

Since Brooks [1986] proposed the behavior control architecture, the idea has been adopted to solve the navigation problem in an unknown environment. Unlike the traditional navigation architecture [Saffiotti, 2000] which decomposes the navigation task using a sense-model-plan-act (SMPA) framework and connects each module serially, the behavior control method decomposes the navigation system into special task-specific behavior modules, e.g., obstacle avoidance, goal seeking, etc., which are connected directly to sensors and actuators and operate in parallel. Simple behaviors are then combined in order to produce a complex strategy able to pursue the strategic goals while effectively reacting to any contingencies. Therefore, this architecture can act in real-time and has good robustness. As the behavior control architecture tackles the navigation problem in an on-line manner and requires no environment model, it is efficient in dealing with navigation in an unknown environment.

In the behavior control architecture, behavior modules are usually constructed as reactive systems, which map the perceived situations to the correct actions. Fuzzy logic method [Lee & Wu, 2003; Seraji & Howard, 2002; Saffiotti et al, 1999; etc.] is an efficient way of representing this mapping relationship as it is able to represent human expert’s knowledge and requires no mathematical model.

This chapter proposes a framework for a behavior-based navigation strategy of autonomous robots. The framework includes the steps of goal determination, preprocessing, behavior design, behavior arbitration, and command fusion. It is practical and has been shown experimentally to be reliable.

The rest of this chapter is organized as follows. Section 4.2 proposes the framework of behavior-based autonomous navigation. Section 4.3 describes an example of the proposed framework. Section 4.4 makes the experiments. Section 4.5 discusses the proposed framework. The last section summarizes the chapter.

4.2 A framework of behavior-based autonomous navigation

A framework of a behavior-based autonomous navigation is proposed as shown in Figure 4.1. It is independent of robotic development platform. In every robot control cycle, the robot's reasoning system outputs the next set of motor control commands by performing an inference process *I*. This inference process can be defined as a relationship between the input space *U* and the output space *Y*. The input space *U* is multidimensional, with each dimension corresponding to a particular input data mode, e.g., distance to front obstacle, direction to the goal. Similarly, the output space *Y* is multidimensional, with each dimension corresponding to a particular type of output, which normally includes motor speed and delta turn angle. Thus it is expressed by:

$$I: \quad U(u_1, u_2, \cdots, u_i, \cdots, u_n) \rightarrow Y(y_1, y_2, \cdots, y_j, \cdots, y_m)$$

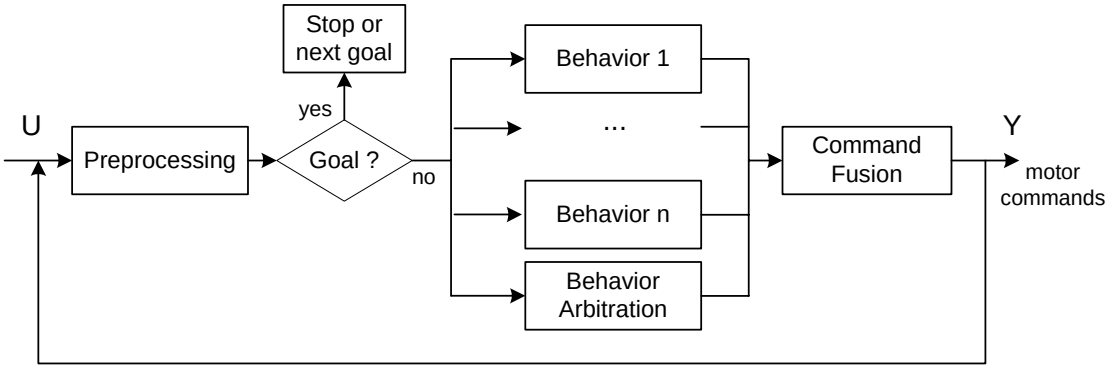


Figure 4.1: A framework of behavior-based autonomous navigation

4.2.1 Preprocessing

In the preprocessing module, the input space *U*, including the exteroception and proprioception sensing data, should be gathered and updated. The robot uses exteroceptive sensors, like a camera or a sonar or laser sensors, to observe the state of the environment. It uses proprioceptive sensors, like a compass or shaft encoders on the wheels, to monitor the state of its own body.

If the input space *U* is too large, the computational complexity should be controlled by reducing the number of dimensions. A common way to do this is to use a "situation clustering" approach [Goodridge & Kay, 2000] in which the complexity of the input space is reduced by introducing a limited number of intermediate

variables. These variables classify the different "perceptual situations" relevant to the robot's behavior. Possible intermediate variables are statements such as "*facing_obstacle*" or "*distance_to_left_obstacle*". These variables are then used by the consequent behavior design. Typical example is provided in Section 4.3.

4.2.2 Goal determination

In our view, a task possesses two types of goal determination: a *determined goal* and a *non-determined goal*.

When a goal is determined, its exact coordinate is given. A typical example is the task for goal-oriented navigation, in which the robot must move to a given target while autonomously avoid any static or dynamic obstacles in its path. The robot succeeds only if it arrives at the goal without any collisions.

When a goal is non-determined, it does not have an exact coordinate location. Instead, the goal is defined by a termination criterion. An example is the task for GOTOEND, in which the robot is required to avoid any lateral obstacles and to stop only if the distance to front obstacle is less than a threshold. Another example is the task for WANDER. The robot is required to wander randomly without exact goal location. A possible termination criterion is that the total distance of wandering is over a given value.

4.2.3 Behavior design

A complex task can be decomposed into multiple simpler behaviors which can subsequently be coordinated. The framework allows multiple individual behaviors and the module of behavior arbitration to be executed in parallel. This makes full use of precious computational resources and results in the best real-time efficiency.

Classically, robot behaviors are of two types: global (e.g., path-tracking and goal-seeking) and local (e.g., obstacle-avoidance, wall-following, door-crossing, and light-reaching). For example, to realize a mobile robot’s path-tracking behavior, a controller is given a path in some internal reference frame, and it generates motor commands in order to follow the path as closely as possible. Local behaviors are actually sensor-based behaviors, which implement a control strategy based on external sensing.

There are three common ways to design a behavior. One is to use an analytic algorithm with a determined model (see the example in Section 4.3). The second way uses a machine learning technique based on supervised learning or reinforcement learning [Godjevac & Steele, 2000; HuaNan Yu et al, 2002; Hagras et al, 2001; Na & Oh, 2003; etc.]. We will discuss it in Section 4.5. The third way uses a pure fuzzy logic controller [Seraji & Howard, 2002; etc.]. Since the critical problem for behavior design is to guarantee robust operation in the presence of uncertainty, we focus on the way using fuzzy logic controller in the following paragraphs.

In our fuzzy logic controller, reasoning is embodied in the rules operating on linguistic input and output variables, as in

If u_1 *is* A_1 *and* u_2 *is* A_2 *and* $\cdots\cdots$ *and* u_n *is* A_n

Then y_1 *is* B_1 *and* y_2 *is* B_2 *and* $\cdots\cdots$ *and* y_m *is* B_m

Where the u_is are input linguistic variables taking linguistic values A_i, each linguistic value being defined by a membership function $\mu_{A_i}(u_i)$; the y_is are output linguistic variables taking linguistic values B_i, each linguistic value being defined by a membership function $\mu_{B_i}(y_i)$.

Given two linguistic values A and B defined on the same universe of discourse, the AND and OR operation are defined respectively as Eq. (4.1) and Eq. (4.2).

$$\mu_{A\cap B} = \min_{u\in U}(\mu_A(u), \mu_B(u)) \qquad (4.1)$$

$$\mu_{A\cup B} = \max_{u\in U}(\mu_A(u), \mu_B(u)) \qquad (4.2)$$

The best well-known Centroid method is chosen as the defuzzification method. For the continuous output space, we obtain

$$y^* = \frac{\int y \cdot \mu_Y(y)dy}{\int \mu_Y(y)dy} \qquad (4.3)$$

where $\int$ is the classical integral. So this method determines the center of the area below the combined membership function.

Algorithm 4.1 shows the fuzzy inference process. The examples of behavior design are presented in Section 4.3.

Algorithm 4.1 (fuzzy inference process):

Input: u_is = crisp numerical values of the input variables.

Output: y_is = crisp numerical values of the output variables.

BEGIN:

Step 1: Fuzzification of the input variables u_is;

Step 2: Application of the fuzzy operator (AND or OR) as Eqs.(4.1)-(4.2) in the antecedent of the rules;

Step 3: Implication from the antecedent to the consequent using the AND operation as Eq. (4.1);

Step 4: Aggregation of the consequents across the rules using the OR operation as Eq.(4.2);

Step 5: Defuzzification into output variables y_is using Eq.(4.3).

END Algorithm 4.1

4.2.4 Behavior arbitration and command fusion

As suggested by Figure 4.2, behavior coordination problems can be approached as two conceptually different problems: (i) how to decide which behavior, obstacle-avoidance (OA) or goal-seeking (GS) for example, should be activated at each moment - and, possibly, to what extent; and (ii) how to combine the results from different behaviors into one command to be sent to the robot's motors - possibly, taking weightings into account. These sub-problems are, respectively, called the *behavior arbitration* and the *command fusion* problems [Saffiotti, 2000].

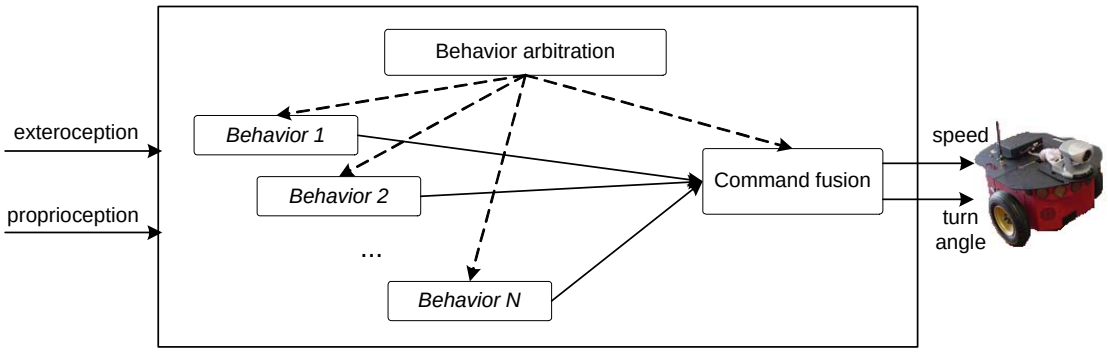


Figure 4.2: Behavior coordination problem decomposing into two subproblems: behavior arbitration and command fusion.

Some [Gat, 1998; etc.] of the behavior arbitration methods adopt High-Priority-Take-All or Winner-Take-All selection strategies but these strategies come with two disadvantages: their performance in certain situations is inefficient, and the

desirability of each behavior cannot vary from situation to situation. Other strategies have employed fusion methodologies in which each behavior is allowed to affect the final output based on the situational context. One such strategy is context-dependent blending (CDB) [Saffiotti et al, 1999] in which fuzzy logic is applied so that a decision between behaviors can be made in a prevailing situation.

Our behavior arbitration strategy is similar to the CDB approach. It uses fuzzy context rules to express a behavior arbitration strategy. When the obstacle is close, both OA and GS behaviors are partially activated. Each behavior is assigned a weighting factor, and these factors are adjusted dynamically according to the fuzzy weight rules. The weighting factors determine the degree of influence of each behavior on the final motion command. The weight rules continuously update the behavior weighting factors during robot motion.

The strategy adopted in our approach is simpler than that of the CDB approach. The CDB approach uses a fuzzy preference combination to carry out command fusion but we first use a behavior arbitration module to calculate the defuzzified weight factors of all behaviors, and then carry out command fusion directly using these weight factors in Eqs. (4.4) and (4.5). One advantage of this coordination strategy is that the defuzzified weight factors can be visualized (refer to Section 7.6.1 in Chapter 7). As a result, the tuning of the fuzzy logic controller is easier because the contributions by different behaviors are clearly visualized.

$$v = \frac{\sum v_i \cdot w_i}{\sum w_i} \tag{4.4}$$

$$\theta = \frac{\sum \theta_i \cdot w_i}{\sum w_i} \tag{4.5}$$

where, v and θ are the desired final speed and the delta turn angle values respectively while v_i and θ_i are the speed and angle preference values suggested by each individual behavior respectively. w_i is the defuzzified weight factors that are output by the behavior arbitration module.

4.3 An example of behavior-based autonomous navigation

In this section, an example (i.e. goal-oriented navigation) of behavior-based autonomous navigation is given using the proposed framework. The goal-oriented navigation is a common robot navigation task. It calls for a robot to be given a goal

location and for the robot to then reach the goal autonomously. Here we assume that: 1) the robot is located in an environment but without prior knowledge about it; 2) the robot knows the coordinates of current location and goal location; 3) the robot senses the environment depending on its ultrasonic sensors (i.e. sonars).

We decompose the task of goal-oriented navigation into two elementary behaviors: obstacle-avoidance (OA) and goal-seeking (GS). The OA behavior is a sensor-based local behavior which implements a control strategy based on external sensing. It is activated if obstacles are close. The GS behavior is a global behavior which does not rely on external sensory data, but seeks for the globally exact goal location. The two behaviors are coordinated to select the final motor control values that steer away from the obstacle while maintaining the goal direction.

In the preprocessing module, we reduce the complexity of input space by grouping the robot's sonar readings into three sectors (left, front, right). For example, our robot has a ring of eight forward ultrasonic sonars that produce a set of obstacle distances {*d0, d1, d2, d3, d4, d5, d6, d7*}. We obtain three groups of obstacle distances by the following equations.

$$d_{left} = min(d0, d1); \quad (4.6)$$

$$d_{front} = min(d2, d3, d4, d5); \quad (4.7)$$

$$d_{right} = min(d6, d7). \quad (4.8)$$

The OA behavior is designed using fuzzy logic controller in order to deal with uncertainties from sonar readings. The obstacle distance of each sector is represented by three linguistic fuzzy sets {VERYNEAR, NEAR, FAR}, with the membership functions shown in Figure 4.3 (a). The weight of OA behavior w_{oa} is represented by three linguistic fuzzy sets {SMALL, MEDIUM, LARGE} with the membership functions shown in Figure 4.3 (b). The motion control variables of the mobile robot are the translational speed and the rotational turn angle. The robot speed is represented by three linguistic fuzzy sets {STOP, SLOW, FAST}, with the membership functions shown in Figure 4.3 (c). The robot delta turn angle is represented by five linguistic fuzzy sets {NB, NS, ZE, PS, PB}, with the membership functions shown in Figure 4.3 (d), where NB is negative-big, NS negative-small, ZE zero, PS positive-small, and PB positive-big. The positive and negative terms stand for the robot turning to the left and right, respectively.

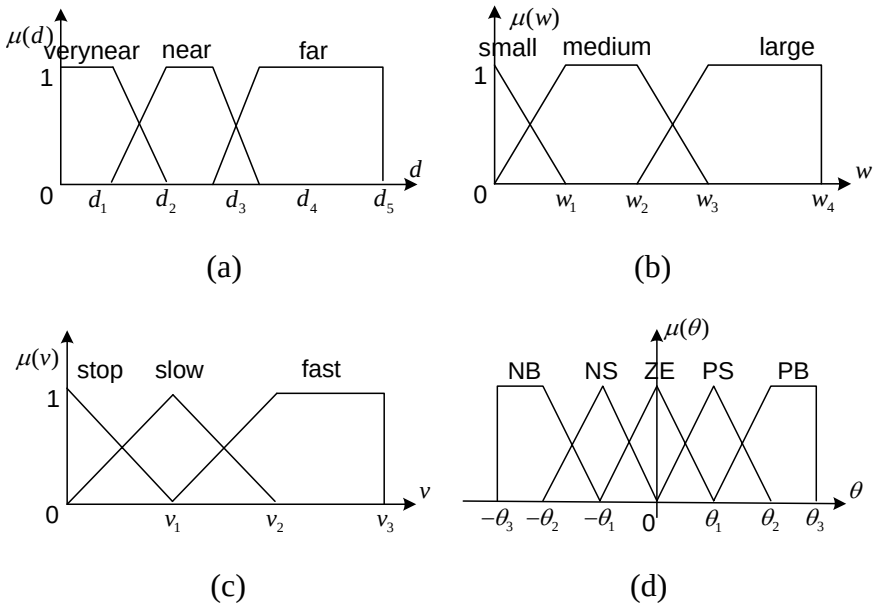


Figure 4.3: Membership functions for (a) obstacle distance; (b) weight; (c) speed; (d) delta turn angle.

The OA navigation rules are presented below. The turn rules for the OA behavior are summarized in Table 4.1. The rules exhibit such a behavior characteristic: if the obstacle distance in any sector is VERYNEAR, the robot should turn away to find a safer direction. For instance, the (1,3) element of the bottom layer in Table 4.1 can be written out as the rule:

IF d_{front} *is FAR AND* d_{left} *is FAR AND* d_{right} *is VERYNEAR, THEN* θ_{oa} *is PS.*

Note that when the three sectors have the same *VERYNEAR* obstacle distance as shown in the (3,3) element of the top layer in Table 4.1, a large left turn (PB) angle is recommended. This turn rule enables the robot to escape from its current embarrassed situation.

Table 4.1: Turn rules for the OA behavior.

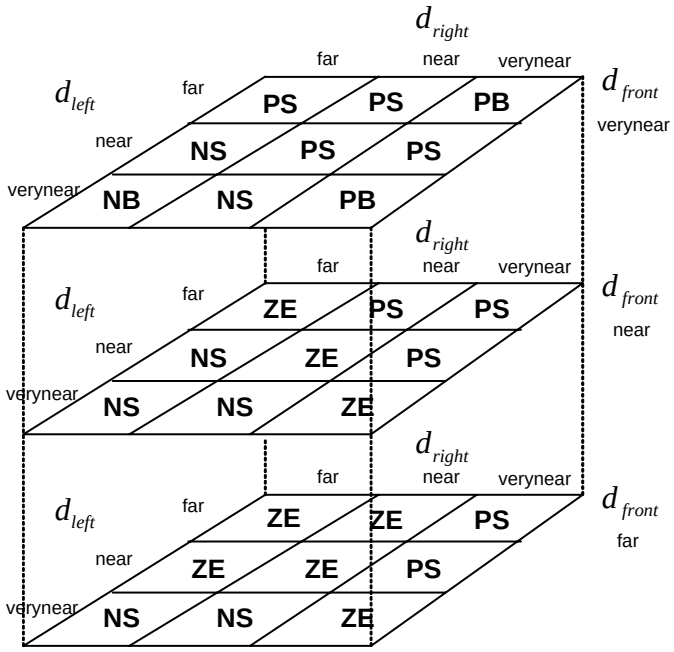
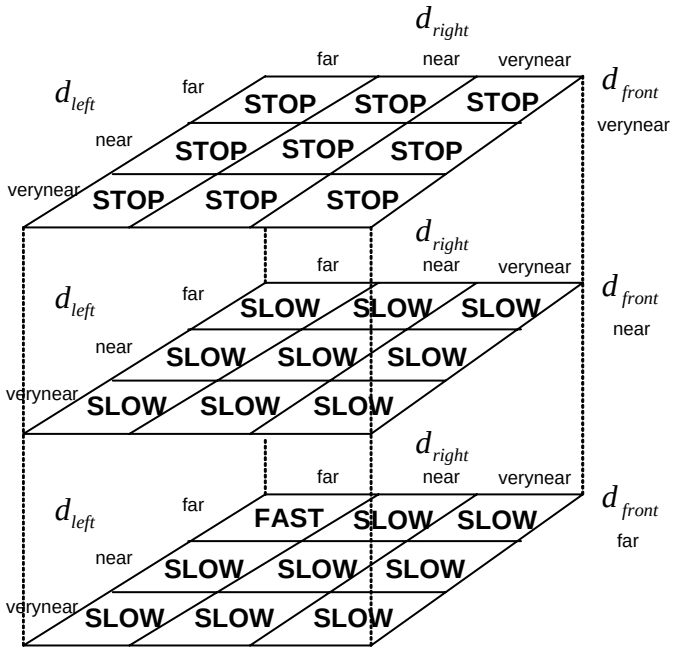


Table 4.2: Move rules for the OA behavior.



The move rules of the OA behavior is summarized in Table 4.2. The rule enables the robot to decrease its speed when an obstacle is approaching. In fact, the elements of the bottom layer in Table 4.2 can be written out as two rules:

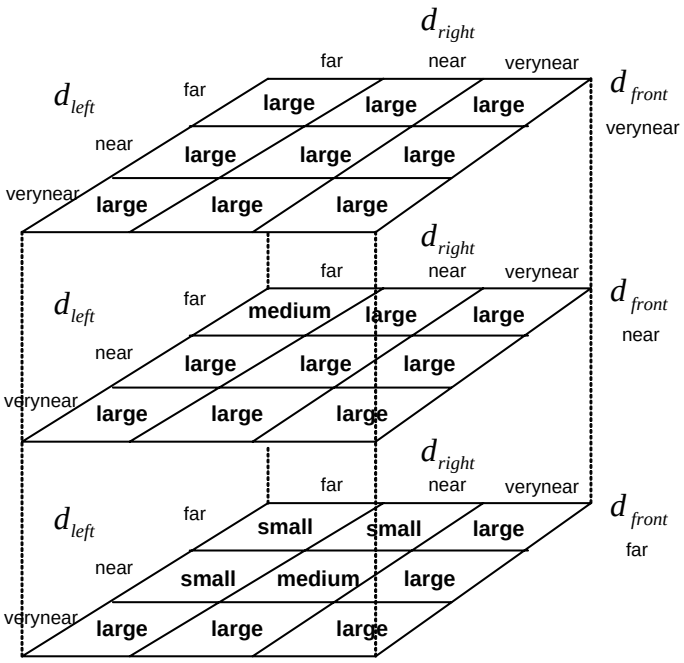
1) IF d_{front} is FAR AND d_{left} is FAR AND d_{right} is FAR, THEN v_{oa} is FAST.

2) IF d_{front} is FAR AND (d_{left} is VERYNEAR OR d_{left} is NEAR OR d_{right} is VERYNEAR OR d_{right} is NEAR), THEN v_{oa} is SLOW.

Table 4.3 summarizes the weight rules of the OA behavior. The weight is derived directly from obstacle distances in the three sectors.

Note that the fuzzy logic navigation and weight rules developed in this chapter can be applied to any mobile robot, regardless of robot characteristics such as wheel size. These characteristics are reflected only in the definition of the membership functions used in the fuzzy rules.

Table 4.3: Weight rules for the OA behavior.



The GS behavior is designed using a precise analytical model. We first assume that the GS behavior does not influence the speed of the robot, and contributes only to the rotational turn angle. Second, we use a very simple analytical model rather than a set of fuzzy logic navigation rules. So,

$$v_{gs} = 0 \qquad (4.9)$$

$$\theta_{gs} = \varphi 1 \qquad (4.10)$$

where, v_{gs} and θ_{gs} are the speed and delta turn angle respectively recommended by the GS behavior. $\varphi 1$ is the heading error between the current robot heading and goal direction as shown in Figure 4.4. Thus, the value domain of θ_{gs} is (-180^0, 180^0]. Similarly, the positive and negative terms have implied that the robot turns to the left

and right respectively. The calculation of $\varphi 1$ requires that we take into account all situations in a system of coordinates, in which the robot and the goal are located in different quadrants.

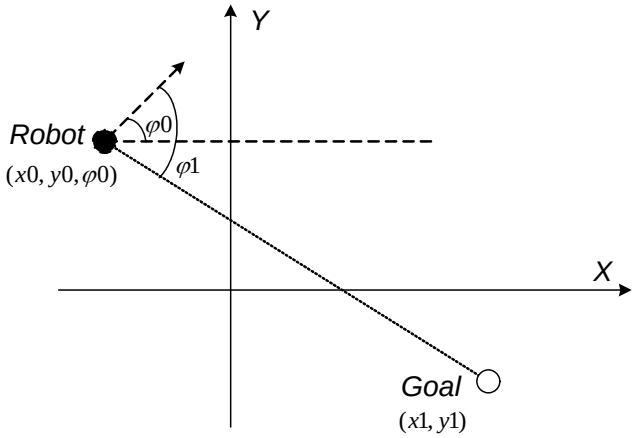


Figure 4.4: Heading error between the current robot heading and goal direction.

There are only three rules for the weight of the GS behavior. The weight w_{gs} is derived directly from the weight w_{oa} of the OA behavior.

1) IF w_{oa} is SMALL, THEN w_{gs} is LARGE.
2) IF w_{oa} is MEDIUM, THEN w_{gs} is MEDIUM.
3) IF w_{oa} is LARGE, THEN w_{gs} is SMALL.

Algorithm 4.2 gives the control algorithm for the task of goal-oriented navigation.

Algorithm 4.2: (Goal-oriented navigation)
Input: (x1, y1) = goal location; (x0, y0) = current robot location;
φ0 = current robot heading angle;
(d0, d1, d2, d3, d4, d5, d6, d7) = sonar readings.
Output: (v, θ) = speed and delta turn angle
BEGIN:

Step 1. To update sensory data including (x0, y0), φ0 and (d0, d1, d2, d3, d4, d5, d6, d7);

Step 2. To preprocess the sonar readings using Eqs. (4.6), (4.7), and (4.8);

Step 3. IF the distance from current robot location to goal location is less than a predefined threshold (i.e. distance tolerance), THEN the goal is reached and the robot is stopped, OTHERWISE go to the Step 4;

Step 4. To calculate v_{oa} and θ_{oa} recommended by the OA behavior using Algorithm 4.1 and the turn rules as in Table 4.1, the move rules as in Table 4.2;

Step 5. To calculate v_{gs} and θ_{gs} recommended by the GS behavior using Eqs. (4.9) and (4.10);

Step 6. To calculate the weight w_{oa} of the OA behavior using Algorithm 4.1 and the weight rules as in Table 4.3;

Step 7. To calculate the weight w_{gs} of the GS behavior using Algorithm 4.1 and the three weight rules of the GS behavior;

Step 8. To calculate (v, θ) by the command fusion using Eqs. (4.4) and (4.5);

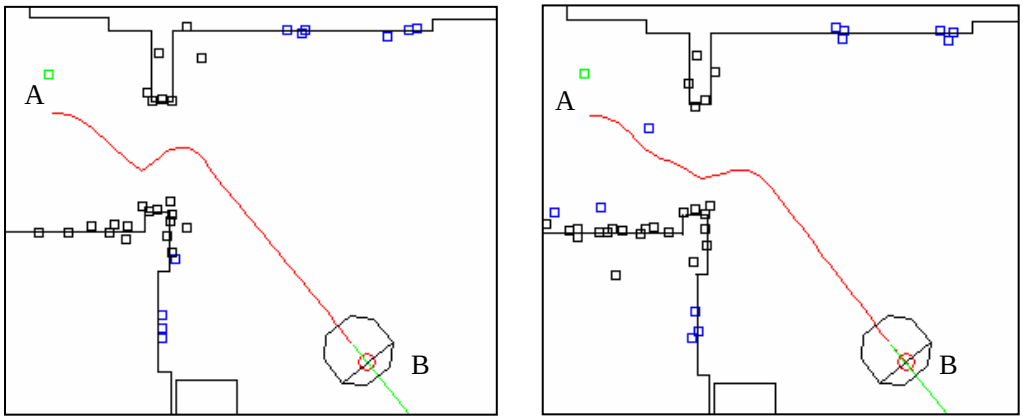
Step 9. To execute the motor control commands (v, θ), and go to the Step 1 again.

END Algorithm 4.2

4.4 Experimental results

4.4.1 Experiment for goal-oriented navigation

First we perform the simulated experiments for goal-oriented navigation in unknown environment. Two methods are implemented to complete the task of goal-oriented navigation for comparison. The first one is “exclusive OA+GS” that we name. In this method, the task is decomposed into two behaviors: obstacle-avoidance (OA) and goal-seeking (GS). But the OA behavior is designed using a precise mathematic model (threshold control) instead of fuzzy logic controller. Moreover, the OA and GS behavior are exclusive each other since only one behavior is activated in a situation. The second method is implemented using Algorithm 4.2. We call it “OA+GS” method. Figure 4.5 shows the experimental results for comparison.



(a) (b)

Figure 4.5: Performance comparison for goal-oriented navigation in unknown environment. The continuous curve is the actual trajectory of the robot movement. (a) Exclusive OA+GS, 22s; (b) OA+GS, 19s.

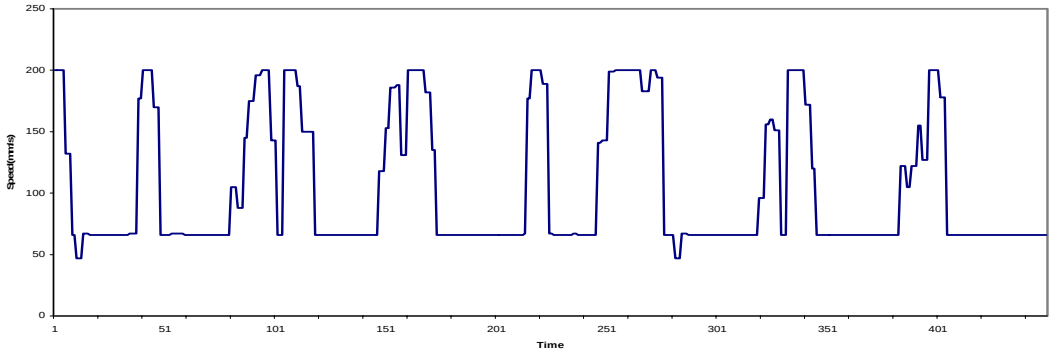
The performance (i.e. time efficiency and trajectory) of the "OA+GS" approach is superior to that of the "Exclusive OA+GS" approach. Starting from A, the robot is required to reach the goal B. Two methods consume 22 seconds and 19 seconds respectively. The differences in the performance arise from the process in which the robot avoids the obstacles and looks for the safe path. The performance of the "Exclusive OA+GS" approach is poor. When the robot is very close to the obstacles, the OA behavior is activated under a threshold control and replaces the GS behavior. When the OA behavior is in operation, the GS behavior can not make a contribution. This is why the turn angle of the trajectory is large and the time efficiency is low. The "OA+GS" method outperforms the "Exclusive OA+GS" method. Because the weights of the two behaviors are being adjusted in real time, both the OA and GS behaviors can be activated simultaneously. Moreover, fuzzy logic provides a good means for mobile robot to handle uncertainties derived from sensory data.

4.4.2 Experiment for robot wander

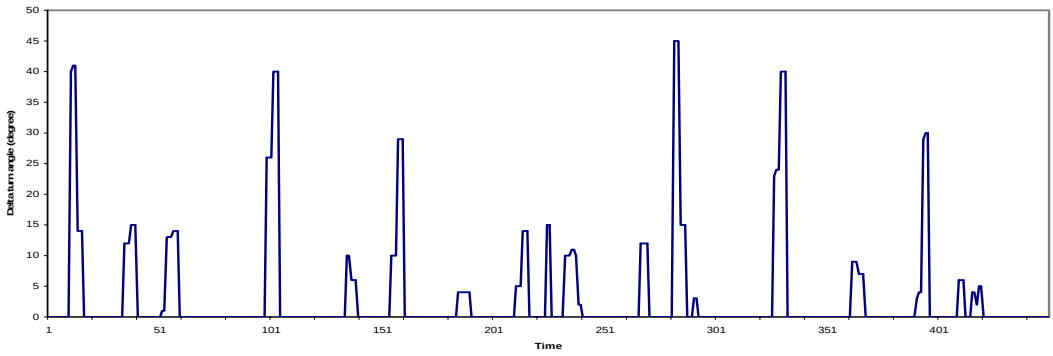
Next we perform the experiment for robot wandering in a real world. The robot's wander is implemented only using an elementary behavior: obstacle-avoidance (OA). The OA behavior is designed using fuzzy logic controller as described in Section 4.3. Figure 4.6 shows a series of pictures captured from a camera during the experiment. The experiment involves allowing the robot to wander within a small circular area

provided with both static obstacles (e.g. boxes and walls) and dynamic obstacle (e.g. moving human). Figure 4.7 shows the variations of robot's speed and turn angle during wandering. From the Figure 4.7(a) and (b), we know that the robot would decrease its speed and turn a degree of angle for safety when it is closing to obstacles. Note that the robot always turns left because the obstacles are always approaching on its right. The experiment demonstrates that the robot's wandering based on the reactive OA behavior using fuzzy logic is feasible and reliable, even in a dynamic environment (i.e. with moving humans).

Figure 4.6: Robot wandering in a circular small area set with static obstacles and moving human.



(a)



(b)

Figure 4.7: The speed and turn angle of mobile robot during wandering. (a) speed; (b) delta turn angle. Positive degree implies turning left.

4.5 Discussion

The success of fuzzy logic controller is owed in a large part to the ability of technology which can convert qualitative linguistic descriptions into complex mathematical functions. It appears very useful when the processes are too complex for analysis by conventional quantitative techniques or when the available sources of information are interpreted qualitatively, inexactly, or uncertainly, which is the case with mobile robots. Given the uncertain and incomplete information an autonomous robot has about the environment, fuzzy rules provide an attractive means for mapping sensor data to appropriate control actions in real time. However, fuzzy logic controller does not have the self-learning capability and is difficult to tune. Also, as the number of input variables increases (which is the case with mobile robots), the number of rules increases exponentially, which creates much difficulty in determining large numbers of rules.

Therefore, the machine learning techniques, such as neural network or neuro-fuzzy controller, are used to design a behavior or a whole robotic system in recent years [Chen et al, 2001; Na & Oh, 2003; Yang & Meng, 2003; etc.]. The arbitrary determination of the structure and initial weight of neural network have great impact on the performance of neural network controller. As an alternative, neuro-fuzzy controllers [Rusu et al., 2003], which combine the learning ability of the neural network with the advantage of the rule-based structure of fuzzy logic, have been extensively studied. In many cases of neuro-fuzzy control, the back-propagation algorithm has been widely used. However, being a gradient descent method, such algorithm has many drawbacks, which include slow convergence, local minimum, and so on. The evolutionary algorithm [Yamada, 2005], for example Genetic Algorithm (GA), is another well-accepted technique to design fuzzy controllers. Unfortunately, most of the work using evolutionary algorithm were undertaken using simulation as it takes a large number of iterations to develop a good controller in conventional GA. Thus, it is not feasible for a simple GA to learn online and adapt in real time. The situation is worsen by the fact that most evolutionary methods developed so far assume that the solution space is fixed (i.e. the evolution takes place within a predefined problem space, not in a dynamically changing and open one), thus preventing them from being used in real-time applications.

From another perspective, reinforcement learning and supervised learning [Tan et

al, 2002; Cang Ye et al, 2003; Kaelbling & Littman, 1996] are commonly used to construct the neural or neuro-fuzzy controller automatically. Reinforcement learning method seems quite promising as it requires no training data. However, it usually leads to a heavy learning phase as the gradient information is not provided explicitly. For example, due to the large number of the input space for learning obstacle avoidance, the search space becomes too large and the performance evaluation surface becomes too complex to allow efficient learning. Therefore, it is not easy to apply the reinforcement structural and parameter learning methods to learn obstacle avoidance, since it is difficult to tell that an incorrect response is due to a mismatch antecedent part or due to an incorrect consequent part. Furthermore, the phenomenon of premature convergence (e.g., trap situation) and ill behavior (e.g., circumnavigate around an obstacle closely and slowly) further undermines the practicality of these methods. On the contrary, supervised learning method has the advantages of fast convergence and is suitable for structure and parameter learning. However, it is very difficult to obtain sufficient training data, which contain no conflict input/output pairs. Insufficient training data may result in an incomplete fuzzy rule base, while the conflicts among the training data may cause incorrect fuzzy rules.

In this thesis, we construct fuzzy logic controllers using a "trial-and-error" approach by human designer to tune the parameters and fuzzy rules. Because the complexity of input space is greatly reduced by introducing a limited number of intermediate variables (e.g. d_{left} , d_{front} , d_{right}), we can easily guarantee the consistency and completeness of the fuzzy rule base. Moreover, it is highly desirable that we can easily realize the desired behavior characteristics by explicitly expressing the linguistic rules using a common natural language. More examples can be seen in the behavior designs in Chapter 6 and Chapter 7.

The fuzzy control approaches for robot navigation have been widely used in literatures. The main difference between our approach and the existing ones is that they have different fuzzy-rule based inference model (e.g. Mamdani model or Takagi-Sugeno-Kang model), defuzzification method (e.g. Centroid or Maximum defuzzification), membership functions (e.g. triangular or Gaussian), or fuzzy rules. Fuzzy logic is just a tool for the proposed framework to design a behavior or make behavior arbitration.

The following summarizes the key attributes of the proposed framework for behavior-based autonomous navigation.

a) *Linguistic representation*: The framework allows the capture of human commonsense knowledge, intuitive reasoning and decision making. The navigational logic uses linguistic terms from a common natural language.
b) *Uncertainty Management*: Fuzzy logic provides a systematic framework for dealing with imprecise and uncertain information. Thus errors arising from sensor noise are effectively handled by the navigation system.
c) *Reliability*: Fuzzy logic can deal with imprecise and uncertain sensing information. While one behavior may produce unreasonable control outputs, it can be made more reliable by coordinating multiple behaviors.
d) *Parallelity*: A complex navigation task can be divided into multiple independent behaviors. Each individual behavior can be executed in parallel.
e) *Computational Efficiency*: The fuzzy logic controller can be implemented using series of min- and max-gates in hardware, with all rules operating in parallel [Watanabe et al., 1990]. Other calculations, such as command fusion, are computationally efficient as well. All these facilitate their use on a real-time mobile robot.
f) *Extensibility*: The behavior-based approach makes it easy to add new modules that represent additional behaviors to the navigation system. The framework makes the navigation logic easily extensible while it does not rely on any specific robotic development platform.

4.6 Summary

This chapter proposes a framework of autonomous navigation for mobile robot. Note that the behaviour-based navigation is not a fresh idea or concept. The work in this chapter focuses on the development of a simple and practical navigation framework that can be easily realized to build robust control programs.

The framework includes the preprocessing, goal determination, behavior design, behavior arbitration, and command fusion. A complex task can be decomposed into multiple simpler behaviors for coordination. The intermediate variables are introduced in the preprocessing module in order to reduce the complexity of input space, so that

fuzzy logic controller can be easily constructed to implement the behavior design and behavior arbitration. Section 4.5 has discussed why we choose fuzzy logic as one of the tools to construct a controller for robot navigation. An example, goal-oriented navigation in unknown environment, is realized to demonstrate that the proposed framework is practical and feasible. The framework has several desirable attributes, including linguistic representation, uncertainty management, reliability, parallelity, computational efficiency, and extensibility.

We think that the proposed framework is simple and practical. For example, as mentioned in Section 4.2.4, we first use a behavior arbitration module to calculate the defuzzified weight factors of all behaviors, and then carry out command fusion directly using these weight factors. As a result, the tuning of fuzzy logic controller is easier because the contributions by different behaviors are clear by visualizing the defuzzified weight factors in the tests.

CHAPTER 5. TELECOMMANDING: A NEW INTERACTIVE TELEOPERATION APPROACH

Chapter 3 presents an Internet-based robot teleoperation using direct control. The direct control could only support some simple tasks for mobile robot teleoperation, because this kind of control mode makes the teleoperation very inefficient and dangerous due to the high latency of the Internet, such as restricted bandwidth and uncertain time delay. An interactive teleoperation approach, which is able to provide sufficient functionality and easy-to-use user interface, is highly desired. This chapter proposes a new teleoperation approach, which implements an interactive control interface and a complete framework for control management and command processing.

5.1 Introduction

Robots can now not only make basic motions but can also closely interact with people. Internet robots can provide many different remote services with potential applications in many areas: consumer home pet services, entertainment, telemedicine, distance learning, and the sharing of laboratory resources, as well as industry automation, military and security applications [Luo et al, 2003]. On the other hand, the Internet also entails a number of limitation and difficulties, such as restricted bandwidth, arbitrarily large transmission delays, and packet lost or error, all of which influence the performance of Internet-based telerobotics systems [Brady & Tarn, 2002; Luo et al, 2003].

Existing online robots are of two types: mobile manipulators with haptic or force feedback [Taylor & Trevelyan, 1995; Goldberg et al, 2000; Elhajj et al, 2003; Stein, 2003; Li & Lu, 2002], and mobile vehicles used for navigation [Simmons et al, 2000; Thrun et al, 1999; Saucy & Mondada, 2000; Siegwart &Saucy, 1999; Huang et al, 2001]. Because manipulated robots and wheeled robots have different characteristics, in the context of Internet-based teleoperation, they call for different control paradigms. The manipulated robots are often located in a limited or known workspace. Direct control is the popular control paradigm for Internet-based manipulated robot

teleoperation. To alleviate the problem of uncertain time delay, three approaches [Luo & Su, 2003] are often used in such systems: the predictive aiding approach, the simulating and planning display approach [Sayers, 2002; etc.], and the event-based approach [Elhajj et al, 2003]. Our focus is on the field of wheeled robot teleoperation.

For Internet-based wheeled mobile robot teleoperation, some systems have used direct control [Han et al, 2001; etc.]. The typical example is the KhepOnTheWeb system [Saucy & Mondada, 2000], in which Web users, via clickable images fed back from a camera, are able to control the robot's movements as it moves within a small wooden maze. Obviously, the direct control is not suitable for Internet-based mobile robot teleoperation because of the high latency derived from the Internet, such as restricted bandwidth, uncertain time delay, packet lost or error, and so on.

Supervisory control paradigm is commonly used for Internet-based mobile robot teleoperation [Luo & Chen, 2000; Simmons et al, 2000; etc.]. In this case, problems derived from the Internet are alleviated by giving the robot local intelligence. Unfortunately, most such systems lack adequate interaction between human operator and robot. We refer to this type of control paradigm as *passive supervisory control*. Passive supervisory control is inadequate in four ways: (1) The control interface is only able to provide single or limited available control methods (e.g. using a mouse to click a map); (2) The human operator can issue only very high-level instructions to the robot, and it is difficult to obtain the robot’s running status or information about the events the robot has encountered; (3) The robot has considerable autonomy but lacks the interaction with the human operator; (4) It is often needed to let the robot know some environmental knowledge in advance for path planning or self-localization, which causes that it is difficult to be applied in an unknown and highly dynamic environment. The typical examples are Xavier, an office-exploring robot at CMU [Simmons et al, 2000], and the museum tour-guide robot RHINO and MINERVA [Thrun et al, 1999]. They allow Web users to take the goal control, but the robots must know some global environment knowledge in advance. The control mode used on the Mars lander [Backes et al, 2002] can also be categorized as passive supervisory control. The human operators on earth use a Web-based tool to specify multiple waypoints as the navigation subgoals in 3D views of the landing site. These waypoints were generated from images obtained using stereo cameras on the lander.

One important characteristic between human and robot is interactivity. Simmon et al summarized their lessons from the 5 year (Dec. 1994 – Dec. 1999) public Xavier experiment as follows [Simmons et al, 2000].

"Autonomy can help in reducing the bandwidth requirements for control but this introduces problems of its own, particularly in the area of interactivity. People seem to prefer 'hands on' control. ….. The only real negative impact of autonomy on web-based interaction is that commanding at a high level is not as interactive as (conventional) teleoperation."

Saucy et al also pointed out the significance of interactivity over the 1 year (May 1997 – May 1998) KhepOnTheWeb system that was accessible to the public [Saucy & Mondada, 2000].

"Another problem is obviously the delay that prevents people from having a good interaction and from taking interests in the site. That's one reason why users do not come back."

Researchers are attempting to add more interaction between humans and robots [Chung et al, 1998; etc.], such as behavior-programming control [Luo & Chen, 2000], supervised autonomy [Cheng & Zelinsky, 2001], shared control [Rybski & Stoeter, 2002], cooperative control [Bourhis & Agostini, 1998], collaborative control [Fong et al, 2003], and fitting autonomy [Vieira et al, 2001]. We refer to these control modes as *active supervisory control* or *interactive control*. Section 5.6 provides a detailed analysis and comparison of these control modes. The main deficiencies of these control modes are: (1) They lack a complete framework to process the commands that can be sent continuously from human operator; (2) They are difficult to evaluate the online running performance and provide the corresponding response actions; (3) Their components are interdependent, which means that one poor component may cause multiple tasks fail; (4) The interfaces are not sufficiently human-friendly and they cannot provide a multi-modal control interface. In this chapter, we attempt to address the problems faced by the existing passive supervisory control and interactive control

methods by proposing a new interactive control approach, telecommanding, for Internet-based mobile robot teleoperation.

The rest of the chapter is organized as follows. Section 5.2 proposes the theoretic framework of telecommanding. Section 5.3 introduces our teleoperation experimental platform. Section 5.4 presents the simulation and real world experiments. Section 5.5 gives a comparison between telecommanding and other approaches. Section 5.6 summarizes the chapter.

5.2 The proposed teleoperation approach

The section describes in detail a proposed teleoperation approach: telecommanding. Section 5.2.1 proposes a theoretic framework of telecommanding, which involves two different but complimentary teleoperation commands: joystick commands and linguistic commands. Section 5.2.2 describes the design of joystick commands, and Section 5.2.3 the design of linguistic commands.

5.2.1 The framework of telecommanding

Central to the telecommanding framework is its use of two kinds of control commands: joystick commands and linguistic commands. Imagine a complex navigational task guided by a human. It may be, for example, to guide a bewildered person out of a maze (see Figure 5.1) from the start A to the goal D. The human guide may use three possible methods. One is that the guide directly guides the person step by step. The second method is that the guide may simply give directions, just like someone giving instructions on how to reach the nearby post office: “Turn right and move forward 50 meters, then turn right and go to the end. Next take a right, move forward 100 meters. And there’s the post office.” The third method is to use a map and point out the coordinates of three waypoints (B, C and D) or only the coordinate of goal D with respect to the start A. We call the instructions used in the first method “*joystick commands*”, and the instructions used in both the second and third methods “*linguistic commands*”.

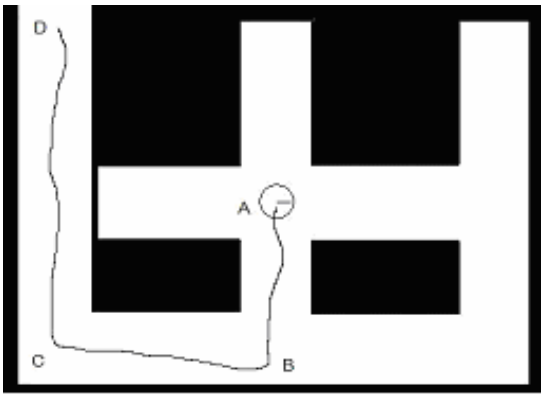


Figure 5.1: The robot in a maze. A is the start, D is the goal.

Figure 5.2 illustrates the framework of robot telecommanding with its multimodal (joystick/linguistic) control interface. Beginning on the left, we can see that the remote human operator issues joystick commands either via the computer keyboard or a real joystick device. As in our example in the preceding paragraph, linguistic commands may be issued in two ways: via an interactive command window in a graphical display interface (words) or via a computer mouse by clicking in the graphic window of a display interface (a map). The display interface shows visual feedback from a camera mounted on the robot, the history and current status, as well as the visualized pose and the obstacles. The Command Parser is responsible for parsing the joystick or linguistic commands from the local computer, then transferring them to the corresponding Command Processor for further command processing and to be passed on for execution at the Command Executor. The Sensing Update Module captures raw exteroception and proprioception sensory data from the robot's sensors. The Command Executor, the robot, and the Sensor Update Module form a reaction loop which enables the robot to react rapidly to unexpected events. Expected events are detected by the Sensing Transformation Module, which transforms raw sensing data into high-level data (e.g. total distance travelled). These expected events provide data from which the command processor can autonomously make a deliberative plan, allowing the robot to respond to the current situation. The Command Processor, Command Executor, Robot, Sensing Update, and Sensing Transformation Modules together constitute a deliberative loop. The larger loop, which includes the human operator, forms a complete telecommanding system.

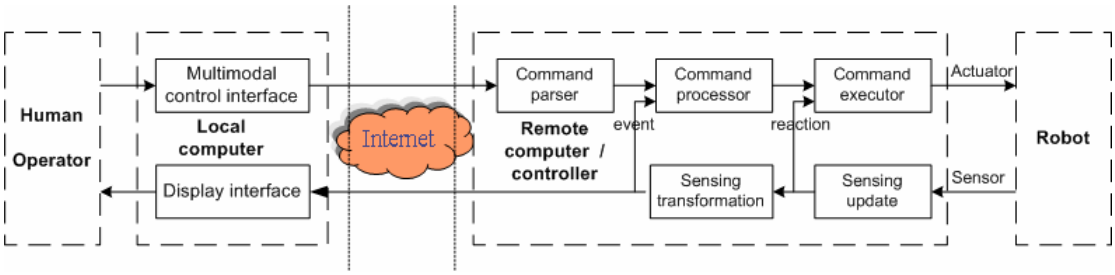


Figure 5.2: The framework of Telecommanding

Joystick commands and linguistic commands are exclusive. When the human operator sends a joystick command, all previous linguistic commands are discarded. Similarly, linguistic commands invalidate previous joystick commands. Linguistic commands, however, can be sent continuously, are stored in an ordered command queue and applied according to a FIFO (first-in-first-out) policy.

In every robot control cycle (e.g. 100 ms in our robotic system), the deliberative loop executes an inference process to output the next set of low-level motor controls. The inference process I can be defined as a relationship between the input space U and the output space Y. The input space U is multidimensional, with each dimension u_i corresponding to a particular input data mode derived from the sensing transformation module, e.g., distance to front obstacle, direction to the goal, the total moving distance, or distance to the goal. Similarly, the output space Y is multidimensional, with each dimension corresponding to a particular type of output, normally motor speed v and delta turn angle ω. This is expressed by $I: U(u_1, u_2, \cdots, u_i, \cdots, u_n) \rightarrow Y(v, \omega)$

We define some terms in the following, which will be used in the design of both joystick and linguistic commands.

Definition 5.1 (Event): Let e_i be a subset of input space $U(u_1, u_2, \cdots, u_j, \cdots, u_n)$, $i \in \{ 1, 2, \ldots m \}$, and $U = e_1 \cup e_2 \cup \cdots \cup e_i \cdots \cup e_m$, so e_i is called an event.

Definition 5.2 (Event occurs): Let e_i be an event, and x_t an input vector at the time t, $x_t \in U(u_1, u_2, \cdots, u_i, \cdots, u_n)$. If $x_t \in e_i$, so the event e_i occurs, otherwise the event e_i does not occur.

Definition 5.3 (Response function): Let e_i be an event. When the event e_i occurs, the robot should output the response actions y_t according to a function $f_{e_i}(x_t, y_{t-1})$, where x_t is the input vector at the current time t, $x_t \in U(u_1, u_2, \cdots, u_i, \cdots, u_n)$, y_{t-1} is the output vector at the time *t-1*, $y_{t-1} \in Y(v, \omega)$, so $f_{e_i}(x_t, y_{t-1})$ is called the response function of the event e_i.

Definition 5.4 (Command function): Let *N* be a joystick or linguistic command. The corresponding command execution function $f_N(x_t, y_{t-1})$ is called the *command function* associated with the command N, where x_t is the input vector at the current time t, $x_t \in U(u_1, u_2, \cdots, u_i, \cdots, u_n)$, and y_{t-1} the output vector at the time t-1, $y_{t-1} \in Y(v, \omega)$.

For example, suppose u_1 is an input variable, denoting the distance between the robot and the frontal obstacle. An event e_i can be defined as $e_i = \{(u_1, \cdots, u_n) \mid 0.5 \le u_1 < 1\}$. If $x_t \in e_i$, the event e_i occurs, whose physical mean is such that if the distance between the robot and the frontal obstacle is in a range of 0.5 to 1.0 meters, the robot should not respond to the command from the human operator but should autonomously calculate the motor outputs in accordance with the response function $f_{e_i}(x_t, y_{t-1})$. For instance, a response function is simply defined as $v_t = f_{e_i}(x_t, y_{t-1}) = v_{t-1} - 100$, where v_{t-1} is the speed of the robot at the time t-1. This response function does not affect the turn angle of the robot.

5.2.2 Telecommanding using joystick commands

Telecommanding using joystick commands is in some ways similar to what we find in a car driven by a human. Like a driver using a steering wheel, the human operator uses <Left and Right> joystick commands to steer the robot. Like the driver using the

accelerator and brake, the operator uses <Up and Down> joystick commands to accelerate or decelerate, even to stop or to reverse the robot.

In other ways, the use of joystick commands substantially differs from the use of a human driving a car with the principal difference being that the latter applies a traditional direct control: the driver receives environmental information in real-time through the human vision and through the car instrument, and from this simultaneously builds a real world model; the driver can respond immediately to any contingency and the driver's actions are immediately effective; the car typically lacks autonomous intelligence and relies on the driver to handle unexpected events. In short, a human driver must continuously provide input about steering or acceleration. But given issues such as restricted bandwidth, uncertain time delay in Internet-based teleoperation, it is desirable that human operators send the remote control as few commands as possible and that the robot should have an autonomous capability to respond some expected events as well as to react rapidly to contingencies, so that human operators do not need to handle the control details.

In the telecommanding, joystick commands enable the human operator to send as few commands as possible since the robot uses local intelligence. As the robot would continue to execute a joystick command until otherwise instructed, the operator sends such commands only if necessary. This greatly reduces the number of commands an operator must issue. In addition, the robot may autonomously make judgments about situations. If it becomes aware of some impending danger, for example, nearby obstacles, the robot autonomously decreases the speed to a reasonable value while turning toward a safer direction. If the danger is immediate (e.g. someone suddenly blocks the path), the robot stops. In such situations, the internal autonomous behavior of the robot dominates the control privilege. Potentially, the robot may not respond to the human's joystick commands until it thinks the current danger has passed or unless the joystick command makes it safe.

In the implementation of our telerobotic system, we define four joystick commands (UP, DOWN, LEFT, RIGHT) and the corresponding joystick command functions as the Eqs. (6.1)-(6.4).

$$v_t = f_{UP}(x_t, y_{t-1}) = \begin{cases} v_{t-1} + \Delta v & ,\text{if } v_t < v_{max} \\ v_{max} & ,\text{otherwise} \end{cases} \qquad (6.1)$$

$$v_t = f_{DOWN}(x_t, y_{t-1}) = \begin{cases} v_{t-1} - \Delta v & ,\text{if } v_t > v_{min} \\ v_{min} & ,\text{otherwise} \end{cases} \quad (6.2)$$

$$\omega_t = f_{LEFT}(x_t, y_{t-1}) = \Delta\theta \quad (6.3)$$

$$\omega_t = f_{RIGHT}(x_t, y_{t-1}) = -\Delta\theta \quad (6.4)$$

where, v_t, ω_t are the output variables at the current time t, respectively denoting the speed and delta turn angle of the robot. v_{max} and v_{min} are respectively the minimum and maximum bounds of the speed. Because our robot does not have rear sensors, we set v_{min} =0, meaning that the robot is not allowed to reverse. Δv is a constant parameter (mm/s), and $\Delta\theta$ is a constant parameter (degree/s). The joystick commands UP and DOWN affect the speed of the robot while LEFT and RIGHT affect only the robot's steering angle.

In addition, we need to define the corresponding joystick events and response functions associated with these four joystick commands. For example, we define several joystick events associated with UP as $\{e_{U1}, \cdots, e_{Uk}\}$, and the corresponding joystick response functions as $\{f_{e_{U1}}, \cdots, f_{e_{Uk}}\}$, where $e_{Ui} \cap e_{Uj} = \varnothing$, i, j $\in$ { 1, 2, …, k }, which guarantees that there is only one joystick event associated with a joystick command that occurs.

$$e_{U1} = \{(\cdots, u_3, u_4, u_5, \cdots, u_n) \mid ((0.15 \le u_3 < 0.5) \vee (0.15 \le u_4 < 0.5)) \wedge (u_5 > 0)\},$$

$$e_{U2} = \{(\cdots, u_3, u_4, u_5, \cdots, u_n) \mid ((0 \le u_3 < 0.15) \vee (0 \le u_4 < 0.15)) \wedge (u_5 > 0)\},$$

where u_3 is a distance value (meter) to front obstacle, u_4 is a distance value to lateral obstacle, and u_5 is a currently actual speed of the robot. Obviously, $e_{U1} \cap e_{U2} = \varnothing$. Correspondingly, we simply define two joystick response functions as the Eq.(6.5).

$$v_t = \begin{cases} f_{e_{U1}}(x_t, y_{t-1}) = \dfrac{\Delta v}{2}, & \text{if } x_t \in e_{U1} \\ f_{e_{U2}}(x_t, y_{t-1}) = 0, & \text{if } x_t \in e_{U2} \end{cases} \quad (6.5)$$

In the actual implementation of joystick response function, we define such functions that enable the robot to autonomously decrease the speed to a reasonable value while to turn toward a safer direction.

In every robot control cycle, Algorithm 5.1 is called once. In this algorithm, we ignore the feedback of the running events but they should be displayed on the display interface of human operator's monitor.

Algorithm 5.1: JOYSTICKCOMMANDPROCESSOR( )

Input: $x_t(u_1, u_2, \cdots, u_i, \cdots, u_n)$, $y_{t-1}(v_{t-1}, \omega_{t-1})$

Output: $y_t(v_t, \omega_t)$

BEGIN:

Step1. IF there is a new joystick command, THEN

To calculate $y_t = f_N(x_t, y_{t-1})$ according to the corresponding Eqs.(6.1)-(6.4);

ELSE

$v_t = v_{t-1}$; $\omega_t = 0$; /* maintain the previous robot speed */

END IF

Step2. Detect and respond the joystick events associated with joystick command UP:

IF $x_t \in e_{U1}$, THEN $v_t = f_{e_{U1}}(x_t, y_{t-1})$; /* execute the response function */

… …

IF $x_t \in e_{Uk}$, THEN $v_t = f_{e_{Uk}}(x_t, y_{t-1})$;

Step3. Detect and respond to the joystick events associated with joystick commands DOWN, LEFT and RIGHT, similar to Step 2.

Step4. IF no events occur AND there is no joystick command, THEN

To maintain the current status of the robot;

ELSE

Output the low-level motor command $y_t(v_t, \omega_t)$;

END IF

END Algorithm 5.1

5.2.3 Telecommanding using linguistic commands

By telecommanding using linguistic commands, human operators do not care about the low-level control details. As an example in Section 5.2.1, a person gives a stranger a series of high level instructions to guide him to the nearby post office. The robot must follow these instructions ("linguistic commands") while at the same time autonomously handle unexpected events and avoid any static or dynamic obstacles (e.g. humans) in its path. Therefore, telecommanding using linguistic command can reduce the influence of the high latency of the Internet.

Robotics researchers are able to design any variety of linguistic commands and integrate them into the telecommanding framework to be adapted to specific tasks. In our research examples, we have designed linguistic commands (MOVE, TURN, WANDER, GOTOEND, COORDINATE, and MAPPING) to realize specific tasks. The human operator is able to continuously input these commands from the interactive command window, or by clicking on the special command COORDINATE in the graphical window, without the need of having to wait until the previous linguistic command is finished. If the command is correct and there are no exceptional events, the robot may follow these commands to reach the goal state. Otherwise, the robot enters the command exception handle module automatically.

Every linguistic command is stored in an ordered command queue that adopts the policy of FIFO (first-in-first-out). This command queue is a two dimensional array: *commandQueue[M][N]*, where

- *m* : denotes the m_{th} command. $m \in M = [0, +\infty)$
- *n* : denotes the n_{th} parameter of the m_{th} command. $n \in N = \{ 0, 1, 2, 3, 4\}$
- *commandQueue[m][0]* : the index number of this command type. E.g. MOVE_INDEX, or COORDINATE_INDEX.
- *commandQueue[m][1]*, *commandQueue[m][2]*: two working parameters of this command, e.g. (x, y) coordinate. The parameters are set to adapt flexibly to the real world model.
- *commandQueue[m][3]*, *commandQueue[m][4]*: two performance evaluation parameters of this command. Using the two parameters, the robot can evaluate the performance (success or failure) of execution result of current command, in order to make a decision to enter either the command exception handle module or the next command execution module.

  For the design of linguistic commands, we define the following terms.

Definition 5.5 (Target event): Let e_T be an event associated with a linguistic command N. If the event e_T occurs, the robot has reached the goal state. So e_T is called a *target event* associated with the linguistic command N.

Definition 5.6 (Overrun event): Let e_O be an event associated with a linguistic command N, and e_T a target event associated with N. If the event e_O occurs and e_T does not occur, the robot has met an overrun exception. So e_O is called an *overrun event* associated with the linguistic command N.

Definition 5.7 (Underrun event): Let e_U be an event associated with a linguistic command N, and e_T a target event associated with N. If both the events e_U and e_T occur, the robot has met an underrun exception. So e_U is called an *underrun event* associated with the linguistic command N.

In fact, each linguistic command can be defined as a seven-element tuple $\{\mathrm{N}, e_T, e_O, e_U, f_{e_O}, f_{e_U}, f_N\}$. N is the definition of this linguistic command, involving its name and parameters. The target event e_T is determined by two working parameters (*commandQueue[m][1], commandQueue[m][2]*) of the linguistic command. The underrun event e_U is determined by the first performance evaluation parameter *commandQueue[m][3]*. The overrun event e_O is determined by the second performance evaluation parameter *commandQueue[m][4]*. f_{e_O} and f_{e_U} are the corresponding overrun and underrun response functions respectively.

We explain the physical mean of these events using a linguistic command GOTOEND. When the linguistic command GOTOEND is running and the actual moving distance of the robot has already exceeded the expected maximum distance, the overrun event occurs. This means that the robot has received an incorrect command or encountered an exception (e.g., the goal is too far away or it is not reachable). When the command GOTOEND is finished because of satisfying the target event but the actual moving distance does not exceed the expected minimum distance, the underrun event occurs. This means that the robot has also encountered an incorrect command or an exception (e.g. someone suddenly blocks the path). In such two situations, the robot should then enter the command exception handle module to execute the corresponding overrun or underrun response function.

For the purpose of presenting how to design a linguistic command in terms of {N, e_T, e_O, e_U, f_{e_O}, f_{e_U}, f_N}, we will make use of a linguistic command MOVE. The formal definition of the linguistic command MOVE in our telerobotic system is as follow:

MOVE(double Distance, double minDistanceScale = 0.5, double maxDistanceScale =1.5)

If the human operator does not input *minDistanceScale* and *maxDistanceScale*, the default values are used. MOVE is a complex linguistic command. Its function is to enable the robot to arrive at a goal lying ahead of the current location, and to avoid any static or dynamic obstacles (e.g. box and human). MOVE is converted into the following style in the command queue for execution.

commandQueue(MOVE_INDEX, Distance, 0, minDistanceScale, maxDistanceScale).

The current location of the robot in the robot internal coordinate can be represented as a vector (x^0, y^0, φ^0), where φ^0 denotes the current absolute heading angle. The command goal location is (x^T, y^T, φ^T). The current location of the robot is (x', y', φ'). The expected minimum and maximum moving distance are respectively $d_{\min}$ and $d_{\max}$. Therefore,

$d_{\min}$ = minDistanceScale × Distance;

$d_{\max}$ = maxDistanceScale × Distance;

$x^T = x^0 + \text{Distance} \times \cos(\varphi^0)$;

$y^T = y^0 + \text{Distance} \times \sin(\varphi^0)$;

$\varphi^T = \varphi^0$

So a target event e_T can be defined as $e_T = \{(u_1, u_2, u_3, \cdots, u_n) \mid (u_1 < \lambda) \wedge (u_2 < \beta)\}$, where u_1 is the distance between current location of the robot and the goal, and $u_1 = \sqrt{(x' - x^T)^2 + (y' - y^T)^2}$. u_2 is the angle difference between current heading angle of the robot and the heading angle of the goal. $u_2 = |\varphi' - \varphi^T|$. λ is a constant that denotes the minimum tolerance for u_1. β is a constant that denotes the minimum tolerance for u_2. An underrun event e_U can be defined as $e_U = \{(u_1, u_2, u_3, \cdots, u_n) \mid u_3 < d_{\min}\}$. An overrun event e_O can be defined as $e_O = \{(u_1, u_2, u_3, \cdots, u_n) \mid u_3 > d_{\max}\}$, where u_3 is the actual moving distance of the robot.

The overrun and underrun response function (i.e. f_{e_O} and f_{e_U}) can be simply designed to enable the robot to cancel all subsequent linguistic commands in the command queue while to stop the movement of the robot. A more sophisticated strategy is to enable the robot to reschedule the command queue. The command function f_N is the most important element of a linguistic command. The command function of MOVE is implemented to realize a goal-oriented navigation. The main idea of such function is to decompose the task into two behaviors (i.e. goal-seeking, and obstacle-avoidance) and navigate the robot to the goal location through the coordination of the two behaviors. The details about goal-oriented navigation are previously described in Section 4.3 of Chapter 4.

In every robot control cycle, Algorithm 5.2 is called once.

Algorithm 5.2: LINGUISTICCOMMANDPROCESSOR( )

Input: $x_t(u_1, u_2, \cdots, u_i, \cdots, u_n)$, $y_{t-1}(v_{t-1}, \omega_{t-1})$

Output: $y_t(v_t, \omega_t)$

BEGIN:

Step1.

IF a target event e_T of current linguistic command occurs, THEN

IF an underrun event e_U occurs, THEN

$y_t = f_{e_U}(x_t, y_{t-1})$; /* execute the underrun response function */

ELSE

IF the next linguistic command can be obtained from the command queue via FIFO, THEN

To set the e_T, e_U e_O of this command, and go to the Step 1 again;

ELSE

$v_t = 0$; $\omega_t = 0$; /* stop the robot */

END IF

END IF

ELSE

IF an overrun event e_O occurs, THEN

$y_t = f_{e_O}(x_t, y_{t-1})$; /* execute the overrun response function */

ELSE

$y_t = f_N(x_t, y_{t-1})$; /* execute the linguistic command function */

END IF

END IF

Step2. Output the low-level motor command $y_t(v_t, \omega_t)$.

END Algorithm 5.2

Each linguistic command should be designed to autonomously perform an independent task. More linguistic commands, such as light-seeking, door-crossing, wall-following, can be designed to perform more complex tasks. The linguistic command MOVE has been described in the above paragraphs. The definitions of other linguistic commands in our telerobotic system are as follows:

TURN(double deltaAngle)

GOTOEND(double minDistance = 0, double maxDistance = INFINITE)

WANDER(double totalDistance, double minDistanceScale=0.5, double maxDistanceScale=1.5)

COORINDATE(double x, double y, double minDistanceScale=0.5, double maxDistanceScale=1.5)

MAPPING(double totalDistance, double minDistanceScale=0.5, double maxDistanceScale=1.5)

TURN is a simple linguistic command, whose function is to enable the robot to rotate. GOTOEND is an interesting linguistic command that enables the robot to reach the end of a routeway (e.g. corridor end) while to avoid any lateral obstacles. Three types of routeway shown in Figure 5.3 are particularly suitable for the use of this linguistic command. The linguistic command WANDER enables the robot to wander randomly without collision with any obstacles. It is simply realized using an obstacle-avoidance behavior. Fuzzy logic is used for the behavior design. COORDINATE is a complex linguistic command that enables the robot to move from the current location to a given goal location. The robot does not have any a priori known environmental knowledge. As it happens, MOVE and COORDINATE share the same linguistic command function as presented in Section 4.3 of Chapter 4. Unfortunately, such realization of goal-oriented navigation makes it easy that the robot gets trapped in a dead end, which is the local minimum problem encountered by autonomous robots in unknown environments. Chapter 7 will present a new navigation method (i.e. an enhanced COORDINATE linguistic command) to address this problem. MAPPING is a complex linguistic command that enables the robot to

autonomously explore unknown environment and build a map based on robot's sensory information. We will present the implementation of MAPPING in Chapter 6.

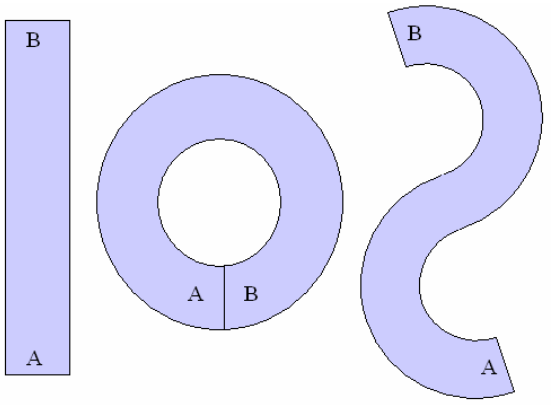


Figure 5.3: Three types of routeway that are suitable for the use of linguistic command GOTOEND. A is the start, B is the end.

5.3 Teleoperation platform

A platform for Internet-based teleoperation using telecommanding is shown in Figure 5.4. The research is tested on a mobile robot with eight forward ultrasonic sensors. The control commands transfer through radio Ethernet devices, and the video data is feedback through a set of A/V transmitter-receiver from a pan-tilt-zoom camera mounted on the robot deck. The work style and the use of streaming video are previously presented in Section 3.2 of Chapter 3.

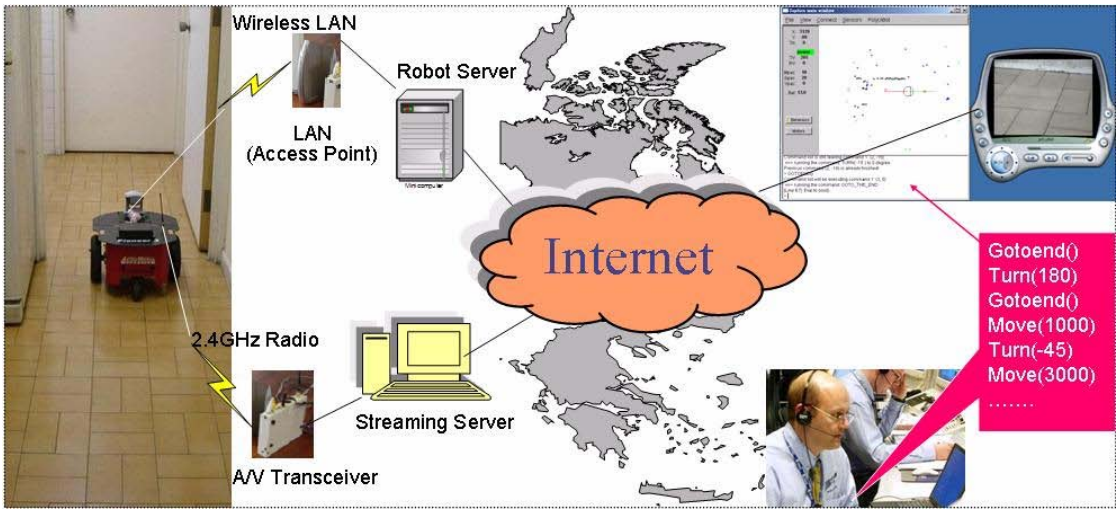


Figure 5.4: A platform for Internet-based teleoperation using telecommanding

The data visualization (e.g. sonar reading visualization, virtual trajectory display) is developed to complement the video transmission technique. The streaming video provides global environment feedback, and truly improves the quality of services over the low-bandwidth and unreliable Internet, producing a more stable system, higher image resolution, and smoother images. On the other hand, the video transmission time delay is large (about 8 seconds through the campus Internet, and about 12 seconds through the 33.6 Kbps Internet connection over a telephone line). This time delay is caused mainly by the encoder and decoder buffers, which are used to guarantee quality of service. Therefore we develop the data visualization using sonar readings and dead reckoning data in order to obtain more timely perceptual feedback (less than 1 second to transfer in our campus Internet). The human operator can obtain the robot's global context information through the video feedback, and obtain the local context information through the data visualization. This enables the operator to easily predict the next control command, and improves the efficiency of teleoperation.

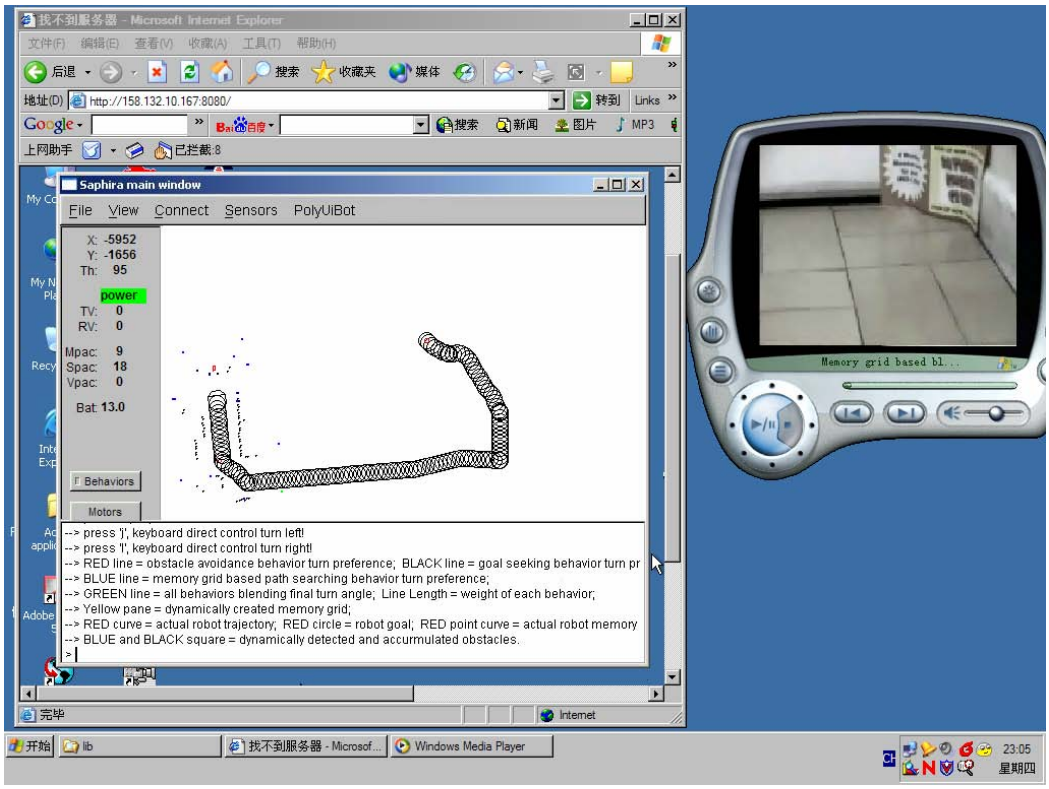


Figure 5.5: The display and control interface. Joystick commands are sent via the computer keyboard; linguistic commands are sent via the bottom command window or clicking in the graphic window. The commands and sensory information are transferred via the VNC Web service, and video images are transferred based on streaming technology.

Currently, unlike the other existing Internet telerobotics projects [Taylor & Trevelyan, 1995; etc.], we have not built our own Web-based data transmission system. Instead, we use an Ultr@VNC service [http://ultravnc.sf.net] for simplification of development workload, which is a reliable and convenient way for Web users to connect with the robot server. The display and control interface is shown in Figure 5.5. On the other hand, the VNC service is not an efficient teleoperation service since it consumes extra bandwidth for unnecessary data transmission. Moreover, through this service, the human operator actually dominates the control privilege of the robot server so that it is not safe for real public application.

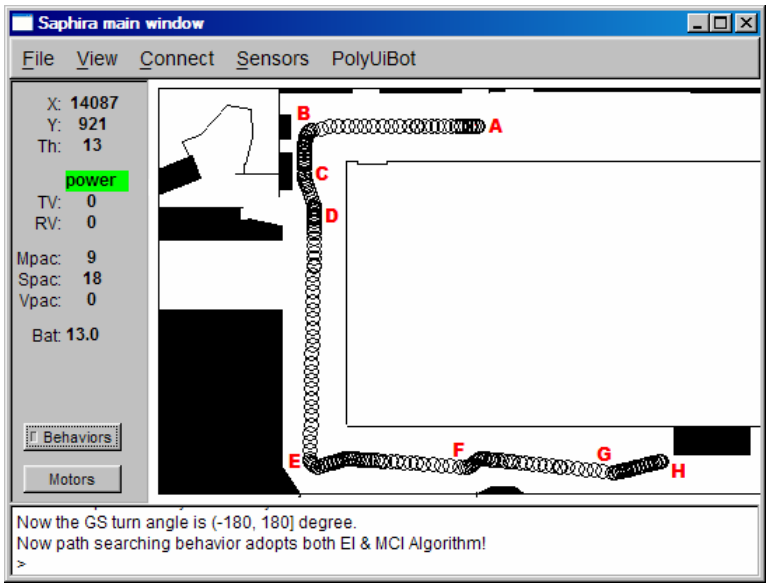
5.4 Experimental results

In this section, we perform the simulated and real world experiments to test the performance of the proposed telecommanding approach. In Section 5.4.1 Internet-based teleoperation uses joystick commands in both the simulation and the real world. In Section 5.4.2, teleoperation applies linguistic commands. Section 5.4.3 presents a robot teleoperation over a long distance. Section 5.4.4 provides a performance and stability analysis.

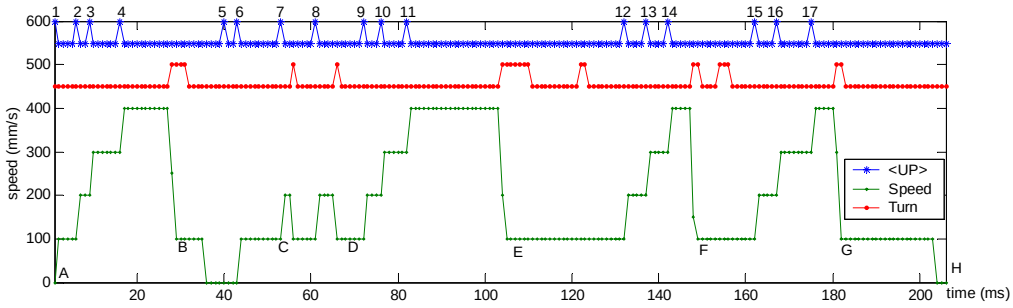
5.4.1 Teleoperation using joystick commands

We conducted a simulation to test the performance of joystick commands as shown in Figure 5.6(a). To make the robot move from the start A to the goal H, the human operator uses the joystick commands <UP, DOWN, LEFT, RIGHT>. The trajectory is indicated by a chain of black circles. The program draws a circle once every 0.5 second. A denser concentration of circles (e.g. B to C in Figure 5.6(a)) thus indicates that the robot is traveling more slowly. Figure 5.6(b) shows the relationship of the robot speed, the robot turn and the joystick command <UP>. The robot begins to increase its speed from location A. Every time the robot receives an <UP> joystick command (see Figure 5.6(b) 1^{st}, 2^{nd}, 3^{th}, 4^{th}, etc. <UP> command), the speed increases 100 mm/s until it reaches the predefined maximum bound (400mm/s). When the robot is approaching obstacles, it may not respond to the <UP> command from the human operator and may autonomously decrease its speed to a reasonable value 100mm/s and turn in a safer direction (see B to G in Figure 5.6(a) and (b)). When the robot is very close to obstacles, its speed is autonomously set to 0 mm/s (see B and H in

Figure 5.6). For simplicity of implementation, in this test, we simply define the joystick events and corresponding joystick response functions.



(a)



(b)

Figure 5.6: Teleoperation simulation using joystick command. (a) the robot is moved from the start A to the goal H. (b) The relation of the robot speed, robot turn and joystick command <UP>. Each peak (*) in the curve represents an <UP> joystick command. Each peak (•) in the curve represents a turn action that is automatically produced by the control program. There are 17 <UP> commands in total. (A-H) correspond to the locations in (a).

Next we test the Internet-based teleoperation in our department corridor using joystick commands (see Figure 5.7). A remote human operator controlled a real robot through the campus Internet. The remote operator used the experimental platform and control interface discussed in Section 5.3 and found it convenient to control the robot using joystick commands. The operator did not need any robotic expertise as it is just

doing like playing a game using the keyboard's <Up, Down, Left, Right> keys. The task is completed in about 180 seconds. The maximum speed was 400mm/s. The actual average speed of the robot was 45000/180 = 250mm/s. During the process, the robot autonomously decelerates and turns in a safer direction if it is approaching an obstacle, and stops if danger is imminent (e.g. someone suddenly blocks the path). As a result, the robot was able to avoid collisions with obstacles (e.g. walls, desks, boxes, and humans), even though we purposely disconnected the network cable in order to lose the Internet connection for a period.

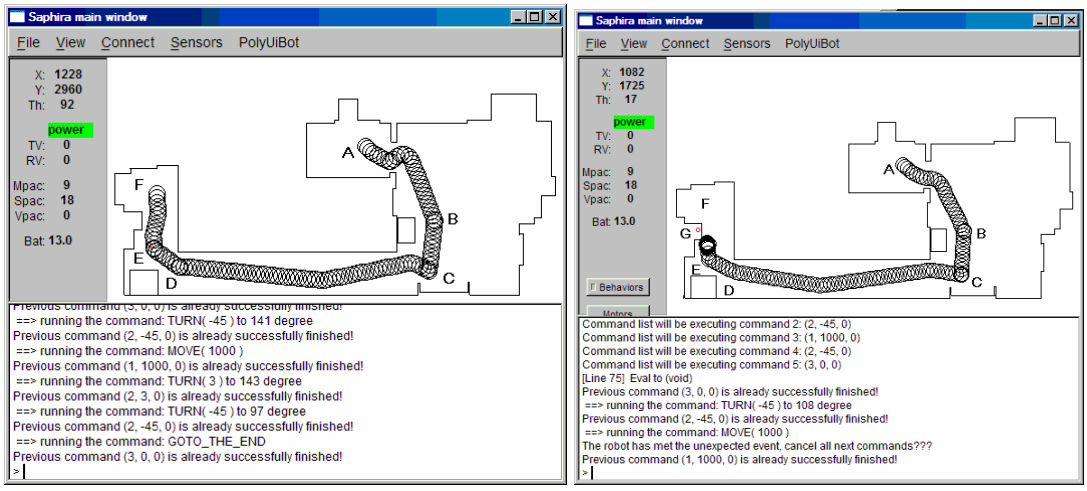
Figure 5.7: Joystick commands for use in Internet-based teleoperation. The robot moves in our department corridor under remote control. The corridor is about 45 meters long and contains two corners and a number of obstacles.

5.4.2 Teleoperation using linguistic commands

First we conduct a simulation experiment to demonstrate how to control a robot so that it can navigate in a complex and unknown space using linguistic commands. As shown in Figure 5.8(a), the remote operator guides the robot by continuously sending a series of linguistic commands. The instructions are to first move forward (MOVE) to the location B; then take a right turn 45^0 (TURN) and go to the end C (GOTOEND); next turn right 90^0 (TURN) and go to the end D (GOTOEND); turn right 45^0 (TURN) and move forward (MOVE) to the location E; finally turn right 45^0 (TURN) and go to the end (GOTOEND) to reach the final goal F. Those commands are stored in a command queue and allowed to execute only after completion of the previous command. This test was successful without any occurrence of overrun or underrun events. By storing all of those linguistic commands in the robotic system, we obtain a new linguistic command GOTO_ROOM_F. The learned linguistic command GOTO_ROOM_F was reissued from location A again as shown in Figure 5.8(b). Because the sensing data are not identical with data of the previous test, the trajectory of the robot was a little different. When the command GOTOEND( ) is running from location C, the target event does not occur at location D, but occurs at location E. The

subsequent MOVE(1000) is wrong because it causes the robot to move toward location G, which is not reachable. This leads to an overrun event when the robot tries to move around to reach location G. This triggers the corresponding command exception handle module. In our telerobotic system, the robot is simply stopped autonomously so that the human can send further commands.

Those linguistic commands reduce the length and complexity of the command list, making it suitable for Internet-based teleoperation. For example, the robot is expected to move forward three meters from location A to location B. But a corner blocks its path (see Figure 5.8). Our MOVE command provides a convenient way to reach the goal. To avoid the obstacle, the operator needs only send the command MOVE(3000) instead of a lot of low-level commands. The linguistic commands are particularly useful in the dynamic real world.



(a) (b)

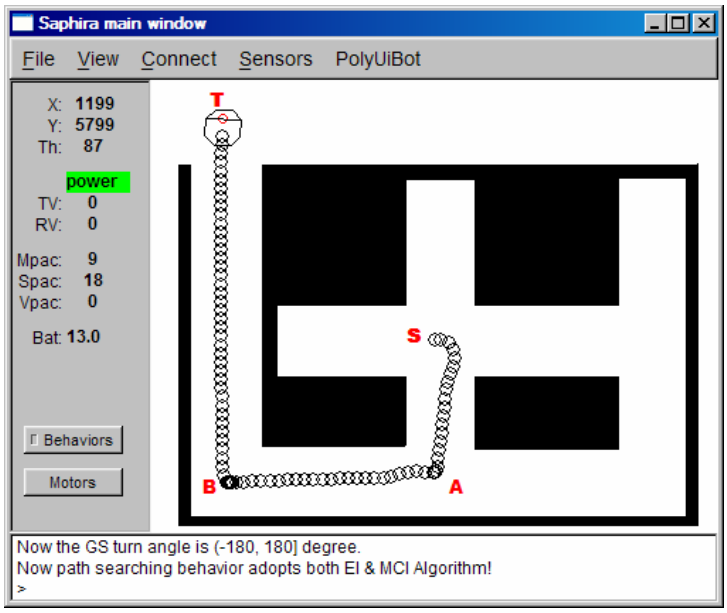
Figure 5.8: Simulation using linguistic commands from the start A to the goal F. (a) GOTO_ROOM_F= {MOVE(3000),TURN(-45),GOTOEND(),TURN(-90), GOTOEND(),TURN(-45),MOVE(1000),TURN(-45),GOTOEND()}; (b) Learned linguistic command GOTO_ROOM_F is executed. An overrun event occurs when MOVE(1000) is running at the location E.

Next we test an Internet-based teleoperation in real world using linguistic commands to navigate a robot in a complex house (see Figure 5.9). The remote operator observes the robot's surroundings through the streaming video feedback. The perceptual data visualization provides more timely local information. The remote operator sends a series of linguistic commands to let the robot move to the end of the

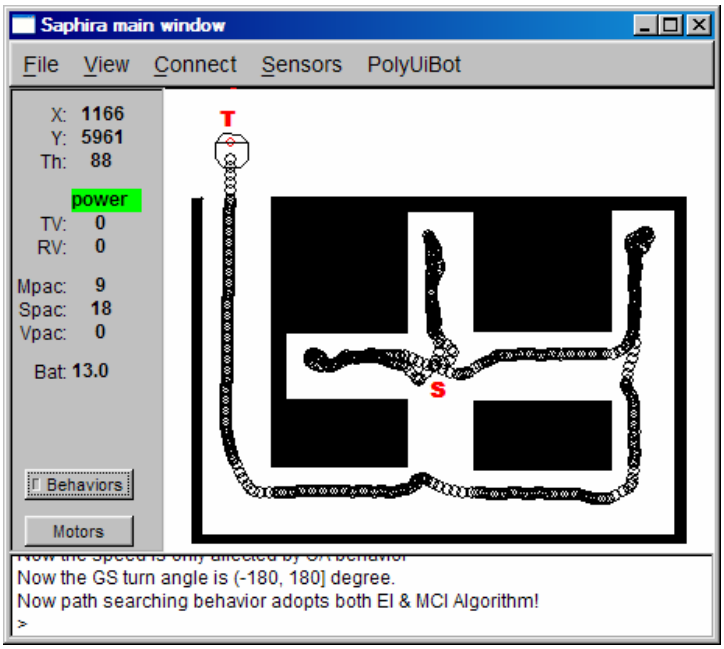
path, return to the cross, then turn and go to the final goal. As the robot is highly autonomous, its maximum speed is set at 200mm/s. The experimental task took about 100 seconds without any collisions, and the trajectory was about 13 meters long, making the average speed 13000/100 = 130mm/s. We ran the same Internet-based teleoperation test using direct control without robot intelligence. The task took over 300 seconds and on several occasions the robot collided with walls or doors because of the large time delay derived from the video feedback.

Figure 5.9: The use of linguistic commands in real world for Internet-based teleoperation. The remote operator sends a series of linguistic commands: {GOTOEND(), TURN(180), GOTOEND(), TURN(40), MOVE(2000), TURN(90), MOVE(1000), TURN(50), MOVE(3000)}

Finally we conduct the experiments to demonstrate the COORDINATE linguistic command. As shown in Figure 5.10, the robot does not have a priori knowledge about the map of a maze. It is required to move from the start S to the goal T. By clicking a mouse, the human operator is able to send the COORDINATE command. The simplest way to do this is to continuously give out three COORDINATE commands (see Figure 5.10(a)). This allows the robot to pass by the waypoints A and B from the start S to the goal T, thereby escaping from the dead ends. Another way is using an enhanced COORDINATE command to simply point out the final goal location T. The robot is able to autonomously look for the safest regional direction and escape from the dead end. The experimental results are encouraging. The robot autonomously finds the correct path out of the maze (see Figure 5.10(b)). More tests are shown in Figure 5.11.

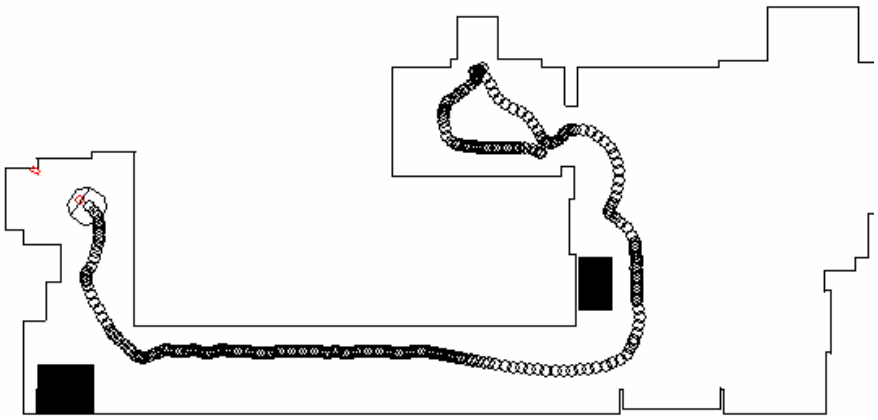


(a)

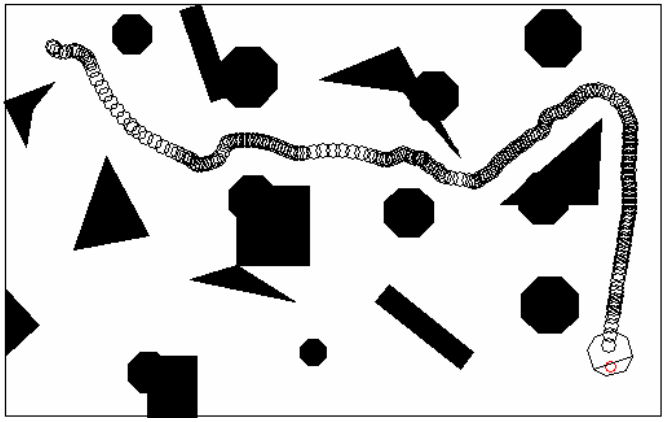


(b)

Figure 5.10: The goal-oriented navigation in a maze from the start S to the goal T. (a) Three COORDINATE command (S->A->B->T) by the use of the command queue; (b) One enhanced COORDINATE command (S->T). The robot autonomously searches the solution path by coordinating three behaviors: path searching, obstacle avoidance, and goal seeking behaviors.



(a)



(b)

Figure 5.11: The goal-oriented navigation using one enhanced COORDINATE command. (a) in a structured environment. (b) in a cluttered and unstructured environment.

5.4.3 Robot teleoperation over a long distance

We tested the Internet-based teleoperation using telecommanding over a long distance from Beijing to Hong Kong (over 1500 kilometers). A remote human operator (located in Beijing) connects with the robot server (located at our department in Hong Kong) through the VNC service, and observes the robot's surroundings (our department corridor) through streaming video (50Kbps). Combining the graphical control interface and local perceptual data visualization, the remote operator can determine and send the telecommanding commands. The experiment has tested the use of joystick commands and linguistic commands. The operator has no robotic expertise and he is told the teleoperation commands only at the beginning of the test. The test demonstrates that the telecommanding is interactive, effective, and easy to

use. The telerobotic system was publicly demonstrated and was well received at the Hong Kong Convention and Exhibition Center in April, 2004. Here are some video clips shown in Figure 5.12.

Figure 5.12: The robot is navigating by telecommanding in the Hong Kong Convention and Exhibition Center. Some audiences think the robot is "alive" and they are interested in testing the response of the robot by blocking its path ahead.

5.4.4 Performance and stability analysis

As shown in the above real-world tests, telecommanding using joystick commands is easy to use and is reliable even in a crowded exhibition center. Joystick commands provide the human operator 'hands-on' control, giving the operator a strong feeling of interaction with the robot. On the other hand, the human operator does have to spend more effort on the control details if joystick commands are used in the highly dynamic environment. It should also be noted that joystick commands are not suitable for carrying out some more skilful tasks (e.g. finding and entering a door located in the lateral wall of the corridor) if an uncertain or long time delay exists (e.g. through the Internet or the space).

Telecommanding using linguistic commands compensates for the disadvantages of using joystick commands. Moreover, it can evolve and obtain more high-level linguistic commands by learning the linguistic commands queue from the human operator. Each linguistic command can be designed and used independently. This means that one poor linguistic command may not affect the performance of others. The use of linguistic commands is easy to be accepted by inexperienced users so that it does not require expertise. Linguistic commands are suitable for the use in the environments affected by uncertain or long time delays. The disadvantages of telecommanding using linguistic commands are that a linguistic command function involves quite complicated design and that we currently lack an explicit standard to define exception detection and responses.

The stability of Internet-based telerobotic system is affected by the uncertain time delay that results in the loss of synchronization of time based action references. In the telerobotic system by telecommanding, the predefined events, e.g. the distances to obstacles, are non-time-based action references that are non-decreasing functions to time. These events are independent of the uncertain time delay, and drive the robot to output actions in accordance with predefined response functions. Xi & Tarn [2000] have previously proven the theoretical stability of non-time referenced Internet-based telerobotic systems, and now it has been demonstrated via our tests in simulation and in the real world.

5.5 Comparison with other control approaches

It is obvious that the direct control is unsatisfactory for use in Internet-based mobile robot teleoperation because of the high latency derived from the Internet such as restricted bandwidth and uncertain time delay. Passive supervisory control is unsatisfactory mainly in that it fails to provide adequate human-robot interactivity. Table 5.1 provides a comparison of these approaches with the telecommanding.

Table 5.1: Comparison of related systems under various teleoperation approaches

	Direct control	**Passive supervisory control**	**Telecommanding (interactive control)**
Command type	low-level speed & angle	determined goal coordinate	joystick & linguistic commands
Command send	Continuous	One by one	Both
Command level	Low	High	High
Task efficiency	Low	High	High
Semi-autonomy	None	High	High
Stationary environment	Feasible	Feasible	Feasible
Dynamic environment	Dangerous	Feasible	Feasible
Internet connection lost	Dangerous	Safe	Safe
Complex task	Difficult	Feasible	Feasible
Human-robot interactivity	Good	Poor	Good
Real world applicability	Good	Poor	Good
Easy to use	Easy	Easy	Easy

Researchers are attempting to add more human-robot interaction in the form of behavior-programming control, fitting autonomy, supervised autonomy, shared control, cooperative control, collaborative control, and so on. We refer to these strategies as active supervisory control or interactive control. Chung et al [1998] propose a control strategy consisting of three major parts: behaviors, planner, and coordinator. The coordinator produces a wake-up table that contains all behaviors which should be scheduled by a real-time robotic system. Each task action must be defined with three conditions: pre_activate, fire_condition, and post_activate. These conditions respectively denote a group of behaviors that must be activated at the corresponding time. This kind of control strategy is too complex and the running performance of the robotic system is difficult to evaluate. One poor behavior could lead to the failure of multiple tasks. The human operator also finds it difficult to provide suggestions for action via the command parameters. Similar problems arise in the behavior-programming control mode proposed by Luo et al [2000]. In behavior-programming control mode, the event derived from a motion assistant is used on the robot to select a behaviour that is suitable in the encountered situation. Vieira et al [2001] proposed a concept of fitting autonomy, which allows the mobile agents to adapt its high level abstract plan to the exact environment it finds in remote places and to execute the adapted plan including execution monitor and error recovery. This concept is evaluated in the field of mobile manipulator teleoperation.

Gordon Cheng et al. [2001] propose a teleoperation paradigm: supervised autonomy. This allows some qualitative instructions (e.g. Go Forward, Go Toward, Go Between, or Keep To) to be implemented using a vision-based approach. These instructions are slightly similar to our linguistic commands, but they lack the performance evaluation and it allows instructions to be sent only one at a time. More importantly, this paradigm does not have a complete framework for command processing and event response. The shared control in literature [Lin et al, 1996] is similar to our joystick command, but it is rather simple and lacks the corresponding events definition and response functions. Bourhis and Agostini [1998] used cooperative control to control an intelligent wheelchair. This allows, at certain times, both the robot and the human to become the supervisor. Three types of behaviors are defined: skill-based, rule-based, and knowledge-based behaviors. The collaborative control in [Fong et al, 2003] is a model based on human-robot dialogue. Both the

cooperative control and collaborative control are difficult to apply in Internet-based mobile robot teleoperation because the Internet causes uncertain time delays and the robot cannot obtain timely suggestions from the human operator.

In summary, the differences between the proposed telecommanding and the existing passive or active supervisory control methods (including the event-based control) are as follows.

First, the proposed telecommanding can provide a complete framework to process different types of commands (e.g. joystick commands or linguistic commands) and allow these commands to be sent continuously. Most of other control methods are only able to provide single or limited available control commands (e.g. using a mouse to click a map).

Second, in most existing Internet telerobotic systems, the robots have considerable autonomy but lack the interaction with the human operator. For example, the operator is only able to send very high-level commands to the robot, without the capability of using the running parameters to influence the robot's execution process. Also, it is difficult to obtain the robot's running status or information about the events the robot has encountered. The proposed telecommanding provides such linguistic commands with flexible working parameters, and allows that the robot can respond and feedback predefined expected events as well as react to unexpected events.

Third, most robots under passive supervisory control need to know environmental knowledge in advance for path planning or localization, which cause that it is difficult to be applied in an unknown and dynamic environment. The proposed telecommanding is proposed to fully address the teleoperation of remote robot that explores unknown and dynamic environments.

5.6 Summary

This chapter proposes a new teleoperation approach namely telecommanding, which involves two different but complementary commands: joystick commands and linguistic commands. The commands are designed to perform different independent tasks. Each joystick or linguistic command is defined with multiple events (non-time action references) and the corresponding response functions. Some events (e.g. overrun event or underrun event) are used to evaluate the performance of the task when the robot is executing a linguistic command. The approach allows the robot to

deliberately respond to expected events while to reactively respond to unexpected events. Telecommanding ensures the safety of Internet robot that is navigating in an unknown and highly dynamic real world, and alleviates the problems of arbitrary network delays and restricted bandwidth. Moreover, it eases the work load of the human operator, reduces the operation sequence and its complexity in the command queue, and improves the interactivity and reliability of Internet telerobotics. The experiments have demonstrated the promising performance and the advantages of telecommanding over direct control and passive supervisory control.

CHAPTER 6. REAL-TIME MAP BUILDING AND ACTIVE EXPLORATION

Chapter 5 presents a new teleoperation approach, which provides a complete framework of control management and command processing. Both joystick and linguistic commands are designed to help human operators remotely control the mobile robot to explore unknown environments. One of the linguistic commands is MAPPING, which allows a mobile robot to be able to actively explore the unknown environment and to build a map autonomously. This chapter realizes such command function by proposing a new map learning approach.

6.1 Introduction

To perform fully autonomous tasks, it is necessary for mobile robots to model an a priori unknown environment. The optimal way to do this is for the mobile robot to actively explore the environment and construct a map based on its sensory information. This is the problem of active map learning [Arleo et al., 1999], which is a little different from the SLAM (Simultaneous Localization And Mapping) problem [Filliat & Meyer, 2003; Chong & Kleeman, 1999]. The former focuses more on the active exploration strategy for sensing the environment, while the latter focuses more on the localization strategy for estimating robot's accurate position. This chapter addresses the problem of active map learning.

There are some practical limitations on a robot's ability to learn accurate map models including the perceptual limitations of most sensors (e.g. ultrasonic sensors, cameras), sensor noise, drift or slippage, environmental complexity and dynamics, as well as real-time requirements [Thrun, 1998b]. In addition, two fundamental requirements must be satisfied for effective active map learning: first, the robot must have an efficient map model for representing the environment, and second, the robot must incorporate a fast path-planning algorithm based on this representation for actively exploring the environment.

There are a variety of map learning approaches, as described in Chapter 2. They use different grid-based map models to represent the environments, and update a map

based on probability [Thrun, 1998b; Yamauchi et al., 1998; Wallner & Dillmann, 1994; Dieguez et al., 2003; Song & Chang, 1999; etc.], fuzzy possibility [Oriolo et al., 1998], or frequency concept [Borenstein & Koren, 1991; Edson et al. 2004]. Some active exploration methods are developed to navigate a mobile robot to least known environments. Almost all of these methods adopt the strategy of global path planning and path tracking, typically belonging to a SMPA (Sense-Model-Plan-Act) approach [Saffiotti, 2000]. In these methods, a target representing least known environment is selected at first based on an already modelled environment, then an optimal path from current robot position to the selected target is obtained by global search, finally the robot follows the planned path to reach the target and chooses another target again. One drawback of such exploration approach is that the computational complexity of path planning rapidly increases as the environmental complexity or the scale of learned map increases, making real-time computation in practical applications infeasible. Moreover, this approach encounters the problem that the plan built from the modelled map will be inadequate to the environment actually faced during execution, particularly in a dynamic environment.

This chapter proposes a new approach called "*memory grid mapping*" for active map learning in unknown indoor environments. The proposed map model adopts a grid-based representation and uses frequency values to measure the confidence that a cell is occupied by obstacle. The map model allows that more information about the environment and the robot's history of experience (e.g. its trajectory) can be kept in a map. The exploration strategy adopts a behavior-based approach as previously presented in Chapter 4. In each control period, the proposed exploration method recommends a direction that provides minimum risk in a predetermined region in order to drive the robot greedily moving toward less visited environment. This minimum risk involves both minimum collision risk with obstacles and minimum iteration risk toward previously visited area. The proposed map postprocessing method, including a threshold operation, a template operation, and an insert operation, is useful to improve the accuracy of learned map. The approach makes no assumptions about environmental complexity or the shape or size of obstacles, but we assume in this chapter that the robot obtains an accurate position by localization without odometric errors.

The rest of the chapter is organized as follows. Section 6.2 proposes a new map

learning approach, involving a grid-based map model and a framework for real-time map building and active exploration. Section 6.3 presents a map update method, Section 6.4 an exploration method, and Section 6.5 a map postprocessing method. Section 6.6 provides the results of our simulations experiments. Section 6.7 gives some discussions. Section 6.8 summarizes the chapter.

6.2 The proposed approach

6.2.1 The model of memory grid map

The proposed map represents an environment by using evenly-spaced grid cells. A map shown in Figure 6.1 can be defined as a vector $V(x_{GridHead}, y_{GridHead}, M, N, \ell)$, where $(x_{GridHead}, y_{GridHead})$ is the coordinate of the top-left-corner cell in the internal coordinate systems of the robot; (M, N) are respectively the rows and columns; ℓ is a constant denoting the length of the cell size. *Coordinate mapping* is a transform process from the internal coordinate (x', y') of the robot to the coordinate (m', n') of the grid cell. By using a coordinate mapping, current physical position of the robot is mapped into a position of the grid-based map so that corresponding information can be saved in a map.

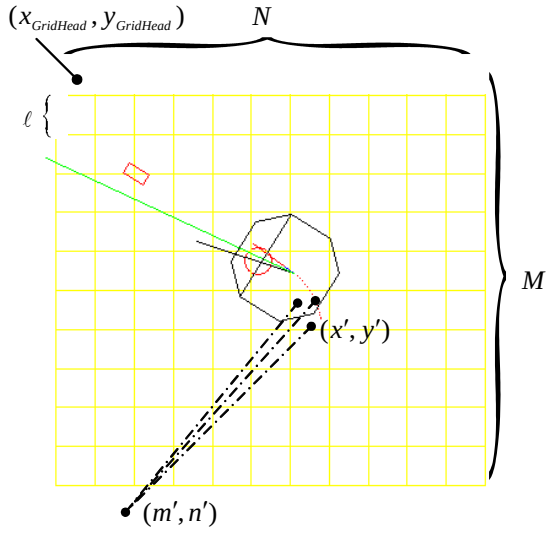


Figure 6.1: A memory grid map and its coordinate mapping

The equations of coordinate mapping are as follows.

$$m' = \begin{cases} \overline{y} & , \quad \text{if } 0 \le \hat{y} < 0.5 \\ \overline{y} + 1, & \text{if } 0.5 \le \hat{y} < 1 \end{cases} \qquad (6.1)$$

$$n' = \begin{cases} \bar{x} \quad , & \text{if } 0 \le \hat{x} < 0.5 \\ \bar{x} + 1, & \text{if } 0.5 \le \hat{x} < 1 \end{cases} \qquad (6.2)$$

where,

$$\bar{x} = \mathrm{Int}\left(\frac{x' - x_{GridHead}}{\ell}\right) \qquad \hat{x} = \frac{x' - x_{GridHead}}{\ell} - \mathrm{Int}\left(\frac{x' - x_{GridHead}}{\ell}\right)$$

$$\bar{y} = \mathrm{Int}\left(\frac{y_{GridHead} - y'}{\ell}\right) \qquad \hat{y} = \frac{y_{GridHead} - y'}{\ell} - \mathrm{Int}\left(\frac{y_{GridHead} - y'}{\ell}\right)$$

We call the proposed map *memory grid map* because each grid cell of the map contains two kinds of memorized information that we call *memory dot*. One is *Obstacle Memory Dot* (OMD). The other is *Trajectory Memory Dot* (TMD). The OMD's value $V_{OMD}(i, j)$ indicates the measure of confidence that an obstacle exists within the cell (i, j) area, where i = 1, 2, …, M, and j=1, 2, …, N. The TMD's value $V_{TMD}(i, j)$ indicates the number of occurrence, i.e. how many times the robot traverses the cell (i, j) area. The TMD is designed to record the previously traversed trajectory as well as the time consumed by the robot that traverses the cell area. The information about TMD can be used for robot online path-planning. The information saved in the map appears as matrix, such as OMD matrix $O_{M\times N}$ and TMD matrix $T_{M\times N}$. Every control period (100ms in our robotic system), we update $O_{M\times N}$ and $T_{M\times N}$. The update algorithm is described in Section 6.3.

6.2.2 A framework of map building and active exploration

A framework of the proposed approach is shown in Figure 6.2 for map building and active exploration. In this approach, a memory grid map is built based on robot's sensory information, so that we call the approach *memory grid mapping*. The approach includes three modules: map update, environmental exploration, and map postprocessing. This section provides a short description of their design ideas.

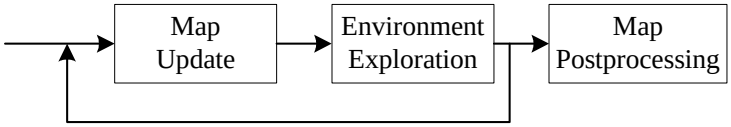


Figure 6.2: A framework of the proposed approach

1) Map update

The module of map update is to interpret the sensor readings and integrate them over time into a map that models the environment. In this module, the sonar readings are mapped into frequency values (i.e. OMD's values) which represent the confidence

of the cells where they are occupied by obstacles or not. These values are integrated over time to yield a single, combined estimate of occupancy in a map (i.e. OMD matrix $O_{M\times N}$) by simple addition or subtraction of frequency values. For the update of TMD matrix $T_{M\times N}$, only one cell where the robot is currently located is incremented in each control period. The detail is presented in Section 6.3.

2) Environmental exploration

The module of environmental exploration is to make online path planning in order to actively explore the least known environment. In this module, the path planning method adopts a strategy of multi-behavior coordination, in which a novel regional path-exploring behavior is developed to recommend the regional direction toward less visited environment, and a local environment-detecting behavior is developed to detect the environment details while to avoid obstacles. The TMDs in the memory grid map are used by the path-exploring behavior to evaluate the risk whether or not the robot is iterating the previously visited areas. Each behavior is assigned a weighting factor, and these factors are adjusted dynamically by weighting functions during robot motion. The weighting factors determine the degree of influence of each behavior on the final motion command. The final command output is obtained by coordinating these two behaviors using a command fusion equation. The detail is presented in Section 6.4.

3) Map postprocessing

The module of map postprocessing is to filter the learned map offline in order to remove some misclassified cells and to obtain a more consistent and complete environment map. At first we use a threshold operation in order to remove some misclassified cells from the perspective of cell's intensity (i.e. magnitude of OMD value). Next we use a template operation in order to remove most misclassified cells from the perspective of neighboring correlation. Finally we use an insert operation in order to add some undetected cells. The detail is presented in Section 6.5.

6.3 The map update

The map update is done real time in order to build a map based on robot's sensory information. In general, a map is updated in two steps. First, sensor readings are interpreted to draw a local map (i.e. a map that only keeps the obstacle information

derived from current sensor readings). Then the local map is integrated into a global map (i.e. a map that keeps global obstacle information throughout the entire control period) and the corresponding cells are updated. Thrun's method [1998b] trains an artificial neural network using Back-Propagation to map sonar readings to occupancy values. Multiple sonar interpretations are then integrated over time using Bayes rule to form a global metric grid. This approach requires many calculations. Arleo et al [1999] use a similar neural network technique to obtain the local grid-based map, but this local map is subsequently used only to identify obstacle boundaries in order to build a variable-resolution partitioning map. Song and Chang's method [1999] extends from heuristic asymmetric mapping (HAM) [Song & Chen, 1996], in which a sonar reading indicates the probabilities of multiple cells that correspond to physical occupied region and empty region. The probability of each cell is then integrated into a global grid map through a first-order digital filter to generate a certainty value from -1 to 1. Oriolo et al. [1997; 1998] provide a fuzzy reasoning method to update the map. Borenstein and Koren [1991] uses a simple metric sonar model that increases the cell value measured by the sonar and decreases the cells corresponding to free areas.

The update method of the proposed memory grid map involves two parts: one is to update the OMD matrix $O_{M\times N}$, another is to update the TMD matrix $T_{M\times N}$. Initially, both $O_{M\times N}$ and $T_{M\times N}$ are set to zero matrixes. The details are as follows.

The update method of OMD matrix $O_{M\times N}$ increments only one cell for each range reading. At the same time it decrements those cells that represent "empty" areas in this range reading. This design makes the update algorithms simple and fast. For sonar sensors as shown in Figure 6.3, the incremental cell is the one *Sd* that corresponds to the measured distance *d* and lies on the acoustic axis of the sonar *S0*. The incremental cell is updated by Eq. (6.3)

$$V_{OMD}(i,j)=\begin{cases} V_{OMD}(i,j)+I^{+}, & \text{If } V_{OMD}(i,j)<V_{O-MAX} \\ V_{O-MAX} & \text{Otherwise} \end{cases} \qquad (6.3)$$

where, $V_{OMD}(i, j)$ is the OMD value of grid cell (i, j), i = 1, 2, …, M, and j=1, 2, …, N, $V_{O\text{-}MAX}$ is a constant for a grid cell's maximum OMD value. The increment I^{+} is 3 and $V_{O\text{-}MAX}$ is 25, experimentally determined in our robotic system.

The decremental cells are located on the line of the acoustic axis except the incremental cell *Sd*. They are upated by Eq. (6.4).

$$V_{OMD}(i,j) = \begin{cases} V_{OMD}(i,j) - I^{-}, & \text{If } V_{OMD}(i,j) > V_{O-MIN} \\ V_{O-MIN} & \text{Otherwise} \end{cases} \quad (6.4)$$

where $V_{O\text{-}MIN}$ is a constant for a grid cell's minimum OMD value. The decrement I^{-} is 1 and $V_{O\text{-}MIN}$ is 0. These values are determined experimentally. Note that I^{-} must be smaller than I^{+} because only one cell is incremented whereas multiple cells are decremented for one reading.

Finally, we only update the cells that are located inside a circular sector of radius centered at the sonar position. This circular sector is called the "*confidence sector*". The radius r_c of this sector is 1 metre, which is an acceptable value that we have confidence to obtain the correct sonar readings in our robotic system. This reduces artifacts produced by sonar noises (e.g. noises from false reflections). Because of this update strategy, a likelihood distribution of occupancy is actually obtained by continuously and rapidly sampling each sensor as the robot is moving, in which high values are obtained in cells close to the actual location of the obstacle.

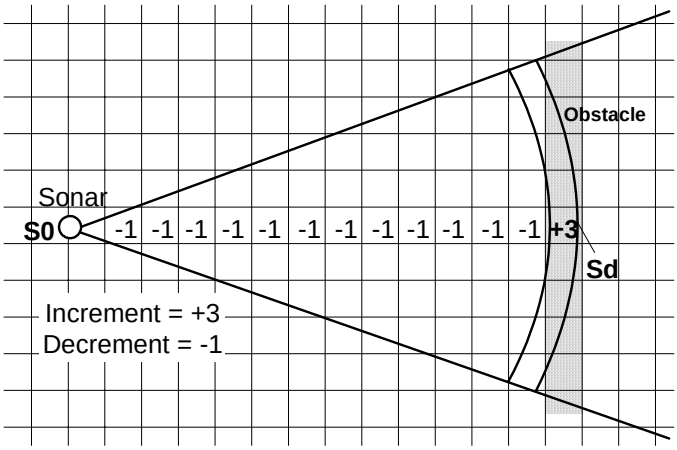


Figure 6.3: The update of OMD matrix in a memory grid map. The incremental cell Sd, corresponding to measured distance d, is incremented by I^{+}, and other decremental cells between S0 and Sd are decremented by I^{-}.

The update method of the TMD matrix $T_{M\times N}$ is very simple. Only one cell where the robot is currently located is incremented in each control cycle.

$$V_{TMD}(i,j) = \begin{cases} V_{TMD}(i,j) + 1, & \text{If } V_{TMD}(i,j) < V_{T-MAX} \\ V_{T-MAX} & \text{Otherwise} \end{cases} \quad (6.5)$$

where $V_{TMD}(i, j)$ is the TMD value of grid cell (i, j), i = 1, 2, …, M, and j=1, 2, …, N, $V_{T\text{-}MAX}$ is a constant for a grid cell's maximum TMD value. This maximum value is 50, experimentally determined in our robotic system. There is no decrement for

TMD matrix, which means that the trajectory experienced by the robot might not be forgotten.

During every control cycle (100ms in our robotic system), Algorithm 6.1 is called once to update a memory grid map.

Algorithm 6.1: MAPUPDATE( )

Input: (x0, y0) = current robot location; φ0 = current robot heading angle;

d_i (i = 0, 1, …, 7) = sonar readings from eight forward sonars.

Output: $O_{M\times N}$ = The OMD matrix; $T_{M\times N}$ = The TMD matrix.

BEGIN:

Step 1. Update the TMD matrix $T_{M\times N}$.

Step 1.1. Do the coordinate mapping to transform current robot coordinate (x0, y0) into coordinate (m^0, n^0) of memory grid map by Eqs. (6.1) and (6.2);

Step 1.2. Update the TMD value $V_{TMD}(m^0, n^0)$ of corresponding cell (m^0, n^0) in $T_{M\times N}$ By Eq. (6.5);

Step 2. Update the OMD matrix $O_{M\times N}$.

FOR every sonar S_i (i=1 to 8), Do the same jobs as the following:

IF the sonar reading d_i is less than the radius r_c of confidence sector, THEN

Step 2.1. Calculate the coordinate (x_{Sd}, y_{Sd}) of incremental cell Sd as in Figure 6.3 based on the sonar's coordinate (x_{S0}, y_{S0}) and sonar reading d_i;

Step 2.2. Do the coordinate mapping to transform (x_{Sd}, y_{Sd}) into grid coordinate (m_{Sd}, n_{Sd}) by Eqs. (6.1) and (6.2);

Step 2.3. Increment the OMD value $V_{TMD}(m_{Sd}, n_{Sd})$ of cell (m_{Sd}, n_{Sd}) in $O_{M\times N}$ By Eq. (6.3);

Step 2.4. Calculate the grid coordinates of all decremental cells between S0 and Sd;

Step 2.5. Decrement the OMD value of all decremental cells in $O_{M\times N}$ By Eq. (6.4);

OTHERWISE

Step 2.6. Calculate the grid coordinates of all decremental cells within confidence sector between S0 and Sd;

Step 2.7. Decrement the OMD value of all decremental cells in $O_{M\times N}$ By Eq. (6.4);

END IF

NEXT FOR

END Algorithm 6.1.

6.4 The environmental exploration

The environmental exploration methods are developed for the mobile robot to actively explore the least known environment. Thrun [1998b] resorts to an exploration scheme that allows the robot to drive towards unexplored areas, i.e. areas where cell probabilities have never been updated. For each cell, this scheme updates a value representing the distance to the closest unvisited cell area using a value-iteration algorithm, so that performing a gradient descent on these values leads to unexplored areas. Instead of using value-iteration, Yamauchi et al. [1998] implement the exploration by directing the robot toward the closest *frontier* between explored and unexplored areas. The path to this frontier is computed using a depth-first search in known open-areas. Arleo et al. [1999] develops a technique called *counter-based exploration with decay* [Thrun, 1992], in which a counter keeps track of the number of occurrences for each partition (i.e. how many times that partition has been visited). The counter is multiplied by a decay factor in order to take into account when a partition has been visited. The exploration is directed toward the partitions that have been less often and less recently visited.

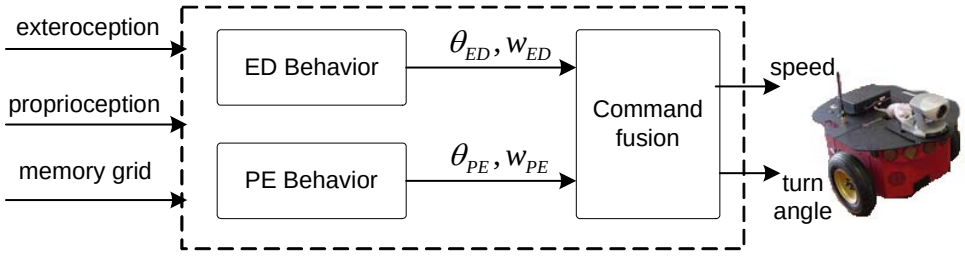
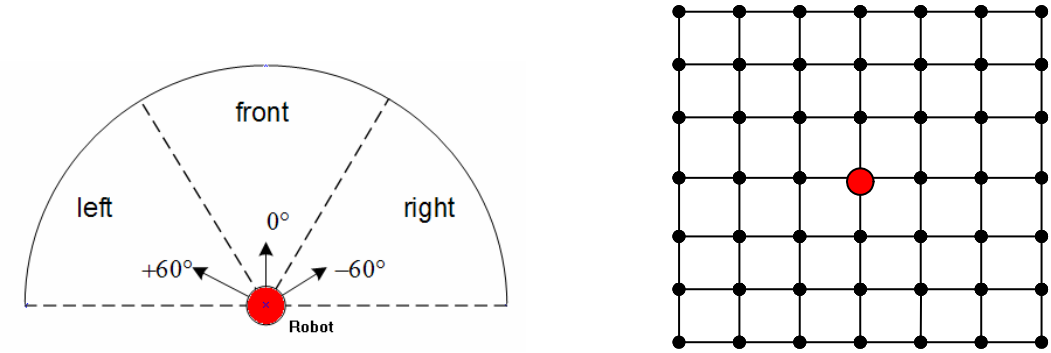


Figure 6.4: The proposed exploration method by ED and PE behavior coordination.

The proposed exploration method adopts a strategy of multi-behavior coordination as shown in Figure 6.4, which comprises two elementary behaviors, path-exploring (PE) behavior and environment-detecting (ED) behavior. The PE behavior's role is to

navigate a mobile robot to a less visited region. This region is among the LEFT, RIGHT, FRONT regional sectors as shown in Figure 6.5(a), which we call "*turn detection region*". The total values of OMDs of a turn detection region would represent the risk that the robot could collide with obstacles in this region. Similarly, the total values of TMDs of a turn detection region would represent the risk that the robot is moving to its previously visited areas. Therefore, the region with minimum risk is the one with the minimum values of both TMDs and OMDs. Such regional direction is the best choice for the robot in trying to avoid both obstacles and previous trajectory, and consequently safely explore new environment. The local ED behavior is a sensor-based behavior, which detects the environment while making the robot safe without collision with obstacles. It's desired that the ED behavior enables the robot to follow the boundary of obstacles as near as possible in order to detect more environmental details.



(a) (b)

Figure 6.5: Detection regions for the PE behavior. (a) Arc-shaped turn detection regions (i.e. LEFT, FRONT, RIGHT). (b) Square-shaped weight detection region. The center is the robot location.

In each control period, the final motion command is obtained by fusing two behaviors' weighting output. The rotational turn angle θ and the speed v are obtained by Eqs. (6.6) and (6.7) respectively.

$$\theta = \frac{w_{PE} \cdot \theta_{PE} + w_{ED} \cdot \theta_{ED}}{w_{PE} + w_{ED}} \qquad (6.6)$$

$$v = v_c \qquad (6.7)$$

where, θ_{ED} and θ_{PE} are respectively the delta turn angle recommended by the ED and PE individual behavior. w_{ED} and w_{PE} are respectively the weighting factor of the ED and PE behavior. v_c equals to 100mm/s in our robotic system. The robot's speed v

is set to this small constant value so that the robot has enough time to detect the environment. We shall describe the design of two behaviors in the following.

To calculate the turn angle and weight of the PE behavior, we define the following terms at first.

Definition 6.1 (Iteration Risk): *Iteration Risk* (IR) of a region *A* is defined as $\alpha(A) = \sum_{(i,j)\in A} V_{TMD}(i,j)$, where *A* is an arc-shaped turn detection region (see Figure 6.5(a) left, front, right regions), $V_{TMD}(i, j)$ is the TMD's value of the cell (*i, j*) involved in the region *A*.

In fact, Iteration Risk is defined as the total values of TMDs saved in a turn detection region. Similarly, we define the following terms.

Definition 6.2 (Collision Risk): *Collision Risk* (CR) of a region *A* is defined as $\beta(A) = \sum_{(i,j)\in A} V_{OMD}(i,j)$, where *A* is an arc-shaped turn detection region (see Figure 6.5(a). left, front, right regions), $V_{OMD}(i, j)$ is the OMD's value of the cell (*i, j*) involved in the region *A*.

Definition 6.3 (Trajectory Dot Intensity): *Trajectory Dot Intensity* (TDI) of a region *B* is defined as $\kappa(B) = \sum_{(i,j)\in B} V_{TMD}(i,j)$, where *B* is a square-shaped weight detection region (see Figure 6.5(b)), $V_{TMD}(i, j)$ is the TMD's value of the cell (*i, j*) involved in the region *B*.

TDI and IR have different detection regions. We call the region B *weight detection region* as shown in Figure 6.5(b). The regions available for robot traversal (i.e. turn detection regions) are three circular side sectors as shown in Figure 6.5(a). The radius of the circular sector is the robot's *regional perception range*, i.e. the distance at which we wish the robot to react to the regional risk features. The size of the radius is 1 metre in our robotic system since the robot updates the OMD's value only in those cells that are located inside a circular sector of 1 metre in radius. These regional sectors are labelled left, front, and right, and have the central angular values of $+60^\circ$,

0°, - 60° respectively. The weight detection region is no directionality (i.e. it is same whatever angle the robot's heading is). This design makes the computation of TDI very simple and fast. In our robotic system, the size of this squared region is 2m×2m (i.e. a half of the side is 1 metre) so that it is similar with the size of the turn detection region.

Algorithm 6.2 is used to calculate the turn angle of the PE behavior. The output only contains three angle values $\{60^0, 0^0, -60^0\}$ that respectively correspond to the LEFT, FRONT and RIGHT regional direction.

Algorithm 6.2: (Calculate the turn angle of PE behavior)

Input: $O_{M\times N}$ = The OMD matrix; $T_{M\times N}$ = The TMD matrix.

Output: θ_{PE} = delta turn angle of the PE behavior, $\theta_{PE} \in \{60^0, 0^0, -60^0\}$

BEGIN:

Step 1. Update the iteration risk and collision risk of all turn detection regions, including $\alpha(A_{front})$, $\alpha(A_{left})$, $\alpha(A_{right})$, $\beta(A_{front})$, $\beta(A_{left})$, and $\beta(A_{right})$;

Step 2. Find out all turn detection regions (among A_{left}, A_{front}, A_{right} regions) whose collision risk $\beta(A)$ are less than a threshold T1. IF so, THEN the corresponding regions are reserved and go to Step 3, OTHERWISE the weight of PE behavior is forced to zero (i.e. $w_{PE}=0$) and RETURN;

Step 3. IF the front region is one of the reserved regions AND its iteration risk $\alpha(A_{front})$ is less than a threshold T2, THEN the direction toward front region is recommended to move (i.e. $\theta_{PE} = 0$) and RETURN, OTHERWISE go to Step 4;

Step 4. The regional direction, whose iteration risk $\alpha(A)$ is minimum among the reserved regions, is chosen as the recommended turn angle θ_{PE}, and RETURN.

END Algorithm 6.2

The purpose of Step 2 is to guarantee that the recommended regional direction has minimum CR. Because of the uncertainty from sensor errors, it is reasonable to assume that the region is safe when it has a small CR value less than the threshold *T1*. This threshold (25 in our robotic system) is mainly determined by the size of turn detection region. Similarly, we assume that the region has a safe IR if its value is less than the threshold *T2*. This threshold (30 in our robotic system) is mainly determined by the size of turn detection region and robot's speed. The Step 3 guarantees that the

front region that has a safe IR is recommended as the moving direction, in spite of that the left or right region has smaller IR than that of front region. This step allows the robot to avoid frequent variations of the turn angle in order to decrease the odometric error. The Step 4 guarantees that the PE behavior recommends a less visited region (i.e. with minimum IR).

The weight of the PE behavior is calculated by Eq. (6.7).

$$w_{PE} = \begin{cases} 0, & \text{if } \kappa(B) \le T3 \\ \kappa(B) - T3, & \text{if } T3 < \kappa(B) \le T3 + 100 \\ 100, & \text{if } \kappa(B) > T3 + 100 \end{cases} \qquad (6.7)$$

where, *k(B)* is the TDI of the weight detection region *B*. *T3* is a threshold that represents how many times the region *B* has been visited by the mobile robot. The PE behavior is activated only when *k(B)* is larger than *T3* (200 in our robotic system). Note that, in Step 2 of Algorithm 6.2, when all turn detection regions whose CR are not less than the threshold *T1*, the weight w_{PE} of PE behavior is forced to zero. At this time, the robot depends on the ED behavior to escape from this puzzle.

The ED behavior is designed using fuzzy logic controllers as presented in Chapter 4 in order to deal with uncertainties from sonar readings. The sonar readings of the robot are grouped into three sectors (left, front, right). It is similar as represented by Eqs. (4.6) (4.7) (4.8) in Chapter 4. The obstacle distance of each sector is represented by three linguistic fuzzy sets {VERYNEAR, NEAR, FAR}. The robot turn angle is represented by five linguistic fuzzy sets {NB, NS, ZE, PS, PB}, where NB is negative-big, NS negative-small, ZE zero, PS positive-small, and PB positive-big. The weight of ED behavior w_{ED} is represented by three linguistic fuzzy sets {SMALL, MEDIUM, LARGE}. The membership functions of obstacle distance, turn angle, and weight are referred to the Figure 4.4 in Chapter 4.

Table 6.1 summarizes the turn rules of the ED behavior. For instance, the (1,1) element of the top layer in Table 6.1 can be written as the rule:

IF d_{front} *is VERYNEAR AND* d_{left} *is FAR AND* d_{right} *is FAR, THEN* θ_{ED} *is PS.*

The turn rules of the ED behavior govern the following behavior characteristics: if the obstacle is not very near, the robot still keeps moving forward (i.e. turn angle is 0), otherwise the robot only turns left or right a small angle to avoid the obstacle. Note that, when the three sectors have the same *VERYNEAR* obstacle distance as shown in the (3,3) element of the top layer in Table 6.1, a large left turn (PB) angle is

recommended. This turn rule enables the robot to escape from its current embarrassed situation.

Table 6.2 summarizes the weight rules of the ED behavior. The weight is derived directly from obstacle distances in the three sectors. Note that the weight's range is 0 to 100, same with that of the weight of PE behavior. On the other hand, the defuzzified minimum weight of the ED behavior is a small non-zero value. As a result, when the weight of the PE behavior is zero, the ED behavior might dominate the final motion output although its weight possibly is small.

Table 6.1: Turn rules for the ED behavior.

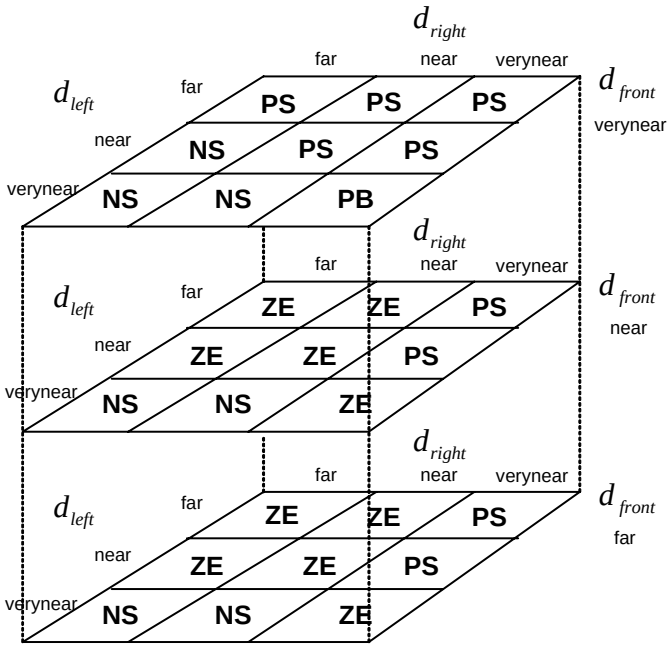
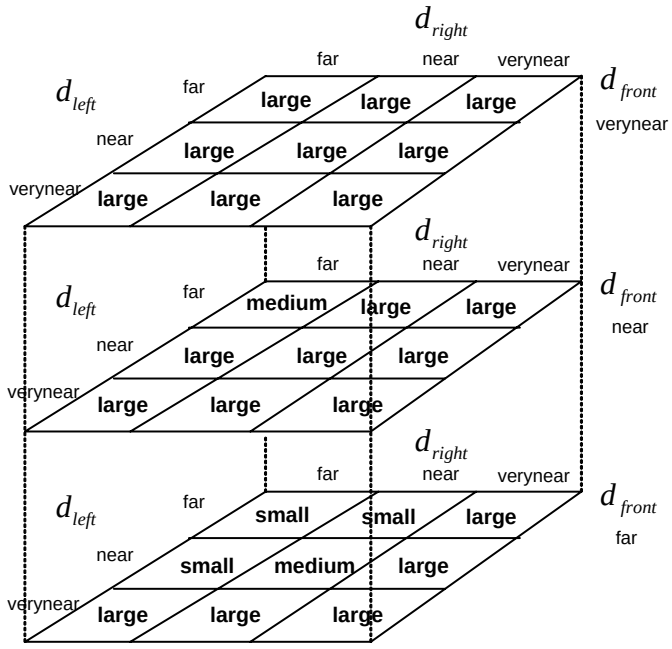


Table 6.2: Weight rules for the ED behavior.



For every control cycle, Algorithm 6.3 is called once for environmental exploration.

Algorithm 6.3: ENVIRONMENTEXPLORATION( )

Input: (x0, y0) = current robot location; φ0 = current robot heading angle;

(d0, d1, d2, d3, d4, d5, d6, d7) = sonar readings.

Output: (v, θ) = speed and delta turn angle of the robot

BEGIN:

Step 1. To preprocess the sonar readings using Eqs. (4.6), (4.7), and (4.8) in Chapter 4;

Step 2. IF the total distance the robot travels is greater than a given threshold OR the robot receives a STOP command, THEN the robot is stopped and RETURN, OTHERWISE go to the Step 3;

Step 3. To calculate the weight w_{PE} of the PE behavior using Eq. (6.8);

Step 4. To calculate the delta turn angle θ_{PE} recommended by the PE behavior using Algorithm 6.2;

Step 5. To calculate the delta turn angle θ_{ED} recommended by the ED behavior using Algorithm 4.1 in Chapter 4, and using the turn rules as in Table 6.1;

Step 6. To calculate the weight w_{ED} of the ED behavior using Algorithm 4.1 in Chapter 4, and using the weight rules as in Table 6.2;

Step 7. To calculate (v, θ) by the command fusion using Eqs. (6.6) and (6.7);

Step 8. To execute the motor control commands (v, θ).

END Algorithm 6.3

6.5 The map postprocessing

The purpose of map postprocessing method is to filter the constructed map offline in order to remove noises and obtain a more consistent and complete environment map. In histogramic in-motion mapping [Borenstein & Koren, 1991], the permanent map is obtained by simple threshold comparison. The certainty values of the cells are set to zero if they are less than a predetermined threshold, otherwise they are reserved in the permanent map. In Edson's method [2004], cells have three states: *occupied, free*

space, and *not explored*. Cells are changed to *occupied* if their immediate neighbors on both sides are *occupied*, and cells classified as *not explored* are changed to *free space* if most of their neighboring cells are explored (either *free space* or *occupied*). Dieguez et al. [2003] developed a mechanism called *propagation* to increase or decrease the confidence value of each cell according to the total values of this cell's neighbors.

This chapter proposes a method for map postprocessing as shown in Figure 6.6. The final map is obtained after the raw learned map (i.e. the OMD matrix $O_{M\times N}$) is orderly processed by the modules of a threshold operation, a template operation, and an insert operation.

Figure 6.6: A framework of the proposed method for map postprocessing

First the threshold operation eliminates some misclassified cells from the perspective of cell's intensity (i.e. magnitude of OMD's value). Note that the cells belonging to free area (whose OMD value is zero) are called *free cells*, and the cells occupied by obstacles (whose OMD value is non-zero) are called *occupied cells*. The *misclassified cells* are those free cells but they are mistakenly classified as occupied cells because of the errors of sonar readings. By threshold operation, the OMD's value of each cell is set to zero if it is not larger than a threshold *T4*, otherwise it is set to the maximum value $V_{O\text{-}MAX}$ (25 in our robotic system, see Section 6.3). This threshold *T4* in our robotic system is 3, which implies that each occupied cell is eligible to reserve in the final map only if the cell's area is detected at least twice by any of robot's sensors.

Next the template operation eliminates most of the misclassified cells from the perspective of neighboring correlation. The nature of this operation is to realize the following heuristic rule: Isolated cells (i.e. cells whose neighbors are not occupied as they have small frequency values) come mostly from erroneous sonar readings. We have defined eight templates shown in Figure 6.7. Every cell of the processed map is matched with the eight templates. For example, the first template (see Figure 6.7(1)) is used to match. If all neighboring cells are occupied (i.e. the OMD's values are larger than zero in these neighboring cells), this template is matched successfully,

otherwise it fails. The OMD's value of the cell is maintained in the final map if any one template is matched successfully. If all templates fail to match, the OMD's value of the cell is set to zero (i.e. a free cell).

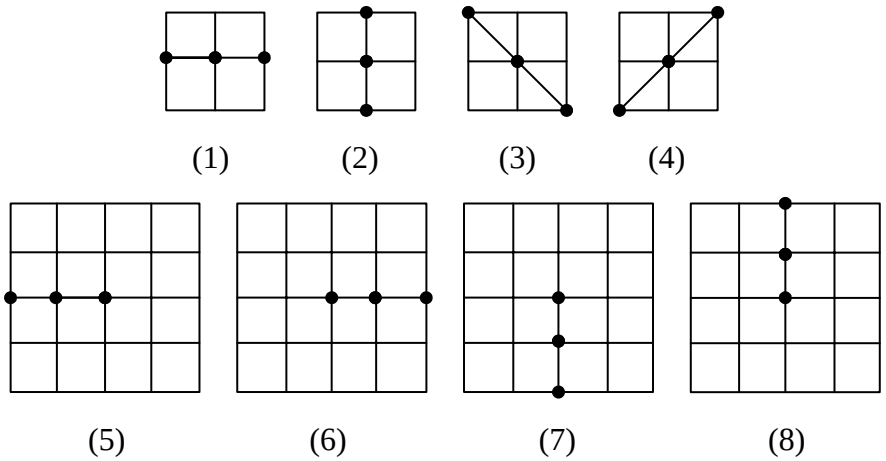


Figure 6.7: Eight templates for map postprocessing. (1-8) The black dot in template center is the cell that is being matched. The other black dots are neighboring cells of the matched cell.

Finally the insert operation adds some undetected cells. The *undetected cells* are those occupied cells that are mistakenly classified as free cells because the sonars miss those cell areas due to the robot moving. The purpose of this operation is to realize the following heuristic rule: the cells, whose neighbors on both sides are occupied, should be also occupied. The insert operation makes use of the former four templates (see Figure 6.7(1-4)) to match every cell of the processed map. If any template is matched successfully, the OMD value of the matched cell is set to the maximum value $V_{O\text{-}MAX}$, otherwise its value is maintained.

In summary, Algorithm 6.4 gives the process of map postprocessing.

Algorithm 6.4: MAPPOSTPROCESSING( )

Input: $O_{M\times N}$ = The OMD matrix

Output: $O_{M\times N}$ = The OMD matrix

BEGIN:

Step 1. Do the threshold operation.

FOR every cell (i, j) in $O_{M\times N}$, i = 1, 2, …, M, and j=1, 2, …, N, DO

IF $V_{OMD}(i, j) > T4$, THEN

$V_{OMD}(i, j) = V_{O\text{-}MAX}$

ELSE

$V_{OMD}(i, j) = V_{O\text{-}MIN}$

END IF

NEXT FOR

Step 2. Copy $O_{M\times N}$ *into* $O'_{M\times N}$ *, then do the template operation.*

FOR every cell (i, j) in $O'_{M\times N}$*, i = 1, 2, …, M, and j=1, 2, …, N, DO*

IF all templates as in Figure 6.7(1-8) are failed to match, THEN

Update the cell (i, j) in $O_{M\times N}$ *using* $V_{OMD}(i, j) = V_{O\text{-}MIN}$

END IF

NEXT FOR

Step 3. Copy $O_{M\times N}$ *into* $O'_{M\times N}$ *, then do the insert operation.*

FOR every cell (i, j) in $O'_{M\times N}$*, i = 1, 2, …, M, and j=1, 2, …, N, DO*

IF any one template as in Figure 6.7(1-4) is successful to match, THEN

Update the cell (i, j) in $O_{M\times N}$ *using* $V_{OMD}(i, j) = V_{O\text{-}MAX}$

END IF

NEXT FOR

END Algorithm 6.4

6.6 Experimental results

Section 6.6.1 shows a simulation test to analyze the robot's exploration process and to evaluate the learning efficiency of the proposed approach. Section 6.6.2 evaluates the map accuracy after map postprocessing. Section 6.6.3 evaluates the map accuracy if the size of cells is different. Section 6.6.4 will give the simulation tests in more complex environment.

6.6.1 Performance analysis of exploration process

The purpose of this simulation experiment is to analyze the exploration process in which how the robot makes decision to explore unknown environment. At the beginning of map learning, the values of all TMDs and OMDs in a memory grid map are initialized to zero. The memory grid map is updated by Algorithm 6.1. The cell size is 100mm×100mm. When the robot begins to explore unknown environment

(see Figure 6.8(1)), the weight of PE behavior is zero because the TDI in weight detection region B is smaller than the threshold *T3*. At this moment, the ED behavior makes dominant contribution to the final motion output. When the robot closes to obstacles (see Figure 6.8(2)(3)), the weight of ED behavior becomes larger. The ED behavior recommends a small turn angle to make the robot following the boundary of obstacles in order to detect more environmental details. When the weight of PE behavior becomes larger with the increase of TDI (see Figure 6.8(4)), both behaviors coordinate to drive the robot moving toward less visited and safe area. When the robot is far away from obstacles, the weight of ED behavior becomes smaller. The PE behavior is dominant with the increase of TDI (see Figure 6.8(5)(6)(7)), which enables the robot to avoid visiting previously traversed area and to move toward less visited environment. We manually stop the map learning when the result is acquired as shown in Figure 6.8(7). Observe that the learned map (here only the OMD matrix $O_{M\times N}$ is taken into consideration) contains a number of misclassified cells that are derived from sensor errors. It is necessary to post process the learned map. Figure 6.8(8) shows the simulation interface and the result of map postprocessing, in which many misclassified cells are removed. Section 6.6.2 will analyze the performance of map postprocessing. In addition, observe that some corners in the environment are not modelled. The main reason is that our robot only gets equipped with eight forward sonar sensors but without backward sensors. When the robot turns at the corners, the forward sensors do not have enough time to detect the environmental details. To install some backward sonars will effectively improve the robot's detection capability.

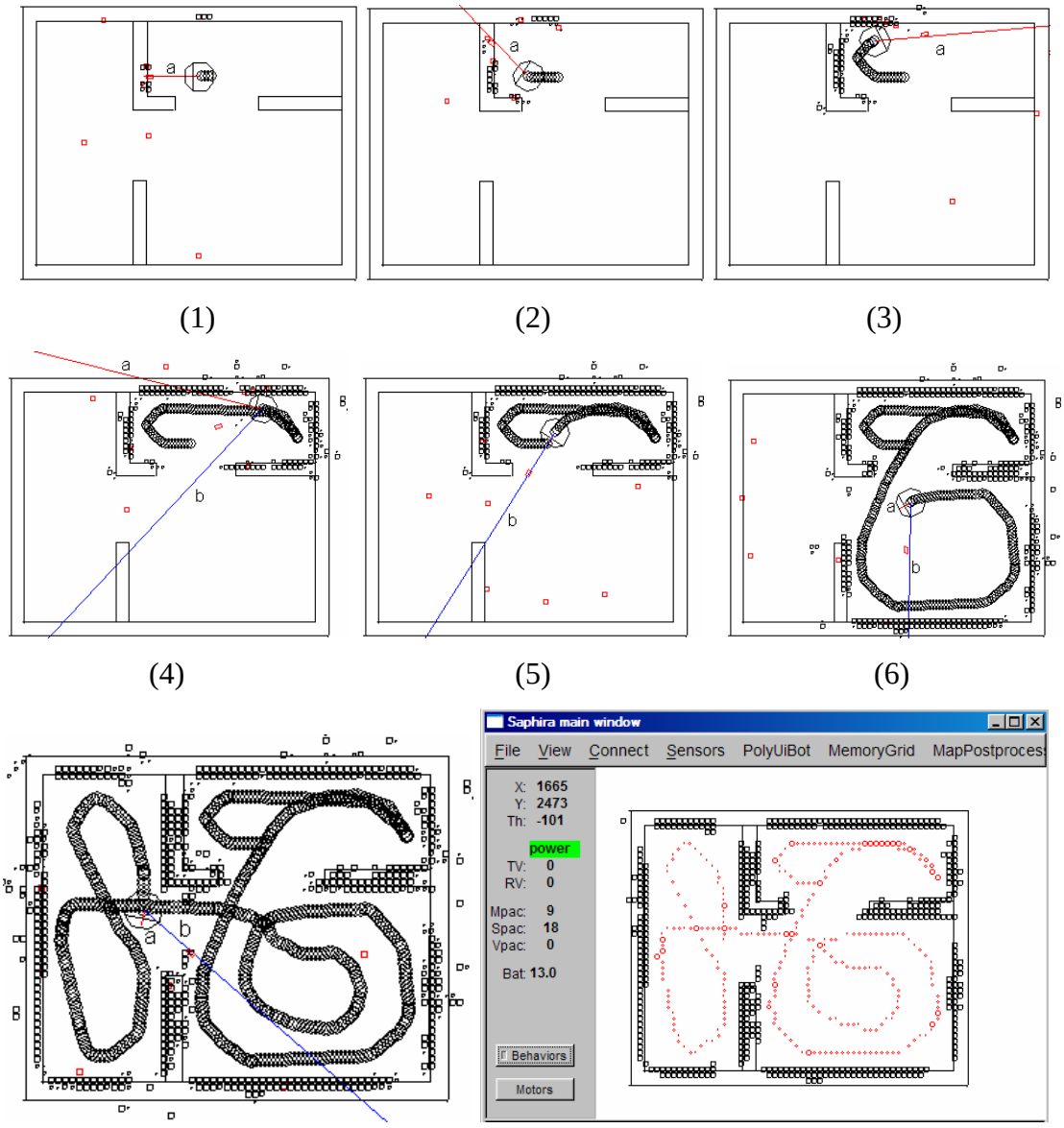


Figure 6.8: Real-time map building and active exploration in unknown indoor environment. Note that to exhibit the different contributions to the final control output provided by the different behaviors, data visualization is developed. Each behavior produces a turn angle recommendation while its weight represents the degree of influence on the final angle output. The different lines "a" and "b", drawn automatically by the control program, respectively represent the turn angles recommended by the ED, PE behaviors. The length of each line represents the weight value of each behavior. The trajectory is indicated by the chain of circles. The program draws the circle once every 0.5 second. (1-7) exploration process; (8) simulation interface.

In order to evaluate the learning efficiency, we define the following utility function:

$$U(t) = \frac{O(t)}{\frac{d(t)}{\ell} + O(t)}$$

where, *O(t)* is the total number of cells whose OMD values are not zero, which represents how much environmental knowledge the robot has already known. *d(t)* is the actual total distances(mm) the robot has already travelled. ℓ is the length of cell size (mm). It is desired that the robot could obtain the environmental knowledge as much as possible while it travels the distance as short as possible. As a result, the larger the value of *U(t)*, the better the learning efficiency is. Figure 6.9 compares the performances of the active exploration and of a random exploration during the map learning process within the environment shown in Figure 6.8. The diagram shows the active exploration outperforms the random walk. The main reason is that the robot randomly walking is easier to get trapped in local minima and it often visits previous traversed areas. If the test environment contains more complex local minima, the active exploration would obtain much better learning efficiency than the random exploration.

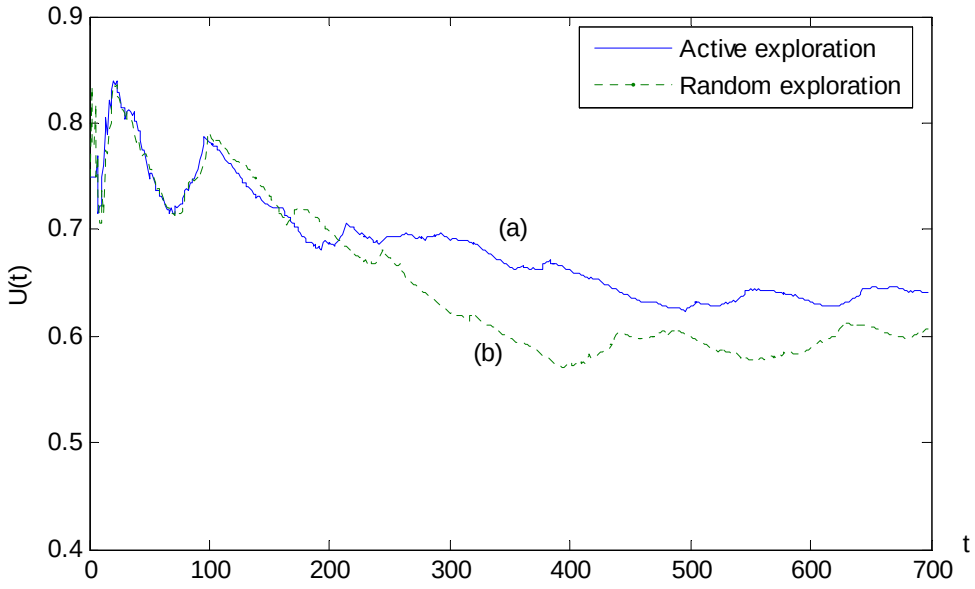


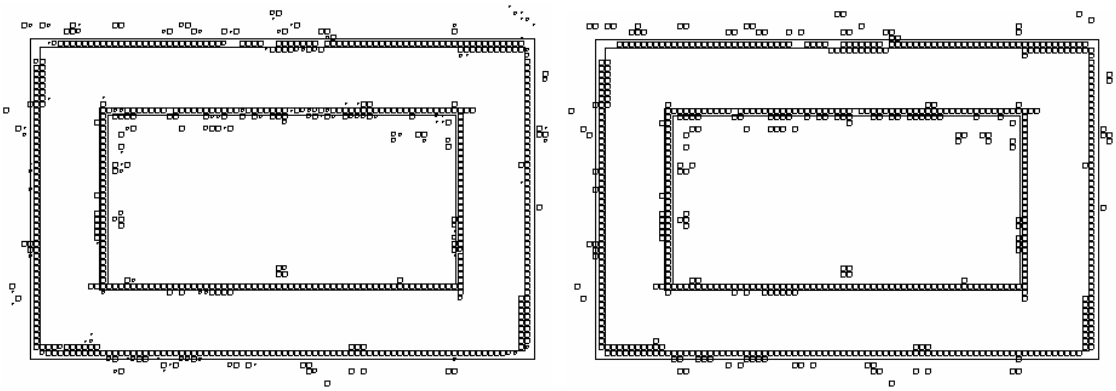
Figure 6.9: Performance comparison between active exploration and random exploration. (a) active exploration; (b) random exploration.

6.6.2 Performance of map postprocessing

In order to evaluate the map accuracy after map postprocessing, we define a simple index e to measure the misclassified cells (i.e. free cells that are misclassified as occupied cells) of total classified cells. Let A_e be the total number of misclassified cells, and A_{tot} be total number of cells whose OMD values are not zero. The error e is:

$$e = \frac{A_e}{A_{tot}},$$

Figure 6.10 shows the simulation results of map postprocessing. Obviously, the processed map has higher accuracy compared with the unprocessed map. The template operation is particularly useful to greatly eliminate misclassified cells. On the one hand, the insert operation adds some undetected occupied cells. On the other hand, it adds some misclassified cells as well. These misclassified cells are often close to the correct occupied cells. From the perspective of acquiring the environmental knowledge as much as possible, the insert operation is useful.



(a) e=17.4%; (b) e=13.9%;

(c) e=2.6%; (d) e=2.9%.

Figure 6.10: Map postprocessing. (a) unprocessed map (100mm×100mm cell size); (b) map after threshold operation; (c) map after template operation; (d) the final map after insert operation.

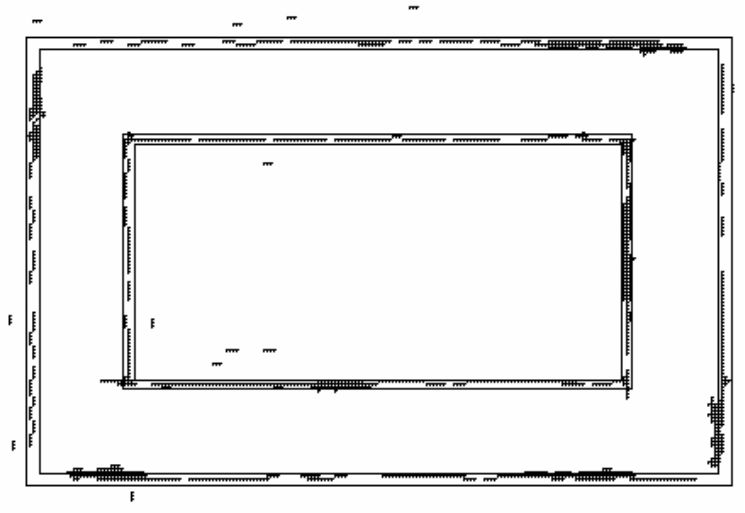
6.6.3 Performance of map with different cell sizes

In order to evaluate the map accuracy when the size of cells (i.e. granularity) is different, we adopt the Index of Performance (IOP) proposed by Raschke and Borenstein [1990]. The purpose of this index is to quantitatively express the quality of matching between a learned map and a reference map.

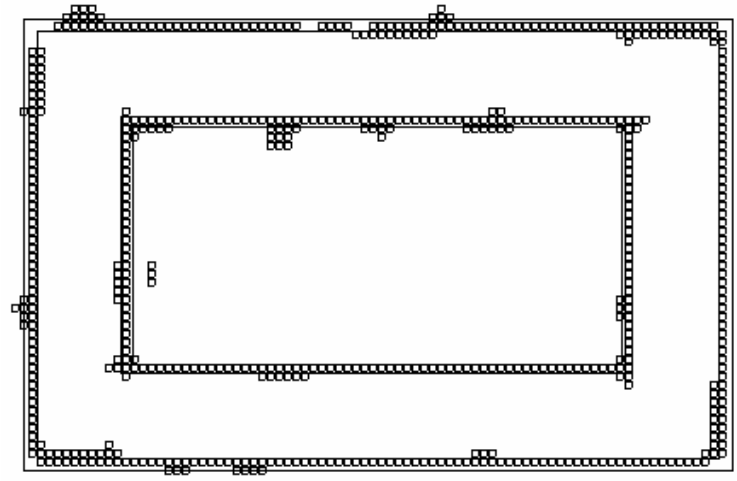
$$IOP = \frac{\sum [D_{\min}(i, j) \cdot CV(i, j)]}{\sum CV(i, j)}$$

where, $D_{min}(i,j)$ is the distance (millimetre) from cell (i,j) to the nearest occupied cells, and $CV(i,j)$ is the certainty value of cell (i,j) in learned map. Here, when the cell's OMD value is zero, the certainty value of the cell is equal to 0, otherwise it is equal to 1. The meaning of this index is the average error distance between the represented and the actual obstacles. It is independent of the cell size, the adopted map representation, and the environment range. The smaller the IOP is, the smaller the error between a learned map and a reference map will be. In other words, the learned map has higher accuracy.

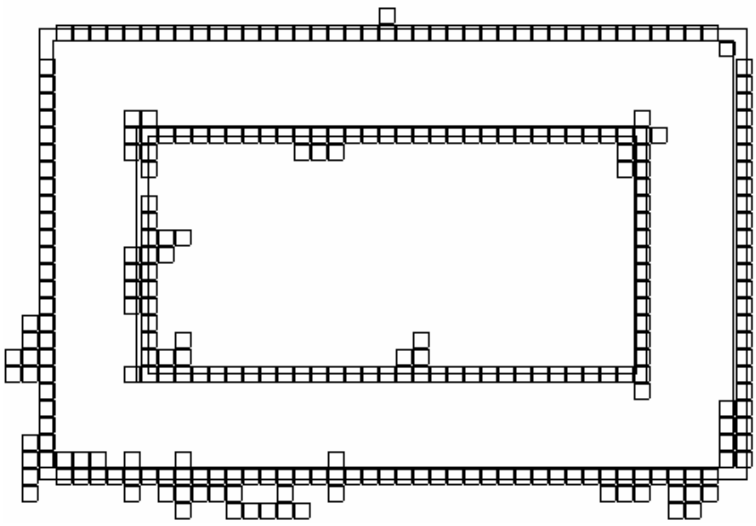
Figure 6.11 shows the results of map postprocessing with different cell sizes. In this test, the IOP of former two are close, and the IOP of latter one is relatively larger. We think that the 100mm×100mm cell size is a good compromise between map accuracy and space requirement of map storage.



(a) IOP = 4.98mm;



(b) IOP = 4.93mm;

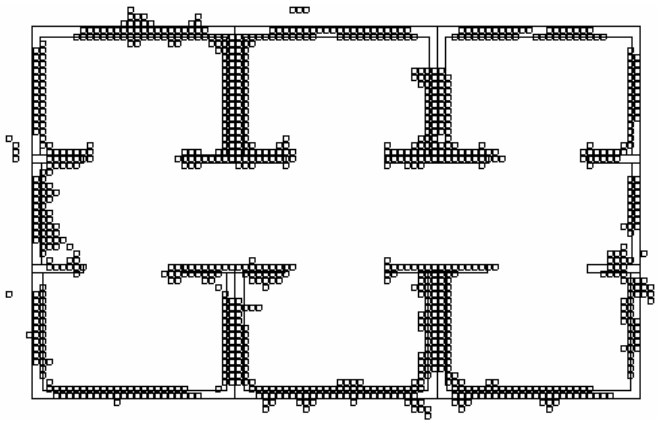


(c) IOP = 17.08mm.

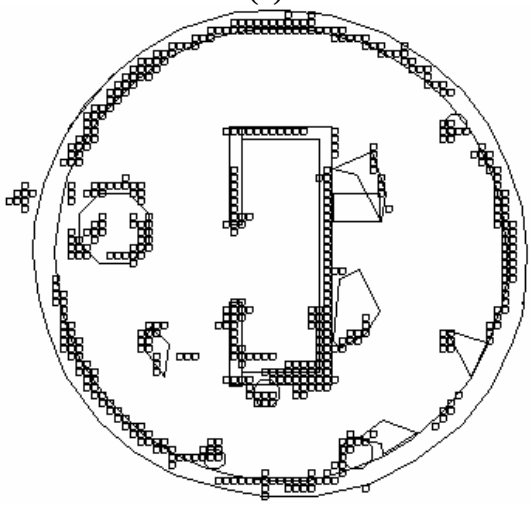
Figure 6.11: Maps with different cell size. (a) map with 40mm×40mm cell size; (b) map with 100mm×100mm cell size; (c) map with 200mm×200mm cell size.

6.6.4 Performance in complex environments

We perform the simulation tests of active map learning in more complex environments. Figure 6.12 shows the learned results, which demonstrates that the proposed memory grid mapping approach is able to model not only structured environments but also unstructured even cluttered environment. The approach does not need any assumption with the environmental complexity or obstacle's shape or size. Note that some corners in the environment are not modelled because our robot is only equipped with eight forward ultrasonic sonar sensors as described in Section 6.6.1.



(a)



(b)

Figure 6.12: Map learning in complex environments. (a) Structured office-like environment; (b) Unstructured and cluttered environment.

6.7 Discussion

Here we discuss the proposed map learning approach and compare it with existing approaches in literatures from the following several aspects.

1) **Map model**. The idea of obstacle memory dot (OMD) of the proposed memory grid map is similar to the map of histogramic in-motion mapping approach proposed by Koren & Borenstein [1991], which uses frequency values to indicate the measurement of a confidence that a cell is occupied by obstacles. The update of frequency values is simple and fast, different from the most ones based on probability [Moravec, 1988; Thrun, 1998; Dieguez et al., 2003]. One special of the proposed map is that the trajectory memory dot (TMD) is designed to record previously traversed trajectory and the time consumed by the robot that traverses the cell area. In a short, the proposed map itself is not a novel idea, but it is suitable for our online path planning (i.e. exploration) method, making it possible that the proposed approach has a low time complexity.

2) **Time complexity**. Almost all of others adopt the strategy of global path planning and path tracking in order to find an optimal exploration path and guarantee global convergence. The drawback of such approach is that the time complexity of both path planning and map update rapidly increases as the environmental complexity or the scale of learned map increases, making real-time computation in a large scale practical application infeasible. Our mapping approach takes use of a small range of sensory data and map information, making the time complexity of both map update and exploration algorithms low. The limitation of the proposed exploration method is that it is difficult to guarantee global convergence because the decision is based on local information.

3) **Learning efficiency and map accuracy**. Since it is short of standard test map and standard robotic hardware configuration in the field of robotics, it is quite difficult to compare the mapping efficiency and accuracy among different map learning approaches. Our robot is only equipped with eight forward inaccurate sonar sensors. It is difficult to compare the mapping performance with those robots that are equipped with more advanced sensors such as laser. Possibly those robots are better suited for the type of mapping application.

4) **Granularity (i.e. different cell size) evaluation**. Most mapping approaches

have adopted the cell size 100mm×100mm, but they do not explain why this cell size is chosen. We have experimentally evaluated the map accuracy under different cell sizes (e.g. 40mm×40mm, 100mm×100mm, 200mm×200mm), which quantitatively obtains the result that the 100mm×100mm cell size is a good compromise between map accuracy and space requirement of map storage.

5) **Performance of map postprocessing method**. Few literatures have proposed the techniques of map postprocessing or evaluated their performance. We have quantitatively evaluated the map representation accuracy when different map postprocessing technique is used. The proposed map postprocessing method is able to improve the representation accuracy from the original error index e = 17.4% to e = 2.6%.

6) **Exploration of dynamic environment**. Almost all of other exploration methods typically belong to a SMPA (Sense-Model-Plan-Act) approach. This approach encounters the problem that the plan built from the modelled map will be inadequate to the environment actually faced during execution, particularly in a dynamic environment. The proposed exploration method is based on real-time behavior coordination, enabling the robot to explore a dynamic environment (i.e. with humans) safely.

7) **Localization**. One short of the proposed approach is that we assume that an ideal localization technique can estimate robot's position accurately. However, it is unrealistic for real robot. The self-localization technique using odometry data is not enough, which results in serious odometric errors in a large space area. Our robot cannot do the accurate map learning in a real world at this stage because of two reasons: (1) The accumulated odometric errors have not been corrected. Especially our test environments (e.g. corridor and office) are covered with carpets, making the errors worse. (2) The robot's sonar sensors often obtain wrong sonar readings in our test environments with smooth walls. On the other hand, it is more difficult to find a suitable localization technique for teleoperated mobile robots, especially which can be applied in structured and unstructured even outdoor environments. Possibly, to limit the environment the teleoperated robot works or to equip with more advanced sensors such as laser, compass, or GPS, might help improve the accuracy of localization.

6.8 Summary

This chapter proposes a new map learning approach namely *memory grid mapping*. The approach includes a map model, a map update method, an exploration method, and a map postprocessing method. The map adopts a grid-based representation and uses frequency value to measure the confidence that a cell is occupied by an obstacle. The fast map update and path planning (i.e. the exploration method) make the approach a candidate for real-time implementation on mobile robots. The proposed map postprocessing method, including a threshold operation, a template operation, and an insert operation, is useful to improve the accuracy of the learned map.

CHAPTER 7. GOAL-ORIENTED NAVIGATION IN UNKNOWN ENVIRONMENT WITH LOCAL MINIMUM

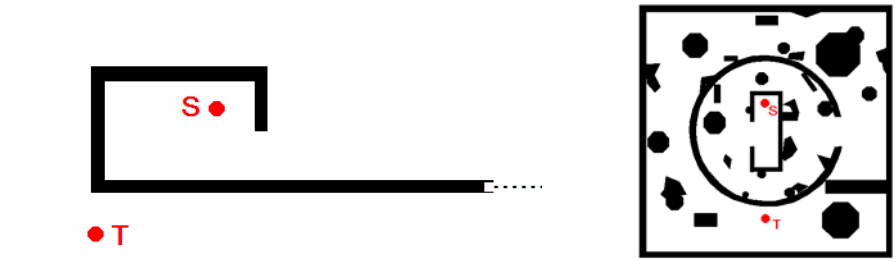
Chapter 4 has realized a goal-oriented navigation by coordinating two elementary behaviors: obstacle-avoidance (OA) and goal-seeking (GS). Such navigation method makes it easy that the mobile robot gets trapped in a local minimum (i.e. dead end) of the environment. This is the reason that the OA behavior is trying to get away from the local minimum while the GS behavior is making the robot to move back toward the goal. For a teleoperated mobile robot that is exploring unknown indoor environments, it is desired that the robot is able to autonomously arrive at a given goal location, even though the environments involve all kinds of complex situations, such as long-wall, large concave, recursive U-shape, unstructured, cluttered, maze-like, or dynamic (i.e. with moving human) environments. This chapter realizes this function, which is an enhanced COORDINATE linguistic command.

7.1 Introduction

For the goal-oriented navigation in unknown environments, it is difficult to apply the approach of global path planning and path tracking because it is short of a prior known knowledge for global environment. Moreover, the dynamics of real-world environments are typically complex and unpredictable, making a planned path rapidly out of date. Other approaches, such as potential-field [Tsourveloudis et al., 2001] or neural-fuzzy approach [Rusu et al., 2003; Godjevac & Steele, 2000], however, are difficult to guarantee global convergence to the goal because the mobile robots are susceptible to get trapped in local minima (or dead ends) of the environments.

Two types of approaches, i.e. boundary-following and virtual subgoal approach as described in Chapter 2, are specially developed to address the local minimum problem in the literatures [Huang & Lee, 1992; Kamon & Rivlin, 1997; Lim & Cho, 1998; Krishna & Kalra, 2001; Maaref & Barret, 2002; Chatterjee & Matsuno, 2001; Pin & Bender, 1999; Xu, 2000; Xu & Tso, 1999]. Section 7.7 provides a detailed

comparison of these approaches. Unfortunately, they are still difficult to guarantee global convergence in complex environments. The following, a-f, are just some of the difficulties that have to be overcome in solving local minimum problem. (a) When the goal is always at the side of the wall, a long-wall environment (Figure 7.1(a)) may cause a robot to be trapped in a wrong boundary-following direction. (b) Unstructured and cluttered environments (Figure 7.1(b)) invalidate methods that recognize typical landmarks. (c) A dynamic environment may lose preserved information, resulting in an inability to satisfy detection or escape criterion. (d) Recursive U-shape or maze-like environments (Figure 7.1(b)) may cause a robot to regress into the old local minimum. (e) Inaccurate localization estimation derived from the odometry drift problem may result in an inability to satisfy detection or escape criterion. (f) The sensing capability (e.g. sonar sensors) and sensing noises make it difficult to determine the size or location of obstacles when this information is required for the escape criterion.



(a) (b)

Figure 7.1: Two environment maps. S is the start of the robot, T is the goal location. (a) long-wall environment; (b) unstructured, cluttered, and maze-like environment

This chapter proposes a new navigation method that we call *minimum risk method* to address local minimum problem for goal-oriented robot navigation in unknown environments. The method is an application of the memory grid map proposed in Chapter 6. The key of the method is to design a novel regional Path-Searching (PS) behavior that complements the local OA and global GS behaviors commonly used in behavior-based navigation systems. The framework of behavior-based navigation using fuzzy logic proposed in Chapter 4 is used in this method. The mobile robot is required to reach a given goal by coordinating three elementary behaviors: PS, OA, and GS.

The rest of the chapter is organized as follows. Section 7.2 describes the design of

our path-searching behavior, Section 7.3 obstacle-avoidance behavior, and Section 7.4 goal-seeking behavior. Section 7.5 provides a detailed discussion about global convergence, the complexity of the method as well as the performance influenced by the localization technique. Section 7.6 shows the experimental results for both our simulated and real world tests. Section 7.7 categorizes and compares the existing methods with the proposed method. The final section summarizes this chapter.

7.2 The regional path searching behavior

This section designs a regional Path-Searching (PS) behavior that navigates a mobile robot to the safest (i.e. minimum risk) region in order to move away from the local minima. This region is among the LEFT, RIGHT, FRONT turn detection regions as shown in Figure 6.5 of Chapter 6. The region with minimum risk is the one with minimum values of both TMDs and OMDs in a memory grid map (The definitions are referred to Section 6.2.1). This map is updated every control cycle based on robot's sensory information as described in Section 6.3. Such regional direction with minimum risk is the best choice for the robot in trying to avoid both obstacles and previous trajectory, and consequently escape from the local minima. That's why we call our navigation method *minimum risk method*.

The use of a memory grid map for the PS behavior is similar to the use for the path-exploring (PE) behavior as described in Chapter 6, but the two behaviors are different at least on two aspects. First, the PE behavior is activated only at the time the robot is visiting its previously traversed areas so that the robot is driven to explore less visited environment. However, it is desired that the PS behavior is activated as long as the robot encounters the obstacles so that the robot can detect potential dead ends and escape from them. Therefore, the calculation of the PS behavior's weight must take the obstacle dot intensity (we define it in the following) into consideration. Second, the output space (i.e. turn angle) of the PE behavior only contains three crisp angle values $\{60^0, 0^0, -60^0\}$ that respectively correspond to LEFT, FRONT and RIGHT three regional direction. It is desired, however, that the output space of the PS behavior is continuous within the range $(-90^0, 90^0]$ so that the robot turns smoother. Therefore, we design the PS behavior using fuzzy logic controller rather than using an analytic algorithm like the design of the PE behavior.

To realize such PS behavior, we do at first by inferring a Risk Index for each turn

detection region. Then we develop the PS behavior's fuzzy turn rules based on the Risk Index, and develop a complementary algorithm. Finally, we develop a fuzzy logic to obtain the weight of the PS behavior. The details are as follows.

When the robot begins to move, it constructs a so-called memory grid map based on sensory information. During every control period, the OMD matrix $O_{M\times N}$ and the TMD matrix $T_{M\times N}$ are updated to represent the current environmental obstacles and the previously traversed trajectory. At the same time, advanced data features, i.e. iteration risk (IR), collision risk (CR), trajectory dot intensity (TDI), and obstacle dot intensity (ODI), are extracted from $O_{M\times N}$ and $T_{M\times N}$ for each turn and weight detection regions in order to aid the robot in making decision to turn next. In fact, the minimum risk means the minimum IR and CR. The IR and CR are used to infer the Risk Index for the fuzzy navigational rules of PS behavior. The TDI and ODI are used in combination with fuzzy logic to calculate the weight of the PS behavior. We have defined IR, CR, and TDI in the Definitions 6.1, 6.2, 6.3 respectively in Chapter 6. Now we define the ODI.

Definition 7.1 (Obstacle Dot Intensity): *Obstacle Dot Intensity* (ODI) of a region *B* is defined as $\tau(B) = \sum_{(i,j)\in B} V_{OMD}(i,j)$, where *B* is a square-shaped weight detection region (see Figure 6.5(b) in Chapter 6), $V_{OMD}(i,j)$ is the OMD's value of the cell (*i, j*) involved in the region *B.*

The magnitude α of IR is converted into three linguistic fuzzy sets {LOW, MEDIUM, HIGH}, with the membership functions shown in Figure 7.2(a). The magnitude β of CR is converted into three linguistic fuzzy sets {LOW, MEDIUM, HIGH} with the membership functions shown in Figure 7.2(b). The magnitude κ of TDI is converted into three linguistic fuzzy sets {SMALL, MEDIUM, BIG} with the membership functions shown in Figure 7.2(c). The magnitude τ of ODI is converted into three linguistic fuzzy sets {SMALL, MEDIUM, BIG} with the membership functions shown in Figure 7.2(d).

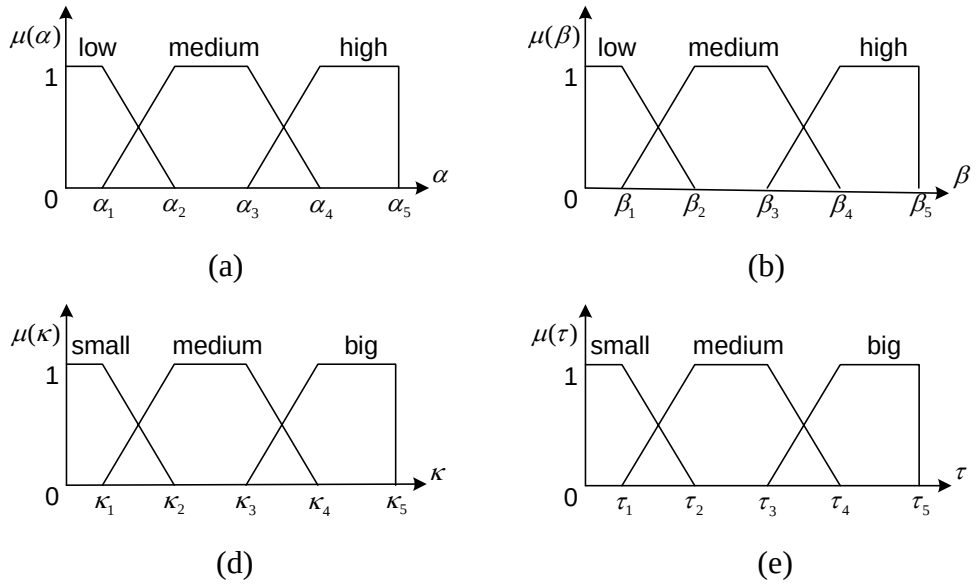


Figure 7.2: Membership functions (a) for iteration risk. (b) for collision risk. (c) for trajectory dot intensity. (d) for obstacle dot intensity.

7.2.1 Regional Risk Index

The Fuzzy Rule-Based Risk Index combines the two regional risk parameters into a single indicator of how safe it is for the mobile robot to traverse the region. The Risk Index *r* is represented by three linguistic fuzzy sets {DANGEROUS, UNCERTAIN, SAFE} with the membership functions shown in Figure 7.3(a). The Risk Index *r* is defined in terms of both the iteration risk α and the collision risk β by a set of simple intuitive fuzzy logic relations as Table 7.1. For instance, the (3, 3) element of Table 7.1 can be written as one rule: IF α is LOW AND β is LOW, THEN *r* is SAFE. Naturally, *we define that the region with minimum risk is the region that has a "SAFE" Risk Index*. The Risk Indices for three turn detection regions, r_{left} , r_{front} and r_{right}, are inferred using the fuzzy rules of Risk Index.

Table 7.1: Fuzzy rules of regional Risk Index *r*

α \ β	HIGH	MEDIUM	LOW
HIGH	DANGEROUS	DANGEROUS	DANGEROUS
MEDIUM	DANGEROUS	DANGEROUS	UNCERTAIN
LOW	DANGEROUS	UNCERTAIN	SAFE

Note that multiple rules can be active at the same time and the fuzzy classes have overlaps. Hence, the Risk Index can have, for instance, both 0.5 UNCERTAIN and 0.5 SAFE membership values. The multivalued nature of the proposed fuzzy logic representation of regional risk offers significant robustness and tolerance to the large amount of uncertainty and imprecision inherent in sonar sensing of a region. This robustness is due to the fact that the output of a rule-based system depends on the *fuzzy* values of the input variables.

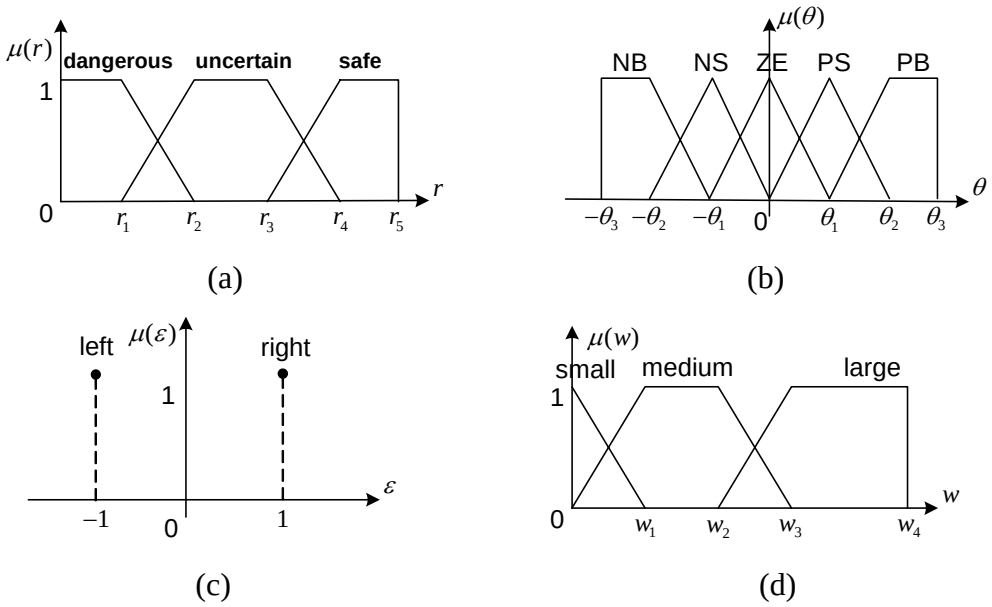


Figure 7.3: Membership function. (a) for Risk Index. (b) for turn angle. (c) for goal location. (d) for behavior weight.

7.2.2 Turn rules

Here the Risk Index is used to develop a fuzzy turn rules of the PS behavior. The motion control variables of the mobile robot are the translational speed v and the rotational turn angle θ. The robot's safety is influenced mainly by the OA behavior that is able to detect the environmental obstacles real time. Therefore we assume that the robot speed v is determined only by the OA behavior rather than by the PS or GS behavior. In addition, we assume that the robot can move only in the forward direction (i.e., reverse motion is not considered) because our robot does not have backward sensors. The robot turn angle θ is represented by five linguistic fuzzy sets {NB, NS, ZE, PS, PB}, with the membership functions shown in Figure 7.3(b), where NB is negative-big, NS negative-small, ZE zero, PS positive-small, and PB positive-big. The positive and negative terms have implied that the robot turns to the left and

right, respectively.

The turn rules of the PS behavior are summarized in Table 7.2. The rules have a tendency to select the direction that is closest to the forward direction, so that the robot does not make unnecessary rotations. As shown in Table 7.2, when the robot needs to turn but the left and right sectors have the same Risk Indices, then the recommended turn angle θ_{ps} is GOAL, where GOAL implies that the recommended turn angle should be toward the direction close to the goal location. For instance, the (3, 3) element of the top layer in Table 7.2 denotes two rules:

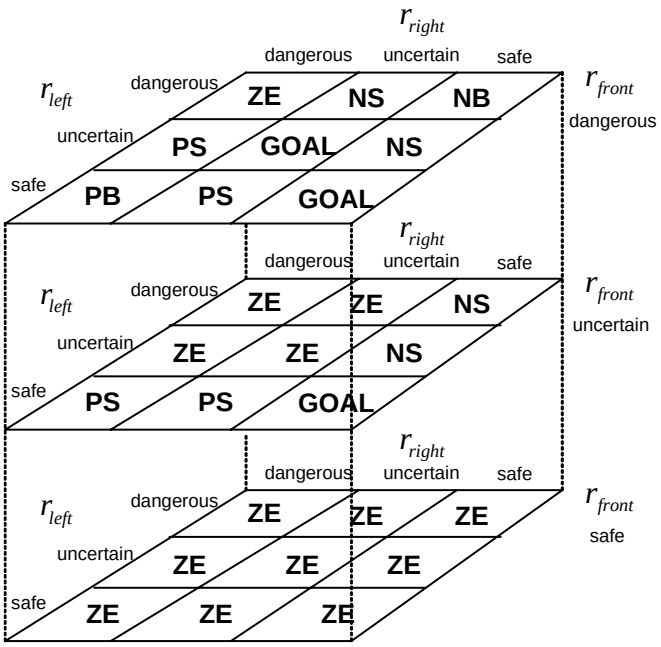
IF r_{front} is DANGEROUS AND r_{left} is SAFE AND r_{right} is SAFE AND ε is LEFT, THEN θ_{ps} is PS;

IF r_{front} is DANGEROUS AND r_{left} is SAFE AND r_{right} is SAFE AND ε is RIGHT, THEN θ_{ps} is NS;

where, ε is the goal location, with the membership functions shown in Figure 7.3(c).

Another important note: a turn maneuver is not initiated when the three sectors have the same *dangerous* risk indices as shown in the (1, 1) element of the top layer in Table 7.2. The turn rule does not force the robot to arbitrarily choose between left and right, but maintain the turn angle at zero at this stage. The final selection will be made using a complementary algorithm introduced in the following.

Table 7.2: Turn rules for the path-searching behavior.



The fuzzy turn rules proposed above can work well in most of possible situations so as to make recommendation for the region with the minimum risk. On the other hand, when the robot is located in an extreme situation (e.g. three turn detection regions have the same HIGH iteration risk), the fuzzy turn rules cannot judge the region with real minimum iteration risk. We know that the region with HIGH collision risk cannot be recommended, but under such extreme situation the robot can choose a region that has a HIGH iteration risk but its value is minimum among the three turn detection regions (i.e. left, front and right regions).

The kernel idea of the complementary algorithm as seen in Algorithm 7.1 is that if the turn regional direction recommended by the turn rules does not have a SAFE Risk Index (by threshold comparison), the collision risk and iteration risk of all three sectors are compared again (by threshold comparison) so as to recommend a regional direction that has a safe collision risk and a minimum iteration risk. The thresholds for IR and CR are α_1 and β_1 as shown in Figure 7.2(a) and (b) respectively. The complementary algorithm is exact, not fuzzy. Both the turn rules and the complementary algorithm are comprised of a complete framework for calculating the turn angle of PS behavior as shown in Figure 7.4. Figure 7.5 shows the architecture of the fuzzy logic controller whose details are described in Chapter 4. Therefore, the turn angle recommended by the PS behavior can prevent the robot from iterating its previous trajectory as few as possible, so that the robot chooses to explore a new region as a means of escaping from the local minimum.

Algorithm 7.1: (A complementary algorithm for turn rules)

Input: θ_{ps} = original crisp turn angle output by turn rules of the PS behavior;

$\alpha(A_{front})$, $\alpha(A_{left})$, $\alpha(A_{right})$ = IR values of LEFT, FRONT, and RIGHT turn detection regions respectively;

$\beta(A_{front})$, $\beta(A_{left})$, $\beta(A_{right})$ = CR values of LEFT, FRONT, and RIGHT turn detection regions respectively;

Output: θ_{ps} = final turn angle of the PS behavior

BEGIN:

Step 1: IF the turn region recommended by the turn rules has a lower IR value than the threshold α_1, THEN maintain the original turn angle and RETURN; OTHERWISE go to Step 2;

Step 2: To check whether or not there are regions that have a lower CR value than the threshold β_1. IF not, THEN maintain the original turn angle and RETURN; OTHERWISE go to Step 3.

Step 3: IF only one region that has a lower CR value than the threshold β_1 exists, THEN this region is recommended as the turn direction and its turn angle is returned; OTHERWISE go to Step 4.

Step 4: The region, which has a minimum IR value, is recommended and its turn angle is returned.

END Algorithm 7.1

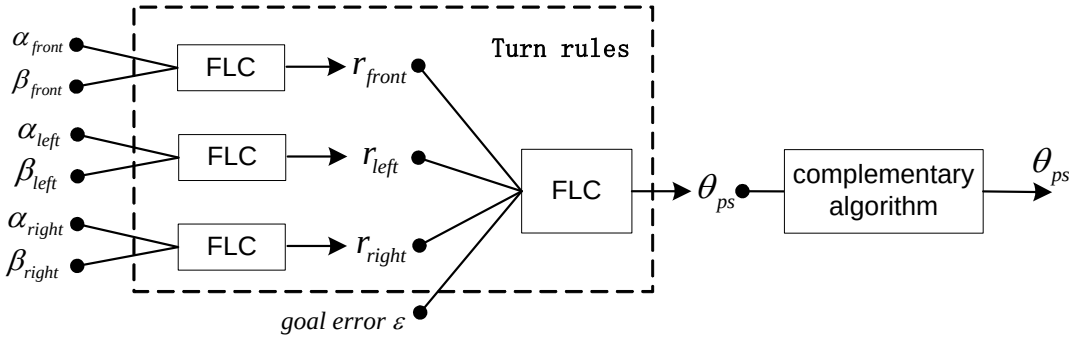


Figure 7.4: The determination of the turn angle recommended by the PS behavior using both Turn Rules and a complementary algorithm.

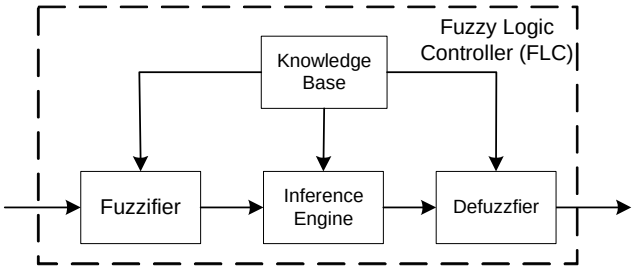


Figure 7.5: Architecture of fuzzy logic controller (FLC).

7.2.3 Weight rules

The weighting factor w_{ps} represents the strength by which the PS behavior recommendation is taken into account to compute the final motion command. The weight of PS behavior is represented by three linguistic fuzzy sets {SMALL, MEDIUM, LARGE} with the membership functions shown in Figure 7.3(d), and is derived directly from both the TDI and ODI (see Section 3.1) of a square-shaped

region (i.e. weight detection region), using the rule sets as in Table 7.3. For instance, the (1,1) element of Table III represents one rule:

IF κ *is BIG AND* τ *is BIG, THEN* w_{ps} *is LARGE.*

Table 7.3: Fuzzy weight rules of the PS behavior

κ \ τ	BIG	MEDIUM	SMALL
BIG	LARGE	LARGE	LARGE
MEDIUM	LARGE	LARGE	MEDIUM
SMALL	LARGE	MEDIUM	SMALL

7.3 The local obstacle avoidance behavior

The local Obstacle-Avoidance (OA) behavior is a sensor-based behavior which makes the robot safe without collision with obstacles. It is activated if obstacles are approaching. We design the OA behavior using fuzzy logic controller, almost the same as that of the OA behavior in Section 4.3 of Chapter 4. The difference is only the turn rules.

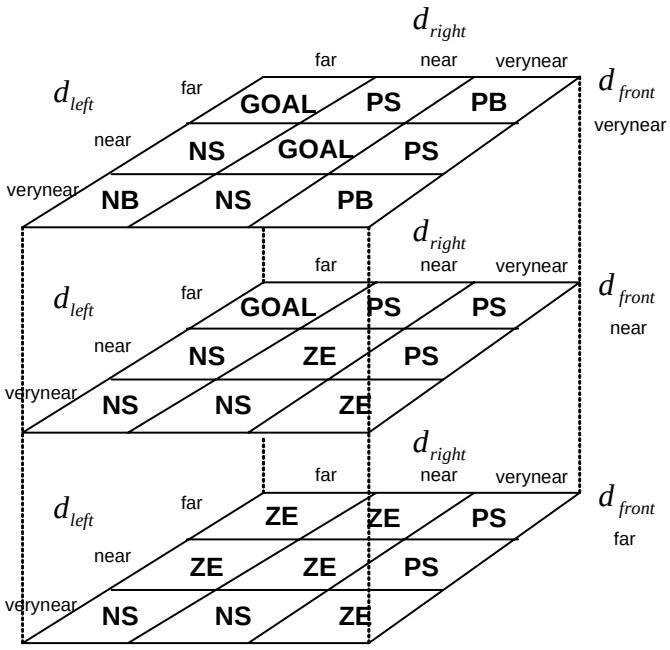
In this navigation method, the turn rules for the OA behavior are summarized in Table 7.4. When the robot needs to turn, but the left and right sectors have the *same* obstacle distance, then the recommended turn angle is GOAL, where GOAL implies that the recommended turn angle should be toward the direction close to the goal location. This is similar to the turn rules for PS behavior. For example, the (1, 1) element of the top layer in Table 7.4 represents two rules:

IF d_{front} *is VERYNEAR AND* d_{left} *is FAR AND* d_{right} *is FAR AND* ε *is LEFT, THEN* θ_{oa} *is PS;*

IF d_{front} *is VERYNEAR AND* d_{left} *is FAR AND* d_{right} *is FAR AND* ε *is RIGHT, THEN* θ_{oa} *is NS;*

where, ε is the goal location.

Table 7.4: The turn rules of the OA behavior.



7.4 The global goal seeking behavior

The Goal-Seeking (GS) behavior is a global behavior which does not rely on external sensing data, but seeks for the exact goal location. The calculation of the speed and turn angle recommended by the GS behavior is same as that of the GS behavior in Section 4.3 of Chapter 4. But their weight rules are different.

The weight w_{gs} of the GS behavior here is based on the weights of both OA and PS behaviors. Figure 7.6 shows the weight determination of three behaviors. Table 7.5 summarizes the weight rules of the GS behavior.

Table 7.5: Fuzzy weight rules of the GS behavior

w_{oa} \ w_{ps}	LARGE	MEDIUM	SMALL
LARGE	SMALL	SMALL	SMALL
MEDIUM	SMALL	SMALL	SMALL
SMALL	SMALL	SMALL	LARGE

Importantly, the weight of GS behavior is suppressed and is small when any

weight of the OA or PS behaviors is not SMALL. When the weights of both OA and PS are SMALL, the GS behavior can make a dominant contribution to the final control command. Although the GS behavior is usually suppressed, the GOAL factor is reflected in the turn rules of both OA and PS behaviors (see Tables 7.2 and 7.4). It is an important factor for our minimum risk method to ensure global convergence. This point is analyzed in Section 7.5.

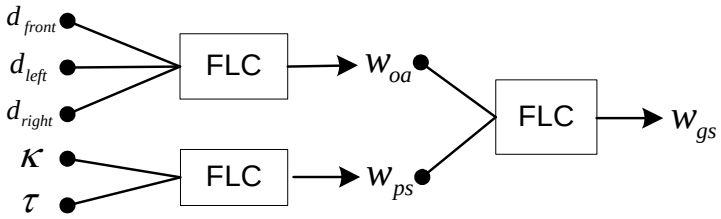


Figure 7.6: Weight determination of OA, PS, GS behaviors.

For every control cycle, Algorithm 7.2 is called once to perform the goal-oriented navigation by coordinating three behaviors in our minimum risk method.

Algorithm 7.2: (Goal-oriented navigation by minimum risk method)

Input: (x1, y1) = goal location; (x0, y0) = current robot location;

φ0 = current robot heading angle;

(d0, d1, d2, d3, d4, d5, d6, d7) = sonar readings.

Output: (v, θ) = speed and delta turn angle

BEGIN:

Step 1. Update sensory data including (x0, y0), φ0 and (d0, d1, d2, d3, d4, d5, d6, d7);

Step 2. IF the distance from current robot location (x0, y0) to goal location (x1, y1) is less than a predefined threshold (i.e. distance tolerance), THEN the goal is reached and the robot is stopped, OTHERWISE go to the Step 3;

Step 3. Preprocess the sonar readings using Eqs. (4.6), (4.7), and (4.8);

Step 4. Update the OMD matrix $O_{M\times N}$ and the TMD matrix $T_{M\times N}$ using Algorithm 6.1;

Step 5. Update the IR and CR of three turn detection regions, including $\alpha(A_{front})$, $\alpha(A_{left})$, $\alpha(A_{right})$, $\beta(A_{front})$, $\beta(A_{left})$, and $\beta(A_{right})$;

Step 6. Update the TDI's value $\kappa(B)$ and ODI's value $\tau(B)$ of weight detection region B;

Step 7. Calculate the Risk Index of three turn detection regions, including $r(A_{front})$, $r(A_{left})$, $r(A_{right})$ using Algorithm 4.1 and the fuzzy rules as in Table 7.1;

Step 8. Calculate the turn angle θ_{ps} recommended by the PS behavior using Algorithm 4.1 and the turn rules as in Table 7.2;

Step 9. Calculate the final turn angle θ_{ps} recommended by the PS behavior using the complementary Algorithm 7.1, and set the speed of the PS behavior to zero;

Step 10. Calculate the weight w_{ps} of the PS behavior using Algorithm 4.1 and the weight rules as in Table 7.3;

Step 11. Calculate the speed v_{oa} and the turn angle θ_{oa} recommended by the OA behavior using Algorithm 4.1 and the turn rules as in Table 7.4, the move rules as in Table 4.2;

Step 12. Calculate the weight w_{oa} of the OA behavior using Algorithm 4.1 and the weight rules as in Table 4.3;

Step 13. Calculate the speed v_{gs} and the turn angle θ_{gs} recommended by the GS behavior using Eqs. (4.9) and (4.10);

Step 14. calculate the weight w_{gs} of the GS behavior using Algorithm 4.1 and the weight rules as in Table 7.5;

Step 15. To calculate (v, θ) by the command fusion using Eqs. (4.4) and (4.5);

Step 16. To execute the motor control commands (v, θ).

END Algorithm 7.2

7.5 Performance Analysis

7.5.1 Convergence analysis

When there exists a region with minimum risk in the turn detection regions (i.e. left, right, and front regions), the PS behavior can be guaranteed to recommend it. The reasons are the following. First, the Risk Index rules guarantee that if a region (among LEFT, FRONT, and RIGHT regional sectors) contains both LOW Collision Risk (CR) and LOW Iteration Risk (IR), it might be labeled as “SAFE” region. The turn rules of PS behavior guarantee that the direction toward a region with “SAFE” Risk Index must be recommended if such a region exists. Second, when no region that has a “SAFE” Risk Index exists, a complementary algorithm is triggered. The

complementary algorithm guarantees that if regions that have a safe CR (less than the threshold β_1) exist, a region that has a safe CR and a minimum IR must be recommended by the exact threshold comparison.

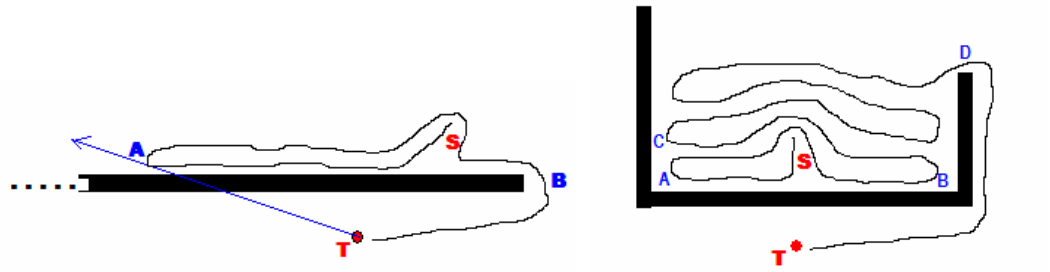
Consequently if a solution path exists for a goal-oriented navigation task in unknown environment, the minimum risk method can guarantee global convergence to the given goal location. This is the reason that the solution path always contains minimum collision risk and iteration risk (i.e. SAFE Risk Index), while the method guarantees that such region with minimum risk can be recommended by the PS behavior so as to escape from potential local minima and the global goal is sought by coordinating three behaviors (i.e. PS, OA, ad GS).

Even though the robot is located in a dynamic environment (e.g. moving humans exist), the minimum risk method can guarantee global convergence if a solution path exists. This is the reason that the robot may detect the environmental change in real-time and update a memory grid map accordingly. Based on the continuously updated memory grid map, the PS behavior can choose the safest direction to escape from potential local minima. At the same time, the OA behavior keeps the robot safe, which is able to respond to any contingency and to avoid the collision with any possible stationary or dynamic obstacles.

7.5.2 Trial-and-return phenomenon

Now we introduce an interesting and important behavioral phenomenon of the robot. We call it "*trial-and-return*" phenomenon, as shown in Figure 7.7. The robot is required to move from the start S to the goal T. At first, the robot moves toward the goal along a straight line, chiefly guided by the GS behavior. When obstacles are encountered, the robot follows the boundary of the obstacles. But the underlying mechanism of this boundary following is totally different from that of other methods [Huang & Lee, 1992; Krishna & Kalra, 2001; Maaref & Barret, 2002]. It is not the result of a single behavior, but of the coordination of the OA, GS, and PS behaviors. Although the weight of the GS behavior is small, the influence of the GOAL is represented in the turn rules of OA and PS behaviors (see Tables 7.2 and 7.4). As indicated in Table 7.4 the OA behavior may recommend a turn angle in order to turn away from the lateral obstacle. Just as shown in Figure 7.7(a), the robot tries to turn right so as to keep itself away from the wall boundary, but the GOAL factor leads the

OA and PS behaviors to recommend the robot to turn left toward the wall because the goal T is at the side of the wall. The robot exhibits the action of following the wall until it moves to the location A as seen in Figure 7.7(a). At location A, the goal error angle between the current robot heading and the goal direction is very large. When the OA behavior tries to turn the robot far away from the wall boundary, this goal error angle increases beyond 180˚, which causes the goal T to change from the left side of the robot to the right side. Thus the GOAL factor enables the robot to turn backward and return, instead of following the wall boundary again. At this time, the PS behavior makes a dominant contribution to enable the robot to move closing to the previous trajectory instead of moving in the same trajectory. The location B is the nearest exit, where the robot might continue to move and reach the goal T under the dominant influence of GS behavior. At location A of the Figure 7.7(b), the goal T is quickly changed from the left side of the robot to the right side because of the forward wall obstacle. Then the robot returns and moves to the location B. A similar situation occurs at the location B. Thus the robot returns again and moves to location C. This kind of "trial-and-return" behavioral phenomenon is maintained until the robot arrives at the nearest exit D.



(a) (b)

Figure 7.7: "trial-and-return" behavior phenomenon. S is the start of the robot. T is the goal location.

Obviously, if there is no obstacle blocking the nearest exit, the "trial-and-return" behavior phenomenon enables the robot to find the exit and escape from the local minimum. It is verified by the experiments in Section 7.6. This property is particularly useful in the local minimum problem. Although sometimes the "trial-and-return" behavior phenomenon looks stupid in an environment such as that in Figure 7.7(b), it is a smart strategy for all kinds of environmental situations because it guarantees that the robot is never trapped in a wrong boundary-following direction. Most of local

minimum problems have a nearer exit to escape from the dead ends. The “trial-and-return” phenomenon thus improves the efficiency of the minimum risk method. More importantly, it guarantees global convergence.

7.5.3 Complexity analysis

A) Space complexity

The minimum risk method requires a fixed memory space to save a memory grid map if the goal location is determined. Importantly, this space requirement does not change with the navigational time or environmental complexity. The map should cover the physical areas that include the start, the goal and the solution path. When the cell size of the map is determined, the size of the whole memory grid map is determined. Assume that the length of the cell size is λ, so that an M×N grid map covers a physical space whose area is $M\times\lambda\times N\times\lambda$. For example, if λ= 0.2 metre (in our robotic system), a 1000×50 grid map may cover an actual space whose area is $(1000\times0.2)\times(50\times0.2)$ = 2000 m^2.

If the memory space of a robot is really not enough or the goal is too distant, a dynamic memory grid technique can be used to obviate the need for a large memory. Note that only those cells that are located inside a circular sector are updated in each control period, and the decision is determined only based on a small range of map information and sensory data. Therefore, the robot needs to save only the necessary memory grid map information into the working memory while other map information is saved in the hard disk. If necessary, the other map information is switched to the working memory. Using the dynamic memory grid technique, it is possible to control the memory space requirement into an acceptable range.

B) Time complexity

The computational time of the minimum risk method is fixed and efficient. As discussed in A), the decision is determined based on a small range of map information and sensory data. Hence, the calculations of four features (iteration risk, etc.) involve very few addition operations. In addition, the fuzzy rule-based navigation algorithm is computationally fast and efficient [Heraji & Howard, 2002]. Other calculations such as command fusion are some simple equations or threshold comparison.

7.5.4 The performance influenced by localization technique

There are two classic problems in robotics: Where is the robot, and how does the robot reach the goal? These two problems correspond respectively to localization [Victorino et al, 2003] and path-planning problems [Meyer & Filliat, 2003]. The self-localization using dead reckoning data is widely used, but it tends to inaccurately estimate the robot's location because of the well-known odometry drift problem caused by wheel slippage, gear backlash and so on. Long-distance movement makes this error even worse. This problem can be improved by adopting global or external localization techniques or special devices [Filliat & Meyer, 2003; Golfarelli et al, 2001]. Although many researchers have addressed this problem, it is still difficult to be fully solved. The localization technique is not our focus in this chapter. We here address only the path-planning problem in terms of the goal-oriented navigation within an unknown indoor environment with local minimum.

The minimum risk method can guarantee global convergence if an ideally accurate localization of the robot exists. It can also work if a self-localization technique using dead-reckoning data is adopted. Consider that the local minimum often occurs in a small space (e.g. within 10 m^2), the accumulated data error from odometry drift is not serious. The memory grid map uses a value of TMD or OMD to save the information of the whole cell area, and the PS behavior uses the information of a whole detection region to make decision, which naturally can tolerate a certain degree of data error. On the other hand, the foundation of fuzzy logic is the representation of, and reasoning with, imprecise information. Fuzzy logic provides a systematic framework for dealing with imprecise and uncertain information. Therefore, the odometry drift problem has little influence on the minimum risk method if it is used in a small space. This point is verified by the simulation and real world tests in Sections 7.6.4, 7.6.5 and 7.6.6. In fact, the main influence of an inaccurate localization technique is that the robot misses the goal's exact location and it arrives at another nearby location. This problem, however, is a matter related to the localization technique, which is beyond the scope of this chapter.

7.6 Experimental results

To exhibit the different contributions to the final control output provided by the different behaviors, we use data visualization. Each behavior produces a turn angle recommendation while its weight represents the degree of influence on the final angle control output. Thus the different lines, drawn automatically by the control program, respectively represent the final turn angle and the turn angles recommended by the OA, PS, and GS behaviors. The length of each line represents the weight value of each behavior. The trajectory is indicated by the chain of circles. The program draws the circle once every 0.5 second. A denser concentration of circles thus indicates that the robot is travelling more slowly.

7.6.1 Performance analysis in long-wall environments

The purpose of this experiment is to analyze the decision-making process when the robot adopts our minimum risk method. The robot is required to move from the start S to the goal T. When the robot starts to move at a normal (maximum) speed, the OMD and TMD values saved in the memory grid map are SMALL, so that the TDI and ODI are SMALL. The weight of the PS behavior is thus small. At this time the weight of the OA behavior is small too because the front obstacle is distant. Consequently, at this time the GS (line c) behavior makes the dominant contribution to the final motion output (Figure 7.8(1)). When in response to a nearby obstacle the robot decreases its speed, the number of memory dots increases and the TDI or ODI becomes MEDIUM or LARGE. Consequently, the weight of PS behavior increases, and the PS behavior (line b) is activated in these cases (see (2)(3)(4)(5)(6) in Figure 7.8); When the robot is approaching the obstacles, the weight of OA behavior (line a) becomes large (see (2)(4)(6) in Figure 7.8); When the OA or PS behaviors are dominant, the GS behavior is suppressed and its weight is small (see (2)(3)(4)(5)(6) in Figure 7.8). When the robot is far from the obstacles and is approaching the goal T at a normal speed, the weight of both OA and PS behavior is small and only the GS behavior is dominant (Figure 7.8(7)). Figure 7.8(9) shows the underlying memory grid map, drawn as horizontal and vertical lines. Figure 7.8(9) also shows our control and display interface. The labels A, B, C, D, E, F, S, and T represent the robot locations, as shown in Figure 7.9(a-d) and Figure 7.8(8).

Figure 7.9 (a) shows the turn angles recommended by different behaviors. The

turn angles recommended by OA and PS behavior are consistent during most of the entire task period. At the locations B, C, and D, the goal T is switched from the left of the robot to the right, or from the right to the left. This is why the robot leaves the wall at location D and turn toward location E. This “trial-and-return” property enables the robot to avoid being trapped in a wrong boundary-following direction.

Figure 7.9 (b) shows the weight values of different behaviors. The weight of GS behavior is suppressed and small when the weight of either OA or PS is larger. Figure 7.9 (c) shows the relation between memory dot intensity and the PS behavior’s weight. The TDI and ODI determine the weight of PS behavior (refer to Section 7.2.3). Figure 7.9 (d) shows the robot’s speed during the task period. The speed may decrease when the robot is approaching to an obstacle.

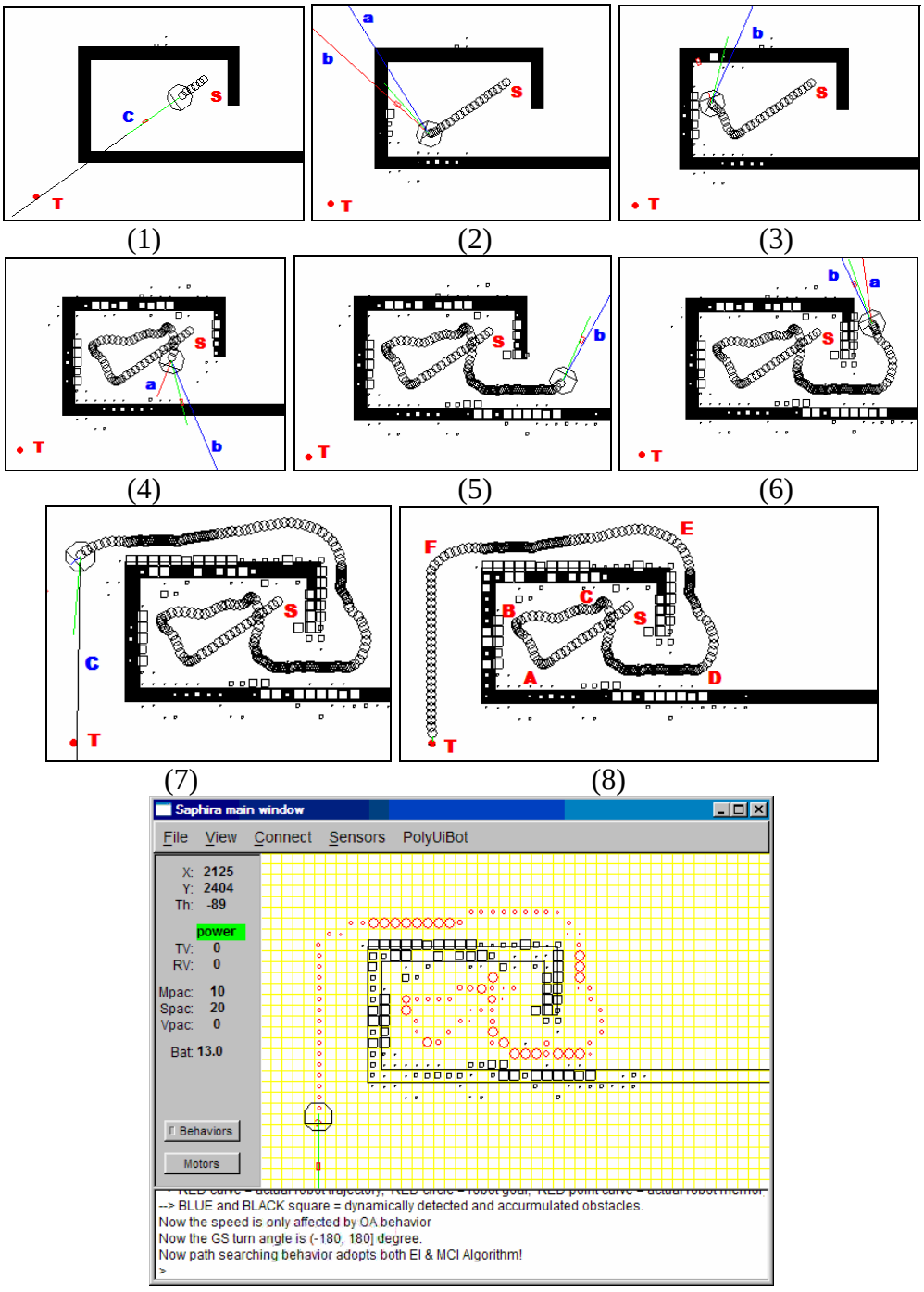


Figure 7.8: Minimum risk method to the long-wall environment with local minimum. (1-8) S is the start, T the goal target. OA is the line "a", PS is the line "b", GS is the line "c". (9) Underlying memory grid map is shown by the spaced horizontal and vertical lines; The obstacle memory dots (OMD) are drawn as the squares of different sizes. The larger is the square's size, the higher the possibility of the obstacle is; The trajectory memory dots (TMD) are drawn as the circles of different sizes. The bigger is the circle, the larger the TMD value is.

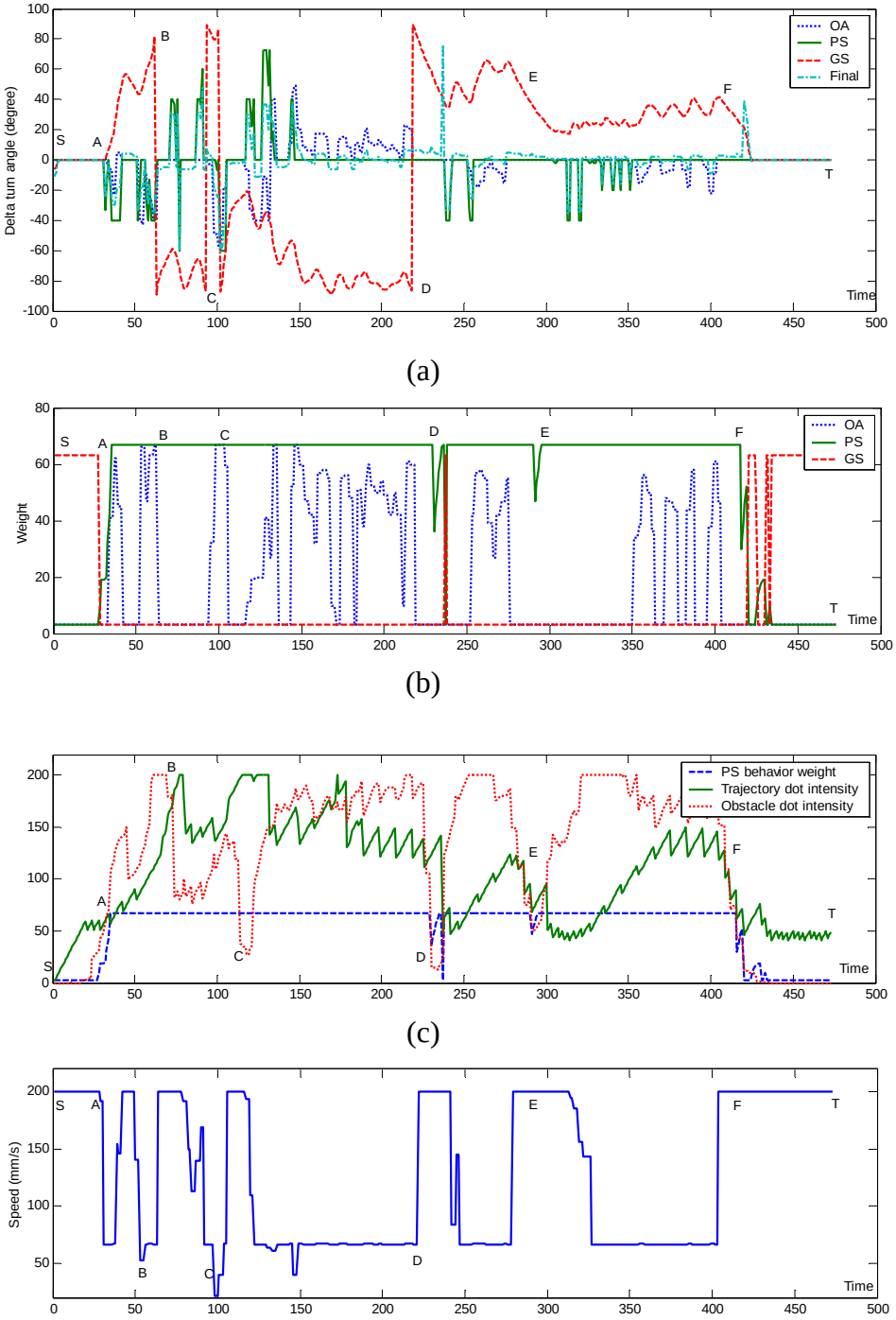
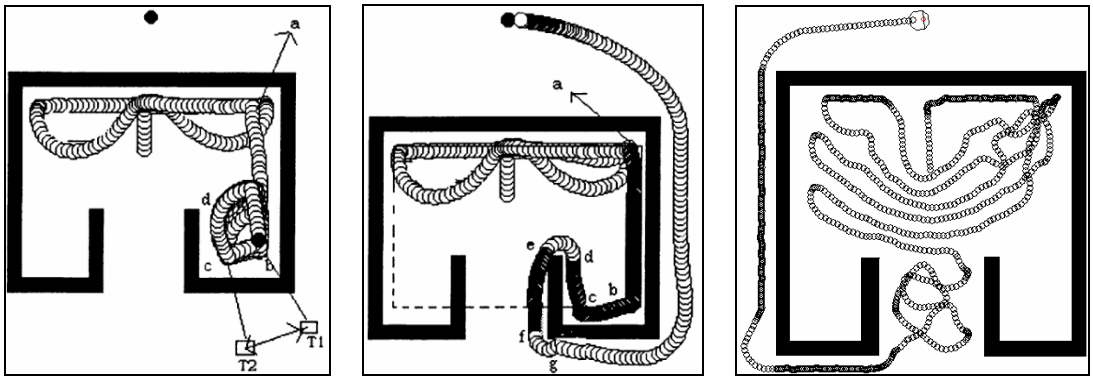


Figure 7.9: (a) Turn angles recommended by different behaviors. For the display, the GS turn angle is a half. (b) Weight values suggested by different behaviors. (c) The relation among the memory dot intensity and PS behavior weight. (d) The speed of the robot during the task period.

7.6.2 Comparison of performance in concave environments

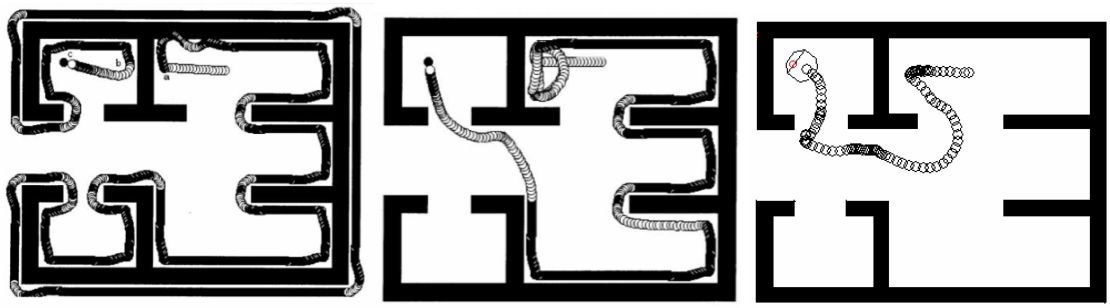
We firstly compare our minimum risk method with the virtual target method [Xu & Tso, 1999; Xu, 2000], Krishna and Kalra's method [2001], and Maaref and Barret's method [2002]. All of them are applied to a large concave and recursive U-shape environment. As shown in Figure 7.10(a), the virtual target method detects the local minimum by using an abrupt change in the goal orientation with respect to current robot heading. Upon detection, the robot continues to navigate using a new virtual goal orientation T1 at the location "a" until it finds an opening. But the robot detects a new local minimum at the locations "b" and "c". As a result, the robot is trapped in a dead cycle as it seeks both T1 and another new virtual target T2. The virtual target method fails to reach the goal in this kind of recursive U-shape environment. A modified strategy proposed by Krishna et al. [2001] can improve the virtual target method, but is still not suitable for complex environments. Figure 7.10(b) shows the result of Krishna and Kalra's method. This method detects the local minimum by recognizing a landmark encountered in the previous navigation. The robot then follows the wall boundary until it goes outside a configured bounding rectangle. This method highly depends on landmark recognition and exact coordination localization. In addition, as discussed in the next paragraph, it is difficult to choose a correct boundary-following direction. Figure 7.10(c) shows the result of our minimum risk method. The robot exhibits a typical "trial-and-return" phenomenon, which helps the robot find the nearest exit to escape from the local minimum and guarantee global convergence. This is further demonstrated in Figure 7.11(c). Maaref and Barret's method fails to reach the goal in this large concave environment because it detects the local minimum using a restricted criterion that all sensors must give small distances to obstacles at the same time.



(a) (b) (c)

Figure 7.10: In a large concave and recursive U-shape environment. (a) virtual target method. (b) Krishna and Kalra's method. (c) our minimum risk method.

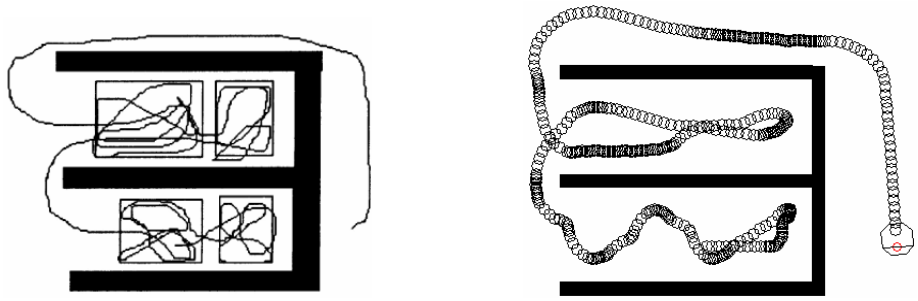
We next compare our minimum risk method with Krishna and Kalra's method [2001], Huang and Lee's method [1992], Distbug [Kamon & Rivlin, 1997; Lim & Cho, 1998], and Virtual-target-side method [Chatterjee et al., 2001]. Figure 7.11(a) shows the result of Huang and Lee's method. This method detects the local minimum by comparing a large difference in rotation of the robot over successive control periods. The robot then follows the wall boundary until an escape criterion is satisfied. As seen in Figure 7.11 (a), when the detection point a, the escape point b and the goal c are collinear and b is between a and c, the robot leaves the wall boundary and seeks for the goal again. This conservative escape criterion produces a longer path than other methods that adopt the boundary-following strategy. Figure 7.11 (b) shows the result of Krishna and Kalra's method. This method has a better escape criterion but it is still difficult to choose the correct boundary-following direction. Similar problems occur using the Distbug method and Virtual-target-side method. The main difference between both is that they have different detection and escape criteria. Figure 7.11 (c) shows our minimum risk method. It finds the nearest exit to reach the goal.



(a) (b) (c)

Figure 7.11: In a concave environment. (a) Huang and Lee's method. (b) Krishna and Kalra's method. (c) our minimum risk method.

Finally, we compare our minimum risk method with the virtual obstacle method [Pin & Bender, 1999]. Figure 7.12 (a) shows the result of the virtual obstacle method. This method finds the local minimum when the robot twice visits the same location with the same orientation. This detection criterion is so difficult to satisfy that the unnecessary iteration is caused. Upon detection, this method sets a virtual obstacle that involves all traversed path, and sets a new subgoal that is located outside the virtual obstacle. When this subgoal is reached, the robot recovers the original goal. In this process, the robot has to memorize all traversed trajectories and this requires a very large memory. This method produces the longest path of all referred methods. Figure 7.12 (b) shows the result of our minimum risk method. Clearly, it is simple and efficient.

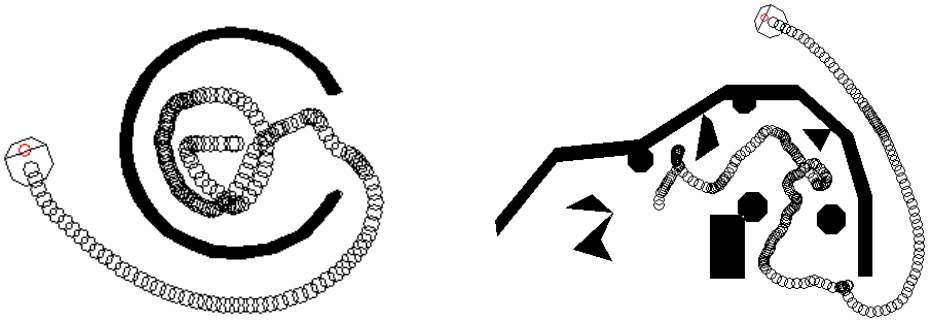


(a) (b)

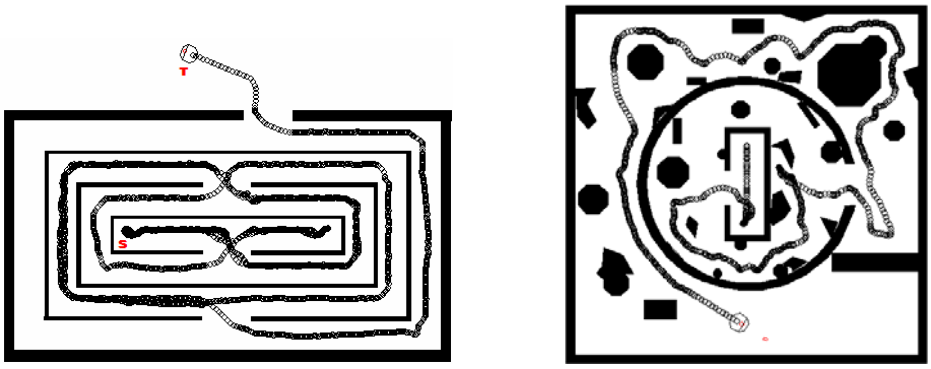
Figure 7.12: In a recursive U-shape environment.
(a) virtual obstacle method. (b) our minimum risk method.

7.6.3 Performance in complex environments

We tested our minimum risk method in unknown complex environments. Figure 7.13 (a), (b), (c) show the results for, respectively, circular, unstructured and cluttered, and maze-like environments. Figure 7.13 (d) shows the result for a highly complex environment that is unstructured, cluttered, recursive U-shape, and maze-like. The underlying mechanism of the results has been analyzed in Sections 7.5 and 7.6.1.



(a) (b)

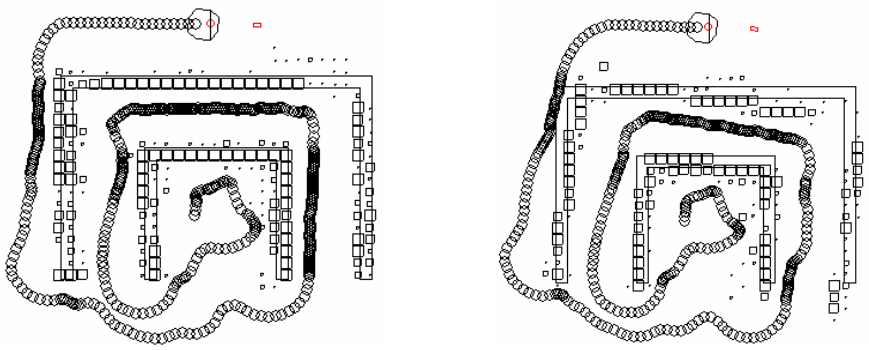


(c) (d)

Figure 7.13: (a) in circle-shape environment. (b) in unstructured and cluttered environment. (c) in maze-like environment. (d) in highly complex environment.

7.6.4 Performance in simulation with odometry drift

This section considers the performance influenced by the odometry drift problem. Odometry drift produces inaccurate location estimation, which is reflected by the obstacle memory dots as seen in Figure 7.14. Figure 7.14 (a) and (b) show the results without and with odometry drift respectively. As discussed in Section 7.5.4, the use of the memory grid map and fuzzy logic makes our minimum risk method tolerant of the uncertainty and errors derived from sensor noise and self-localization. Ultimately, the robot reaches the desired goal.



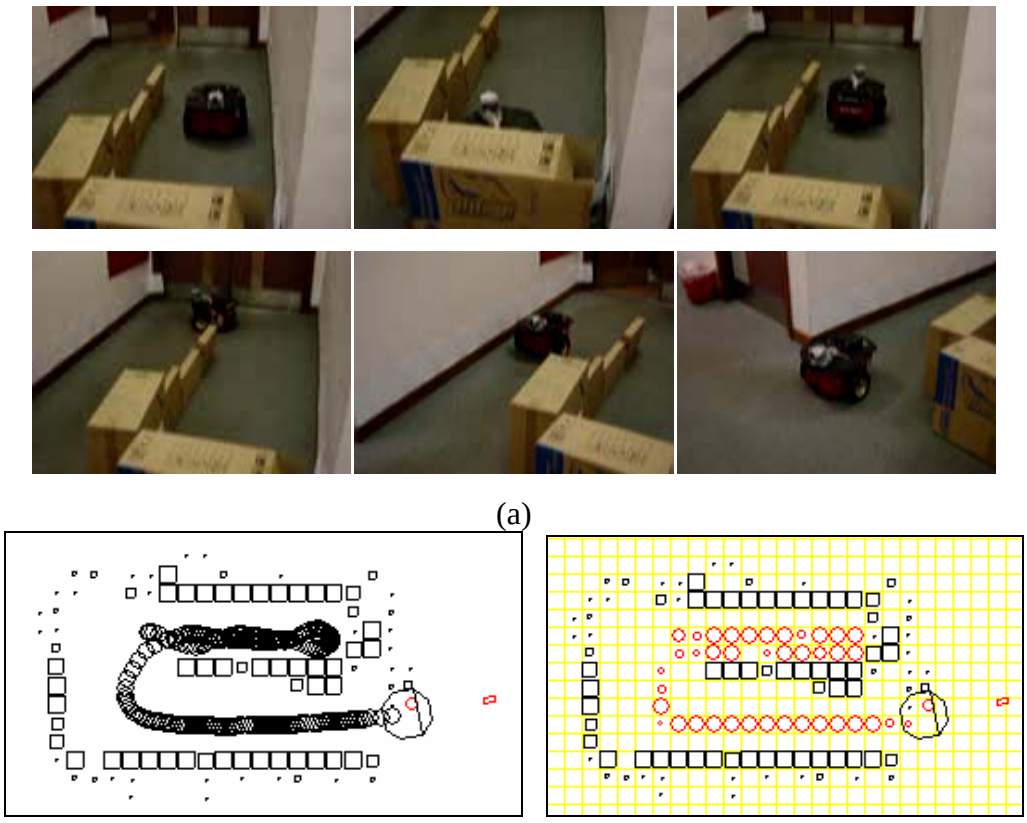
(a) (b)

Figure 7.14: In a recursive U-shape environment. (a) without odometry drift. (b) with odometry drift.

7.6.5 Performance in real world with odometry drift

In this section, we describe a real world test conducted in a corridor located in our department. Most related methods do not adequately consider the performance

influenced by the odometry drift problem, and are little tested in the real world. Here we simply use the dead-reckoning data for self-localization. Figure 7.15 (a) shows a series of pictures captured during robot movement. Figure 7.15 (b) shows the actual trajectory and the OMD. Figure 7.15 (c) shows the memory grid map that saves the TMD and OMD. The result has verified that there is little influence from the odometry drift problem if our minimum risk method is applied in a small space (e.g. less than $10m^2$) where the local minimum occurs. To address the local minimum problem in a large space, the other localization techniques have to be used to compensate for the drift error.



(a)

(b) (c)

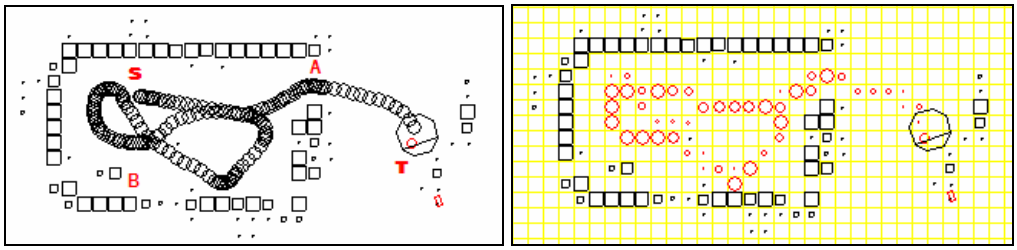
Figure 7.15: Performance in real world with local minimum

7.6.6 Performance in real world with dynamic environment

Here we exhibit a real world test in a dynamic environment (i.e. moving human exists). Figure 7.16 (a) shows a series of pictures of the test. One person first blocks the exit “A” (Figure 7.16 (b)), which forces the robot to turn around in order to look for another exit. Then the person moves to the location “B” where he is approaching

the robot but leaving the exit “A” clear. When the robot avoids the person and it is approaching to the exit “A” again, the OMD in “A” is updated to decrement its value so that “A” becomes a safest regional direction with minimum collision risk and iteration risk. Consequently, the robot finds the exit “A” while avoiding the moving obstacle (i.e. the person). Figure 7.16 (b) shows the actual trajectory and the OMD. Figure 7.16 (c) shows the memory grid map that records the TMD and OMD.

(a)



(b) (c)

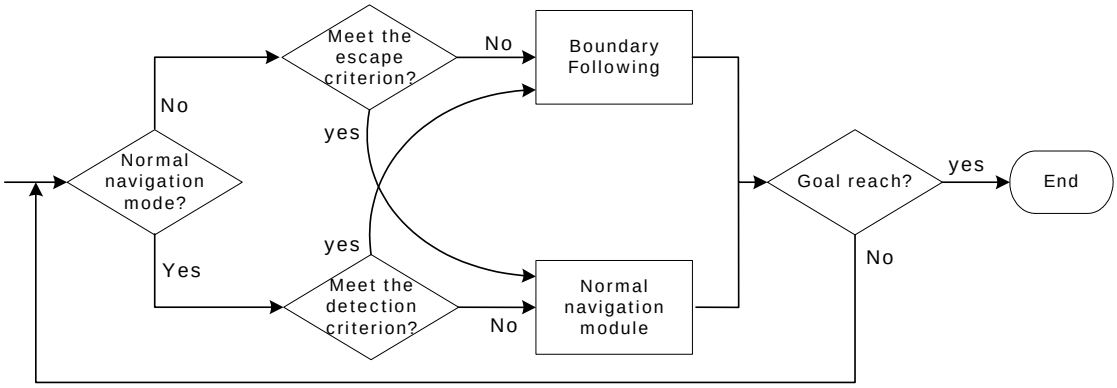
Figure 7.16: Performance in dynamic real world.

7.7 Categorization and comparison with related methods

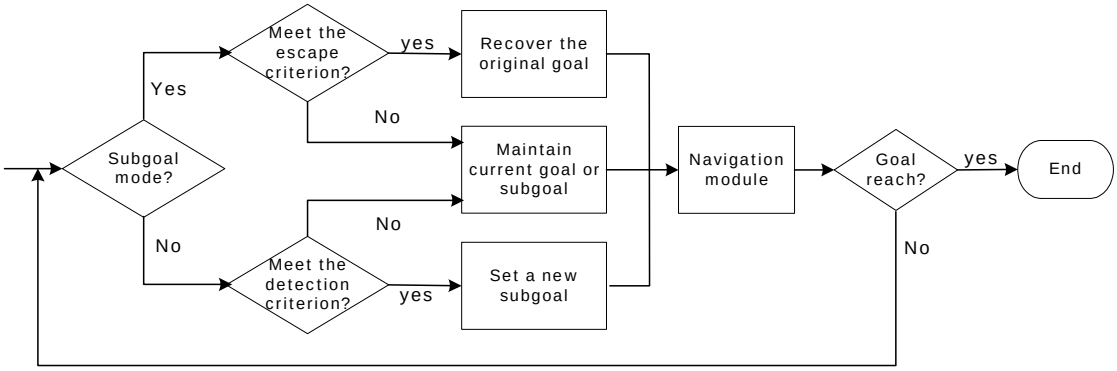
The literatures [Araujo et al, 1999; Seraji & Howard, 2002; etc.] address the robot navigational problem using machine learning or fuzzy behaviors approaches. They do not focus on local minimum problem in unknown indoor environments. As a result, they at most handle very simple environment with local minimum, and cannot go to the goal location in more complex environments with local minima.

For the related methods [Huang & Lee, 1992; etc.] that focus on local minimum problem, we have categorized them as three types of approach: boundary following,

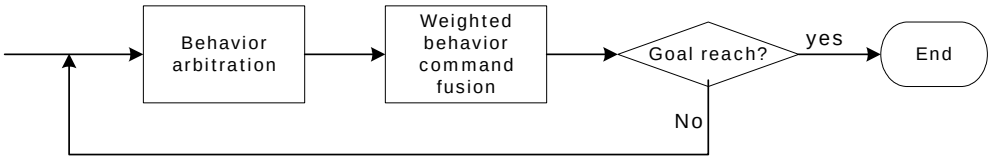
virtual subgoal, and behavior arbitration. Most of related methods belong to the boundary-following approach. Figure 7.17 (a)(b)(c) show the flowcharts of three different approaches. Table 7.6 compares how these methods address the local minimum problem in unknown indoor environments.



(a)



(b)



(c)

Figure 7.17: The flowchart of the approaches for local minimum problem. (a) boundary following. (b) virtual subgoal. (c) behavior arbitration.

Table 7.6: Comparison of related methods that address local minimum problem.

	Methods	Detection and escape criterion, and comments	
Boundary following approach	Huang and Lee	Detection criterion:	When a large difference in rotation of the robot between successive control periods is detected.
		Escape criterion:	When the detection point a, the escape point b, and the goal c are collinear and b is between a and c.
		Comments:	Because of empiric detection it is easy to produce wrong classification of the local minimum. The conservative escape criterion creates a long path.
	Distbug	Detection criterion:	When an obstacle is encountered.
		Escape criterion:	When the goal is visible, or the nearest obstacle toward the goal is closer to the goal than the current obstacle followed.
		Comments:	Escape criterion is dependent on maximal sensor range.
	Virtual-target-side	Detection criterion:	When a large differential change of the goal angle is detected, the obstacle boundary is followed with a virtual goal side.
		Escape criterion:	When the current goal distance is below the minimum distance attained prior to the trap detection, the real target side is used again for navigation.
		Comments:	A better strategy derived from a virtual target side, but still a long path.
	Maaref and Barret	Detection criterion:	When all sensors detect the small obstacle distances.
		Escape criterion:	When the three sensors measure big distances and the goal is not at the side of the obstacle followed by the robot.
		Comments:	Fails to detect most local minima.
	Krishna and Kalra	Detection criterion:	When the robot recognizes the landmarks experienced by previous navigation in a similar environment at the same position.
		Escape criterion:	When the robot reaches a location outside the bounding rectangle where the goal and obstacle are on the same side of the robot.
		Comments:	Detects using spatial and temporal reasoning. Depends on landmark recognition and exact coordinate localization.
Virtual subgoal approach	Virtual target	Detection criterion:	When an abrupt change in robot's turning tendency occurs due to a change in goal orientation.
		Escape criterion:	When an opening in the obstacle is detected.
		Comments:	Regresses into the same infinite loop it tries to avoid, and is unsuitable for recursive U-shape environments.
	Virtual obstacle	Detection criterion:	When the robot twice visits the same location with the same orientation.
		Escape criterion:	When the subgoal created is reached.
		Comments:	Very large memory requirements. Long corridor may create many virtual obstacles that lead to the longest path. Has difficulties detecting the local minimum.
Behavior coordination approach	Our Minimum Risk method	Detection criterion:	When the obstacle or trajectory dot intensity is not small, the weight of path-searching behavior becomes higher.
		Escape criterion:	When both the obstacle and trajectory dot intensities are small, the weight of path-searching behavior is low.
		Comments:	Multiple weighted behaviors coordination, and global convergence guaranteed in all local minimum situations. Able to find the nearest exit to escape from the local minimum.

Now we describe the general problems of the existing related methods. The differences among the boundary-following methods [Huang & Lee, 1992; Kamon & Rivlin, 1997; Lim & Cho, 1998; Krishna & Kalra, 2001; Maaref & Barret, 2002; Chatterjee & Matsuno, 2001] are that they have different detection and escape criteria. There is not any method that can be proved to obtain a shorter path. More importantly, they have no way to choose the nearest exit, and they possibly choose a wrong boundary-following direction leading to a rather inefficient path. Virtual-subgoal methods [Pin & Bender, 1999; Xu, 2000; Xu & Tso, 1999] encounter difficulties in dealing with unstructured or cluttered environments. Moreover, when used in recursive U-shape or more complex environments they may overproduce virtual subgoals, leading to a dead cycle arising from conflict subgoals. In addition, the problem must be taken into account whether or not the subgoal is located in an unreachable place. The above methods adopt an analytical model for detection and escape criteria. These, however, are not suitable for dealing with the uncertainties produced by sensors and the real world, and especially by the odometry drift problem.

7.8 Summary

This chapter proposes a new behavior-based navigation method called “minimum risk method”. The method is an application of the memory grid map, which addresses the local minimum problem for goal-oriented robot navigation in unknown indoor environments. This method is experimentally demonstrated to give global convergence to a given goal location even in long-wall, large concave, recursive U-shape, unstructured, cluttered, maze-like, and dynamic indoor environments. One of the future works is to improve and formulate the method as well as to theoretically prove global convergence.

The proposed minimum risk method for the goal-oriented navigation is not suitable for a long-distance navigation at this stage because the accumulated odometric errors have not been corrected. The method is currently particularly suitable for the applications of short path navigation between waypoints in complex environments with local minima. The Internet-based teleoperation falls into this kind of application where the human operator can give a number of subgoals to enable the remote robot exploring unknown environments. The minimum risk method has not been tested in outdoor environments because of the serious odometric errors as well

as a short wireless communication distance between the robot and the control computer. However, the ideas involved in the memory grid map and navigational algorithm can be applied in outdoor environments as well.

CHAPTER 8. EVALUATIONS AND RESEARCH IMPACT

8.1 Evaluations

We have performed the Internet-based teleoperation for robot navigation by inexperienced users remotely from places overseas (e.g. Canada, Singapore, Chinese Beijing, Shanghai, Xiamen) to Hong Kong. The authorized remote human operators (e.g. located in Canada) connect with the robot server (located at our department in Hong Kong) through the VNC service, and observe the robot's surroundings (our department corridor) through streaming video. The authorized users commonly have no robotic expertise and they learn the joystick and linguistic commands only at the beginning of the teleoperation. By using the telecommanding, the remote users are able to control the real robot to explore areas of interest, and also able to observe details via the camera movement.

The first public show of this Internet telerobotic system was on March 19th to 20th, 2004 during our departmental Demo Day (see Figure 8.1 (a). It was then publicly demonstrated at the International ICT Expo (see Figure 8.1 (b)), which was held at the Hong Kong Convention and Exhibition Center on April 14-17th, 2004. The latest public services were done in our campus during the university's Info Day on 9th October 2004 (see Figure 8.1(c)), and on 8th October 2005 (see Figure 8.1(d)), respectively.

All the remote user operations and public demonstrations have adopted a same teleoperation platform mentioned in Section 5.3, in which the VNC service is used as the interface between human operator and the robot server, the streaming video is transferred to help human operator obtain remote robot's surroundings, and the proposed telecommanding provides both joystick and linguistic commands to human operator in order to control the remote robot. The difference among the different evaluation scenarios is the number of linguistic commands we had completed. In the early period of open evaluations, we have completed four joystick commands and four linguistic commands (i.e. MOVE, COORDINATE, TURN, GOTOEND). During

the next period, we progressively add WANDER, MAPPING, and enhanced COORDINATE linguistic commands.

The remote user operations and public demonstrations show that, our Internet telerobotic system is practical and is feasible to provide the service of Internet-based teleoperation for robot navigation in order to interact with people and to explore unknown environments.

(a)

(b)

(c)

(d)

Figure 8.1: Public demonstration of Internet-based robot teleoperation. (a) Demo Day in our department, March 19th -20th, 2004; (b) International ICT Expo at the Hong Kong Convention and Exhibition Center, April 14-17th, 2004; (c) Info Day in our university campus, Oct. 9th, 2004; (d) Info Day in our university campus, Oct. 8th, 2005

In the following, we draw our lessons and limitations of the developed Internet telerobotic system in this thesis compared with other existing systems in literature.

1) **Interactivity**. Most of existing Internet robots have considerable autonomy but lack the interaction with human operator. For example, the operator is only able to

send very high-level commands to the robot without intermediate feedback. However, the interactivity is an important factor to attract Internet users' interests. Our system can provide more interaction between human operator and online robot through the telecommanding. For instance, the use of joystick commands particularly gives human operator a strong experience of hands-on control. In addition, human operator is able to continuously send linguistic commands with flexible working parameters to influence the robot's execution process, and online robot can respond and feedback predefined expected events to human operator as well as react to unexpected events. The limitation of the proposed telecommanding is that joystick commands are not suitable to handle more skilful tasks and linguistic commands need quite complicated design of a linguistic command function.

2) **Video transmission**. Other Internet telerobotic systems often adopt the techniques of picture transmission or video conference system to transfer the images about online robot's surroundings. Our system has adopted the latest streaming video technology that provides better quality of service (QoS) and extensibility. The limitation of streaming video is that the codec buffer leads to a long time delay (over 10s). Moreover, our streaming video transmission is developed based on Windows operational system. The client must have installed the Windows Media Player to receive the streaming video. These prevent the developed telerobotic system being remotely controlled by mobile devices (e.g. mobile phones or PDA).

3) **Usability**. There are two factors that mainly influence the usability of our telerobotic system: wireless connection, and battery recharging. Other existing Internet mobile robots have encountered the same problem. The robot server is a computer which directly controls the mobile robot and provides the Internet connection. The distance of wireless connection between the robot and the robot server is too short. For example, the distance that can provide a good quality communication is less than 50 metres in our robotic system. In addition, the batteries of the robot are only able to support the robot moving continuously for a limited few hours. The two factors highly influence the mobility of a mobile robot and the continuous teleoperation service to the public. In the last public demonstration during Info Day (see Figure 8.1(d)), we tried to place the robot server computer onboard the robot via the cable connection. This configuration of the telerobotic system is still

restricted by the wireless connection to the Internet, but it is useful for the mobile robot to perform some autonomous tasks.

4) **Data transmission**. Most Internet telerobotic systems have developed a private Web-based client interface for command and status data transmission. At this stage, we do not spend much time on the development of Web-based data transmission for workload simplification. We make use of an existing tool, i.e. the VNC service, for the Web users to send control commands and receive the information transferred from the robot server. The VNC service is indeed a convenient way for Web users to connect with the robot server, but it is inefficient because it consumes extra bandwidth for unnecessary data transmission.

5) **Time delay**. All Internet telerobotic systems have encountered the time delay problem caused by the Internet. Although it has been addressed in literatures, it is still the most difficult problem that influences the practical use of an Internet telerobotic system. Our research allows that a long and uncertain time delay exists. The solution is that the mobile robot is equipped local intelligence to handle expected events while to react to unexpected events, or to perform some tasks autonomously.

6) **Application environment**. Most Internet telerobotic systems need to know environmental knowledge in advance for path planning or localization. Our system is realized to fully address the Internet-based teleoperation of a remote robot that explores unknown and dynamic environments.

7) S**ensors.** Most online robots are equipped with many sonar sensors at 360 degree angles, even more advanced sensors such as laser, compass and so on, in order to detect the environment more accurately. Our robot is only equipped with eight forward sonars, which weakens the robot's capability of detecting obstacles, such as smooth walls, chairs, human feet.

Furthermore, we discuss the scalability of the proposed approaches in this thesis to larger interactions, more complex tasks, and multirobots.

1) **Scalability to larger interaction**. The proposed telecommanding has provided a multimodal and multifunctional framework, enabling human operator to more actively participate in remote robot's task completion and environmental exploration. The predefined events and response functions enable the robot to respond expected events and feedback information to human operator. For scalability to a larger interaction, the key issues are how to realize the proposed response functions and how

to send both joystick and linguistic commands. For example, to design a response function, we let the robot feedback force or haptic information according to distance to obstacles. Or human operator uses a real joystick device to send joystick commands, or uses human language or voices to send linguistic commands.

2) **Scalability to more complex tasks**. The proposed telecommanding is able to take advantage of human's intelligence through multiple joystick or linguistic commands and their working parameters, in order to help remote robot complete more complex tasks. In addition, the behavior-based navigation framework proposed in Chapter 4 provides good scalability, which can make navigational logic easily extensible. Fuzzy logic makes it easy to realize the desired behavior characteristics by explicitly expressing linguistic rules using a common natural language. The work in Chapter 7 to address the local minimum problem is an example for our approach scaling to more complex tasks.

3) **Scalability to multirobots**. The proposed navigation method in this thesis is only suitable for a single robot. We do not consider the key issues of multirobots application: cooperation and communication. For scalability of the proposed telecommanding, the key point is how to design a command function associated with a linguistic command in order to decompose a task and let multirobots cooperate to complete.

8.2 Research impact

The developed Internet telerobotic system (its name is PolyUiBot) was demonstrated to the public four times. Our robot received very positive responses from audiences, and especially it was reported by two magazines and one newspaper in Hong Kong during the period of International ICT Expo (see Appendix B). We have obtained a certain degree of research impact.

In order to enhance our research impact while to observe that whether or not potential students or related researchers could be attracted by our research, we have built a website to introduce the developed system since March 2004. We shared some contents about our research on the website, in which some experiments were recorded as video movies (see Appendix C). The visitors are able to freely download or online playback these video movies. The details can be accessed on the website: http://www4.comp.polyu.edu.hk/~csnkliu/polyuibot . In order to make statistics for

website visitors, the webpage records the visitor's IP address and its date. In this section, we take statistics based on the visitor's information from April 2004 to October 2005. There were 805 visitors in total.

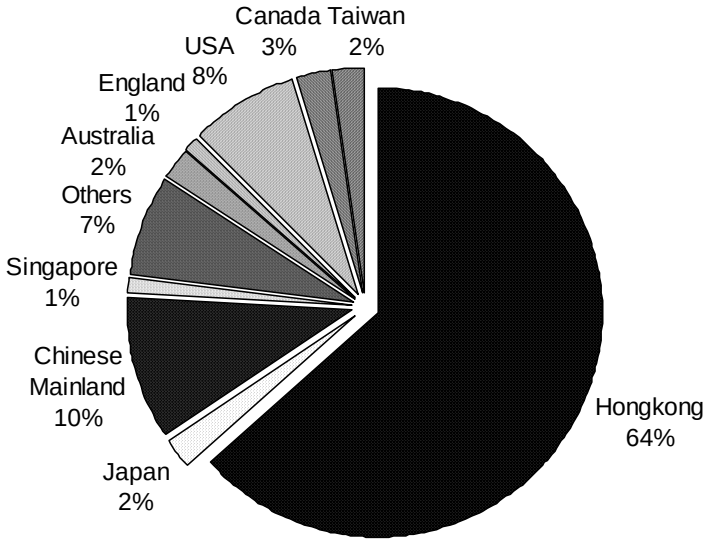


Figure 8.2: Statistics of countries or regions where the visitors come from.

First, we take statistics of countries or regions where the visitors came from, which we obtained by localizing their IP address. The result is shown in Figure 8.2. The visitors from Hong Kong win the most visits (64% of the total 805 visits), in which most visitors came from the author's university, The Hong Kong Polytechnic University (PolyU), and some visitors came from other universities in Hong Kong. We believe that some visitors from PolyU were postgraduate and undergraduate students, and that the others were the academic researchers worldwide among related fields (e.g. artificial intelligence or robotics). This is the reason that we distribute the website address to the public mainly through two ways: (1) teaching materials for students; (2) publications and presentation for international conferences. The Chinese mainland and the USA are the countries that have relatively most visits, 10% and 8% respectively. The other visitors came from the following countries or regions: Canada, Japan, Australia, Taiwan, Singapore, England, Indonesia, Vietnam, South Korea, Germany, Brazil, Philippines, Macao, Norway, French, Malaysia, Thailand, Ukraine, Russia, Mexico, and so on. Some of the visitors visited our website through the recommendation of their friends or colleagues. We know that because some persons have sent emails to us for enquiring research methods. The statistics result shows that

a number of related researchers worldwide are interested in our research. In addition, the research has attracted a number of potential research students.

Next we take statistics of the visit quantity for every month, from April 2004 to October 2005. The result is shown in Figure 8.3. It shows an average of 43 visits every month. There were a quite large amount of visitors during the first three months (i.e. April, May, June) in 2004. This was the reason that the website is initially built and we distribute the website address to the public during Demo Day and International ICT Expo. The website attracted the visitors, including the staffs and research students from PolyU. There were relatively small amount of visits in July and August. This could be due to a long academic holiday. During October 2004 to February 2005, we distributed the teaching materials and published some conference papers that possibly led to the increase of visit quantity in these months. The visit quantity peaked in April 2005 soon after we had updated the website contents.

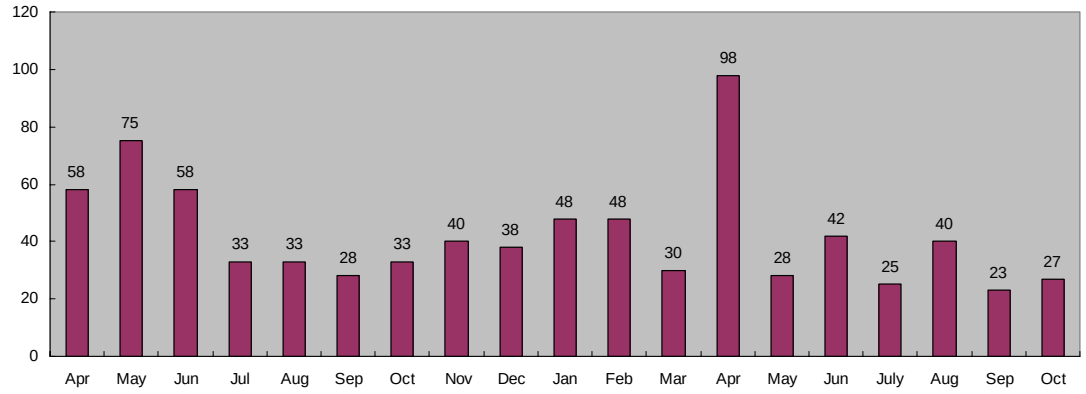


Figure 8.3: Statistics of the visit quantity for every month.

CHAPTER 9. CONCLUSIONS AND FUTURE WORK

9.1 Conclusions

In the thesis we have developed a telerobotic system that supports Internet-based teleoperation for robot navigation. Any inexperienced users are able to remotely control a mobile robot through the Internet in order to explore unknown and dynamic (i.e. with moving humans) environments.

The video transmission over the Internet has been investigated and implemented. It is a prerequisite to develop a practical teleoperation system, which allows that the Internet users can see the remote robot's surroundings through the images captured from an onboard camera. Traditional approach is via the picture transmission (e.g. JPEG or GIF), which leads to a very poor quality of service (QoS) because of the high latency of the Internet, such as long time delay, data error or restricted bandwidth. The thesis investigates and develops an existing streaming technology based approach for streaming video transmission. The streaming video improves the QoS by producing a more stable system, higher image resolution, and smoother image streams, even though it is used over a low-bandwidth Internet (e.g. 33.6Kbps dial-up modem). Moreover, it has better extensibility to integrate more multimedia information, and it allows that any client users can watch the continuous image streams simultaneously without reducing the QoS or increasing network bandwidth. But the time delay for streaming video is still large (over 10 seconds) since both the encoder buffer and the decoder buffer are used to guarantee the QoS. The time delay makes it necessary to equip a mobile robot local intelligence to perform some tasks autonomously.

Thus a framework for autonomous robot navigation using fuzzy logic has been proposed, which includes goal determination, preprocessing, behavior design, behavior arbitration, and command fusion. The traditional framework for autonomous navigation is SMPA (Sense-Model-Plan-Act) approach, which is inadequate for dealing with unknown and dynamic real world. The behavior-based approach can act in real-time and has good robustness in such environments. The behaviour-based navigation is not a fresh idea or concept. The thesis focuses on the development of a simple and practical navigation framework that can be easily realized to build robust

control programs. The preprocessing module is used to reduce the complexity of input space by introducing a limited number of intermediate variables. The elementary behavior is designed using fuzzy logic controller or a precise analytic algorithm. A behavior arbitration module is used to calculate the crisp weighting factors of each elementary behavior. The final robot motion output is obtained by the command fusion for a weighting combination of all elementary behaviors. Fuzzy logic is indeed a good tool, which allows that we can easily realize the desired behavior characteristics by explicitly expressing the linguistic rules using a common natural language.

A new teleoperation approach is proposed to provide an interactive control interface and a complete framework for control management and command processing. The traditional direct control reduces the stability of control loop because the controlled robot has no local intelligence and it needs to maintain continuous connection. The existing supervisory control methods are inadequate mainly in that they fail to provide human-robot interactivity. The proposed approach, namely telecommanding, involves two different but complementary commands: joystick command (e.g. LEFT, RIGHT, UP, and DOWN) and linguistic command (e.g. MOVE, TURN, GOTOEND, WANDER, COORDINATE, and MAPPING). Each command is designed to perform an independent task, which is defined with multiple events (non-time action references) and the corresponding response functions. The approach allows the robot to deliberately respond to expected events while to reactively respond to unexpected events. Thus the reliability for teleoperation is improved by equipping local intelligence of the robot even though the user's commands are lost or mistaken due to the unreliable Internet. Telecommanding provides human operators hands-on control, giving them a strong experience of interaction with the robot. Any inexperienced users can easily use the joystick commands or linguistic commands to remotely control a mobile robot.

A map learning approach, namely memory grid mapping, has been proposed for the mobile robot to model a priori unknown environment autonomously. The robot builds a map based on robot's sensory information and actively explores the unknown environment. The approach includes a map model, a map update method, an exploration method, and a map postprocessing method. The map adopts a grid-based representation. A so-called obstacle memory dot (OMD) matrix is designed to record

the frequency values which measure the confidence that a cell is occupied by an obstacle. A so-called trajectory memory dot (TMD) matrix is designed to record the trajectory traversed by the robot in order to facilitate the online path planning. Two behaviors, path-exploring behavior and environment-detecting behavior, are coordinated to make the robot exploring a least known environment. The increase of the learned map scale or environmental complexity has little influence on the computational time of our path planning method (i.e. exploration). This is the reason that the robot makes the path plan based on a small range of map information and sensory data. Thus our approach is a candidate for real-time implementation on mobile robots. It is verified that the 100mm×100mm cell size is a good compromise between map accuracy and space requirement of map storage. In addition, the proposed map postprocessing method, including a threshold operation, a template operation, and an insert operation, is able to improve the map representation accuracy from the original error index e = 17.4% to e = 2.6%.

For a teleoperated mobile robot that is exploring unknown indoor environments, it is desired that the robot is able to autonomously arrive at a given goal location, even though the environments involve all kinds of complex situations with local minima. The thesis has proposed a new navigation method, namely minimum risk method, to realize such function. The method makes use of the proposed memory grid map. When a mobile robot is performing the goal-oriented navigation, it updates a memory grid map in real-time. A novel path-searching behavior is developed to use the map information and to recommend a safest regional direction that can enable the robot to detect potential local minima and escape from them. The method is experimentally demonstrated to give global convergence to a given goal location, even though it is used in the long-wall, large concave, recursive U-shape, unstructured, cluttered, maze-like, or dynamic (i.e. with moving human) environments. Compared with the existing boundary-following or virtual-subgoal approach, the proposed method can deal with more complex environments and is able to find the nearest exit to escape from local minimum. The method is particularly suitable for the applications of short path navigation between waypoints in complex environments with possible local minima. The Internet-based teleoperation falls into this kind of application where the human operator can give a number of subgoals to enable the remote robot exploring unknown environments.

The developed Internet telerobotic system has been demonstrated to the public successfully, while it has been used by remote inexperienced users overseas (e.g. Canada, Singapore, Chinese Beijing, Shanghai, Xiamen). It turns out to be practical and be feasible to provide the service of Internet-based teleoperation at university campus or exhibition center.

9.2 Future work

It is impossible for a thesis to cover many issues about Internet telerobotics. We have implemented a primary prototype system for Internet-based robot teleoperation. Further research is required. Some of the possible problems and direction of solutions are given in the following.

1) Develop a localization technique that is suitable for telerobotic purpose.

The self-localization technique using dead-reckoning data from odometry is inadequate, which would lead to serious odometric errors in a large space area. To engineer the environment where the robot works is one possible means. For example, all the walls are orthogonal and the environments have not any unstructured objects. Such environments make it possible to permit local map matching and efficient correction of the robot's position estimate. But this means is not suitable for the practical use.

Another way is to adopt perception-based localization techniques, in which firstly the sensors detect an artificial or natural landmark in the environment and estimate the relative position of this landmark with respect to the robot. Then a robot's location is estimated by matching the detected characteristics of landmarks with those stored in a model of the environment. The artificial landmark (e.g. specific objects or colors) detection methods are well developed and have proved to be reliable, but natural landmark detection methods are not yet sufficiently developed [Meyer & Filliat, 2003].

Integrated localization techniques [Meyer & Filliat, 2003] are possible solution and make the telerobotic system usable in real world. They are absolute positioning methods which require external absolute references (e.g. artificial beacons, GPS) to estimate robot's position and orientation.

2) Develop techniques for scalability to more interaction between human operator and teleoperated robot.

One idea is to develop techniques to realize the predefined response functions associated with both joystick and linguistic commands. For example, to design a response function, we let the robot feedback force or haptic information according to distance to obstacles. A good example is seen in literature [Lo and Liu et al, 2004]. They developed a system that enables multiple operators at different sites to cooperatively control multiple robots with real-time force reflecting via the Internet.

Another idea is to develop techniques to enable that human operator uses a real joystick device to send joystick commands, or uses human language or voices to send linguistic commands.

3) Develop the image-based or vision-based robot navigation approach, which makes use of the images captured from onboard camera.

At least four research directions can be done.

a. Using the images for goal recognition and identification, which enhance the robot capability for goal seeking behavior;

b. Using the images for detecting and avoiding obstacles, which complement the inaccurate sonar sensors;

c. Using the images to identify the artificial or natural landmarks in order to localize the robot's position;

d. Using the images to identify and track the human body in order to interact more with people.

REFERENCES

[1] Araujo R., Almeida AT. (1999), "Learning sensor-based navigation of a real mobile robot in unknown worlds", *IEEE Trans. On SMC*, Part B, Vol.29, No.2, pp.164-178

[2] Arkin R.C. (1998), *Behavior-Based Robotics*. MIT Press, Cambridge, MA

[3] Arleo A., Millan J.D.R., Floreano D. (1999), "Efficient learning of variable-resolution cognitive maps for autonomous indoor navigation", *IEEE Trans. on Robotics and Automation*, Vol.15, No.6, pp.990-1000

[4] Backes P.G., Tso K.S., Norris J.S., Tharp G.K. (2002), "Internet-based Ground Operations for Mars Lander and Rover Missions", In K.Goldberg and R.Siegwart, eds, *An Introduction to Online Robots*, The MIT Press, Cambridge, Massachusetts, London, England, pp.227-240

[5] Barbera H.M., Izquierdo M.A.Z., Skarmeta A.F.G. (2001), "Web-based supervisory control of mobile robots", *Proceeding 10th IEEE International Workshop on Robot and Human* Interactive Communication, pp.256-261

[6] Baruch J.E.F., Cox M.J. (1996) "Remote control and robots: an Internet solution", IEE Computing Contr. Eng.J., pp.39-44

[7] Beom H.R., Cho H.S. (1995), "A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning", *IEEE Transactions on Systems, Man and Cybernetics*,Vol.25, Iss.3, pp.464-477

[8] Beom H.R., Cho H.S. (2000), "Sonar-based navigation experiments on a mobile robot in indoor environments", *Proceedings of the 2000 IEEE International Symposium on Intelligent Control*, pp.395-401

[9] Borenstein J., Koren Y. (1991), "Histogramic in-motino mapping for mobile robot obstacle avoidance", *IEEE Trans. on Robotics and Automation*, Vol.7, No.4, pp.535-539

[10] Bourhis G., Agostini Y. (1998), "Man-machine cooperation for the control of an intelligent powered wheelchair", Journal of Intelligent and Robotic Systems, Vol.22, pp.269-287

[11] Brady K., Tarn T.J. (2001), "Internet-Based Teleoperation", *Proc. Of the 2001 IEEE International Conference on Robotics & Automation*, Korea, vol. 1, pp.644-649

[12] Brady K., Tarn T.J. (2002), "Handling latency in Internet-based teleoperation", In K.Goldberg and R.Siegwart, eds, *An Introduction to Online Robots*, The MIT Press, Cambridge, Massachusetts, London, England, pp.171-192

[13] Braunl T., Tay N. (2001), "Combining configuration space and occupancy grid for robot navigation", International Journal of Industrial Robot, Vol.28, No.3, pp.233-241

[14] Brooks R. A. (1986), "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation.*, vol. RA-2, no. 1, pp. 14–23

[15] Burgard W., Schulz D. (2002), "Robust Visualization for Online Control of Mobile Robots", In K.Goldberg and R.Siegwart, eds, *An Introduction to Online Robots*, The MIT Press, Cambridge, Massachusetts, London, England, pp.241-258

[16] Cang Ye, Danwei Wang (2000), "A novel behavior fusion method for the navigation of mobile robots", *IEEE International Conference on Systems, Man, and Cybernetics*, Vol.5, pp.3526-3531

[17] Cang Ye, Yung N.H.C., Danwei Wang (2003), "A fuzzy controller with supervised learning assisted reinforcement learning algorithm for obstacle avoidance", IEEE Transactions on Systems, Man and Cybernetics, Part B, Vol.33, Iss.1, pp.17- 27

[18] Chatterjee R., Matsuno F. (2001), "Use of single side reflex for autonomous navigation of mobile robots in unknown environments", Robotics and Autonomous Systems, Vol.35, pp.77-96

[19] Chen L.H.; Chiang C.H; John Yuan, 2001] "New approach to adaptive control architecture based on fuzzy neural network and genetic algorithm", *2001 IEEE International Conference on Systems, Man, and Cybernetics*, Vol.1, pp.347-352

[20] Cheng G., Zelinsky A. (2001), "Supervised autonomy: a framework for human-robot systems development", Autonomous Robots, Vol.10, pp.251-266

[21] Choi B., Kuc T.Y., Choi H. (1997), "Adaptive learning of teleoperating robotic motion", in Proc. IEEE Int. Conf. Systems, Man, and Cybernetics, vol.3, pp.2752-2757

[22] Chong K.S., Kleeman L.(1999), "Feature-based mapping in real, large scale environments using an ultrasonic array", International Journal Robotics Research, Vol 18, No.1, pp.3-19

[23] Chung J. H., Ahuja N. (1998), "An analytical tractable potential field model of free space and its application in obstacle avoidance," *IEEE Trans. Syst., Man, Cybern. B*, Vol. 28, pp.729–736

[24] Chung J., Ryu B.S., Yang H.S. (1998), "Integrated control architecture based on behavior and plan for mobile robot navigation", Robotica, Vol.16, pp.387-399

[25] Cohn D.A. (1996), "Neural network exploration using optimal experiment design", Neural Network, Vol.9, pp.1071-1083

[26] Dalton B. (2002) "A distributed framework for online robots", In K.Goldberg and R.Siegwart, eds, *An Introduction to Online Robots*, The MIT Press, Cambridge, Massachusetts, London, England, pp.37-59

[27] Dieguez A.R., Sanz R., Lopez J. (2003), "Deliberative on-line local path planning for autonomous mobile robots", Journal of Intelligent and Robotic Systems, Vol.37, pp.1-19

[28] Dongbing Gu; Huosheng Hu; Spacek, L. (2003), "Learning fuzzy logic controller for reactive robot behaviours", 2003 IEEE/ASME Conference on Advanced Intelligent Mechatronics, Vol.1, pp. 46- 51

[29] Dongbing Gu; Huosheng Hu; Reynolds, J.; Tsang, E. (2003), "GA-based learning in behaviour based robotics", Proceedings. 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation, Vol.3, pp.1521- 1526

[30] Dudek G., Jenkin M. (2000), *Computational Principles of Mobile Robotics*, Cambridge University Press, United Kingdom

[31] Edson, Idiart M.A.P., Trevisan M., Engel P. (2004), "Autonomous Learning Architecture for Environmental Mapping", Journal of Intelligent and Robotics System, Vol.39, pp.243-263

[32] Elfes A. (1987), "Sonar-based real-world mapping and navigation", IEEE Trans. on Robotics and Automation, Vol.3, pp.249-265

[33] Elhajj I., Ning Xi, et al. (2003), "Supermedia-enhanced Internet-based telerobotics", *Proceedings of the IEEE*, Vol.91, Iss.3, pp.396- 421

[34] Fabrizi E., Oriolo G., Ulivi G. (2000), "Accurate Map Building via Fusion of and Ultrasonic Range Measures". In D. Driankov and A. Saffiotti, eds, *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*, Physica- Verlag, Heidelberg, New York, 2000, pp.257-280

[35] Fernandez J.L., Sanz R., Benayas J.A., Dieguez A.R. (2004), "Improving collision avoidance for mobile robots in partially known environments: the beam curvature method", *Robotics and Autonomous Systems*, Vol.46, pp.205-219

[36] Filliat D., Meyer J.A. (2003), "Map-based navigation in mobile robots: I. A review of localization strategies", *Cognitive Systems Research*, Vol.4, pp.243-282

[37] Freirer E., Teodiano B.F., et al. (2004), "A new mobile robot control approach via fusion of control signals", *IEEE Trans. on SMC*, Part B, Vol.34, No.1, pp.419-429

[38] Fiorini P., Oboe R. (1997), "Internet-based telerobotics: problems and approaches", 8th International *Conference on Advanced Robotics*, pp.765-770

[39] Fong T., Thorpe C., Baur C. (2003), "Robot, asker of questions", Robotics and Autonomous Systems, Vol.42, Iss.3-4, pp.235-243

[40] Fung W.K., Ning Xi, Lo W.T., Liu Y.H. (2002), "Improving efficiency of Internet based teleoperation using network QoS", *IEEE International Conference on Robotics and Automation (ICRA '02)*, Vol.3, pp.2707-2712

[41] Fusiello A., Caprile B. (1997), "Synthesis of indoor maps in presence of uncertainty", Robotics and Autonomous Systems, Vol.22, pp.103-114

[42] Gasos J., Rosetti A. (1999), "Uncertainty representation for mobile robots: perception, modeling and navigation in unknown environments", Fuzzy Sets and Systems, Vol.107, pp.1-24

[43] Gasos J. (2000), "Integrating linguistic descriptions and sensor observations for the navigation of autonomous robots". In D. Driankov and A. Saffiotti, eds, *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*, Physica-Verlag, Heidelberg, New York, pp.313-340

[44] Gat E. (1998), "Three-layer architectures", In R.P. Bonasso D. Kortenkamp and R. Murphy, editors, *Artificial intelligence and mobile robots*, MIT Press, Cambridge, MA, pp.195-210

[45] Godjevac J., Steele N. (2000), "Neuro fuzzy control for basic mobile robot behaviours". In D. Driankov and A. Saffiotti, eds, *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*, Physica-Verlag, Heidelberg, New York, pp.98-118

[46] Goertz R., Thompson R. (1954), "Electronically controlled manipulator", *Nucleonics*

[47] Goldberg K., Santarromana J., et al. (1995), "The Telegarden", in Proc. ACM SIGGRAPH, pp135

[48] Goldberg K., Chen B., Solomon R., et al. (2000), "Collaborative teleoperation via the Internet", *in Proc. IEEE Int. Conf. Robotics and Automation*, Vol.2, pp.2019-2024

[49] Goldberg K., Gentner S., Sutter C., Wiegley J. (2000), "The Mercury Project: a feasibility study for Internet robots", *IEEE Robotics & Automation Magazine*, 7(1), pp.35-40

[50] Goldberg K., Gentner S., Sutter C., Wiegley J., Farzin B. (2002), "The Mercury Project: A Feasibility Study for Online Robots", In K.Goldberg and R.Siegwart, eds, *An Introduction to Online Robots*, The MIT Press, Cambridge, Massachusetts, London, England, pp.17-36

[51] Goldberg K., Siegwart R. (2002), "Introduction", In K.Goldberg and R.Siegwart, eds, *An Introduction to Online Robots*, The MIT Press, Cambridge, Massachusetts, London, England, pp.XV-XXi

[52] Golfarelli M., Maio D., Rizzi S. (2001), "Correction of dead-reckoning errors in map building for mobile robots", *IEEE Trans. on Robotics and Automation*, Vol.17, No.1, pp.37-47

[53] Goodridge S. G., Kay M. G. (2000), "Multi-layered fuzzy behavior fusion for reactive control of autonomous robots". In D. Driankov and A. Saffiotti, eds, *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*, Physica- Verlag, Heidelberg, New York, pp.179-204

[54] Guangyu Lian; Jinshi Cui, et al. (2001), "Telemanipulation via Internet based on human-robot cooperation", International Conferences on Info-tech and Info-net, Beijing. Vol.4, pp.256-262

[55] Habert O., Pruski A. (1997), "Cooperative construction and maintenance of maps for autonomous navigation", Robotics and Autonomous Systems, Vol.21, pp.341-353

[56] Hagras H., Callaghan V., Colley M. (2000), "Online learning of fuzzy behaviour co-ordination for autonomous agents using genetic algorithms and real-time interaction with the environment", *The Ninth IEEE International Conference on Fuzzy Systems*, Vol.2, pp.853-858

[57] Hagras H., Callaghan V., Collry M. (2001), "Outdoor mobile robot learning and adaptation", IEEE Robotics & Automation Magazine, Vol.8, Iss.3, pp.53-69

[58] Halme A., Suomela J., Savela M. (1999), "Applying telepresence and augmented reality to teleoperate field robots", Robotics and Autonomous Systems, Vol. 26, pp.117-125

[59] Han K.H., Kim S., Kim Y.J., Kim J.H (2001), "Internet control architecture for Internet-based personal robot", Autonomous Robots, Vol.10, pp.135-147

[60] Hashimoto H., Ando N., Lee J.H. (2002), "The performance of mobile robots controlled through the Web", In K.Goldberg and R.Siegwart, eds, *An Introduction to Online Robots*, The MIT Press, Cambridge, Massachusetts, London, England, pp.137-154

[61] Hoffmann F. (2000), "The role of fuzzy logic control in evolutionary robotics". In D.Driankov and A. Saffiotti, eds, *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*, Physica-Verlag, Heidelberg, New York, pp.119-148

[62] Huang H.P., Lee P.C. (1992), "A real-time algorithm for obstacle avoidance of autonomous mobile robots", Robotica, Vol.10, pp.217-227

[63] Huh D. J., Park J.H., Huh U. Y., Kim H.I. (2002), "Path planning and navigation for autonomous mobile robot", IEEE 28th Annual Conference of the Industrial Electronics Society, Vol.2, pp.1538- 1542

[64] HuaNan Yu; JiaMin Zhao; YuRu Xu (2002), "Tuning of neuro-fuzzy controller by real-coded genetic algorithm with application to an autonomous underwater vehicle control system", *International Conference on Machine Learning and Cybernetics*, Vol.2, pp.735-738

[65] Huosheng Hu, Lixiang Yu, Pui Wo Tsui, Quan Zhou (2001), "Internet-based Robotic Systems for Teleoperation", *International Journal of Assembly Automation*, Vol. 21, No. 2, pp.1-10

[66] Ismail I.I., Nordin M.F. (2002), "Reactive navigation of autonomous guided vehicle using fuzzy logic", Student Conference on Research and Development, pp.153- 156

[67] Jinshi Cui, Sun Zengqi; Li Ping (2002), "Visual technologies in shared control mode of robot teleoperatioin system", *Proceedings of the 4th World Congress on Intelligent Control and Automation*, Vol.4, pp.3088- 3092

[68] Kaelbling L.P., Littman M.L. (1996), "Reinforcement Learning: A Survey", Journal of artificial intelligence research, iss.4, pp.237-285

[69] Kamon I., Rivlin E. (1997), "Sensory-based motion planning with global proofs", IEEE Transactions on Robotics and Automation, Vol.13, Iss.6, pp.814-822

[70] Kawanaka H., Yoshikawa T., Tsuruoka S. (2000), "Acquisition of fuzzy control rules for a mobile robot using genetic algorithm", Proceedings. 6th International Workshop on Advanced Motion Control, pp.507-512

[71] Kiendl H., Ruger J. J. (1995), "Stability analysis of fuzzy control systems using facets functions". *Fuzzy Sets and Systems*, Vol.70, pp.275-285

[72] Kikuchi J., Takeo K., Kosuge K. (1999), "Teleoperation system via computer network for dynamic environment", in Proc. IEEE Int. Conf. Robotics and Automation, Vol. 4, pp.3534-3539

[73] Kim J.W., Choi B.D., Park S.H., et al. (2002), "Remote control system using real-time MPEG-4 streaming technology for mobile robot", *International Conference on Consumer Electronics*, pp.200- 201

[74] Kortenkamp D., Bonasso R.P., Murphy R. (1998), Artificial Intelligence and Mobile Robots, AAAI Press/The MIT Press, California, Cambridge, London, England

[75] Krishna K.M., Kalra P.K. (2001), "Perception and remembrance of the environment during real-time navigation of a mobile robot", Robotics and Autonomous Systems, vol.37, pp.25-51

[76] Kuipers B. J. (2000). "The spatial semantic hierarchy". Artificial Intelligence, Vol.119,pp.191-233

[77] Lam S.K., Srikanthan T. (2001), "High-speed environment representation scheme for dynamic path planning", Journal of Intelligent and Robotic Systems, Vol.32, pp.307-319

[78] Lee T.L., Wu C.J. (2003), "Fuzzy motion planning of mobile robots in unknown environments", *Journal of Intelligent and Robotic Systems*, Vol.37, pp.177-191

[79] Lee Y.G.; Zak S.H. (2002), "Genetic fuzzy tracking controllers for autonomous ground vehicles", *Proceedings of the 2002 American Control Conference*, Vol.3, pp.2144- 2149

[80] Lim J.H., Cho D.W. (1998), "Sonar based systematic exploration method for an autonomous mobile robot operating in an unknown environment", Robotica, Vol.16, pp.659-667

[81] Lin I.S., Wallner F., Dillmann R. (1996), "Interactive control and environment modelling for a mobile robot based on multisensor perceptions", Robotics and Autonomous Systems, Vol.18, Iss.3, pp.301-310

[82] Liu P.X., Meng M., et al. (2002), "An UDP-based protocol for Internet robots", Proceedings of the 4th World Congress on Intelligent Control and Automation, Vol.1, pp.59- 65

[83] Lo W.T., Liu Y.H., Elhajj, I.H.; et al, (2004), " Cooperative teleoperation of a multirobot system with force reflection via Internet", IEEE/ASME Transactions on Mechatronics, Vol.9, Iss.4, pp.661-670

[84] Luger G.F., Stubblefield W. A. (2002), "Artificial Intelligence: Structures and Strategies for Complex Problem Solving", 4th Edition, Addison Wesley

[85] Luo R.C., Chern M.Y., Hwang K.S., et al. (1998), "Development of intelligent electrical wheelchair for hospital automation", *in Proc. IEEE/ASME Int. Conf. Mechatronics (ICMT'98)*, pp.417-422

[86] Luo R.C., Chen T.M. (2000), "Development of a multi-behavior based mobile robot for remote supervisory control through the Internet", *IEEE/ASME Transactions on Mechatronics*, Vol.5, Iss.4, pp.376-385

[87] Luo R.C., Su K.L., et al. (2003), "Networked intelligent robots through the Internet: issues and opportunities", *Proc. of the IEEE*, Vol.91, Iss.3, pp.371- 382

[88] Maaref H., Barret C. (2002), "Sensor-based navigation of a mobile robot in an indoor environment", Robotics and Autonomous Systems, Vol.38, pp.1-18

[89] Mack S. (2002), Streaming Media Bible, New York, Hungry Minds Inc

[90] Maeyama S., Yuta S., Harada A. (2001), "Remote viewing on the Web using multiple mobile robotic avatars", in Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems, Vol.2, pp.637-642

[91] Makatchev M., Tso S.K. (2000), "Human-robot interface using agents communicating in an XML-based markup language", *Proceedings. 9th IEEE International Workshop on Robot and Human Interactive Communication*, pp.270-275

[92] Malinowski A., Booth T., et al. (2001), "Real time control of a robotic manipulator via unreliable Internet connection", The 27th Annual Conference of the IEEE (IECON'01) on Industrial Electronics Society, Vol.1, pp.170-175

[93] Meyer J.A., Filliat D. (2003), "Map-based navigation in mobile robots: II. A review of map-learning and path-planning strategies", *Cognitive Systems Research*, Vol.4, pp.283-317

[94] Min B.K., Cho D.W., et al. (1997), "Sonar mapping of a mobile robot considering position uncertainty", Robotics & Computer-Integrated Manufacturing, Vol.13, pp.41-49

[95] Minguez J., Montano L. (2004), "Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios", IEEE Trans. on Robotics and Automation, Vol.20, pp.45-59

[96] Moravec H.P., Elfes A. (1985), "High resolution maps from wide angle sonar", IEEE Conference on Robotics and Automation, USA, pp.116-121

[97] Moravec H.P. (1988), "Sensor fusion in certainty grids for mobile robots", AI Magazine, Vol.9, pp.61-73

[98] Murphy R. R. (2000), "Fuzzy logic for fusion of tactical influences on vehicle speed control". In D. Driankov and A. Saffiotti, eds, *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*, Physica-Verlag, Heidelberg, New York, pp.73-98

[99] Mut V., Postigo J., Slawinski E., Kuchen B. (2002), "Bilateral teleoperation of mobile robots", Robotica, Vol.20, pp.213-221

[100] Na Y.K., Oh S.Y. (2003), "Hybrid control for autonomous mobile robot navigation using neural network based behavior modules and environment classification", *Autonomous Robots*, Vol.15, pp.193-206

[101] Nehmzow N. (2000), "Mobile robotics: a practical introduction", Springer-Verlag, London

[102] Niemeyer G., Slotine J.J.E. (2002), "Toward Bilateral Internet Teleoperation", In K.Goldberg and R.Siegwart, eds, *An Introduction to Online Robots*, The MIT Press, Cambridge, London, England, pp.193-213

[103] Ning Xi, Tarn T.J. (2000), "Stability analysis of non-time referenced Internet-based telerobotic systems", Robotics and Autonomous Systems, Vol.32, Iss.2-3, pp.173-178

[104] Nojima Y., Kojima F., Kubota N. (2003) "Local episode-based learning of multi-objective behavior coordination for a mobile robot in dynamic environments", The 12th IEEE International Conference on Fuzzy Systems, 2003. FUZZ '03, Vol.1, pp.307- 312

[105] Ollero A., Ferruz J., Sanchez O., Heredia G. (2000), "Mobile robot path tracking and visual target tracking using fuzzy logic". In D. Driankov and A. Saffiotti, eds, Fuzzy Logic Techniques for Autonomous Vehicle Navigation, Physica- Verlag, Heidelberg, New York, pp.51-72.

[106] Oh J.S., Choi Y.H., Park J.B., Zheng Y.F. (2004), "Complete coverage navigation of cleaning robots using triangular-cell-based map", *IEEE Trans. on Industrial Electronics*, Vol.51, No.3, pp.718-726

[107] Oriolo G., Ulivi G., Vendittelli M. (1997), "Fuzzy maps: a new tool for mobile robot perception and planning", Journal of Robotic Systems, Vol.14, Iss.3, pp.179-197

[108] Oriolo G., Ulivi G., Vendittelli M. (1998), "Real-time map building and navigation for autonomous robots in unknown environments", IEEE Trans. on Systems, Man and Cybernetics, Vol.28, No.3, pp.316-333

[109] Overholt J.L., Cheok K.C. (2001), "Hierarchical systems control using threshold fuzzy systems", 2001 IEEE International Conference on Systems, Man, and Cybernetics, Vol.4, pp.2257-2262

[110] Park J.M., Lee J.M. (2001), "Transmission modelling and simulation for Internet-based control", *IECON '01 on Industrial Electronics Society*, Vol.1, pp.165-169

[111] Passino K.M., Yurkovich S. (1998), "Fuzzy control", Addison-Wesley Longman, Inc., California, USA.

[112] Paulos E., Canny J. (2002), "Personal Tele-Embodiment", In K.Goldberg and R.Siegwart, eds, *An Introduction to Online Robots*, The MIT Press, Cambridge, Massachusetts, London, England, pp.155-167

[113] Pin F., Watanabe Y. (2000), "Resolving conflict between behaviors using suppression and inhibition". In D. Driankov and A. Saffiotti, eds, *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*, Physica-Verlag, Heidelberg, New York, pp.151-178

[114] Pin F.G., Bender S.R. (1999), "Adding memory processing behavior to the fuzzy behaviorist approach: Resolving limit cycle problems in mobile robot navigation", Intelligent Automation and Soft Computing, Vol.5, Iss.1, pp.31-41

[115] Ping Li, Wenjuan Lu (2002), "Implementation of an event-based Internet robot teleoperation system", *Proceedings of the 4th World Congress on Intelligent Control and Automation*,Vol.2, pp.1296- 1300

[116] Pirjanian P., Mataric M. (2000), "Multiple objective vs. fuzzy behavior coordination". In D. Driankov and A. Saffiotti, eds, *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*, Physica-Verlag, Heidelberg, New York, pp.235-254

[117] Poncela A., Perez E.J., Bandera A., et al. (2002), "Efficient integration of metric and topological maps for directed exploration of unknown environments", Robotics and Autonomous Systems, Vol.41, pp.21-39

[118] Pradalier C., Hermosillo J., Koike C., et al. (2005), "The CyCab: a car-like robot navigating autonomously and safely among pedestrians", Robotics and Autonomous Systems, Vol.50, pp.51-67

[119] Quoy M., Moga S., Gaussier P. (2003), "Dynamical neural networks for planning and low-level robot control", IEEE Transactions on Systems, Man and Cybernetics, Part A,, Vol.33, Iss.4, pp.523- 532

[120] Raschke, Borenstein (1990), "Comparison of grid-type map-building techniques by index of performance", International Conference on Robotics and Automation, pp.1828-1832

[121] Ruan J.H, Song R., Li Y.B (2002), “Design for intelligent motion controller of unmanned vehicle”, Proceedings of the 4th World Congress on Intelligent Control and Automation, Vol.2, pp.1643- 1646

[122] Rusu P., Petriu E.M., Whalen T.E., et al. (2003), “Behavior-based neuro-fuzzy controller for mobile robot navigation”, IEEE Transactions on Instrumentation and Measurement, Vol.52, Iss.4, pp.1335- 1340

[123] Rybski P.E., Stoeter S.A., et al. (2002), “Sharing control [multiple miniature robots]”, IEEE Robotics & Automation Magazine, Vol.9, Iss.4, pp.41-48

[124] Ryu B.S., Yang H.S. (1999), “Integration of reactive behaviors and enhanced topological map for robust mobile robot navigation”, *IEEE Trans. on SMC*, Part A, Vol.29, No.5, pp.474-485

[125] Saffiotti A., Ruspini E. H., Konolige K. (1995), "A Multivalued Logic Approach to Integrating Planning and Control," *Artificial Intelligence*, vol. 76, no. 1-2, pp 481-526

[126] Saffiotti A., Ruspini E. H., Konolige K. (1999), “Using fuzzy logic for mobile robot control”, in H.J.Zimmermann, Kluwer, eds, *Practical Applications of Fuzzy Technologies*, Academic Publishers, Norwell, Massachusetts, USA, pp.185-206

[127] Saffiotti A. (2000), “Fuzzy Logic in Autonomous Navigation”, In D.Driankov and A.Saffiotti, eds, *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*, Physica-Verlag, Heidelberg, New York, pp.3-22

[128] Salichs M.A., Moreno L. (2000), “Navigation of mobile robots: open questions”, *Robotica*, Vol.18, pp.227-234

[129] Saucy P., Mondada F. (2000), “KhepOnTheWeb: open access to a mobile robot on the Internet”, *IEEE Robotics & Automation Magazine*, Vol.7, Iss.1, pp.41-47

[130] Saucy P., Mondada F. (2002), “KhepOnTheWeb: one year of access to a mobile robot on the Internet”, In K.Goldberg and R.Siegwart, eds, *An Introduction to Online Robots*, The MIT Press, Cambridge, Massachusetts, London, England, pp.99-115

[131] Sayers C. P., Paul R. P., et al. (1998) “Teleprogramming for subsea teleoperation using acoustic communication”, *IEEE Journal of Oceanic Engineering*, Vol.23, Iss.1, pp.60-71

[132] Sayers C. P. (1999), Remote Control Robotics, Springer-Verlag, New York, Inc.

[133] Sayers C.P. (2002), “Fundamentals of Online Robots”, In K.Goldberg and R.Siegwart, eds, *An Introduction to Online Robots*, The MIT Press, Cambridge, Massachusetts, London, pp.3-16

[134] Schulz D., Burgard W., Cremers A.B. (1998), “Predictive simulation of autonomous robots for teleoperation system using the World Wide Web”, *in Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pp.31-36

[135] Schulz D., Burgard W., Cremers A.B. (1999), "Robust visualization of navigation experiments with mobile robots over the Internet", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol.2, pp.942-947

[136] Schulz D., Burgard W., Fox D., Thrun S., Cremers A.B. (2000), "Web interfaces for mobile robots in public places", *IEEE Robotics & Automation Magazine*, Vol.7, Iss.1, pp.48-56

[137] Seraji H., Howard A. (2002), "Behavior-based robot navigation on challenging terrain: A fuzzy logic approach", IEEE Transactions on Robotics and Automation, Vol.18, Iss.3, pp.308-321

[138] Sheridan T.B. (1992), "Telerobotics, automation, and human supervisory control", The MIT Press, London, England

[139] Sheridan T.B. (2002), Forword, In K.Goldberg and R.Siegwart, eds, *An Introduction to Online Robots*, The MIT Press, Cambridge, Massachusetts, London, England, pp.VIIII-Xi

[140] Siegwart R., Saucy P. (1999), "Interacting Mobile Robots on the Web", ICRA'99, Detroit, MI, USA

[141] Siegwart R., Goldberg K. (2000), "Robots on the web", *IEEE Robotics & Automation Magazine*, Vol.7, Iss.1, pp.4

[142] Siegwart R., Balmer P., Portal C., et al. (2002), "RobOnWeb: A Setup with Mobile Mini-Robots on the Web", In K.Goldberg and R.Siegwart, eds, *An Introduction to Online Robots*, The MIT Press, Cambridge, Massachusetts, London, England, pp.117-135

[143] Simmons R., Fernandez J.L., et al. (2000), "Lessons learned from Xavier", *IEEE Robotics & Automation Magazine*, Vol.7, Iss.2, pp.33-39

[144] Simmons R., Goodwin R., et al. (2002), "Xavier: An Autonomous Mobile Robot on the Web", In K.Goldberg and R.Siegwart, eds, An Introduction to Online Robots, The MIT Press, Cambridge, Massachusetts, London, England, pp.81-97

[145] Song K.T., Chen C.C. (1996), "Application of heuristic asymmetric mapping for mobile robot navigation using ultrasonic sensors", Journal of Intelligent and Robotic Systems, Vol.17, pp.243-264

[146] Song K.T., Chang C.C. (1999), "Navigation integration of a mobile robot in dynamic environments", Journal of Robotic Systems, Vol.16, Iss.7, pp.387-404

[147] Stein M.R. (2002), "One Year of Puma Painting", In K.Goldberg and R.Siegwart, eds, *An Introduction to Online Robots*, The MIT Press, Cambridge, Massachusetts, London, England, pp.277-293

[148] Stein M.R. (2003), "The PumaPaint Project", Autonomous Robots, Vol.15, pp.255-265

[149] Surmann H., Peters L. (2000), "MORIA - a robot with fuzzy controlled behaviour". In D. Driankov and A. Saffiotti, eds, *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*, Physica-Verlag, Heidelberg, New York, pp.343-366

[150] Tan K.C., Tan K.K., Lee T.H., et al. (2002), "Autonomous robot navigation based on fuzzy sensor fusion and reinforcement learning", *Proceedings of the 2002 IEEE International Symposium on Intelligent Control*, pp.182- 187

[151] Taylor K., Trevelyan J. (1995), "Australia's telerobot on the Web", in Proc. 26th Int. Symp. Industrial Robots, pp.39-44

[152] Thongchai S., Kawamura K. (2000), "Application of fuzzy control to a sonar-based obstacle avoidance mobile robot", Proceedings of the 2000 IEEE International Conference on Control Applications, pp.425-430

[153] Thongchai S., Suksakulchai S., Wilkes D.M., Sarkar, N. (2000), "Sonar behavior-based fuzzy control for a mobile robot", IEEE International Conference on Systems, Man, and Cybernetics, Vol.5, pp.3532-3537

[154] Thongchai S. (2002), "Behavior-based learning fuzzy rules for mobile robots", Proceedings of the 2002 American Control Conference, Vol.2, pp.995-1000

[155] Thrun S. (1992), "The role of exploration in learning control", in Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches, D.A.White and D.A.Sofge, Eds., New York

[156] Thrun S. (1998a), "A probabilistic approach to concurrent mapping and localization for mobile robots", Autonomous Robots, Vol.5, pp.253-271

[157] Thrun S. (1998b), "Learning metric-topological maps for indoor mobile robot navigation", Artificial Intelligence, Vol.99, pp.21-71

[158] Thrun S., Bennewitz M., et al. (1999), "MINERVA: a second-generation museum tour-guide robot", *IEEE International Conference on Robotics and Automation*, Vol.3, pp.1999-2005

[159] Thrun S. (2003), "Learning occupancy grid maps with forward sensor models", *Autonomous Robots*, Vol.15, pp.111-127

[160] Tomatis N., Nourbakhsh I., Siegwart R. (2003), "Hybrid simultaneous localization and map building: a natural integration of topological and metric", Robotics and Autonomous Systems, Vol.44, pp.3-14

[161] Tsourveloudis N.C., Valavanis K.P., Hebert T. (2001), "Autonomous vehicle navigation utilizing electrostatic potential fields and fuzzy logic", *IEEE Trans. on Robotics and Automation*, Vol.17, No.4, pp.490-497

[162] Tunstel E., Jamshidi M. (1994), "Fuzzy Logic and Behavior Control Strategy for Autonomous Mobile Robot Mapping," *in FUZZ-IEEE World Congress on Computional Intelligence*, Orlando, Florida, pp. 514-517

[163] Tunstel E. (2000), "Fuzzy-behavior synthesis, coordination, and evolution in an adaptive behavior hierarchy". In D. Driankov and A. Saffiotti, eds, *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*, Physica- Verlag, Heidelberg, New York, pp.205-234

[164] Victorino A.C., Rives P., Borrelly J.J. (2003), "Safe navigation for indoor mobile robots. Part II: exploration, self-localization, and map building", The International Journal of Robotics Research, Vol.22, pp.1019-1039

[165] Vieira W.J., Matos M.L.C, Castolo L.O. (2001), "Fitting autonomy and mobile agents", IEEE Conference on Emerging Technologies and Factory Automation, Portugal, Vol.2, pp.471-480

[166] Volpe R., Estlin T., et al. (2000), "Enhanced Mars rover navigation techniques", IEEE International Conference on Robotics and Automation, USA, vol.1, pp.926-931

[167] Wallner F., Dillmann R. (1994), "Efficient mapping of dynamic environment by use of sonar and active stereo-vision", International Symposium on Intelligent Robotics Systems, France, pp.1-13

[168] Wang X.C., Yang X. (2003), "A neuro-fuzzy approach to obstacle avoidance of a nonholonomic mobile robot", 2003 IEEE/ASME conference on advanced intelligent mechatronics (AIM 2003), pp.29-34

[169] Wang X.G., Moallem M., Patel R.V. (2003), "An Internet-based distributed multiple-telerobot system", IEEE Transactions on Systems, Man and Cybernetics, Part A, vol.3, iss.5, pp.627-634

[170] Watanabe H., Dettloff W., Yount E. (1990), "A VLSI fuzzy logic inference engine for real-time process control", IEEE Journal of Solid State Circuits, Vol.25, Iss.2, pp.376-382

[171] Xiang Guoliang; Yun-Hui Liu, et al. (2002), "An Internet based pulse palpation system for Chinese medicine", IEEE*/RSJ International Conference on Intelligent Robots and System*, Vol.2, pp.1481-1486

[172] Xu W.L. (2000), "A virtual target approach for resolving the limit cycle problem in navigation of a fuzzy behavior-based mobile robot", Robotics and Autonomous Systems, Vol.30, Iss.4, pp.315-324

[173] Xu W.L., Tso S.K. (1999), "Sensor-based fuzzy reactive navigation for a mobile robot through local target switching", IEEE Transactions on Systems, Man and Cybernetics, Vol.29, No.3, pp.451-459

[174] Yamada S. (2005), “Evolutionary behavior learning for action-based environment modeling by a mobile robot”, Applied Soft Computing, Vol.5, pp.245-257

[175] Yamauchi B., Schultz A., Adams W. (1998), “Mobile robot exploration and map-building with continuous localization”, IEEE Conference on Robotics and Automation, pp.3715-3720

[176] Yang S.X., Meng M.Q.H (2003), “Real-time collision-free motion planning of a mobile robot using a Neural Dynamics-based approach”, IEEE Transactions on Neural Networks, Vol.14, Iss.6, pp.1541-1552

[177] Ye X.F.; Meng M.Q., et al. (2002), “Statistical analysis and prediction of round trip delay for Internet-based teleoperation”, *IEEE/RSJ Conference on Intelligent Robots and System*, Vol.3, pp.2999- 3004

[178] Zadeh L.A., (1965), “Fuzzy sets”, Information and Control, vol.8, pp.353-383

[179] Zhang J., Knoll A. (2000), “Integrating deliberative and reactive strategies via fuzzy modular control”. In D. Driankov and A. Saffiotti, eds, *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*, Physica-Verlag, Heidelberg, New York, pp.367-387

[180] Zhuang X.D.; Meng Q.C; Yang S.J. (2002), “Mobile robot control in dynamic environments based on hybrid intelligent system”, IEEE Conference on SMC, Vol.3, pp.6

Appendix A. The robotic programming

In general, a robot control program is one that takes the robot's sensory input, processes it, and decides what motor actions the robot will perform. But the mapping between inputs and outputs is a very complex one, and the control task requires some decomposition into simpler elements to make it workable. In recent years there have been some convergences on an architecture (see Figure A.1) for autonomous mobile robots. The bottom control layer is a controller that implements some form of motion control for the robot. The second execution layer initiates and monitors behaviors, taking care of temporal aspects of coordinating behaviors. The top planning layer makes long-term deliberative planning, with the results being passed down to the second layer for execution.

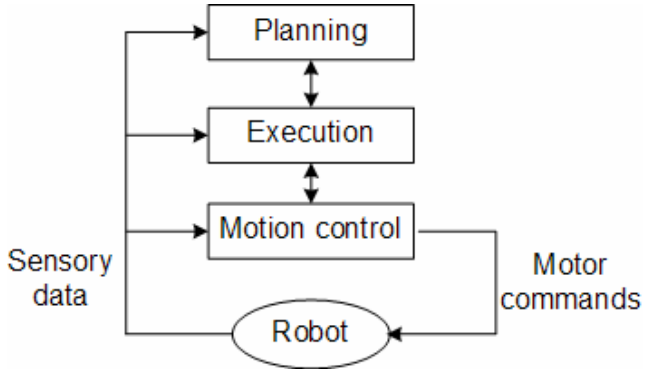


Figure A.1: A hybrid control architecture

We make the robotic programming based on the Saphira development environment (http://www.activrobots.com). The Saphira is an object-oriented, C++ language-based robotic development environment for creating software that intelligently and autonomously control a mobile robot. The Saphira clients work through ARIA (ActivMedia Robotics' Interface for Applications) software to send commands to the robot server, gather information from the robot's sensors, and package them for display in a graphical window-based user interface. ARIA handles the lowest-level details of client-server interactions, including serial communications, command and server-information packet processing, cycle timing, and multithreading, as well as a variety of accessory controls, such as for the PTZ robotic camera. It is possible to call Saphira from any high-level language that has a foreign-function loading facility, LISP and PROLOG, for example. For this thesis, we take use of Visual C++ 6.0 to write and compile the robotic programs based on the Saphira API functions.

Here gives an example of main body programs to realize a WANDER task.

```
#include "PolyUiBot.h"
#include "header.h"

SFEXPORT void                    // define interface to Colbert here
sfLoadInit ()
{
 draw();              // set up drawing object
 mycamera_init();
 SfFrame *ff = (SfFrame *)SfFRAME;  // add menu item, button and key handlers
 ff->Win()->AddButtonHandler(button_fn); // do the mouse thing
 ff->Win()->AddKeyHandler(key_fn); // do the key thing
 sfAddEvalAction("Wander", (void *)SfWanderAction::invoke, 0);
 sfAddEvalFn("WANDER", (void *)wanderCommand, sfVOID, 0);
}

//////////////////////////////////////////
// First obstacle avoidance behavior with wander task using Fuzzy logic controller
//////////////////////////////////////////

class SfWanderAction : public ArAction, public SfArtifact
{
 public:
  SFEXPORT SfWanderAction(); // constructor
  virtual ~SfWanderAction() { FuzzyUnload(); };     // nothing doing
  SFEXPORT virtual ArActionDesired *fire(ArActionDesired currentDesired);
  static SfWanderAction *invoke(); // interface to Colbert
  int FuzzyLoad();
  int FuzzyUnload();
  int FuzzyOutput(double dfront, double dleft, double dright, double goal_error);

 protected:
  ArActionDesired myDesired;          // what the action wants to do
  FIS *fis;
  DOUBLE **fisMatrix, **outputMatrix;
  int data_row_n, data_col_n, fis_row_n, fis_col_n;
  DOUBLE
dataMatrix[OA_NEW_INPUT_NUMBER][OA_NEW_INPUT_VECTOR];
};

// This constructor is a model for all actions. Chains to the basic ArAction class
SFEXPORT
SfWanderAction::SfWanderAction(): ArAction("Wander")
{
    FuzzyLoad();
}
```

```
// What the action does
// Returns and ArActionDesired pointer, containing what the action wants to do

SFEXPORT ArActionDesired *
SfWanderAction::fire(ArActionDesired d)
{

 // reset the actionDesired (must be done)
 myDesired.reset();

 if(bSTOP)   return &myDesired;              // return the desired controls

 double d0, d1, d2, d3, d4, d5, d6, d7;
 SfSonarDevice *sd = Sf::sonar(); // get the device
 if (!sd)
        d3 = d4 = 5000;                 // large value, no obstacle ahead
 else
 {
   d0 = SfROBOT->getSonarRange(0);              d1          =          SfROBOT-
>getSonarRange(1);
   d2 = SfROBOT->getSonarRange(2);              d3          =          SfROBOT-
>getSonarRange(3);
   d4 = SfROBOT->getSonarRange(4);              d5          =          SfROBOT-
>getSonarRange(5);
   d6 = SfROBOT->getSonarRange(6);              d7          =          SfROBOT-
>getSonarRange(7);
 }
 ////////////////// to reduce the distance input dimension /////////////////
 double dfront = MIN(d2,d3);
 dfront = MIN(dfront, d4);
 dfront = MIN(dfront, d5);
 double dleft = MIN(d0,d1);
 double dright = MIN(d7,d6);

  // convert from .mm to .cm
 dfront = dfront/10.;     dleft = dleft/10.;       dright = dright/10.;

 // take the input variable into fuzzy domain
        if(dfront<0  ||  dfront  >  MAX_OBSTACLE_DISTANCE)  dfront  =
MAX_OBSTACLE_DISTANCE; // for FLC normalization
        if(dleft  <0  ||  dleft  >  MAX_OBSTACLE_DISTANCE)  dleft  =
MAX_OBSTACLE_DISTANCE;
        if(dright  <0  ||  dright  >  MAX_OBSTACLE_DISTANCE)  dright  =
MAX_OBSTACLE_DISTANCE;

/////////////// to calculate OA behavior output ///////////////
 FuzzyOutput(dfront, dleft, dright);
 double speedVal = outputMatrix[0][0];   // robot speed
 double angleVal = outputMatrix[0][1];   // robot angle turn
```

```
  myDesired.setHeading(SfROBOT->getTh()+angleVal);     //control the heading
  myDesired.setVel(speedVal);          // moderate speed

  return &myDesired;              // return the desired controls
}


//////////////////////////////////////////////////////////////
// Interface to Colbert
//
// This static function returns a behavioral action object,
// with arguments that can be set from Colbert
//
SfWanderAction * SfWanderAction::invoke()
{
   return new SfWanderAction();
}

//////////////////////////////////////////////////////////////
void wanderCommand()
{
  SfActTask *task;
  SfWanderAction *a;
  task = SfActRegister::getAct("Wander"); // this is the default name
  if (task != NULL)
    {
     a = (SfWanderAction *)(task->action); // get the action object from the task shell
     a->activate(); // activate the behavior action
     SfROBOT->clearDirectMotion();           // lets behavioral actions through
     bSTOP = false;   // true = running, false = no running
    }
}
```

Appendix B. Reports of newspaper and magazines in Hong Kong

PC Focus

04年首個大型資訊博覽展ICT Expo

讓世界各地資訊技術專才匯聚香港

Hall 5

上個4月14至17日，一連四日，假香港會議展覽中心舉辦的「國際資訊科技博覽」(International Information and Communication Technology (ICT) Expo)經已完滿結束。今次展覽吸引了世界不同地區的IT人才及企業來港，實在不少於本地IT公司參展。

在展覽的首日吸引了逾萬名參觀者，場面可謂十分熱鬧，大會安排了在場每一名入場者填寫入場表格然後即時製作入場證，效率十分高，因此並沒有出現混亂的情況，值得一讚。

今年大會把場地分為九大區域，它們分別為電子政府解決方案、電子物流及零售科技、Linux平台應用軟件、E-Security資訊保安、創新科技、本地開發科技、電訊服務、流動及無線科技以及貿易服務。

大會為每一位入場者即時製作入場證，並在進場員在入口處入場看展覽。

A. 電子政府解決方案

香港郵政 - 綜合收遞管理系統

香港理工大學 - 多媒體創新中心

B. 本地開發科技

立體虛擬空間

智能導遊機器人

C. 電訊服務、流動及無線科技

PCCW - 數碼服務大集合

WebEx Asia Limited - 網上會議方案

ISIA資料科技館 - 電子化業管理

金蝶國際軟件(香港)有限公司

D. 貿易服務

賽美巴黎PTX

環球電訊有限公司 - "大力打"透過IDD

PC Station

TECH LIFE

Shuhei (shuheicou@capella.freemail.ne.jp)　攝影：Ringo　美術：Jason

展覽與房地產單位示範，均需要真人作導遊，這是我們一直以來的觀念。不過上周在會展舉行的 ICT Expo 場館，卻有一台三輪機械人四處遊蕩，當見有參觀人士走近它會自動停下來，活像一個機械人導遊，原來它是理工大學一手開發的 PolyUiBot。

港產機械人 PolyUiBot 睇樓、導遊[illegible]

雙[illegible]通訊制式

PolyUiBot 在外觀上，像一台機動三輪車多於一台機械人，不過它身上的無線 LAN 及 2.4GHz 無線電頻譜（Radio Frequency），加上一組鏡頭，卻使它的機能表現與平時在日本出現的機械人差不多。無線 LAN 主要用作配合機械人專用伺服器（Robot Server），對 PolyUiBot 進行遙端操作。至於 2.4GHz 無線電頻譜，則用於傳送鏡頭前影像至串流視訊伺服器（Streaming Server）。

▲ PolyUiBot 與其他設備關係簡介。

於 Robot Server 面前負責操作 PolyUiBot 人士，可透過互聯網直接發動指令，以及接收並觀看實時影像。雖然不少操作是透過手動方式完成，然而 PolyUiBot 也有自動化一面，可預先準備好編程（Scripting），然後經由 Robot Server 發送至 PolyUiBot。比方說我們可寫出一組要求 PolyUiBot 只靠直線行走的 Script 便可。

▲察覺前方的[illegible]，PolyUiBot 立即停止前進。

當日筆者即時在 PolyUiBot 面前，放置一[illegible]，PolyUiBot 的鏡頭與內置感應裝置，發現到前方出現障礙物，隨即減慢速度並停下來。

PolyUiBot 的鏡頭會一邊拍攝前方的畫面（鏡頭可以作擺動），一邊傳送影像至 Robot Server，所以在 Robot Server 面前的人士，能知悉 PolyUiBot 前方的情況。

▲在顯示器上的 PolyUiBot 影像。

遙距睇樓最佳夥伴

理工大學計算學系教授[illegible]表示，PolyUiBot 的其中一個發展理念，是為跨境跨地域的房地產交易，買家未必每次能參加發展商的示範活動，現在可透過 PolyUiBot 充當示範單位導遊，在沿途拍下實時影像後，買家經由互聯網收看，實現「遙距睇樓」。

王氏指出，基於 PolyUiBot 同時具備無線遙控與拍攝實時影像能力，所以它的「身份」更可以是展覽會導遊，而在添加額外裝置後，更可成為 24 小時執勤的保安人員。

▲ PolyUiBot 後輪是無動力，只靠前輪帶動。

▲[illegible] PolyUiBot [illegible]

續篇 又一家居／辦公室自動化成員

[illegible]

參考網站：http://www4.comp.polyu.edu.hk/~csnkliu/polyuibot/

110 e-zone

City Hot 18

同期舉行的「國際資訊科技博覽」，分為九個主題區域，展示香港科技成就，其中將港的大學界展出不少研究成果，全部有概念兼實用導，絕非紙上談兵。

自我檢查遊樂場

有參頭條，又有電腦怪獸，成個森林模樣，以為是兒童遊樂場，實際是理大護理系供小朋友使用的身體檢查站，站在民屋下就可展高體重，在電腦旁依著有聲有畫指示，就可驗血壓、驗血含氧量，讓小朋友在開心的環境下，從小學習身體檢查的重要性。

3D超級實立體眼鏡

科幻片中，戴上特殊眼鏡，會見到古怪事件，在真實世界中，這副3D眼鏡，卻能幫助醫療及幫助工業發展。理大設計的Augmented Reality，把平面的符號，注入特別意思，再配合WebCam及特設的軟件，在眼鏡中看見的，已是會移動的立體影像。可以用作設計遊戲，或者把以往的立體建築圖，變成3D圖案，方便修改。

互聯網機械人

有無想過，在香港也可以遙控身在離細的遊樂術的機械人，再透過視像觀看遠方的世界呢？理大電子計算機學系的智能機器人出（PolyUBot）就可以做到，以後不再只有太空署才能用衛星遙控機械人動作（登陸火星及月球），透過一個網頁瀏覽，經由伺服器，就可以把指令傳到機械人中，控制他們的動作，一般用家，透過電腦畫面就可以即時睇到遠方機械人的環境了。

觸感科技新潮流

XBOX Live大發售，N-Gage來二代亦推出在即，但無論如何新，打機也要用手指不停按動，中大自動化與計算機輔助工程學系研製了觸感技術，以後打機就會像真人觸動一樣，隨機手就可控制得到，助聽器打，比較驚更易操控。

觸感技術已開始在PDA中試用。

掌紋保安系統

為了保安，不少公司都在出入口建立了密碼鎖系統，理大電子計算機學系則創出了掌紋保安系統，把手掌放在識別機，以每人獨特的掌紋為認證，不怕密碼外洩，掌紋比指紋更易管理，就算雙手有「損傷」，亦可使用。此發明已得到2003年香港工業獎科技成就，及2002年韓國機械發明展銀獎。

香港電子展加強版

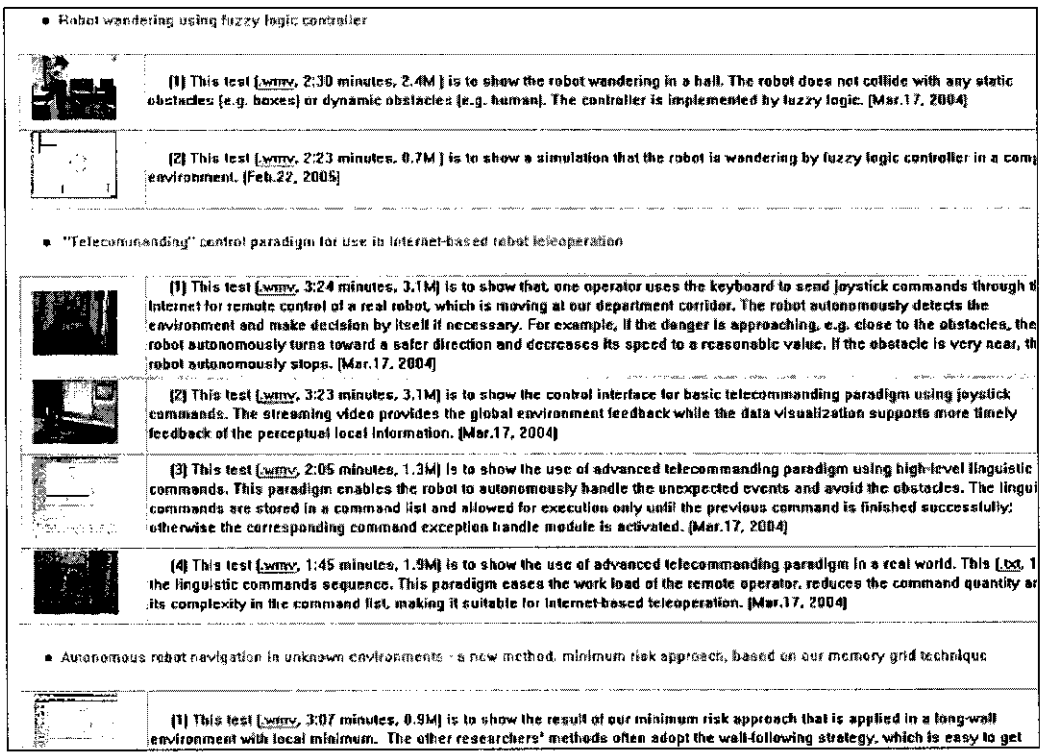
「香港電子產品展」本是每年十月舉行的電子科技產品Show off藝，上年SARS以後的展覽，熱鬧依然。有鑑於場地不足夠，香港貿易發展局增開四月份的春季電子展。香港廠商會設下攤位，展出來年至熱產品，吸引外資生意，約半年後，香港設計的產品，大多數也會在外國換名出現。展覽規模相當大，佔據了灣仔會展多個展館。今年亦同時舉行「國際資訊科技博覽」，一個主攻消費品，一個主攻科技，二合一展出香港科技產品優秀的一面。

Hong Kong Electronics Fair (Spring Edition)

香港設計產品路向

有在廿年歷史的本地公司Kinger，一向出品電子產品，由以往的電子鐘表發展成今日的健體產品。其市場總監張小姐表示，香港出品在不斷轉變中已符合大潮流，所以他們今年進軍健體產品，反應也不錯。她相信今年流行的香港設計，是太空時代感，給人一種未來感覺。金屬加上膠質物料，有如美國Sharper Image般高格調，但售價卻是平民化。

Appendix C. Snapshots of the website



- Robot wandering using fuzzy logic controller

(1) This test (.wmv, 2:30 minutes, 2.4M) is to show the robot wandering in a hall. The robot does not collide with any static obstacles (e.g. boxes) or dynamic obstacles (e.g. human). The controller is implemented by fuzzy logic. (Mar.17, 2004)

(2) This test (.wmv, 2:23 minutes, 0.7M) is to show a simulation that the robot is wandering by fuzzy logic controller in a com environment. (Feb.22, 2005)

- "Telecommanding" control paradigm for use in Internet-based robot teleoperation

(1) This test (.wmv, 3:24 minutes, 3.1M) is to show that one operator uses the keyboard to send joystick commands through t Internet for remote control of a real robot, which is moving at our department corridor. The robot autonomously detects the environment and make decision by itself if necessary. For example, if the danger is approaching, e.g. close to the obstacles, the robot autonomously turns toward a safer direction and decreases its speed to a reasonable value. If the obstacle is very near, th robot autonomously stops. (Mar.17, 2004)

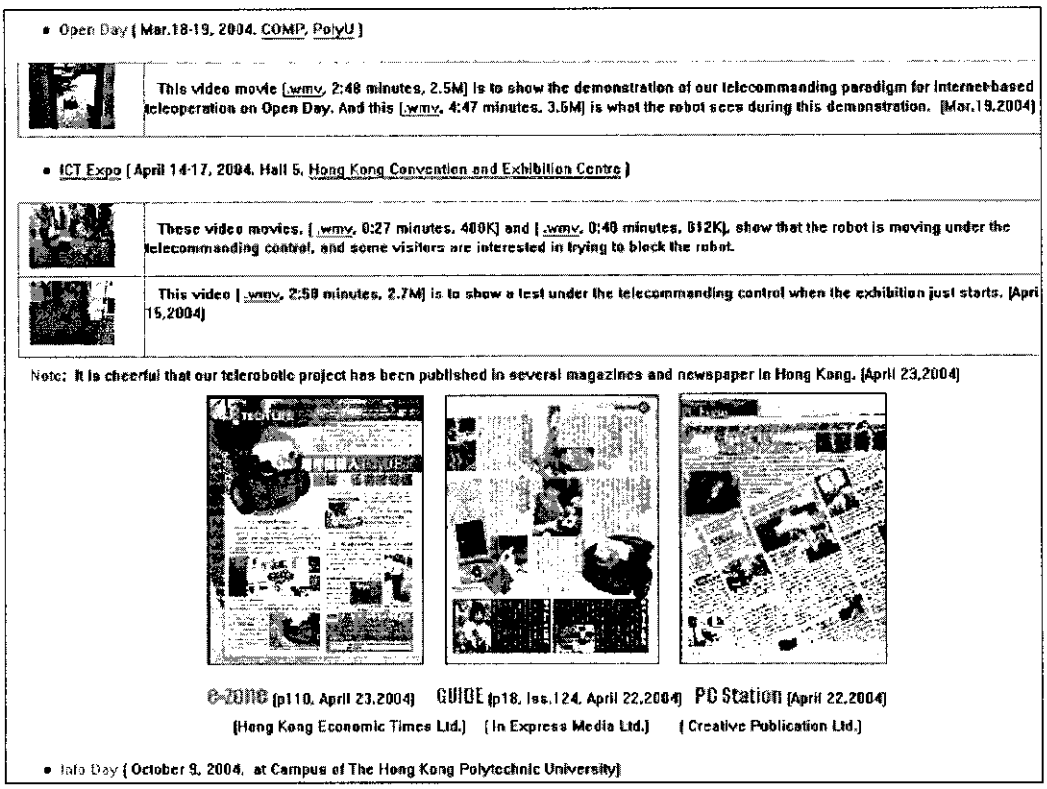
(2) This test (.wmv, 3:23 minutes, 3.1M) is to show the control interface for basic telecommanding paradigm using joystick commands. The streaming video provides the global environment feedback while the data visualization supports more timely feedback of the perceptual local information. (Mar.17, 2004)

(3) This test (.wmv, 2:05 minutes, 1.3M) is to show the use of advanced telecommanding paradigm using high-level linguistic commands. This paradigm enables the robot to autonomously handle the unexpected events and avoid the obstacles. The lingui commands are stored in a command list and allowed for execution only until the previous command is finished successfully; otherwise the corresponding command exception handle module is activated. (Mar.17, 2004)

(4) This test (.wmv, 1:45 minutes, 1.9M) is to show the use of advanced telecommanding paradigm in a real world. This (.txt, 1 the linguistic commands sequence. This paradigm eases the work load of the remote operator, reduces the command quantity a its complexity in the command list, making it suitable for Internet-based teleoperation. (Mar.17, 2004)

- Autonomous robot navigation in unknown environments - a new method, minimum risk approach, based on our memory grid technique

(1) This test (.wmv, 3:07 minutes, 0.9M) is to show the result of our minimum risk approach that is applied in a long-wall environment with local minimum. The other researchers' methods often adopt the wall-following strategy, which is easy to get



- Open Day (Mar.18-19, 2004, COMP, PolyU)

This video movie (.wmv, 2:48 minutes, 2.5M) is to show the demonstration of our telecommanding paradigm for Internet-based teleoperation on Open Day. And this (.wmv, 4:47 minutes, 3.6M) is what the robot sees during this demonstration. (Mar.19,2004)

- ICT Expo (April 14-17, 2004, Hall 5, Hong Kong Convention and Exhibition Centre)

These video movies, (.wmv, 0:27 minutes, 480K) and (.wmv, 0:48 minutes, 612K), show that the robot is moving under the telecommanding control, and some visitors are interested in trying to block the robot.

This video (.wmv, 2:59 minutes, 2.7M) is to show a test under the telecommanding control when the exhibition just starts. (April 15,2004)

Note: It is cheerful that our telerobotic project has been published in several magazines and newspaper in Hong Kong. (April 23,2004)

e-zone (p110, April 23,2004) (Hong Kong Economic Times Ltd.) GUIDE (p18, Iss.124, April 22,2004) (In Express Media Ltd.) PC Station (April 22,2004) (Creative Publication Ltd.)

- Info Day (October 9, 2004, at Campus of The Hong Kong Polytechnic University)

Appendix D. Streaming technologies

Streaming media technologies were introduced in 1995 [Mack, 2002]. Streaming offers a whole new approach to media on the Internet. Instead of waiting for the whole file to be downloaded to a user's computer before playback begins, streaming media playback occurs as the file is being transferred. The data travels across the Internet, is played back and then discarded. Streaming media also offers the user control over the stream during playback, something not possible with a web server.

One of the problems that streaming media systems have to deal with is the stochastic nature of bandwidth on the Internet. It fluctuates wildly between zero and some maximum rate. To deal with this, streaming media player utilizes a buffer. The first few seconds of the file are stored in the computer's memory before playback begins. This gives the media player a reserve of bits to fall back on when the user's bandwidth becomes constricted.

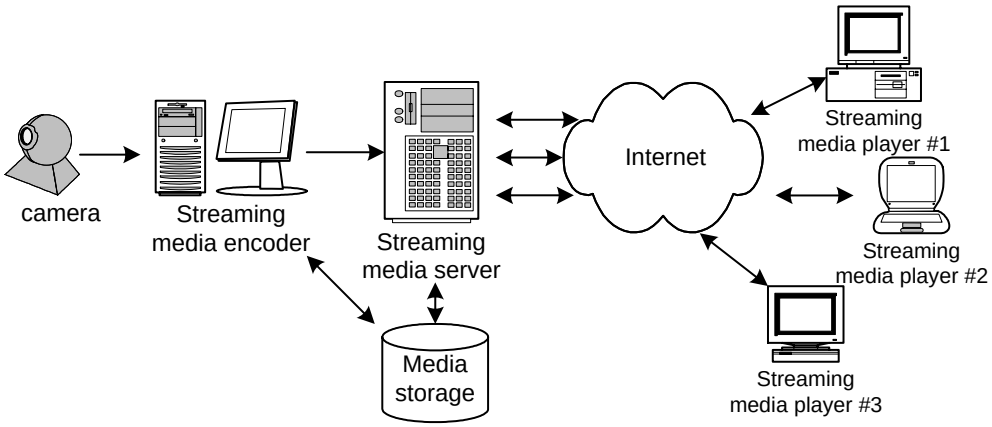


Figure D.1: Basic components of a streaming media system

Streaming media (e.g. video, audio, flash, script, etc.) is made possible by different pieces of software that communicate on a number of different levels. A basic streaming media system has three components [Mack, 2002]. The basic components of a streaming media system are shown in Figure D.1.

- Player. The software that viewers use to watch or listen to streaming media.
- Server. The software that delivers streams to audience members.
- Encoder. The software that converts raw audio and video files into a format that can be streamed.

These components communicate with each other using specific protocols (e.g. RTSP, MMS), and exchange files in particular formats (e.g. RM, WMV, MOV, MP4).

Some files contain data that has been encoded using a particular codec (e.g. MPEG4, Windows Media Video, Real Video, Sorenson Video), which is an algorithm designed to reduce the size of files. Typical architecture of streaming server and client is shown in Figure D.2.

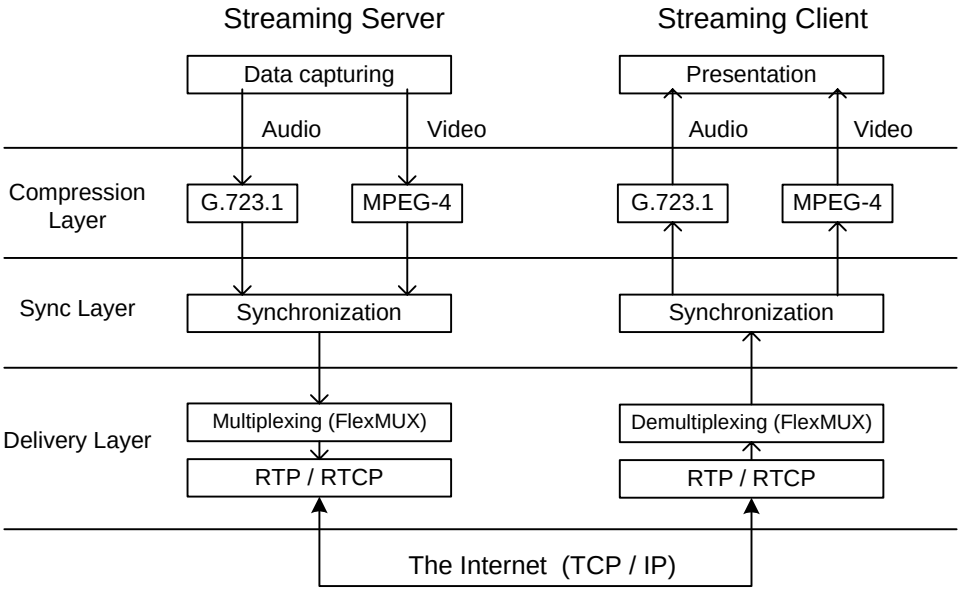


Figure D.2: Typical architecture of streaming server and client

RTSP (Real Time Streaming Protocol) is an application-level protocol developed by IETF (Internet Engineering Task Force) that is used to control the delivery of data with real-time properties [Mack, 2002]. RTSP provides a framework to enable the controlled, on-demand delivery of real-time data, such as audio and video. Sources of data can include both live data feeds and media on-demand. This protocol is intended to control multiple data delivery sessions; provide a means for choosing delivery channels such as UDP, Multicast UDP, and TCP; and provide a means for choosing delivery mechanisms based on Real Time Protocol (RTP). QuickTime and RealSystem use the RTSP protocol. Microsoft uses its own MMS (Microsoft Media System) protocol. Both RTSP and MMS contain a control mechanism to handle client's requests, such as Play, Stop, Fast Forward, or Rewind. Both protocols ensure that media packets arrive in a format recognized by the player. Control requests are always carried over TCP, and data packets are carried over UDP, TCP, or HTTP (HTTP resolves the firewall issues). A growing number of vendors use RTSP for the development of new technologies that deliver streaming content to mobile devices.

The Unicast and Multicast are two methods used to deliver streaming content across networks to end-users. A unicast stream has a one-to-one client-server relationship. When a user makes a request to stream media, the server acts on the request and sends a unique individual stream to that client, one steam for each request. This method maximizes the ability to compensate for lost data and to deliver a better experience to end-users. A multicast stream is more like the experience of watching television. The media server generates one single stream that allows multiple player-clients to connect to it. Users watch the content from the time they join the broadcast. The client connects to the stream, but not to the server. During a multicast stream, the player-client cannot request for the replacement of lost packets. This method saves network bandwidth and is mostly used for live broadcasts.

Appendix E. A brush-up of fuzzy system theory

The theory of fuzzy logic has its roots back in 1965 when Zadeh presented his ideas of fuzzy sets [Zadeh, 1965]. An overview of some of the fundamental concepts in fuzzy systems has been presented here to provide background knowledge used in this thesis. Most of the definitions given in this section have been paraphrased from [Passino & Yurkovich, 1998].

A fuzzy system is shown in Figure E.1, which is static nonlinear mapping between inputs and outputs. The inputs are $u_i \in U_i$, where $i=1,2,\ldots,n$, and outputs $y_i \in Y_i$, where $i=1,2,\ldots,m$. The outputs and inputs are crisp that is real numbers, not fuzzy sets. These crisp inputs are mapped into fuzzy sets by the fuzzification block, in order to activate rules which are in terms of linguistic variables. The variables have fuzzy sets associated with them. The inference mechanism produces conclusions using fuzzy rules in the rule-base. Crisp outputs are obtained from the defuzzification block.

Universe of Discourse

The crisp sets U_i and Y_i are called the *universe of discourse* for u_i and y_i respectively. Generally the universes of discourse are simply the set of real numbers or some interval or subset of real numbers.

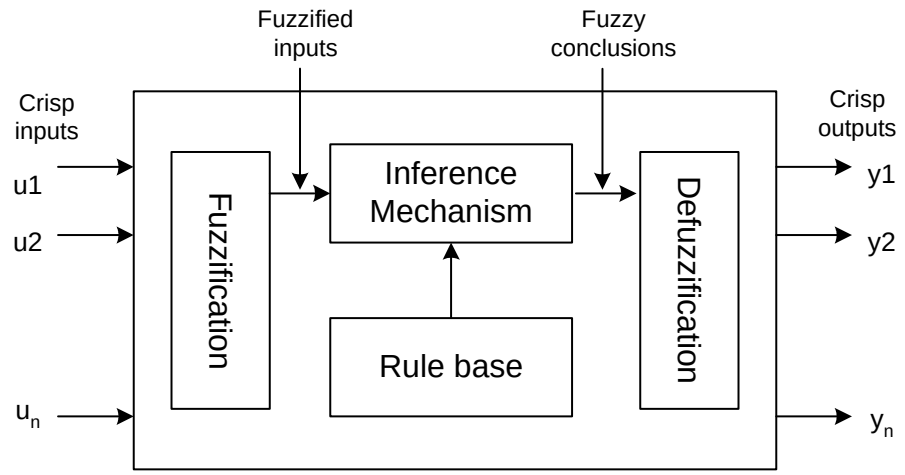


Figure E.1: Fuzzy system.

In classical set theory, an element of any universe can be either a member of the set or not. Fuzzy sets, however, are characterized by the fact that an element of the universe of discourse has a so-called degree of membership, determined by a membership function, i.e. an element can not only belong or not belong to a set, but belong more or less to it. This fuzziness is also characteristic for human beings when they are asked to classify certain elements. The procedure of determining the degree of membership of a crisp input, which is an element of a universe of discourse, is called fuzzification.

Linguistic Variables

These are variables whose values are not number but words or sentences in a natural or artificial language to describe fuzzy system inputs and outputs. Where $\tilde{u}_i$ is the *linguistic variable* that describes the inputs u_i. Similarly $\tilde{y}_i$ is the linguistic variable that describes the output y_i .

Linguistic Values

Linguistic variables $\tilde{u}_i$ and $\tilde{y}_i$ take on *linguistic values* that describe the characteristics of the variable. The set of linguistic values $\tilde{A}_i = \{\tilde{A}_i^j : j = 1, 2, \cdots, N_i\}$ where $\tilde{A}_i^j$ denotes the j^{th} linguistic value of the linguistic variable $\tilde{u}_i$. Similarly $\tilde{B}_i = \{\tilde{B}_i^k : k = 1, 2, \cdots, M_i\}$, where $\tilde{B}_i^k$ denotes the k^{th} linguistic value of the linguistic variable $\tilde{y}_i$.

Linguistic Rules

A set of *condition*$\rightarrow$ *action* rules, or in *modus ponens* (If-Then) rule maps the inputs to the outputs

If antecedent **Then** consequent

Usually, the inputs to the fuzzy system are associated with the antecedent, and the outputs are associated with the consequent, for the multi-input single-output (MISO) the standard rule form is

If $\tilde{u}_1$ *is* $\tilde{A}_1^j$ *and* $\tilde{u}_2$ *is* $\tilde{A}_2^j$ *and* $\cdots\cdots, \tilde{u}_n$ *is* $\tilde{A}_n^j$ **Then** $\tilde{y}_q$ *is* $\tilde{B}_q^p$

This can be in the form of multi-input multi-output (MIMO). Generally the rules in the rule-base are distinct.

Membership Functions

The membership functions $\mu(u_i)$ are subjectively specified in an ad hoc (heuristic) manner, they are associated with the terms that appear in the antecedent and consequent. Many shapes of the membership function are possible (e.g., triangular, trapezoidal shapes), each will provide a different meaning for the linguistic variable.

Fuzzy Sets

Simply a fuzzy set is a crisp set of elements of the universe of discourse paired and coupled with their associated membership value.

$$A_i^j = \{(u_i, \mu_{A_i^j}(u_i)) : u_i \in U_i\}$$

Fuzzification

Fuzzification transforms u_i to a fuzzy set defined on the universe of discourse U_i^* . This transformation is produced by operator f defined by

$$f : U_i \rightarrow U_i^*$$

Where $f(u_i) = \tilde{A}_i^{fuz}$, $\tilde{A}_i^{fuz}$ is the fuzzy set.

Quite often singleton fuzzification is used. Any fuzzy set with the following form for its membership function is called a *singleton*.

$$\mu_{\tilde{A}_i^{fuz}}(x) = \begin{cases} 1 & x = u_i \\ 0 & \text{otherwise} \end{cases}$$

Fuzzy Intersection (AND)

Two methods to define the membership function that represents the intersection of A_i^1 and A_i^2.

1) Minimum, $\mu_{A_i^1 \cap A_i^2} = \min\{\mu_{A_i^1}(u_i), \mu_{A_i^2}(u_i) : u_i \in U_i\}$
2) Algebraic Product, $\mu_{A_i^1 \cap A_i^2} = \{\mu_{A_i^1}(u_i)\mu_{A_i^2}(u_i) : u_i \in U_i\}$

Fuzzy Union (OR)

Two methods to define the membership function that represents the union of A_i^1 and A_i^2.

1) Maximum, $\mu_{A_i^1 \cup A_i^2} = \max\{\mu_{A_i^1}(u_i), \mu_{A_i^2}(u_i) : u_i \in U_i\}$
2) Algebraic Sum, $\mu_{A_i^1 \cup A_i^2} = \{\mu_{A_i^1}(u_i) + \mu_{A_i^2}(u_i) - \mu_{A_i^1}(u_i)\mu_{A_i^2}(u_i) : u_i \in U_i\}$

Fuzzy Implications

It is the fuzzy quantification of the linguistic rule. The implication method shapes the consequent based on the antecedent. The terms in the antecedent and consequent of the If-Then rule are fuzzily quantified to make a *fuzzy implication* (a fuzzy relation).

Aggregation

It is combining the output fuzzy sets into a single fuzzy set in preparation for defuzzification.

Defuzzification

It is a means to choose a crisp output based on the implied fuzzy sets. The most popular defuzzification method is the “centroid” calculation, which returns the center of area under the curve. In the centroid a crisp output is chosen based on the implied fuzzy sets and the point of maximum for each output membership function.

$$y = \frac{\sum_{j=1}^{M}[\bar{y}^j \mu_{A \circ R_j}(\bar{y}^j)]}{\sum_{j=1}^{M}[\mu_{A \circ R_j}(\bar{y}^j)]}$$

Where $A \circ R_j$ is a single implied fuzzy set for the j^{th} fuzzy implication, and $\bar{y}^j$ is the consequent portion of the linguistic rule R_j.