修士論文

Study on a Simulator of Environments with Distributed Sensors

分散配置されたセンサ環境のシミュレータに関する研究

2009 年 2 月 4 日提出

指導教員

橋本　秀紀　准教授

東京大学大学院工学系研究科

電気工学専攻　No.37-076908

鄭　韶華

CONTENTS

INDEXES OF FIGURES

INDEXES OF TABLES

CHAPTER 1

Introduction

1.1 Research Background

As the combination of sensor technology, artificial intelligence, and advanced computer technology, “Intelligent Space” (iSpace) [1] has been walking into our sights in recent years. Different with traditional robotics technology, Intelligent Space is trying to separate sensors from robot to the environment, thus clearly differentiate the role of actuators and sensors. The essence of Intelligent Space is effective fusion of actuators, sensors and environments. Embedded with many kinds of networking sensors, the environment can be observed in several different aspects like human activity, actuator activity etc. All the information can be used in understanding events happening in this environment and the environment itself. After collecting enough information, humanoid robots, mobile robots, robot arms or other kinds of actuators can serve the people in this environment.

This concept is also called “Ambient Intelligence Environment” [2], “smart room” [3] or “ubiquitous environment” [4] in which many devices are working with people who work or live in the environment. They collect information of people’s daily life and try to find useful information behind the surface of human activity. In recent years, so many research groups have been working with the development of “Intelligent Space”, like “Protean Living Project” [5] of UC, Berkeley, “Project Oxygen” [6] by MIT, “Aware Home” [7] Research Initiative of Georgia Tech, “Wabot-House” [8] developed by Waseda University in Japan. Besides, many manufactures and corporations are denoting to the research of this field as well, such as “Experience Lab” of Philips, “Robotic Room SELF” [9] of AIST in Japan.

In a world, the “Intelligence” of Intelligent Space can be summarized as the following: being able to observe environment, being able to recognize human and robot activities, being able to actuate tasks and always applies latest technology in improving the intelligence of environment.

As to the current research of Hashimoto Lab’s Intelligent Space, “Spatial Memory” [10] enables humans to store computerized information such as digital files and commands into the real world by assigning three-dimensional position as the memory address. Using “Active Projector” [11], people can interact with the active projector by human movement. RT-middleware [12] is now being used for packing separate working units into components so that they can be easily reused and distributed. Many latest sensors, like new type laser range finder made by Hokuyo Corp., acceleration sensor, Swiss Ranger [13] have been introduced into our laboratory in order to develop more “intelligent” space.

1.2 Research Purpose

In recent years, with the fast development of sensor technology, the relative researches in Intelligent Space have encountered many problems. For example, currently experiments are the only way to evaluate the correctness of new algorithm during its development. Many experiments are limited by the actual experiment environments, time and human resource. Therefore we consider simulation as an effective way to solve these problems and we propose our environment simulator with distributed sensors to support the further research in Intelligent Space.

The research of environment simulator with distributed sensors is trying to solve two problems. First, in order to develop better robot control algorithms or sensor-use algorithms without redundant and hard experiments, mobile robot and distributed sensor simulation is eagerly expected. Second, the sensor arrangement problem, which is more and more important after so many new sensors have been introduced into our Intelligent Space in recent years, becomes more and more troublesome. Among these sensors, laser range finder is the most widely used one for 2D scanning and its arrangement's result directly influences applications very much. Beside of the above main reasons, other applications like recurrence of the trajectory of human activities, control of multiple actuators are also looking forward to utilizing this simulator.

This simulator includes three tools: Environment Designer, which is used to create specific indoor environment as the basic simulation stage; Sensor Arranger, which is used to help user arrange sensors as well as obstacles and actuators. The last is simulation stage, Simulation Runner.

1.3 Thesis Structure

This thesis is organized as following. Fig.1.1 shows the flow chart of the whole thesis.

In Chapter 2, some relative sensing technology and intelligent space is introduced. After that, other popular robot simulators and ubiquitous environment simulators are introduced, their characteristics are summarized, and the comparison of simulators is made. Besides, the motivation to develop our environment simulator is given and its features are listed.

In Chapter 3, beginning with the framework of our simulator, the implementation is explained in detail. The sequence will be Environment Designer, Sensor Arranger, and Simulation Runner the last.

In Chapter 4, the automatic sensor arrangement mode of Sensor Arranger is discussed in detail. And the executions of automatic arrangement are discussed after that.

In Chapter 5, the simulations of mobile robot and distributed sensors are discussed and the comparison of simulation results with real experiments is after that. Besides, other applications realized by using this simulator are introduced as well.

At last, Chapter 6 is the conclusion and future work.

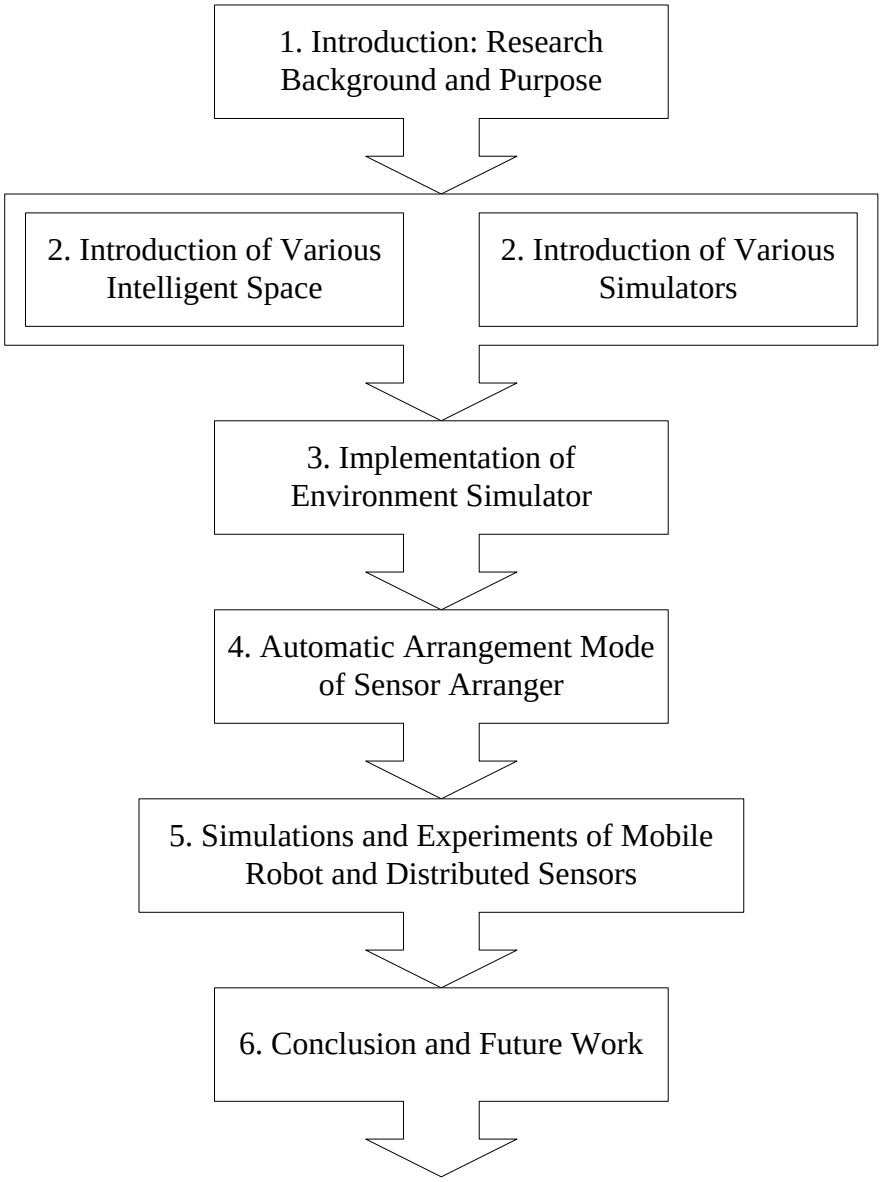


Fig.1. 1 Flow Chart of the Thesis

CHAPTER 2

Intelligent Spaces and Various Simulators

In this chapter, our proposed Intelligent Space and other Intelligent Spaces in the world will be introduced. After that, the current popular robot simulators will be surveyed. Beside of the robot simulators, ubiquitous environment simulators will also be surveyed because their aim is similar with our research. In the end, the comparison of simulators will be given and the characteristics of our simulator will be narrated.

2.1 Various Intelligent Spaces

What is called Intelligent Space is the environment in which a variety of sensors are distributed and installed. The information of environment can be learnt from these sensor data and after analysis service is served in many kinds of ways, for example, like using a robot or showing digital information. The aim of Intelligent Space is to give people better living or working place, to support all aspects of people's life. Here the researches of Intelligent Space will be introduced.

2.1.1 Hashimoto Lab. Intelligent Space

Intelligent Space (iSpace) [1] is an experimental environment developed by Hashimoto Lab. in the University of Tokyo. In this environment, a lot of sensors are embedded and they network with each other as DIND (Distributed Intelligent Network Devices) [14]. These sensors can be cameras, infrared sensors, ultrasonic sensors, laser range finders and so on. By using all these sensors, a lot of information like the most apparent position information can be collected and processed by computers through the highly networked sensor network. After the processing, agents like robots can give people suitable service according to the collected information. The concept of Intelligent Space can be seen in Fig.2.1.

However, in our Intelligent Space, the service is not only actuated by robots but also can be served in a lot of ways. For example, human's actions can be tracked by DIND, the information processing center tries to understand what these actions mean and respond properly. There is this kind of research called Spatial Memory [10], which has been deeply researched and now expanded into many fashions. If people use arm movement to tell that they want to read today's headline on newspaper, this information can be automatically searched and projected on the big screen. Questions like how can that happen can be answered according to the following words. The concept of Spatial Memory can be seen in Fig.2.2. The indoor space is divided into many units like the principle of memory in computer, in which digital

contents are “embedded”.

Continuing with the above example, if the headline of newspaper is embedded in the center of the room, maybe only 20 cm higher than the height of the people, it can be “touched” if people raise their arm to touch that space which means they want to access the contents saved at that place. It is actually a connection of specific place in the space to the contents of the user.

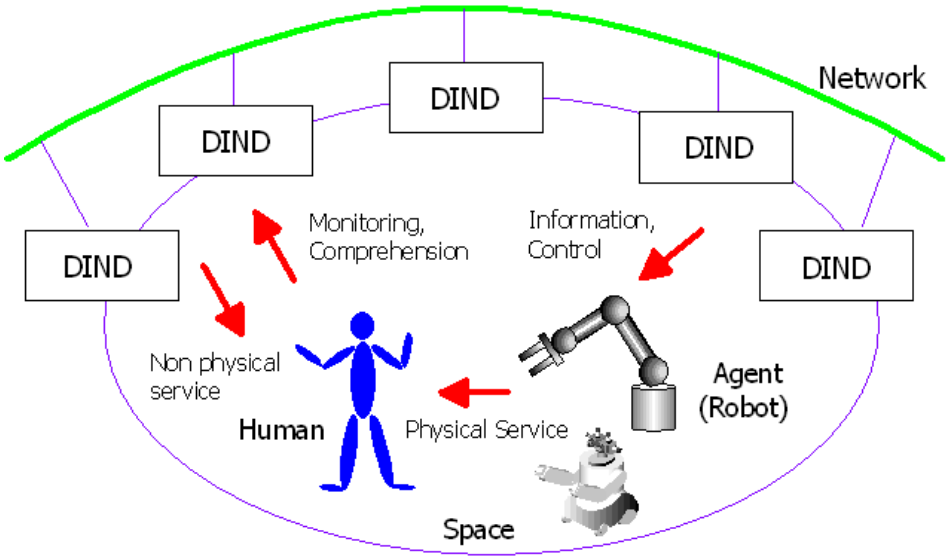


Fig.2. 1 Concept of Intelligent Space

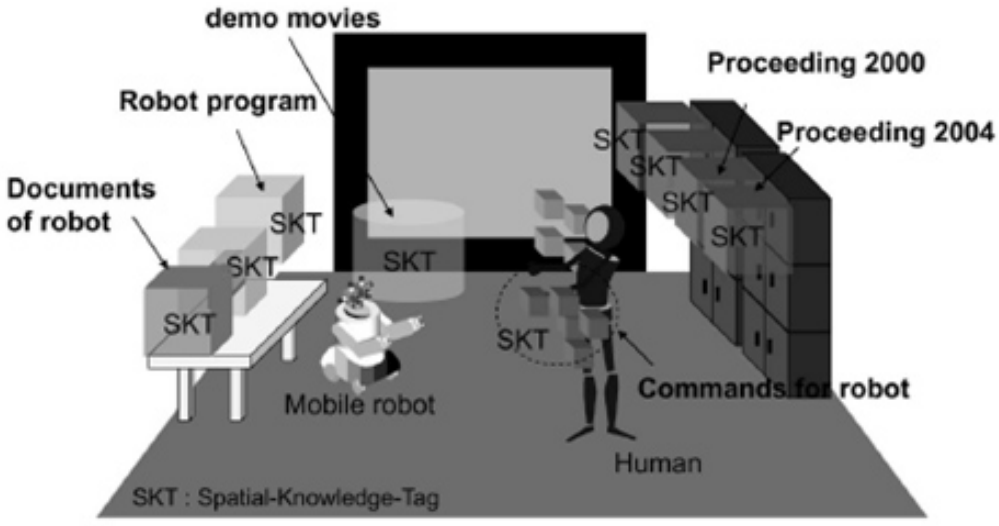


Fig.2. 2 Concept of Spatial Memory

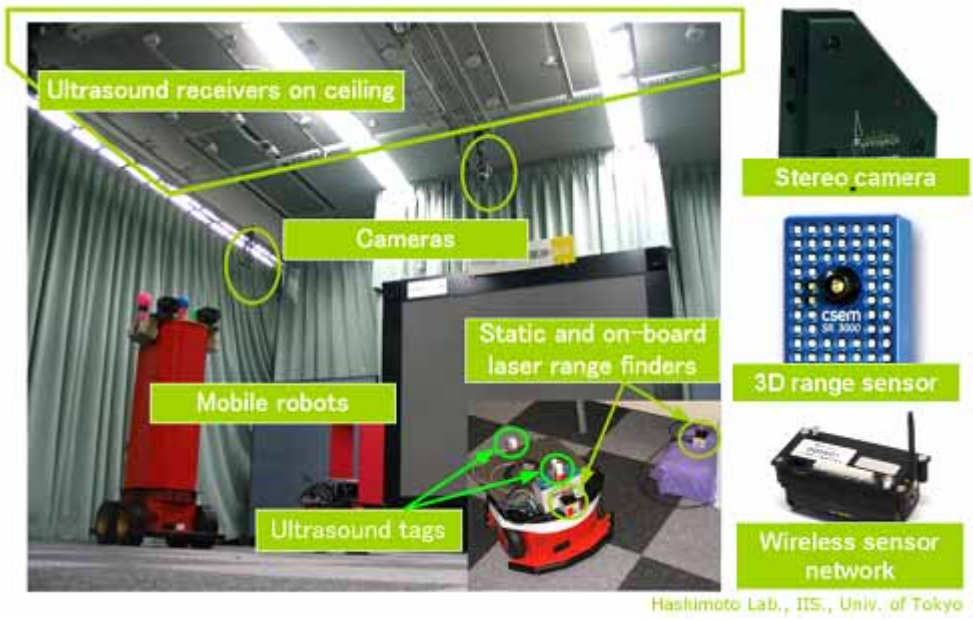


Fig.2. 3 Hashimoto Lab. Intelligent Space

The main sensor system and actuators are shown in Fig.2.3. In the picture, we can see that many high-spec sensors are being used in our lab. By using ultrasonic positioning system installed on the ceiling, the detection accuracy can reach 50 mm and it is enough to detect the arm movement of people and make the research of Spatial Memory possible. As to the specification of sensors and actuators in Intelligent Space, all information necessary is listed in Table.2.1.

Table.2. 1 Specification of Devices in Intelligent Space

Sensors or Actuators	**Production Name**	**Application**
Distributed CCD Camera	VC-C4R (Cannon) Dragonfly2（Point Grey Research）	Observation
Ultrasonic 3D Positioning System	ZPS-3D（FURUKAWA CO.,LTD.）	Observation
Laser Range Finder	URG-04LX（HOKUYO Automatic CO.,LTD.）	Observation
Wireless Sensor Network Device	MOTE（Crossbow）	Observation
3D Range Sensor	SR-3000（MESA Imaging）	Observation
Stereo Camera	Digiclops（Point Grey Research）	Observation
2-Wheel Mobile Robot	Pioneer2-DX（ActiveMedia）	Physical support
4-Wheel Mobile Robot	PioneerP2-AT（ActiveMedia）	Physical support
Active Projector	Projector：V-1100（plus-vision）	Information supply

2.1.2 Protean Living Project

As the creator's pronouncement, the Protean Living Project [5] is exploring technologies to enable a network of distributed devices to seamlessly cooperate to simplify daily tasks. It is started in the University of California, Berkley. The Protean Living Project is an umbrella for several research angles: system integration, video transcoder development, system architecture, programming abstractions, sensor networks, and novel hardware platforms.

Among these researches, the sensor network platform (SNSP) [15] has a service-oriented architecture. The capabilities of the sensor network are abstracted as services. Services may generate content that can be referred to later like historic temperature readings. Another key component of the SNSP is the inclusion of users. In any application space, there will be users that interact with the sensor network. The concept of personae captures preferences of users and their access to certain functionality.

As the number of networks, types of devices, and video coding standards increase, interoperability between different video systems and different networks is becoming more important. Video transcoding [16] is the key technology that enables a seamless interaction between different systems. The target of the research is to develop an efficient real-time video transcoder for Home Gateway Demo. Intra-refresh [17] transcoding architecture with selective encryption scheme [18] is proposed as an efficient solution for low complexity implementation. Dynamic bit-rate and resolution adaptation provide transmission bandwidth saving for various video clients with different processing power and display capabilities. The integrated selective encryption scheme protects the transcoded streams from unauthorized access.

2.1.3 Project Oxygen

Project Oxygen [6] is developed by MIT, the name of this project means the realization of the intelligence of space should be like oxygen we are breathing every time. Space-centered computation embedded in ordinary environments defines Intelligent Spaces populated by cameras, microphones, displays, sound output systems, radar systems, wireless networks, and controls for physical entities such as curtains, lighting, door locks, soda dispensers, toll gates, and automobiles. People interact in intelligent spaces naturally, using speech, gesture, drawing, and movement, without necessarily being aware that computation is present. In Project Oxygen, human-centered computation supports social interaction in its familiar context. People communicate with computation easily, as they do with each other, using shared knowledge and intelligence. Freely-available computation makes it easy for people to monitor and control their environment. Physical objects become more useful and usable, and people and agents in other environments become immensely easier to reach.

Environmental devices, also called an E21 [19], provide a local-area computational and

communication base for this Intelligent Space. E21s are connected to nearby sensors, actuators, and appliances, suitably encapsulated in physical objects. They communicate with each other and with nearby handheld devices (H21s) [19] through dynamically configured networks (N21s) [19]. E21s provide sufficient computational power throughout the environment to communicate with people using natural perceptual resources, such as speech and vision, to support Oxygen's user technologies wherever people may be, and to monitor and control their environment. E21s, as well as H21s, are universal communication and computation appliances. E21s leverage the same hardware components as the H21s so that the same software can run on both devices. E21s differ from H21s mainly in their connections to the physical world, the computational power they provide, and the policies adopted by the software that runs on the devices.

Among researches in Project Oxygen, Intelligent Room is a highly interactive environment that uses embedded computation to observe and participate in normal, everyday events. Microphone and camera arrays enable it to listen to people and observe what they are doing. People can speak with, gesture to, and interact with it in other complex ways. Smart sketching and design tools enable people to express their ideas and the room to record them. Other tools capture and record the essence of meetings in support of human collaboration. A robust agent-based software infrastructure supports the operation of these tools. Fig.2.4 shows the above main technologies in Project Oxygen.

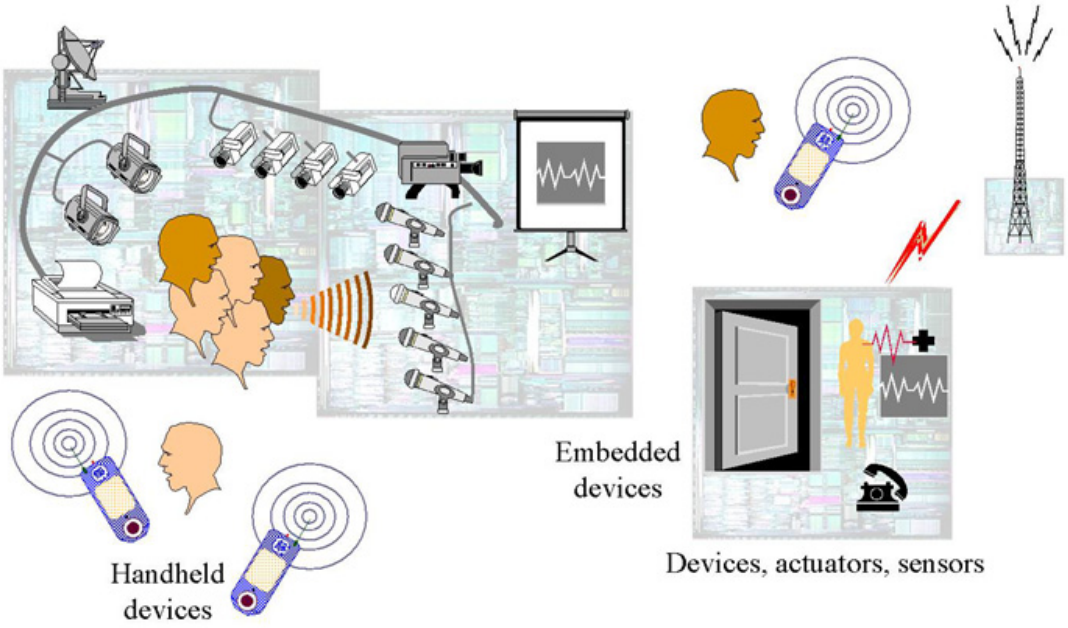


Fig.2. 4 Project Oxygen Device Technologies

2.1.4 Aware Home

The Aware Home Research Initiative (AHRI) [7] is an interdisciplinary research endeavor at Georgia Tech aimed at addressing the fundamental technical, design, and social challenges presented by such problems. The Aware Home Research Initiative at Georgia Institute of Technology is devoted to the multidisciplinary exploration of emerging technologies and services based in the home. Starting in 1998, with specific expertise in health, education, entertainment and usable security, they apply their research to problems of significant social and economic impact. Technology revolutionizes the tools that are used in the home of the near future. Embedded computing, sensing and actuation technologies coupled with new infrastructure in the built environment itself. For example, robots are one category of home tools that people will likely increasingly purchase for their homes. However, to date very little is known about how to design robotic products that fit into the home, do meaningful work there, and are desirable to own. These are critical questions to answer if going to design usable and useful home tools for the future. Understanding how current home tools fit into the work of the home can shed light on how to design the next generation of domestic robots. Such understanding can also inform the creation of new low-level technologies that can be built into tools, or into the home infrastructure itself. In Aware Home, pressure sensors [20] are embedded under floor, RFID tags are put in shoes and slippers, distributed cameras and microphones are installed on the ceiling. They track daily life of people by recording people's positions, sounds and any other kind of information.

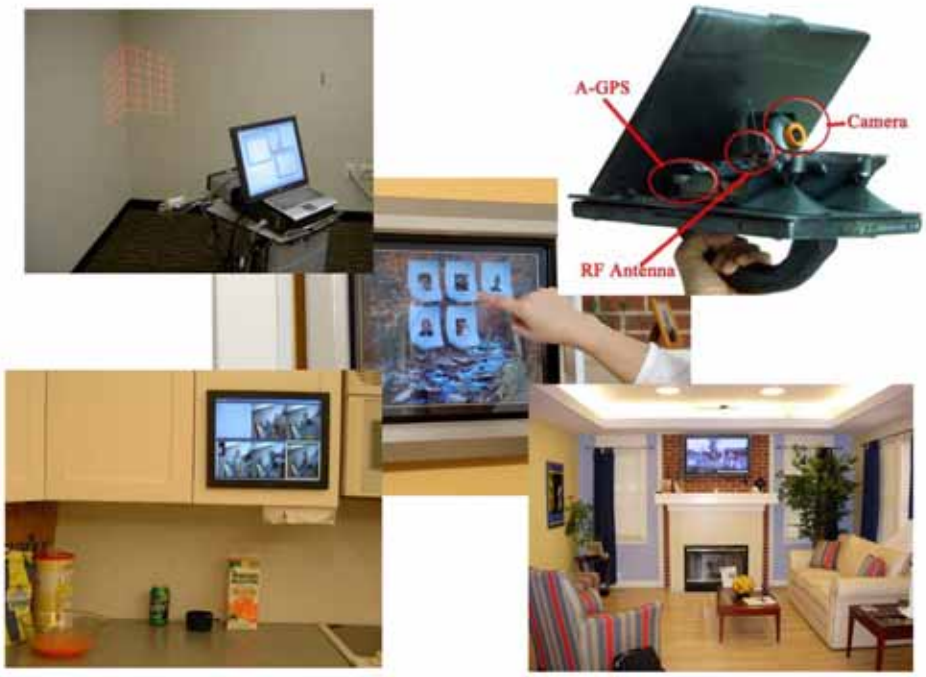


Fig.2. 5 Aware Home Technologies

In Fig.2.5 the applied technologies in Aware Home can be seen. The left top one is called overall projector, used to show position information. The left lower one is called Cook's Collage, used to support cooking. The middle one is Electronic Family Message Board. The right top one is typical sensors used in Aware Home and the right lower picture shows the scene in Aware Home.

2.1.5 WABOT-HOUSE

WABOT-HOUSE [8] is developed mainly by Waseda University, aims at constructing future rooms in which robots and human beings will live together in the future of Robot Times. Its construction is based on society system and theory of many social technologies. It is also named "Robot City" which includes not only human and robots, but also artificial constructions and natural environments. In order to find the proper position of robots and human beings, robots and robots, robots and constructions, many researches including society aspects are being proposed. WABOT-HOUSE Laboratory has two main assignments. The first is the research on robot community, to pursue the symbiotic society where robots and human beings share the same sphere of living embraced by the natural environment. The second is the research on coordination theories to develop a new regional robotics industry. Their research makes the research field expand to society field and consider many future trends of robotics in the near future. In one of their intelligent spaces, RFID and pressure sensors are also used to detect positions of robots and this research provides good accuracy in the positioning of robots. Fig.2.6 shows the appearance of WABOT-HOUSE.

Fig.2. 6 Scene of WABOT-HOUSE

2.1.6 Experience Lab

In 2001, Philips started HomeLab Project, and since then they expanded it into the Bigger Project called Experience Lab. They invite consumers and customers to take part in their research, allowing them to watch their activities and interact with new technologies. In a realistic environment, nature behavior of people can be observed while it can be hardly caught in other indoor places like laboratories.

What Experience Lab is pursuing are two aims: listening to people and making innovations that make sense. Experience Lab is the embodiment of sensor technology, which is not only for technology's sake, but to listen to users' needs and what they really want, then create products that enhance people's lives in a meaningful way. On the other hand, recurring to Philips' powerful developing level, new products and facilities are being created on the base of analyzing user's actions and their advices. Fig.2.7 shows the scene of Experience Lab of Philips.

Fig.2. 7 Experience Lab of Philips

2.1.7 Robotic Room SELF

The Robotic Room SELF (Sensorized Environment for LiFe) [9] is developed by AIST, which is started as to combine scientific calculation with daily life to make the calculation more meaningful. Take use of sensor technology, daily actions and trajectories are collected and analyzed in SELF. Their installation places are considered very clever and try best to not disturb normal life of people. For example, keyboard is hidden in the type of pressure sensor bed; microphone is installed inside lamp; computer display is combined with the washstand display. A digital human model is created and acts as the base for the calculation of sensor data which is used to understand human actions and support their life further. In Fig.2.8 the Concept Picture of SELF can be seen and the characteristic of SELF is making no influence to normal life of the user so that they will not act like that in laboratory of some working places.

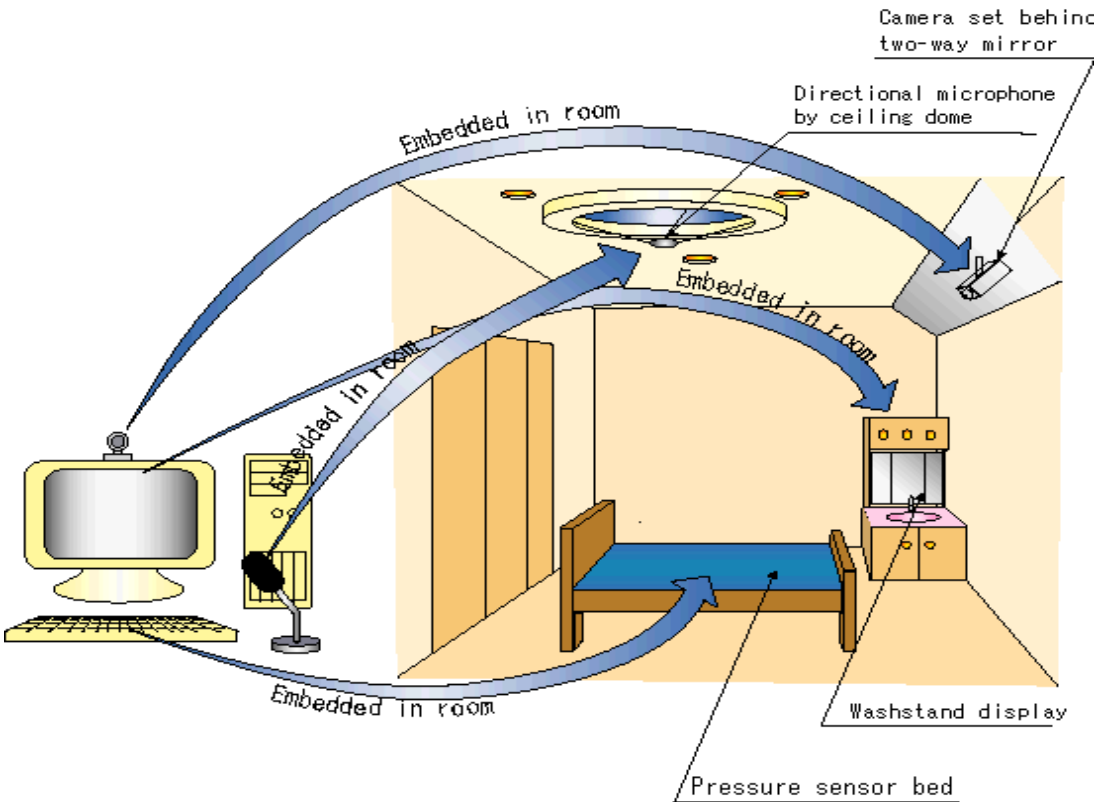


Fig.2. 8 Concept Picture of SELF

2.2 Robot Simulators

There are many robot simulators in the world. If all tools developed by laboratories, institutes and individuals are included, there will be thousands of them. Robot simulator is a simple name, actually any software even if developed for only one part of a robot can be named robot simulator. There are a lot of standards to distinguish these simulators, and the most common ones can be simulators for humanoid robots, simulators for robot mechanical machines, simulators for mobile robots if classified by their target robot types. After entering twenty-first century, the boom of 3D graphics technology has given robot simulators a big favor, thus changed the development of robot simulator a lot. Now started with Microsoft Robotics Studio [21], many institutes and companies have developed common robot simulator platforms and opened to everyone in the world. Like the great development of Information Technology in former years, robotic technology has become a hotter and hotter research field that attracts more and more researchers to devote their wisdom and diligence.

Although there are a lot of useful simulators in the world, we can only use the ones available. Among them, six most popular robot simulators will be introduced as follows.

2.2.1 OpenHRP Series

OpenHRP, which is the abbreviation of “Open Architecture Human-centered Robotics Platform” [22], is a very popular Robot Simulator created by AIST, Japan. Although it focuses on humanoid robots, actually almost all kinds of robots like mobile robots, robot arms can be simulated on this platform.

Accurate dynamics and kinetics model is included in its latest version, OpenHRP3 [23]. Its dynamics model is based on ABA (Articulated Body Algorithm) and ADA (Assembly-Disassembly Algorithm) [24]. The components of OpenHRP include Controller, Dynamics Simulator, Collision Detector, Model Loader, View Simulator and GrxUI. Controller works as a server to control the process of simulation. Dynamics Simulator is in charge of dynamics calculation, integrator calculation. It is the main calculation part in OpenHRP. Collision Detector is used to detect the collision of models in geometrical coordination. All the shapes, structures of models are read by Model Loader, so that they can be combined in the program and used in simulation. The saving medium is as established format text file. View Simulator is used to show the process of simulation. CrxUI is used to coordinate all the different components of OpenHRP. Also, it is the user interface for managing and operating simulation.

By using all these components, serving as independent servers, very complicated simulation can be executed. Therefore it is very suitable for complicated robot simulation like a humanoid and with complicated obstacles. By using its accurate dynamics and kinetics model, the structure of the humanoid can be very well defined, as precise as to every joints of the robot body. The GUI window of OpenHRP can be seen in Fig.2.9.

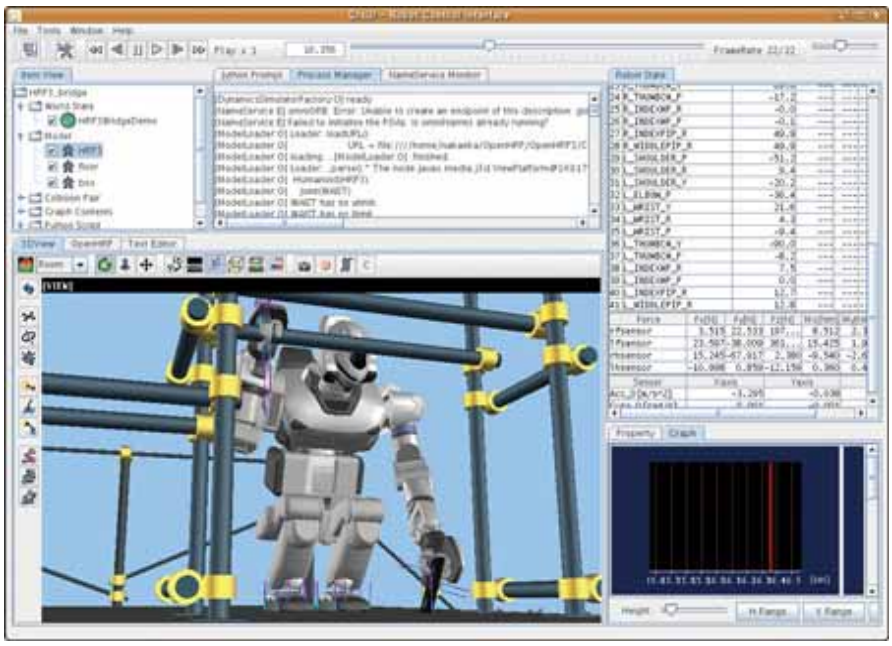
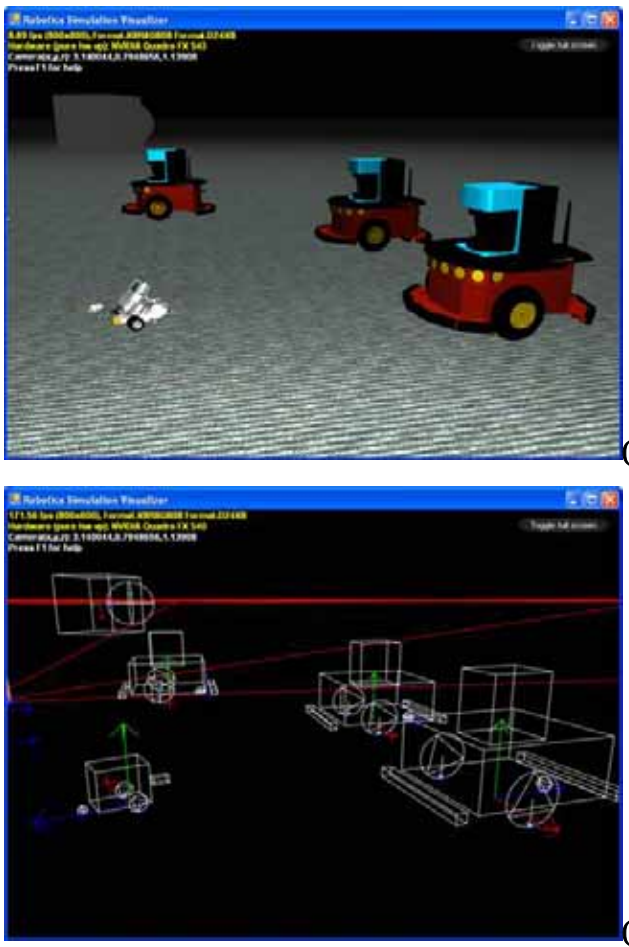


Fig.2. 9 CrxUI, the Sim Management Window of OpenHRP

2.2.2 Microsoft Robotics Studio

The Microsoft Robotics Studio is a windows-based platform for robot simulation. Its aim is academic, developers and hobbyists. Its components include VPL (Visual Programming Language) [25], familiar windows-based interfaces, 3D simulation, abundant models of robot's sensors and actuators and support of most programming languages. Simulation scene can be seen in Fig.2.10. Features of Microsoft Robotics Studio include:

- Simulate robotics applications in 3D physics-based virtual environments
- Interact with robots using Windows or Web-based interfaces
- Non-programmers can create robot applications using a visual programming environment.
- Real-time Monitoring of Robotics Sensors and Response to Motors and Actuators
- Supports both remotely connected (PC-based) and robot-based (autonomous) application scenarios



(a)

(b)

Fig.2. 10 Multiple Robot Simulation in Microsoft Robotics Studio
(a) Simulation Scene; (b) Physical Structure in Collision Detection

2.2.3 Play/Stage/Gazebo

The Player Project (formerly the Player/Stage Project or Player/Stage/Gazebo Project) [26] is a project to create free software for researches in robotics and sensor systems. Its components include the Player network server and Stage and Gazebo robot platform simulators.

Its features include: robot platform independence across a wide variety of hardware, support for a number of programming languages, a minimal and flexible design, support for multiple devices on the same interface, and on-the-fly server configuration.

Player is a network server for robot control. Running on the robot, Player provides a clean and simple interface to the robot's sensors and actuators over the IP network. Client program talks to Player over a TCP socket.

Stage simulates a population of mobile robots, sensors and objects in a two-dimensional bitmapped environment. Stage is designed to support research into multi-agent autonomous systems, so it provides fairly simple, computationally cheap models of lots of devices rather than attempting to emulate any device with great fidelity.

Gazebo is a multi-robot simulator for outdoor environments. Like Stage, it is capable of simulating a population of robots, sensors and objects, but does so in a three-dimensional world. It generates both realistic sensor feedback and physically plausible interactions between objects. Its screenshot can be seen in Fig.2.11. Features include:

- Realistic simulation of rigid-body physics.
- Player compatible: robots and sensors can be controlled through standard Player interfaces.
- Stand-alone operation: programs can interact directly with the simulator.
- Most devices can now be directly controlled / inspected through the simulator GUI.

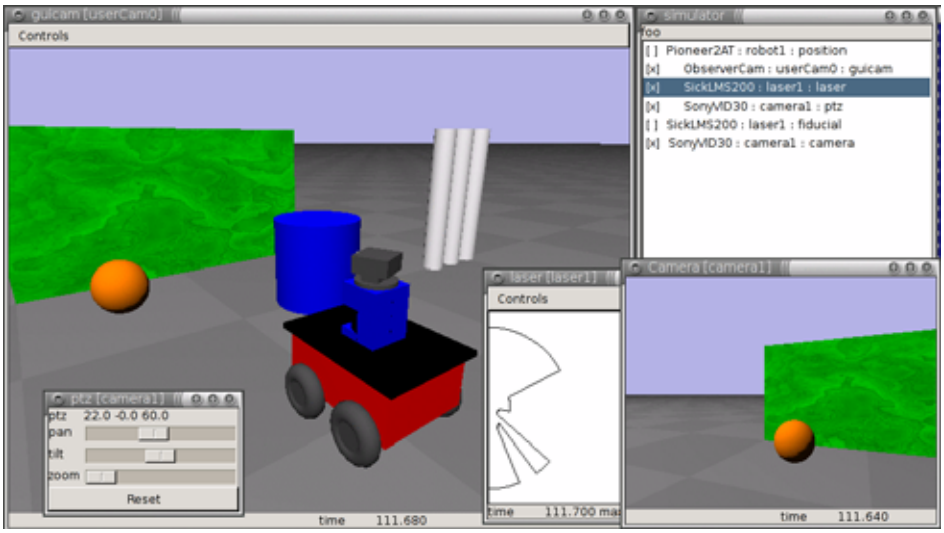


Fig.2. 11 Screenshot of Gazebo with many Control Panels

2.2.4 Simbad 3D Robot Simulator

Simbad [27] is a Java 3D robot simulator for scientific and educational purposes. It is mainly dedicated to researchers/programmers who want a simple basis for studying Situated Artificial Intelligence, Machine Learning, and more generally AI algorithms, in the context of Autonomous Robotics and Autonomous Agents. It is not intended to provide a real world simulation and is kept voluntarily readable and simple. Simbad enables programmers to write their own robot controller, modify the environment and use the available sensors.

Simbad provides the following features: 3D visualisation and sensing, single or multi-robots simulation, vision sensors like color cameras, range sensors like sonars, contact sensors like bumpers and Swing user interface for control. The screenshot of Simbad is shown in Fig.2.12.

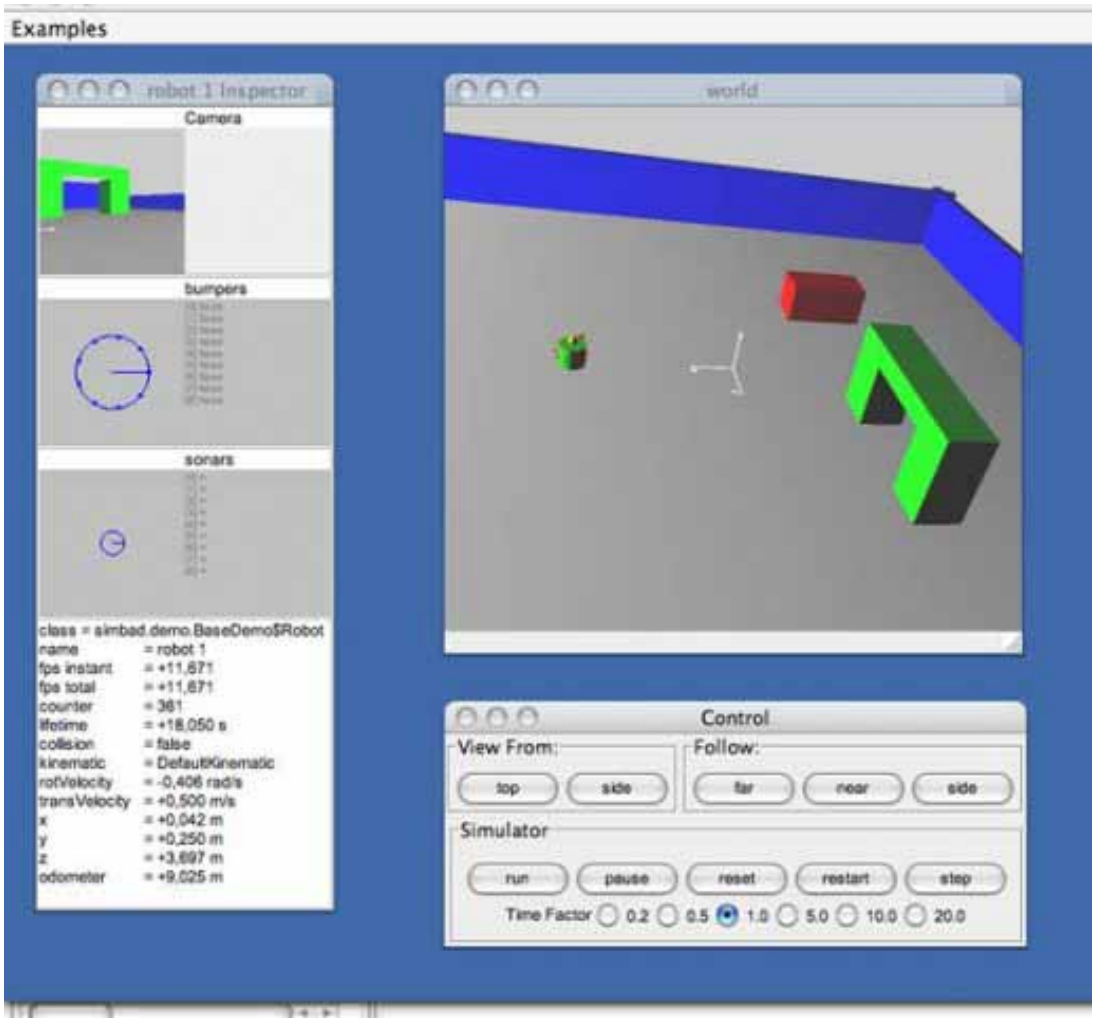


Fig.2. 12 Screenshot of Simbad

2.2.5 MoRoS3D

MoRoS3D [28] is a Java based 3D simulator for mobile robots. This application is able to simulate realistic motion of different wheeled mobile robots including dynamic behavior and collision detection. Typical sensors are also available in order to develop intelligent navigation applications. As this simulator provides CORBA [29] interfaces for every active objects, applications can be written in any language supporting this standard.

The simulator provides the following functionalities: real-time simulation of multiple robots concurrently, 3D real-time visualization of the simulation, user interaction through a GUI, dynamic control of mobile robots, detection of and appropriate reaction to collisions between mobile and fixed objects, simulation of position and distance sensors and CORBA interfaces. The GUI of MoRoS3D can be seen in Fig.2.13. The simulation process is divided into two main steps: the modeling of 3D scenes and robots and the utilization of the modeled objects in the simulator. MoRoS3D allows to place a robot in a 3D environment and to let it interact with that environment in a manner similar to robots situated in the real world. Although the user visualizes the entire surroundings of the robot, the robot software only “sees” the information it collects through its sensors, just like a real robot would do.

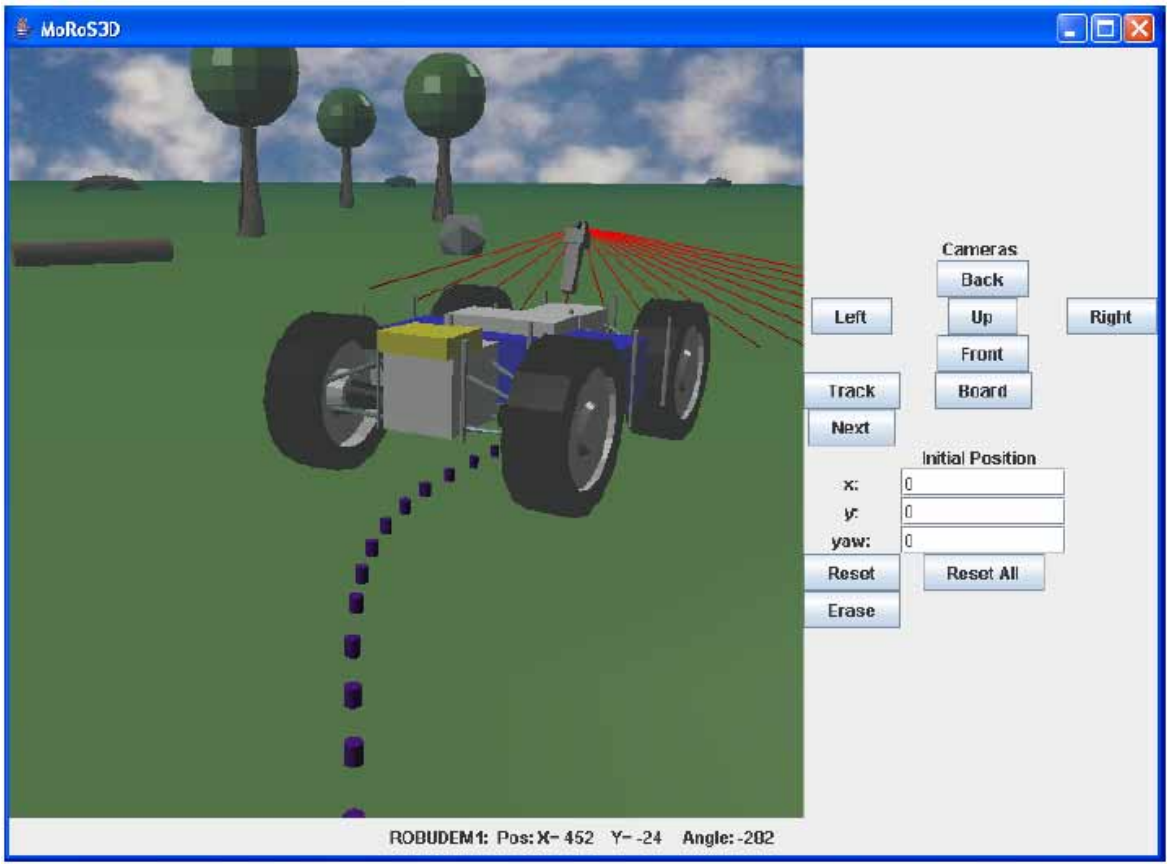


Fig.2. 13 GUI of MoRoS3D

2.2.6 Webots

Webots [30] is a professional robot simulator widely used in academic and education. The Webots project started in 1996, initially developed by Dr. Olivier Michel at the Swiss Federal Institute of Technology (EPFL) in Lausanne, Switzerland.

Webots uses the ODE (Open Dynamics Engine) [31] for detecting of collisions and for simulating the rigid body dynamics. The ODE library allows to accurately simulate the physical properties of objects, such as velocity, inertia, friction etc.

A large collection of robot models comes in the software distribution. These models can be modified whenever needed. In addition it is also possible to build new models from scratch. When designing a robot model, the user specifies both the graphical and the physical properties of the objects. The graphical properties are: the shape, the dimensions, the position and orientation, the colors, the texture etc. of the object. The physical properties are: the mass, the friction factor, the spring and damping constants etc.

Webots includes a set of sensors and actuators frequently used in robotic experiments, e.g. proximity sensors, light sensors, touch sensors, GPS, accelerometers, cameras, emitters and receivers, servo motors (rotational & linear), position and force sensors, LEDs, grippers, gyros, compasses etc.

The robot controller programs can be written in C, C++, Java and Python or interfaced with other languages. The AIBO, Nao (robot) and E-puck robot models can also be programmed with the URBI language (URBI license required). The screenshot of webots can be seen in Fig.2.14.

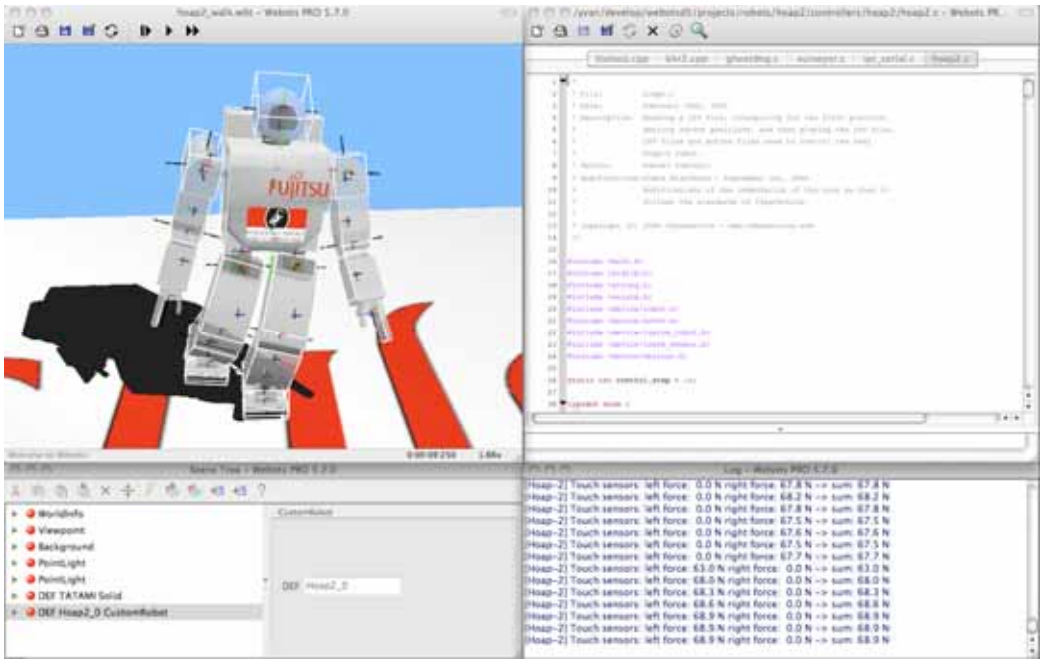


Fig.2. 14 Screenshot of Webots

2.3 Ubiquitous Environment Simulators

Like we discussed in the previous chapter, as a close research field of Intelligent Space, ubiquitous environment as a bright new research field has caught the eyes of many researchers. Among the related researches, there are also many simulators on ubiquitous environment. Here the survey of main ubiquitous environment simulators was made and the introduction will be given.

2.3.1 UbiWise

UbiWise [32] is a simulator for ubiquitous computing developed at Hewlett Packard by Barton et al. The simulator concentrates on computation and communications devices integrated with physical environments or carried by people. It uses a three dimensional model of a physical environment viewed by users on a desktop through two windows. Multiple users can attach to the server to create interactive ubiquitous computing scenarios. UbiWise pays attention on the following aspects:

- To design suitable hardware which fits the ubiquitous computing applications.
- To do experiment with new sensors like building and deploying them.
- To aggregate device functions without connecting real devices.

Main devices can be tested in UbiWise are handheld devices like cameras and PDAs. The screenshot can be seen in Fig.2.15. The device image helps users connect the physical environment view to the device interaction view. The current interface is to adjust the parameters of camera.

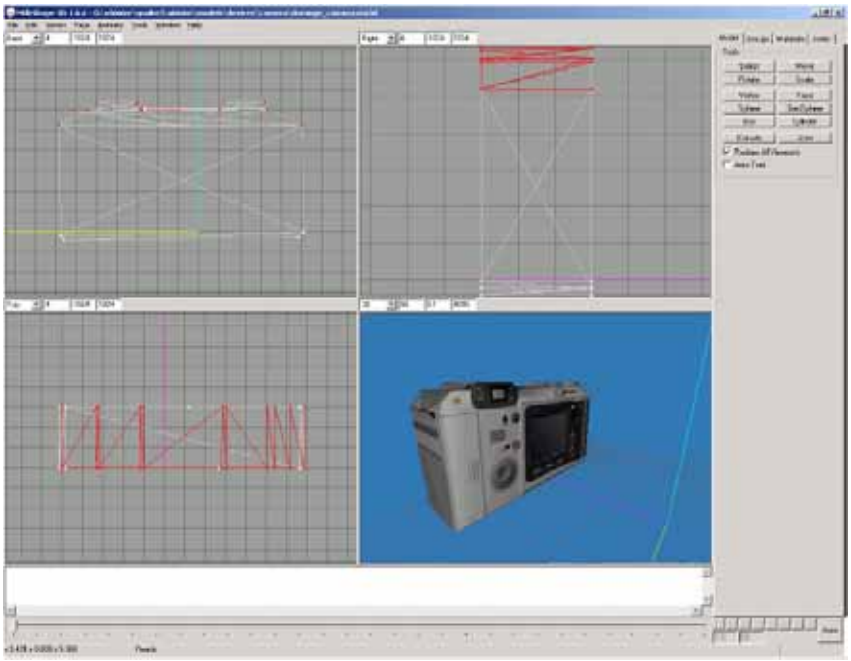


Fig.2. 15 Screenshot of UbiWise Simulator

2.3.2 TATUS

TATUS [33] is a simulator developed to support researchers writing software to control ubiquitous computing environments. A peripheral piece of software-under-test (SUT) connected to the simulator assimilates exported state in order to develop its own representation of the world. Based on the view the SUT holds of the environment, it makes decisions to change the world in reaction to user movements and behavior. The overall effect is to allow the SUT to control the TATUS virtual environment so that the environment behaves intelligently according to the experimental goals of the SUT. The functions which are available are as the following. Fig.2.16 shows the scene in TATUS.

- Allow researchers to connect software-under-test to the simulator.
- Allow researchers to select simulator events of interest.
- Provide researchers with instruction protocol to control actions within the simulator.
- Realistically reflect the current state of ubiquitous computing technology. For example, pressure sensors and RFID tags.

Fig.2. 16 Scene of TATUS Simulator

2.3.3 URSF

Ubiquitous Robot Simulation Framework (URSF) [34] is a simulator framework for simulating ubiquitous computing environments and ubiquitous robots. The simulated world is built by composing a simulation space and placing operational components such as appliances, sensors, persons, robots in it. Each operational component continuously generates context data, which are fed to the plugged platform. This framework can be used to observe how robots interact with environment via ubiquitous network. The ubiquitous robots are like ubiquitous sensors such as pressure sensors that can be in any place of the environment. Furthermore, their number can be very big. In order to execute the simulation fast, 2D

viewer is used in the simulation to give users a top view of the environment. The screenshot of URSF is shown in Fig.2.17.

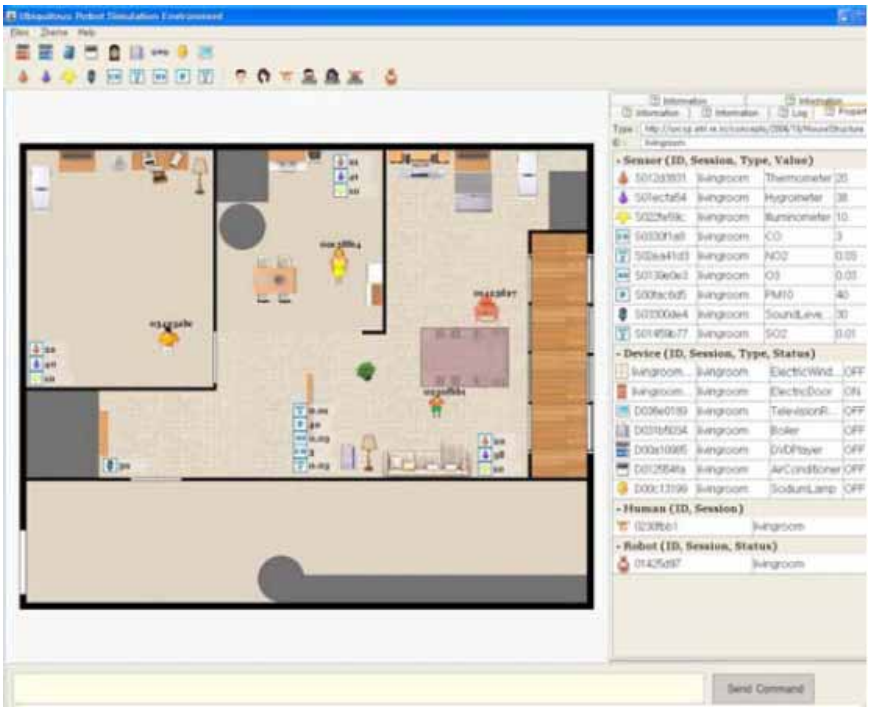


Fig.2. 17 Screenshot of URSF

2.3.4 AE-SIM

AE-SIM [35] is a ubiquitous environment simulator developed by University of Genova. The purpose of AE-SIM is to support the design and simulation of ID and PC agents in the Artificial Ecosystem [36]. It has the following functions:

- Simulate the features of wide, realistic indoor environments like building composed of many floors connected through elevators.
- Test agent software and hardware in order to design new ID and PC agents or add new functionalities to the existing ones.
- Test multiple-agent behaviors.

The system specification is consisted with the following parts:

- World Simulator Server, which simulates all the significant features of the world.
- Visual Interface Client, which provides a visual interface to the use.
- Intelligent Devices Client, which schedules all the simulated ID agents.
- Robot Client, which is one or more applications, each corresponding to a robot acting in the environment. Each robot is constituted by a set of PC agents.

2.4 Our Environment Simulator

In this section, the characteristics of our Intelligent Space are discussed. The motivation and reasons to create our own simulator are given at first. After that the detailed requirements for our simulator are explained. At last, the comparison of the above simulators is given according to the requirements for our simulators.

2.4.1 Motivations

In the recent years, with the fast development of sensor technology, various new sensors have entered the research of Intelligent Space. Therefore nowadays the expense on sensors have decreased a lot and multiple sensor project, sensor fusion have become the new hot topics in this field. However, during the research with new sensors, people find that the sensor arrangement have become a more and more difficult work. Meanwhile, with the development of Intelligent Space research itself, problems during popularization of Intelligent Space should be considered as well. Sensor Arrangement is facing the problems caused by number of sensors [37] and complexity of indoor environment at the same time. Normal user will find the arrangement of sensors difficult if they want to change a specific indoor space into Intelligent Space. Besides of the sensor arrangement, with new low cost sensors, the mobile robot which serves as the typical actuators in Intelligent Space has been updated with new sensors. However, by using new sensors, the control algorithm, how to understand the environment by new sensors have created many new tasks for researchers in Intelligent Space as well. At the moment, any changes of algorithms under development have to be evaluated through experiments. But experiments are not so easy to do especially in some experiments many people are asked to attend, or in some experiments wider place than our laboratory is needed. In a word, experiments have limits in place, time and human resources.

Therefore we are trying to find replacement of experiments in the researches of Intelligent Space. Let me narrate the concept of Intelligent Space. Firstly, real environment is observed by using many types of sensors. Secondly, the sensor data is transferred to DIND [14] processing units. The third, DIND processing units process the sensor data and analyze the activities in the environment. At last, actuators respond to the activities according to the algorithms based on analyzing the sensor data. We find that the part of observing environment, collecting sensor data and controlling actuators can be realized by simulation. The virtual environment can be created for observation. After that, the simulated sensor data is collected and transferred to DIND processing units as the real Intelligent Space does. Besides, actuators can be simulated in virtual environment as well so that the algorithm can be tested in simulator rather than experiments.

2.4.2 Requirements of Simulator

In order to make our environment simulator be able to replace many experiments in developing new algorithms in Intelligent Space, four functions and their responding requirements are summarized as following.

A) Requirements for Indoor Environment Creation Function

In order to build virtual environment for observation, indoor environment creation function is needed in our simulator. In detail, interactive selection of environment elements should be offered to use. Indoor environment elements include walls, floors, ceilings, frames etc.

B) Requirements for Distributed Sensor Arrangement Function

In order to realize the distribution of sensors in Intelligent Space, sensor arrangement function is needed in our simulator. In another word, users should be able to put sensors to any place they want, to add sensors, to delete existed sensors etc. As the solution to sensor arrangement problem for normal users without much experience in sensor arrangement, automatic arrangement function should be realized to make this work easier during the popularization of Intelligent Space. Of course to veteran users, manual arrangement function should be offered to realize their aims.

C) Requirements for Sensor Models in Intelligent Space

In order to realize the simulation of sensor data in Intelligent Space, sensor models in Intelligent Space are needed in our simulator. In detail, there should be sensor models of laser range finder, camera, 3D ranger, ultrasound receiver etc. The creation of sensor model includes the simulation of sensor data which should be the same with real sensor, suitable shapes should be developed to give users a clear look what are sensors look like.

D) Requirements for Interface to DIND Processing Unit

In order to realize the communication with DIND processing unit, communication part is needed in our simulator. In another word, we simulate virtual sensor data and transfer it to outside programs for further use. The interface from simulator to outside program is necessary as well as the receiving APIs of sensor data for outside programs. Our target is to bring minimum changes to the existed program. Therefore the original APIs should be replaced with our new APIs which is communicating with the simulator rather than the real sensors or mobile robots.

2.4.3 Comparison of Simulators

All the above simulators we surveyed are summarized and compared here according to the four functions we require in our simulator. Table.2.2 shows this comparison. Function A~D refer to the four functions we discussed in the last section. "×" means this function is completely unavailable in the simulator. "○" means this function is available in this simulator. "△" in Function C means the simulator does not offer all types of sensors we use in Intelligent Space so that we must develop the corresponding sensor models. From this table, we can find that robot simulators usually do not consider the existence of distributed sensors while ubiquitous environment simulators consider the existence of distributed sensors but offer limited sensor types for use. The reason may be in their different target users. After all, there are no simulators that can be directly used for our research purpose and we are going to develop our own environment simulator.

Table.2. 2 Comparison of Simulators

Simulator Name	Function A	Function B	Function C	Function D
OpenHRP	×	○	△	○
M.R.S	○	×	△	×
Gazebo	×	×	△	○
Simbad	×	×	△	×
MoRoS3D	×	×	△	×
Webots	○	×	△	×
UbiWise	○	○	×	×
Tatus	○	○	△	×
URSF	×	○	×	×
AE-SIM	×	○	△	○

2.5 Summary

In this chapter, the characteristics and specification of our Intelligent Space was introduced. After that many similar simulators were surveyed. At last, the comparisons of all simulators were listed in the forms of tables. The characteristics of our simulators were narrated.

CHAPTER 3

Implementation of Simulator

Before execute simulation to test all kinds of algorithms, design various virtual environments or make a lot of different sensor arrangements, there is a lot of work to construct the infrastructure of the simulator platform. Some functions may be considered things go without saying, but actually creating these functions cost so much time and energy. In this chapter, the three tools of Hashimoto Lab. Environment Simulator: Environment Designer, Sensor Arranger and Simulation Runner will be explained. However the two main tasks: automatic sensor arrangement and mobile robot simulation will be discussed in detail in the next two chapters.

3.1 Overview of Hashimoto Lab. Environment Simulator

The following steps are necessary to change a specific environment into Intelligent Space in the real world. We measure parameters of the environment, apply suitable sensor arranging scheme, look around the changed environment to check if all necessary parts have been covered under the sensor range. This check can be called the first valuation. Then we begin to put actuators into the environment. But before that, we shall develop suitable controlling program for actuators first. This development depends on the way to recognize environment. As we discussed in the introduction, Intelligent Space is named from using many distributed sensors to recognize environment and navigate actuators, so that the recognition of environment depends on the efficiency of sensor arrangement or called sensor placement in many papers [38]. At this point, we develop controlling programs according to the previous observation of the environment and process the first valuation of the sensor arrangement. This is called the first trial of actuator. During actual manipulation of actuators, we make the second valuation of sensor arrangement. At this moment, valuation result is updated because we acquired more information from actual working sensors. After reviewing the valuation, we improve our sensor arrangement first time. Then update the controlling program to fit the latest sensor arrangement, and so on. The above process will be repeated many times until we are content with the final performance of arranged sensors. This is the real work to construct Intelligent Space without simulator.

We can see that the above process is really costly, both in time, money and human resource. It is one of the motivations we propose our Environment Simulator to support it as mentioned before. The simulator is constituted of three tools and the architecture of these tools can be seen in Fig.3.1. Firstly, we construct the specific environment in Environment Designer. After that, we make sensor arrangement in Sensor Arranger based on the created virtual environment, by automatic sensor arrangement or manually. This process is the first valuation of the arrangement in simulator. Then we

switch on Simulation Runner to start the whole system which means we put actuators into the created Intelligent Space and activate all distributed sensors. We make the rest valuation and update any part if necessary. For example, if we need to change the parameters of the sensor, or any other change to the arrangement, we turn to the Sensor Arranger. If the outside program needs to be revised, we update the program, change parameters of robot, strategies and so on. In a word, we finish all the improvement and revision on the computer until we gain the final version of Intelligent Space. Hence we finish the construction of a useful Intelligent Space and now we can begin to build it in real world. Thanks to computer, we surely save a lot of cost, mostly in human resource and time.

In Fig.3.1, blue rectangles denote three developing tools, red rectangle with broken lines denotes the Intelligent Space we are going to construct and the green ellipse denotes controlling program outside. Single direction arrow lines denote operating sequence and double direction arrow lines denote mutual influence as improvement and update.

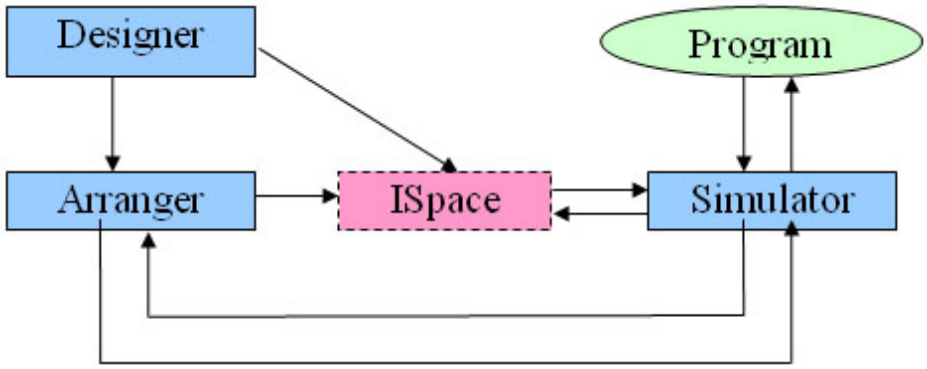


Fig.3. 1 Overview of Hashimoto Lab. Environment Simulator

3.2 Developing Guidelines of Environment Simulator

As an overview, in order to realize the proper and accurate simulation of the main tasks in our Intelligent Space, we have the following developing guidelines for the three tools separately.

Developing Guidelines of Environment Designer

1) Offer simple GUIs to realize all operations.
2) Move environment elements to the suitable places easily and accurately.
3) Offer various elements with different properties.
4) Offer convenient view mode for management placement of environment elements.
5) Offer first-people view to see completion out appearance of the virtual environment.
6) Offer humanistic design and friendly suggestion.
7) Save into files and share with other tools.

Developing Guidelines of Sensor Arranger

1) Offer Hashimoto Lab. specific sensors and facilities to construct the sensor arranging scenario.
2) Offer multiple view modes for better observation and validation.
3) Arrange objects easily and properly.
4) Offer function to change main parameters of sensors.
5) Be able to show sensor data for reference.
6) Offer function to load created environment file.
7) Offer friendly suggestion to help users develop good sensor arrangement.
8) Saved into files for further use.

Developing Guidelines of Simulation Runner

1) Offer functions to load created environment file and sensor arrangement file.
2) Offer interfaces and protocols to communicate with outside program.
3) Offer user-oriented suggestion system with friendly human interface.
4) Apply basic physical principles such as kinematics.
5) Apply collision detection for mutual influence of robot and environment.
6) Offer human model to simulate simple human activities.
7) Offer speedy simulation in best effort.
8) Save important simulation data for further comparison and statistics.

3.3 Environment Designer

This tool is used to create specific environment for user. Because the target environment of Environment Simulator is indoor environment, the useful environment elements can be restricted into a relatively narrow range. Before explain implementation of main parts of Environment Designer tool, its specification is listed in the Table.3.1 and its sample figure is shown in Fig.3.2.

In Fig.3.2, the layout of Environment Designer can be summarized into four sub-windows, neat GUI and menu system. As the figure shows, the left top sub-window is offering bird view or called top view of the whole virtual environment. As the sample figure is the design of a square open room, its simple abstraction outlines can be seen on top of the grids beneath. Grids represent unit length of the virtual space, which is all one meter long for reference. On the left of this sub-window, there is the scalar for zoom function and the current zoom scale is 5 in the middle of limit.

The right sub-window is the first-person view of the virtual environment and currently it supports omni-directional focus which means the focus can all around the camera. This operation is realized by left dragging with mouse, basically changing the focus by reading new x, y value of dragging action. Right dragging with mouse controls the distance to focus, therefore fulfils arbitrary movement in

virtual world. One limit here needs attention is that the movement is can not go under the ground or take off the ground but only move fixed in height at 1.7 meter which is the eye height of normal male adult. This design has considered the necessity of viewing virtual environment and decided that normal human view is the best one for observation in Environment Designer. Because the view camera is free of movement, the recognition of direction becomes difficult for environment design. Therefore the position of camera is represented as a red point in the left sub-window. In order to distinguish different elements, outlines are high-lighted in this window. Yellow lines represent elements' outlines while green lines represent the current chosen element. In this sample figure, the chosen element is a ceiling frame which is often used as the infrastructure to install distributed cameras or ultrasonic positioning system.

Table.3. 1 Specification of Environment Designer

Windows Layout	4 Function Windows: 1) bird view with zoom scalar 2) 3D first-person view 3) movement and operation controller 4) item selection menu
Graphic User Interface	All operations can be executed by mouse. Offer necessary word suggestions for operation and data for reference. Mouse clicking operations include left clicking for selection, right clicking for dragging and double clicking for object operation.
View Engineer	1) Bird view mode with zoom function, move in four directions: X, -X, Z and -Z. 2) First-person 3D view, all operated by mouse. Left drag is used to move the camera focus; Right drag is used to adjust distance.
Menu System	1) All environment elements are classified into 4 types: floor, wall, infra and other items. 2) 2) Sample figure for user's reference. Simple click to select target element and size is adjustable.
Movement Control	1) 4-direction movement in x-z coordination frame. 2) Rotation and partly height control 3) Fast Construct Mode of environments using mouse to create close indoor environments
File Manipulation	Save and load environment files

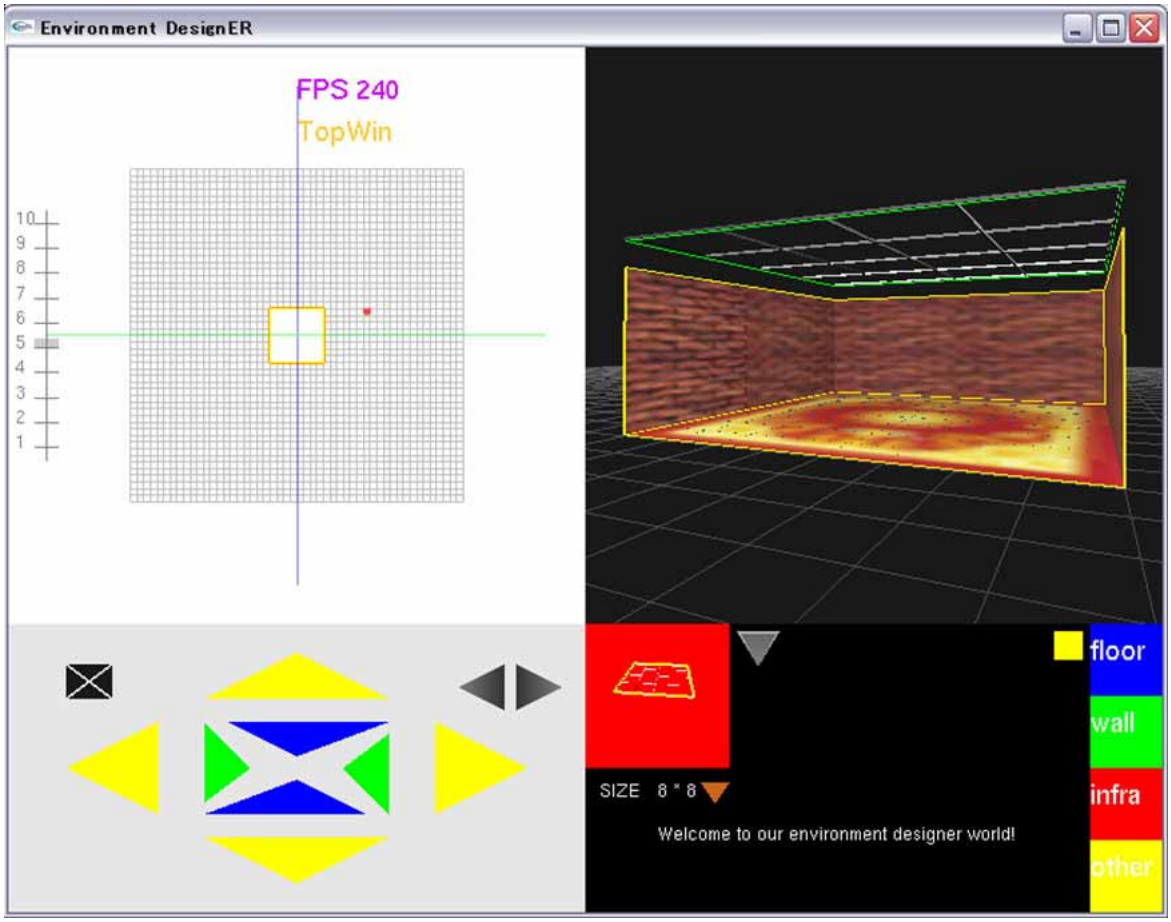


Fig.3. 2 Screenshot of Environment Designer

The left lower sub-window includes movement controller and other function buttons. Basically the movement of environment elements includes rotation and translation. Rotation can be counter-wise or counter counter-wise and translation is simplified into going forward or backward. The outside four yellow arrows realize these operations. The inner arrows are clicked for adjusting height for some elements like the ceiling frames which is high-lighted in green in the first-person view sub-window. Besides, the left top black 'X' button is used to delete environment elements. While creation is very important for environment design, deletion of unnecessary elements is also helpful in arrangement. This process is realized by double clicking target element in the first-person view to select it and then click 'X' button the element can be annihilated. All the process can be done by moving and clicking mouse. The right two black arrows are used to manage files. The left one is for loading default environment file and the right one is for saving environment design into default environment file.

The last sub-window may be the most complicated one. Firstly, there are four tablets on the right to divide the environment elements into four types: floor, wall, infra and other vertically. The sample figure of an element is shown on the left of the sub-window. The background color represents the type

of current showing element and the main outlines are high-lighted for better recognition. More elements are available by click the right down-direction arrow to go page-down. The size selection is available below the figure of current element. The current size is “8 * 8” which means 8 meter wide and 8 meter long object. The creation of element is to click the figure of target element, and hence the target element will appear in the middle of the virtual environment for operations. The small yellow cube near type tablets is the switch for high-light selection. If user does not want to see the outlines of environment elements, switching off is possible.

Since the outside part, layout and GUI have been summarized above, the inside part will be explained in the following sections.

3.3.1 Environment Element Operation

The environment elements are divided into four big types: floor, wall, infrastructure and others. They are main elements in indoor space. Floor works as specific zone for robot to pass, it has attribution like friction to make robot move slower. Wall is the basic barrier to separate indoor space, to create room, maze or any other structure for mobile robot simulation. Infrastructure includes ceiling frame to install sensors, pillar of building and so on. All other elements belong to the other type. By using this classification, most indoor environment can be created by combining these environment elements like Fig.3.3 shows. This is the created environment which has been loaded in Sensor Arranger for sensor arrangement. The left small window is the top view of the created environment and the right small window is the position of sensors and actuators. The right big window shows the scene of environment through omni-directional view which will be discussed in latter section.

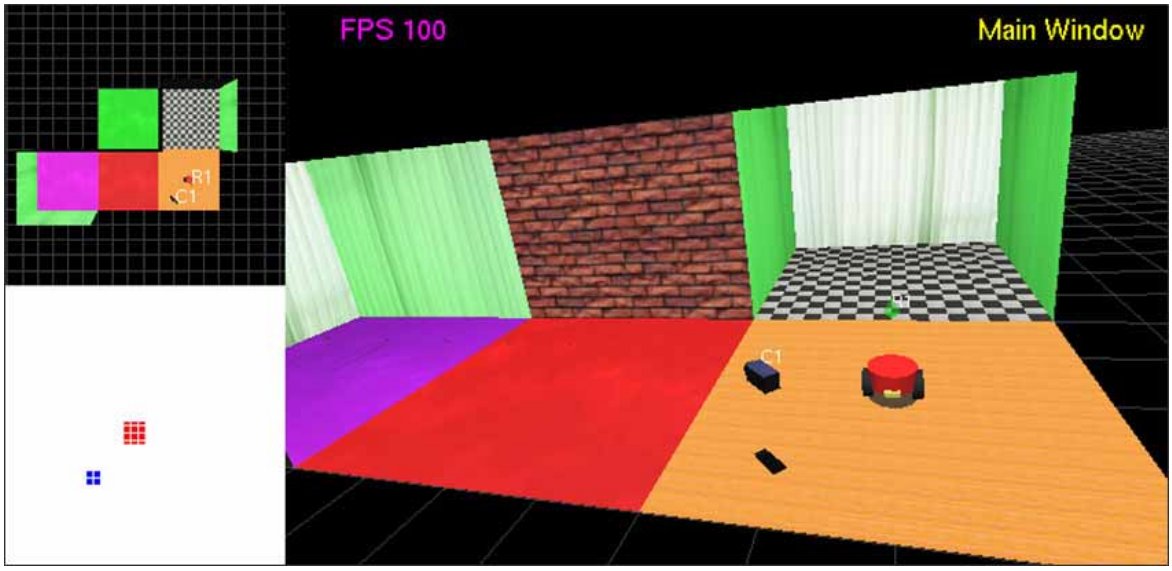


Fig.3. 3 Combination of Environment Elements

The creation of environment elements has the following steps:

1) Click the target element type in item sub-window.
2) Use page down button to browse currently available elements in the chosen type.
3) After decided the target element, choose the size of this element.
4) After decided the size, click the figure of this element to create the element. The created element will be in the middle of virtual space and its direction is south facing.

The movement of environment elements has the following steps:

1) Click the movement control arrows in controller sub-window. Click left or right arrow to rotate the element.
2) Click up or down arrow to move the element along its facing direction.
3) Repeat the above steps to fit the element into expected place.

The deletion of environment elements has the following steps:

1) Drag mouse cursor to find the target element in first-person 3D sub-window.
2) Double click the target element.
3) After the element is high-lighted in green color, it means this element is chosen as the current element.
4) Click the 'X' button in controller sub-window to delete the element.

Beside of the above accurate moving mode, fast construct mode is offered to produce anomaly shaped indoor environment conveniently. The scene of environment modeling using fast construct mode can be seen in the Fig.3.4. In this mode, the steps of creation of environment are as follows:

1) Click the draw button in item-window to start fast construct mode.
2) Begin drawing in the bird-view-window with mouse' left click.
3) Twice Clicks link a line and the line represents a wall in the virtual space.
4) Continue drawing till the expected shape is finished.
5) Click the stop button in item-window to stop fast construct mode and the expected shape of environment is done in very short time.

The merit of fast construct mode is that it is faster and more convenient to create environment elements than the accurate moving method. But as its name implies, this mode draws fast but not accurately. If user wants accurate location of every environment element, the accurate moving mode is suggested. Anyway, since these two methods do not conflict with each other, user can combine these two methods to create discretional indoor environment as he likes.

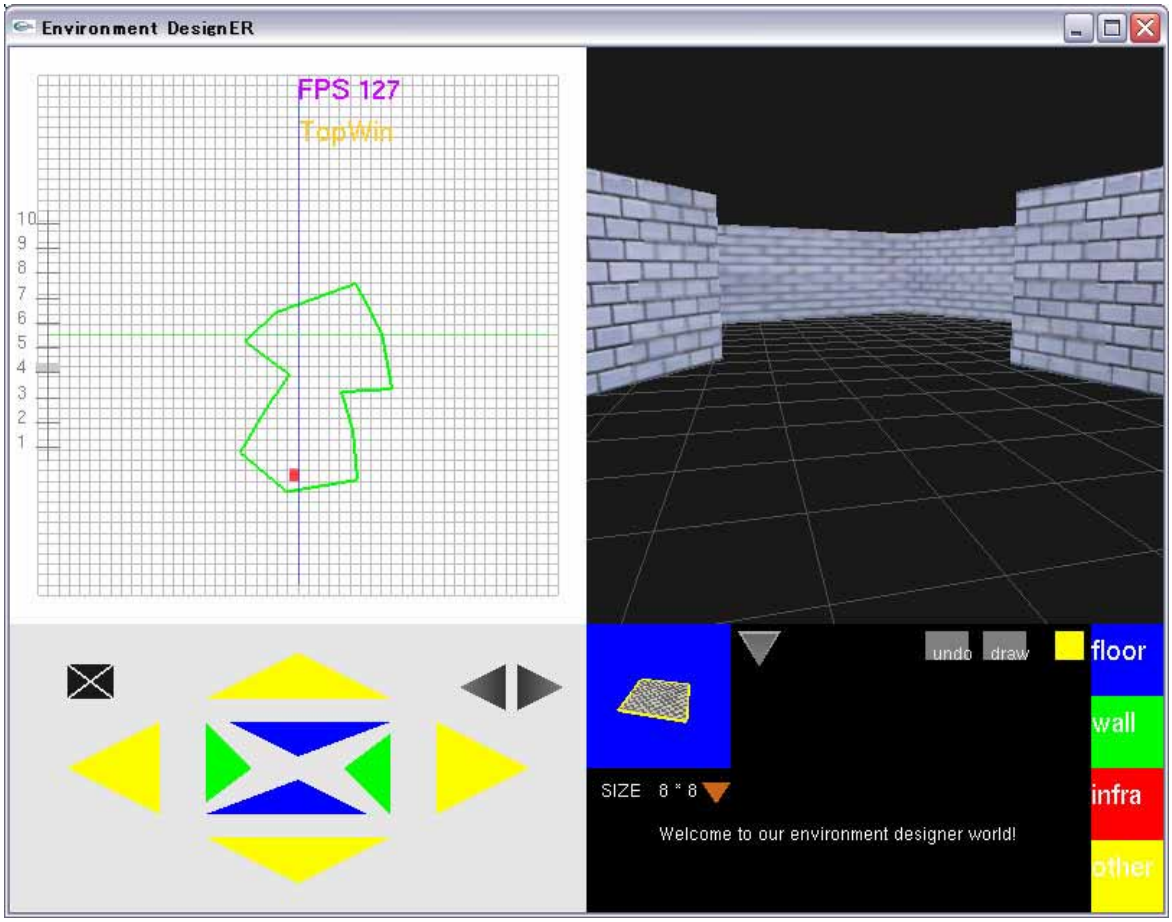


Fig.3. 4 Sample of Fast Construct Mode in Environment Designer

3.3.2 Data Management

The attributions of environment elements are abstracted and managed in structure type. The current data structure of these environment elements are shown in Fig.3.5.

From the figure, we can see the data structure of unitISpace, unitFloor and unitWall is almost the same. Actually the included attributions are the most basic ones for creating these environment elements. The array named pos has two elements, xPos and zPos which are the coordination values on the floor in Environment Designer. The unitFence object has different attributions from other objects. It is the data structure for fast construct mode environment creation. Because the position of walls can be described by two points, aX, aZ and bX, bZ are used to record all these value. Furthermore, unitInfra can be used to describe more complicated objects because the height of the objects is also recorded. For example, the installation frame for distributed cameras and ultrasonic sensors can be described by this structure.

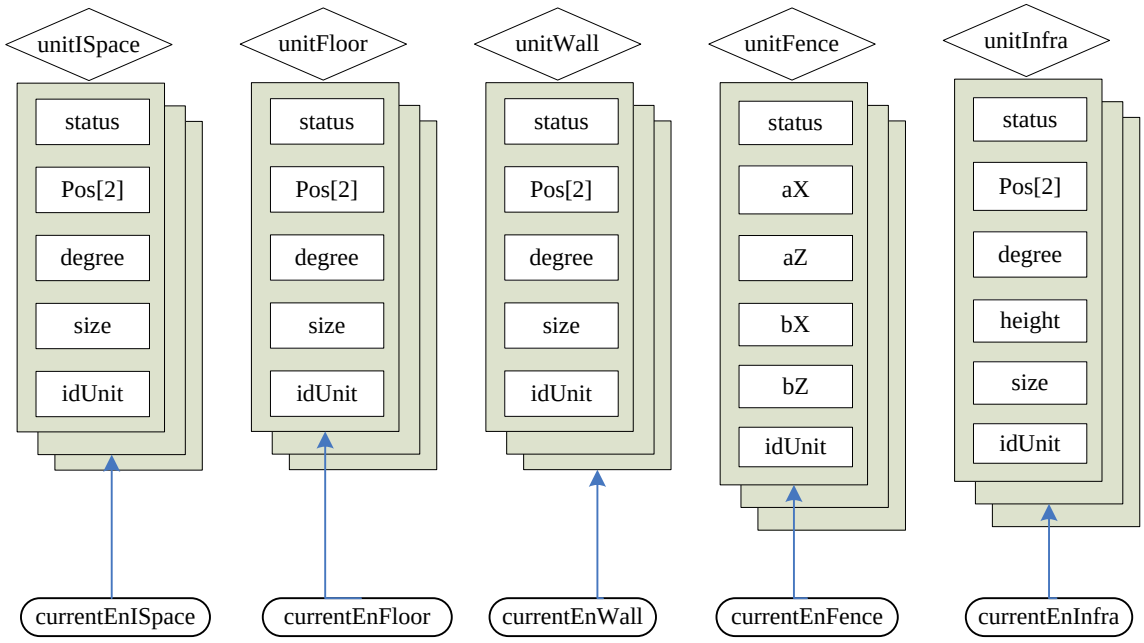


Fig.3. 5 Data Structure of Environment Elements

Every structure has their corresponding current pointer which is pointed to the selected environment element at most time. This pointer is useful when deletion of created environment element is asked to do.

3.3.3 File Exchange

Because the created environment design is used in Sensor Arranger and Simulation Runner, all data is loaded in other two programs through file exchange. The extension file name is ".env" and its structure is shown in Fig.3.6.

The way for us to communicate files between different programs is to include all necessary modeling functions in every program. Therefore the information inside the file only includes most basic and simple form of data. That we can realize this merit is due to that the simple 3D models created in our programs. Whether or not, delicate 3D model is attractive but it does not have very much meaning except the good look. Actually good look does not mean good physical model because in most cases of simulator, physical models are far simpler than the look. Therefore the simplest but accurate 3D model was initially decided to create in the beginning of our program. As that can be seen in the figure of an sample of file format in Environment Designer, only basic information like position, direction and any other necessary data is recorded and communicate with other programs.

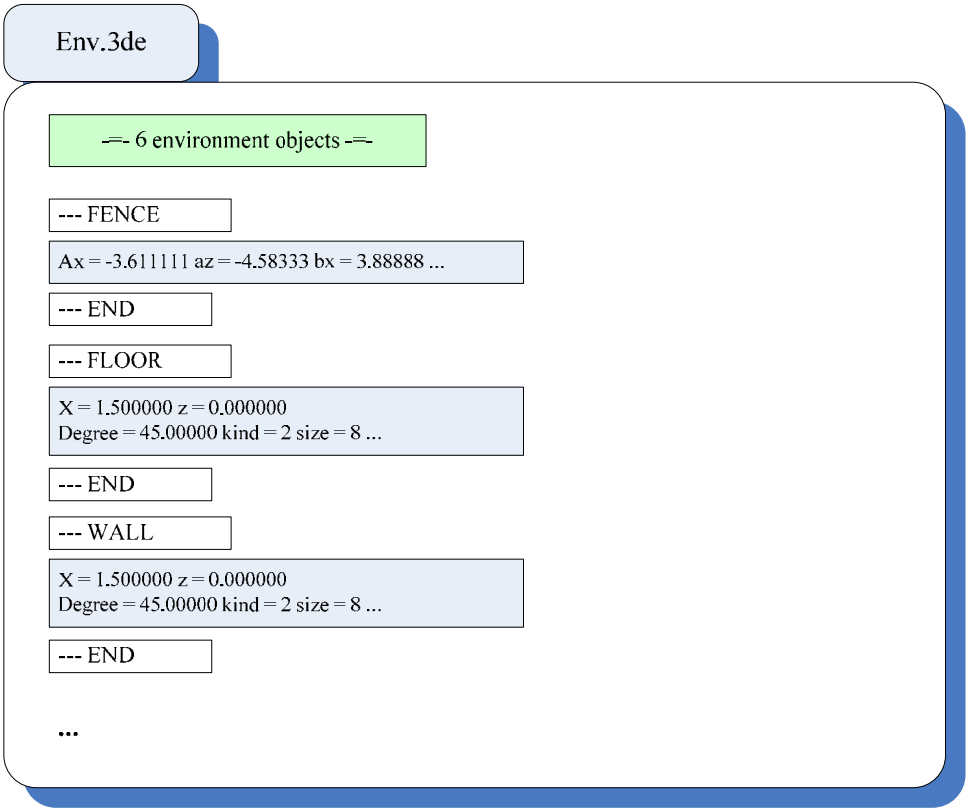


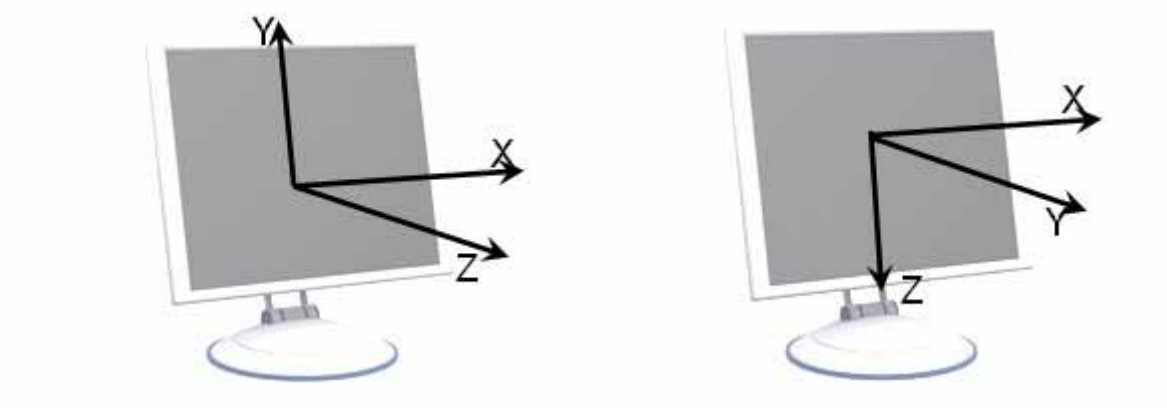
Fig.3. 6 File Format of Environment Designer

3.3.4 View Mode

The view mode used in Environment Designer is bird view mode and first-person view mode. By using only mouse, the view can be moved and zoomed freely. Here the realization of these two view mode will be explained.

i) Bird View Mode

Bird View simulates the view of bird gliding over the indoor environment below. Different with real bird view, the normal of view plane is perpendicular to the horizontal plane. The movement of camera is limited to three directions: x, z and vertical to floor. By using the bird view mode, user can see layout of the created environment clearly. Fig.3.7 shows the transform from normal view to bird view on the screen. Before transform, the Z direction is point to the outside of screen and after transform, the Y direction takes place of Z direction therefore the view is now facing the horizontal plane. Therefore to user, the top view can be available during the construction of indoor environment.



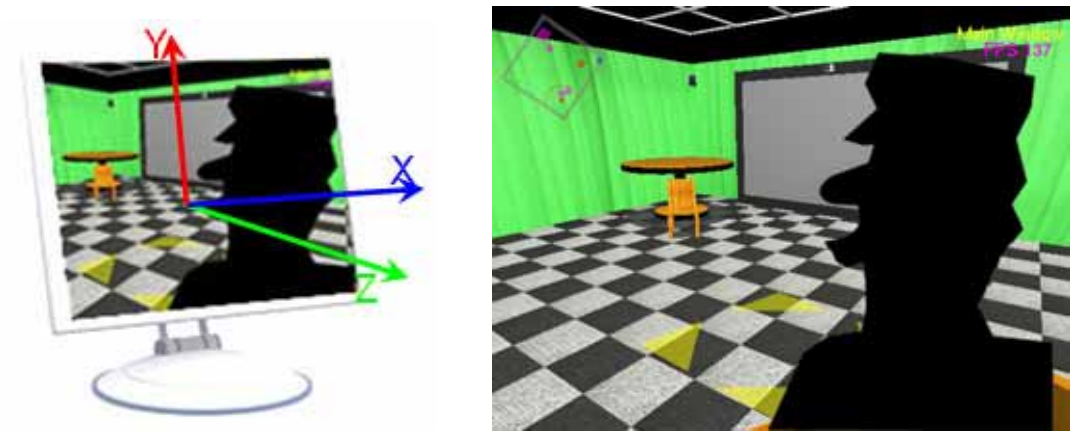
(a) (b)

Fig.3. 7 Transform from Normal View to Bird View

(a) Origin Coordination System; (b) Brid View Mode in Simulator

ii) First-person View Mode

First-person view mode simulates the view of human's eyes. Therefore it has a height above the floor equals to about the height from floor to the eye of a normal person. By moving forward or backward, people going to anywhere on the floor only limited by the exact same height. The merit of first-person view is that it can look at any points in the space. In order to realize that, mouse action is used to move the focus of human's eye. In our tools, the change of focus is by dragging with left button of mouse and the going forward or backward depends on the direction of dragging with right button. First-person view is freer than bird-view because the focus can be changed easily so that almost every aspect of the objects in the space can be observed easily despites the height is limited. Fig.3.8 shows this concept.



(a) (b)

Fig.3. 8 Concept of First-person View Mode in Simulator

(a) Coordination System of First-person View Mode; (b) Scene of First-person View in Simulator

3.4 Sensor Arranger

Sensor Arranger has two main tasks. One is to offer people a variety of sensors and facilities to distribute into the environment. The environment can be the default one which is like our Intelligent Space, or it can be created by Environment Designer first and read into Sensor Arranger. The most objects can be arranged to any place possible as user wishes. Because the arrangement of laser range finder is not so easy to beginners, automatic arrangement function has been developed and helps user a lot. The discussion of automatic arrangement mode will be made in Chapter 4. The other main task is to make sensor arrangement for simulation in Simulation Runner, in which not only the specifications of robots but also the specifications of sensors and facilities will be decided in Sensor Arranger. All these information will be saved as file and loaded by Simulation Runner to use.

In this section, the Sensor Arranger will be introduced in 5 parts: specification, layout and GUI, sensor model, arrangement method, used data structure and some special points.

3.4.1 Specification of Sensor Arranger

The specification of Sensor Arranger is listed in the Table.3.2 in seven aspects. Actually Sensor Arranger is the most complicated tool among the three tools of environment simulator.

Table.3. 2 Specification of Sensor Arranger

Windows Arrangement	5 Windows: bird view, 3D free view, sensor data display, movement controller and object menu
User Interface	Most operations can be done by mouse; Use key board input to change parameters
View Engineer	Bird view for planning; Omni-direction 3D view; Sensor data display
Sensor Type	4 sensors: laser range finder, 3D ranger, camera, ultrasound receiver
Parameters and Menu System	Objects put in pages; Change parameters of sensor in object menu
Movement Control	Object moves in facing direction; Sensors can change their height and pose
File Manipulation	Save function is available

3.4.2 Layout and GUI

In Sensor Arranger, all operations can be finished with mouse actions conveniently. This is because we support users with abundant GUIs. The GUIs are constituted of menus, buttons, feedback figures as which can be seen in Fig.3.9, buttons are used to enable actions like page-down, saving file and so on. Combination of buttons is used here as a controller to move objects. Menus are used to show word prompt and activate events like creating objects. Feedback figures can activate events like choosing expected objects in Sensor Arranger. These figures are highlighted after being clicked to give user clear message of what is happening at the moment.

Take Fig.3.9 as an example. The left four figures in different color are so-called feedback figure that can be clicked to choose their corresponding selection of parameters and arrangement items. The blue figure is laser range finder, the red one is ultrasonic receiver, the green one is camera and the yellow one is pioneer mobile robot. The current sub-menu is of laser range finder and we can see that there are several items to choose. The upper half is called parameter adjustment bar while the lower half is the management bar. In parameter adjustment bar the current page is to adjust specification of the laser range finder. There are three main parameters can be adjusted for the current chosen sensor: effective distance which is 5 meter at the moment, effective range which is 180 degree front of the sensor and resolution is 1 degree. Beside of the parameters, the control of the chosen laser range finder is available. “Show scan” make the laser beam visible. “Scan Once” let the sensor scan once of the environment around. “Keep scan” make the sensor scan continuously. Other settings can be seen in the next page named “ARRANGE”. In the lower bar, red little diamonds means sensors which are able to be created but not being used yet. Meanwhile, blue little diamonds means the created sensors. The blue frame on the diamond means the created sensor is chosen now and in this status all adjustment or scanning control is available. The right part of the GUI window is the movement controller of objects. Up arrow makes sensor move forward while down arrow does the opposite. Left arrow makes sensor rotate left while right one does the opposite. These four arrows make the movement of objects on fixed height possible. Besides, there are other four little arrows inside the outside ones. The blue arrows control the height of sensor and the green arrows adjust the tilt of sensor which is useful when the sensor is used not in horizontal direction. Other than the eight control arrows, the pink down triangle serves as page-down function to browse all items. Other two triangles give user file operation function and view mode change function.

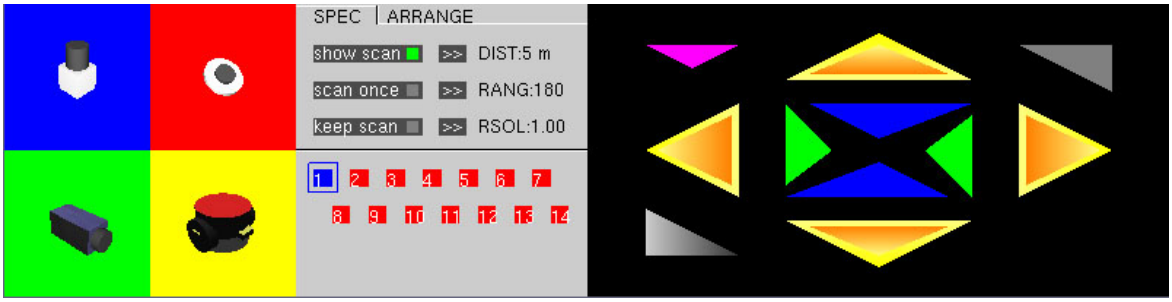


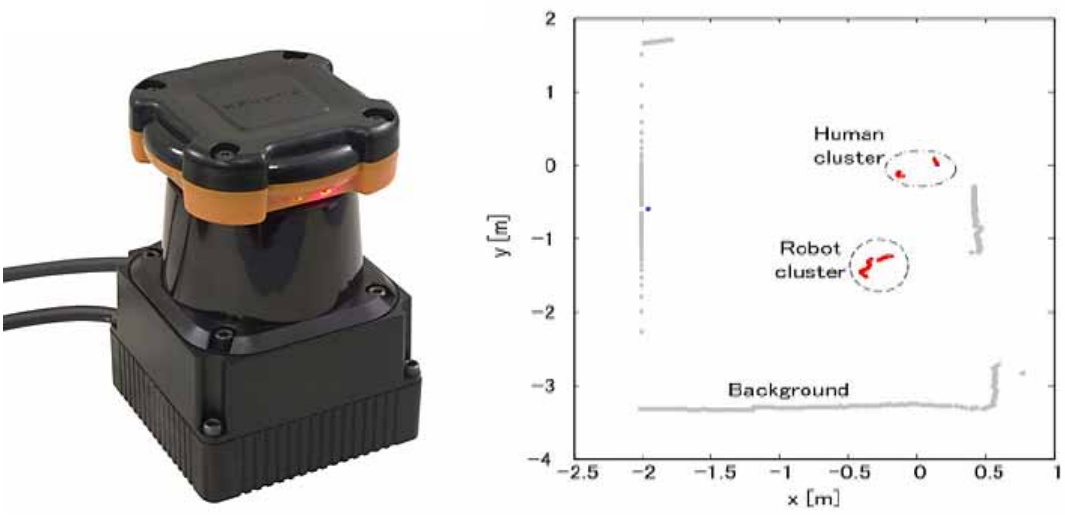
Fig.3. 9 User Interface in Sensor Arranger

3.4.3 Sensor Model

There are four main sensors in Sensor Arranger. Here their properties and sensor model in Sensor Arranger will be discussed in detail separately.

i) Laser Range Finder

Laser range finder may be the most widely used sensor in robotic research. It emits laser beam to detect obstacles around itself based on the principle of TOF (Time Of Flight) [39]. We have adjustable parameters for our laser range finder to have different capabilities. Fig.3.10 shows s sample of laser range finder produced by Hokuyo Corp. and the right figure is the obstacle map created by the sensor data acquired.



(a) (b)

Fig.3. 10 Laser Range Finder

(a) Laser Range Finder UTM-30LX by HOKUYO; (b) Example of scanning result

In real world, the sensor emits laser beam to detect distance of obstacles by calculating the flight time when reflects from obstacles. After the laser beam turns around, we get the distance information in polar coordinates. Apply conversion from polar to orthogonal coordinates to acquire the map of obstacles around the sensor. In kinematics, TOF is the duration in which a projectile is travelling through the air. Given the initial velocity u of the particle (beam particle in laser range finder for example), the downward acceleration a, and the projectile's angle of projection θ (measured relative to the horizontal), the relation can be represented by the following equation.

$$s = vt - \frac{1}{2}at^2 \tag{3.1}$$

Results in the following equation

$$t = \frac{2u\sin\theta}{a} \tag{3.2}$$

Therefore the time of flight can be calculated.

We simulated the above process in our programs. After choosing parameters like range and degree, we make the view of sensor rotates but not the sensor itself. This is because we will use the so-called z-buffer of computer graphics technology. Z-buffer means the depth of one pixel in one view of the scene. We read z-buffer value from the center of the frame while the view is rotating like scanning of real sensor. Therefore we can get the distance information around the laser range finder. After that the data is converted from polar coordinates to orthogonal coordinates to acquire the map of obstacles in the virtual environment.

For usual 2D scanning of laser range finder the distance data can be got easily because there is no consideration of the pose of sensor. Here the pose of sensor means the tilt of sensor like the Fig.3.11 shows. In a word, the sensor can be put at any place in the space which means the position of sensor can be changed as user needs. Despite of the usual floor placement, the sensor should be able to scan on some height. For that reason, the height can be chosen by user. Different from traditional use as scanning in the horizontal direction, the pose of sensor includes tilt which means scanning upwards or downwards. Therefore the coordination transform include more operation than traditional horizontal scanning.

In Fig.3.11 we can see the coordination transform step by step. Firstly, the sensor rotates around y direction by m degree. Then it tilts up by t degree. The scanning plane is represented by yellow semicircle.

In Fig.3.12, the coordination of omni-directional laser range finder scanning can be seen in the form of one laser beam. After the sensor's rotation and tilt, the scanning is consisted of -90~90 degree scanning laser beams. Follow the above algorithm to make the scanning model of laser range finder, the coordination transform should be reached to every laser beam to get distance information.

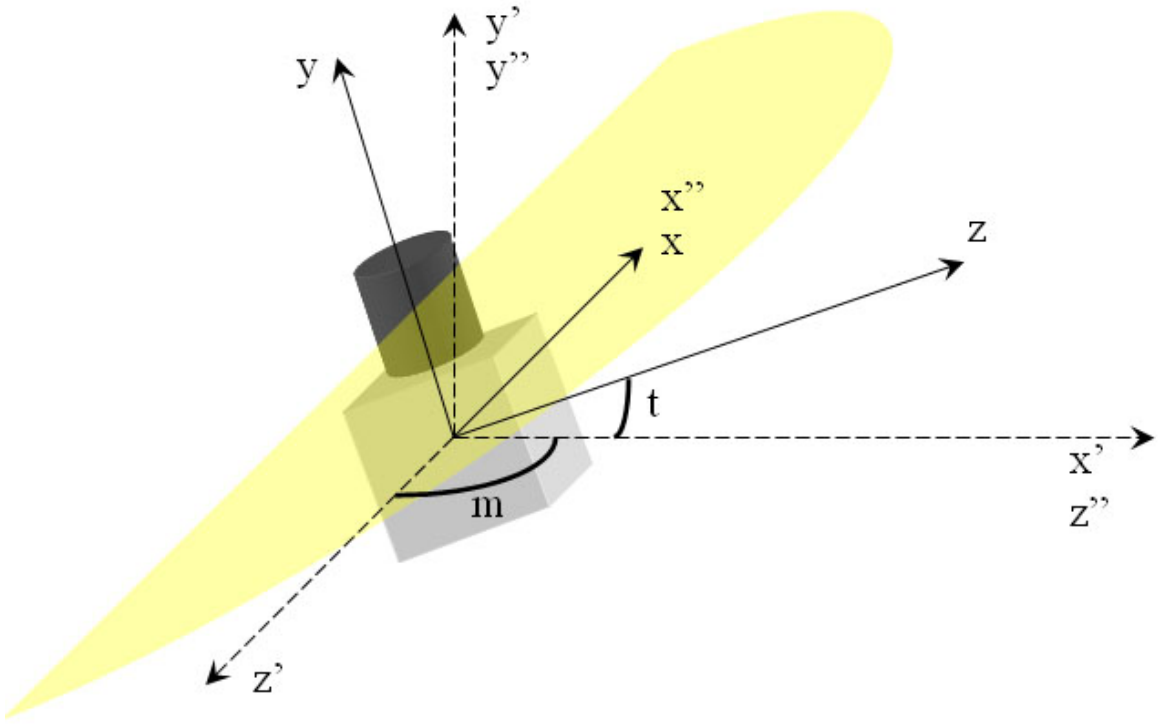


Fig.3. 11 Concept of Omni-direction Laser Range Finder Scanning

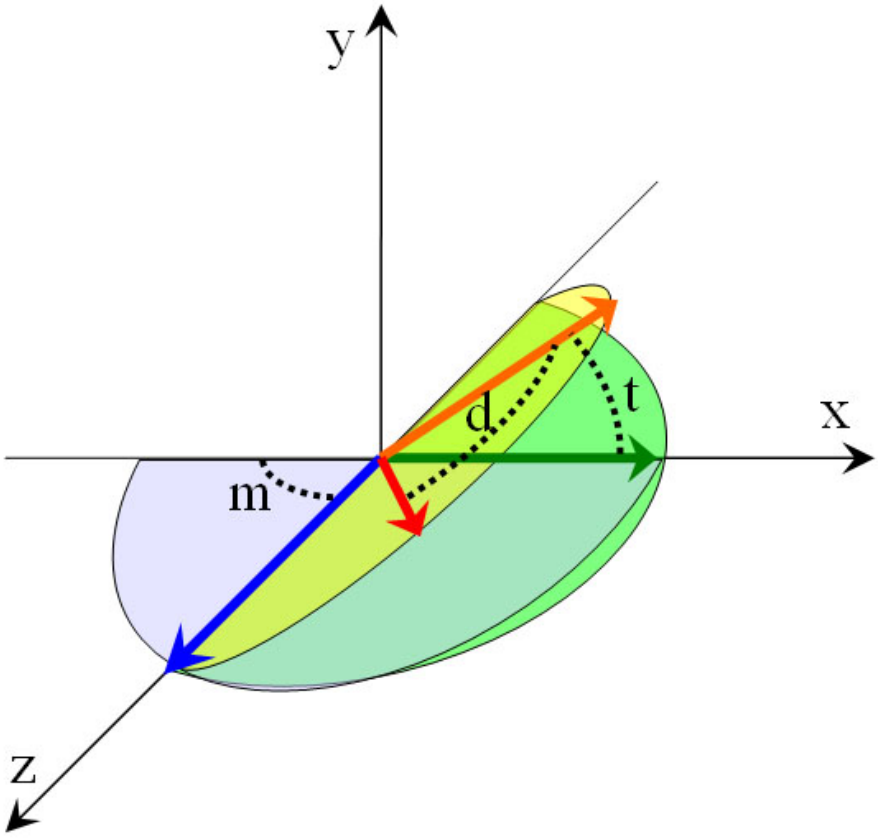


Fig.3. 12 Coordination System of Omni-direction Laser Range Finder Scanning

By using matrix, the above coordination transform can be arranged into the following form.

$$\begin{bmatrix} \cos m & 0 & \sin m \\ 0 & 1 & 0 \\ -\sin m & 0 & \cos m \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos t & \sin t \\ 0 & -\sin t & \cos t \end{bmatrix} \begin{bmatrix} \cos d & 0 & \sin d \\ 0 & 1 & 0 \\ -\sin d & 0 & \cos d \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.3}$$

$$= \begin{bmatrix} \cos m & 0 & \sin m \\ 0 & 1 & 0 \\ -\sin m & 0 & \cos m \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos t & \sin t \\ 0 & -\sin t & \cos t \end{bmatrix} \begin{bmatrix} \sin d \\ 0 \\ \cos d \end{bmatrix} \tag{3.4}$$

$$= \begin{bmatrix} \cos m & 0 & \sin m \\ 0 & 1 & 0 \\ -\sin m & 0 & \cos m \end{bmatrix} \begin{bmatrix} \sin d \\ \sin t \cdot \cos d \\ \cos t \cdot \cos d \end{bmatrix} \tag{3.5}$$

$$= \begin{bmatrix} \cos m \cdot \sin d + \sin m \cdot \cos t \cdot \cos d \\ \sin t \cdot \cos d \\ -\sin m \cdot \sin d + \cos m \cdot \cos t \cdot \cos d \end{bmatrix} \tag{3.6}$$

After implementing the above result, the tilted scanning sample scene can be seen in Fig.3.13. In the figure, the scanning range of laser range finder is 270 degree and the scanning beam is represented in the form of polygon. As the same to TOF theory, the projected laser beam reflects when bumps into obstacle. In the figure, we can see the tangent plane is stopped by the model of mobile robot and the floor of indoor environment.

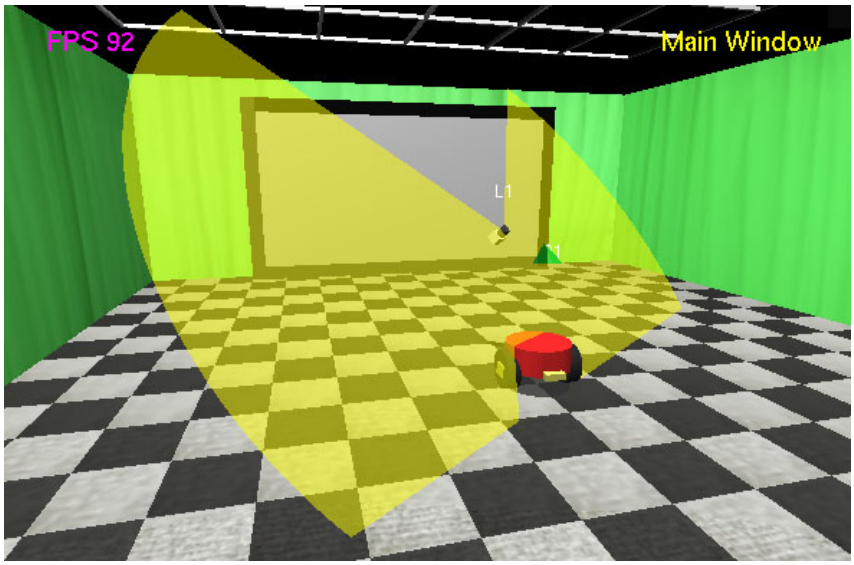


Fig.3. 13 Tilted Laser Range Finder in Sensor Arranger

ii) Distributed Camera

Camera is very useful sensor for two reasons: relative low cost and offer a huge amount of information. Therefore Distributed Camera System is one of the sensor systems in Intelligent Space as well to offer overall monitoring of the indoor environment. While the information is sufficient for research to analysis the situation of environment, in another word, usually in real world, how to use camera efficiently depends on the algorithm of digital image processing rather than sensor itself. In Sensor Arranger camera figure can be easily caught by using coordination transform method. Because in Sensor Arranger, all camera data taken from camera has no influence by noise, image processing algorithm can be simulated in this no-noise environment to test the basic algorithm thought. Actually in most other simulators, the computer graphics image processing is widely used for algorithm evaluation. Because of the purity of sensor data in simulator, the algorithms can be tested hierarchically and improved separately. Fig.3.14 shows the camera view in Microsoft Robotics Studio. In Fig.3.15 the images of distributed camera system of our environment simulator can be seen. The blue transparent radials are the view cubes of every distributed camera. The images of every distributed camera are followed after the overall arrangement graph.

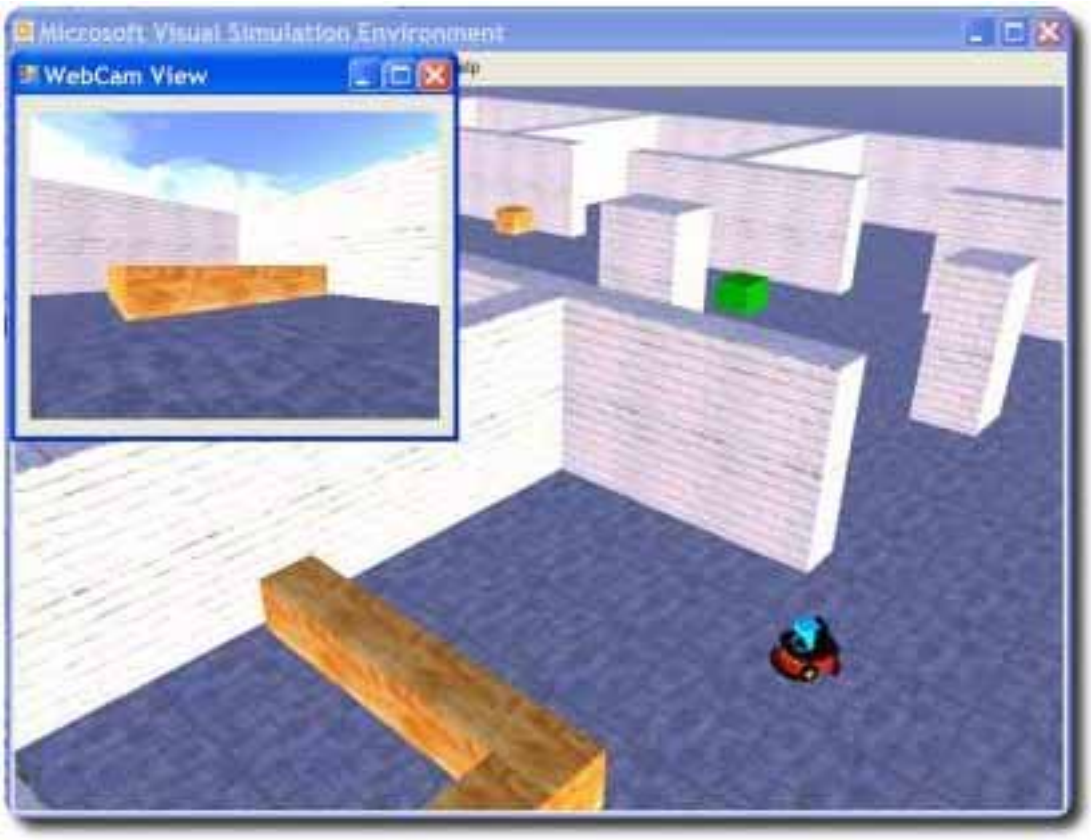
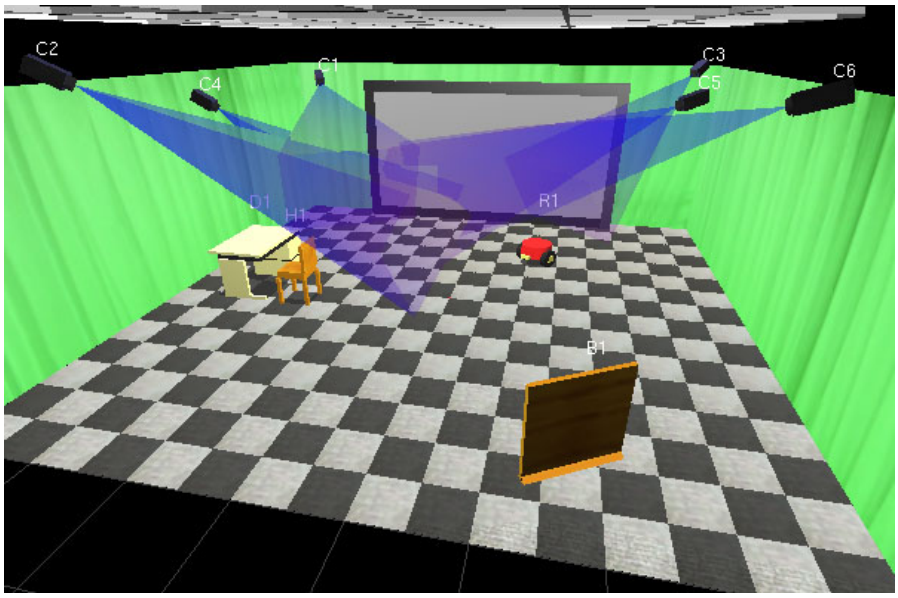
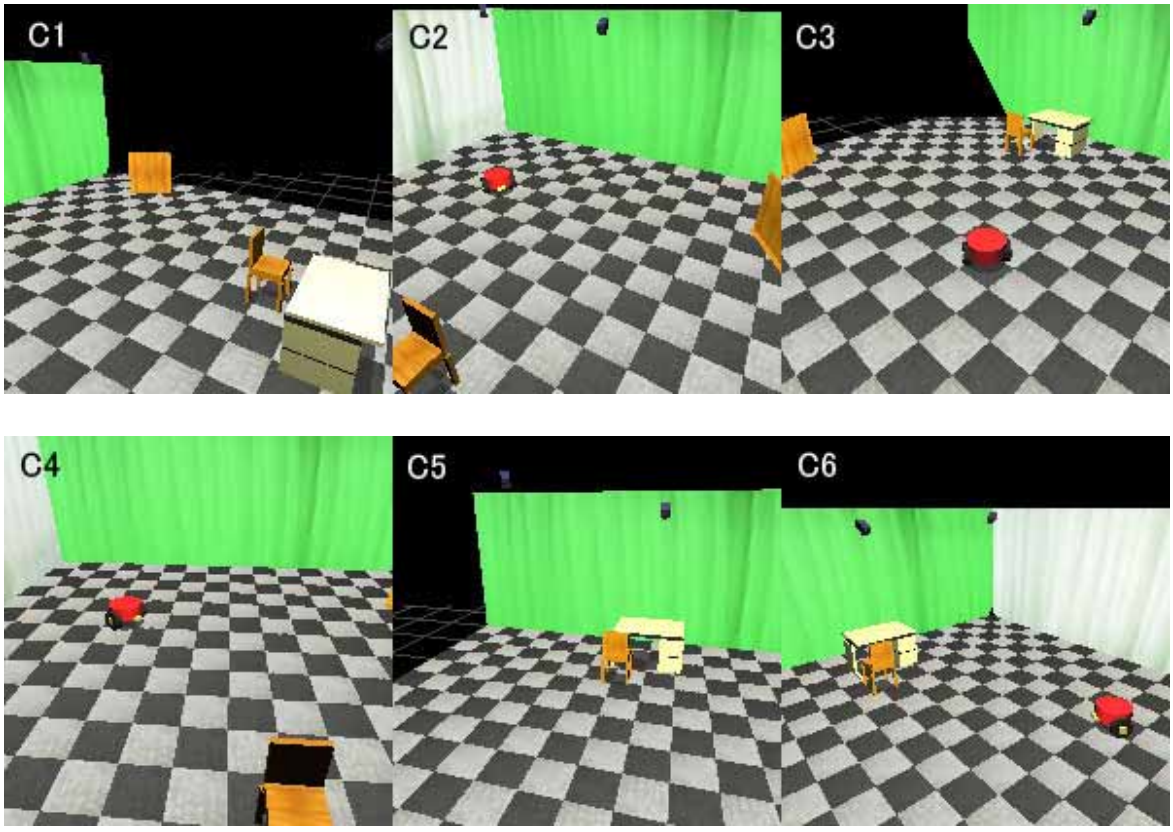


Fig.3. 14 Camera View in MRS



(a)



(b)

Fig.3. 15 Sample of Distributed Camera System in Sensor Arranger

(a) Overall Arrangement of Distributed Camera System;

(b) Captured Images of Every Camera.

iii) Ultrasonic Sensor

Ultrasonic sensing system is a 3D sensing system based on TOF [39] principle as well. The difference is in that ultrasonic sensing system uses many receivers to acquire the direct distance from one ultrasound emitter. Normally, once the emitter started work and the ultrasound was emitted, multiple receivers which are distributed in the same plate but different places receive this ultrasound. By calculating the flight time in the transmission, the distance can be acquired. More than four receivers will receive the ultrasound and make calculation. The reason is that four points are the least to solve the equation in mathematics and extra points can be used to gain better precision. In programs, we can get the position of ultrasound emitter directly so that it is unnecessary to do the above operations as in real world is essential. However, we consider the noise problem that we encounter in real world. Especially when using ultrasonic sensing system, noise is a big program and it should also be simulated. Fig.2.16 shows the concept of triangulation localization.

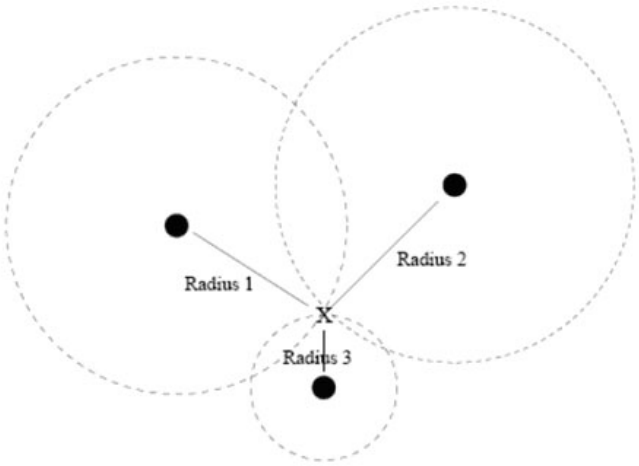


Fig.3. 16 2D Triangulation Localization

The generation of noise for Ultrasonic Positioning System in simulator can be referred to in the following formula. The noise is pseudo white noise. Its distribution belongs to Normal Distribution.

$$x(t) = x(t) + n_1(t) \tag{3.7}$$

$$z(t) = z(t) + n_2(t) \tag{3.8}$$

$$n_1 \sim N(\mu, \sigma^2) \tag{3.9}$$

$$n_2 \sim N(\mu, \sigma^2) \tag{3.10}$$

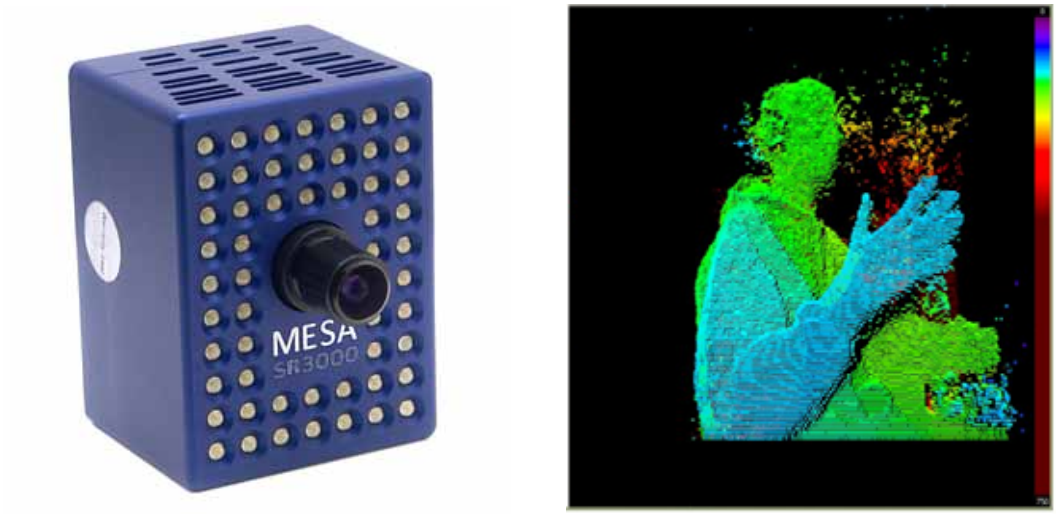
μ is the mean value of the noise, it is zero in simulation.

σ is the standard deviation of the noise, it is 0.1 m in simulation.

iv) 3D Ranger

3D ranger is another application of TOF [39] principal and similar to laser range finder. But mostly, 3D ranger emits infrared light other than laser. Infrared beams are emitted from the same facial plate and reflect if bumping into obstacles. The output is distance image already.

In our programs, we use z-buffer to get the whole scope range information. The operation is similar to distributed camera in that we make translation and rotation from the original coordinates to meet the sensor coordinates. The real work is after getting the data, for convenience, the depth information is usually processed into colorful image in which different color shows different depth. The typical 3D Ranger, MESA SR3000 can be seen in the Fig.3.17. Despite of the main camera lens, it has 55 infrared lighting diodes to support the detection. The right picture is the colored distance picture based on acquired sensor data of the SR3000. The test of 3D Ranger's sensor model in simulator will be discussed in later section and the comparison with real sensor will be given.



(a) (b)

Fig.3. 17 SR3000 Swiss Ranger Camera

(a) SR3000 Swiss Ranger Camera; (b) Sample of Colored Scanning Data

3.4.4 Arrangement Method

The arrangement method is the core part of Sensor Arranger. User controls object to make arrangement through GUIs. Since the operation of GUIs like the control panel has been introduced in previous section, here only the arrangement is explained. However the semi-automatic arrangement mode is put in the next chapter.

For most objects in Sensor Arranger, position and orientation are the basic information. This is the case for most facilities like desk, chair or table etc. For sensors, because sensor can be put in any

position in the whole space, the height of sensor is also considered. Besides, for most sensors, their posture can be controlled by tilt controller.

A typical arrangement has the following steps:

a) Choose the expected sensor or facility, here they are both named object. Adjust their parameters if necessary.

b) Select one object from the potential candidates (blue diamonds in the item-window).

c) The target object should appear in the center of the space, on the floor (in the main-window).

d) Use control panel to move the position on the floor (change the orientation of the object first, put forward or backward afterwards).

e) Use control panel to adjust the height of the object. When adjusting the height, object rises or descends without movement in the horizontal plane.

f) Use control panel to adjust the tilt degree if necessary.

g) Change the view mode or view angles to observe the placement of the object. If adjustment is necessary, repeat the steps beginning from c).

3.4.5 Data Structure

All data of objects are managed in the form of structure like that in Environment Designer. Here this structure can be seen in Fig.3.18.

From the figure, we can see that four sensors' data structure have more attributions than other objects. Their position is defined by three coordination value while other objects have only two because they are put on the floor. The data structure of robot has attributions for simulation while the human model here is not very complicated and its data structure is the same with other facilities. Human model in our simulator is used as active obstacles for mobile robot simulation. Their start position and direction are all defined in Sensor Arranger first, and then it can be used in Simulation Runner. The detail of Human model will be discussed in the section of Simulation Runner.

Every object has the common unit structure and accumulated by list in management. Current pointer always points to the current selected object target by user. The current pointer is important because it is used to operate the position and other attributions of the object. The creation or deletion of objects changes the value of current pointer.

The other main data structures used in Sensor Arranger are in list type, and there are two different kinds of lists that are used in Sensor Arranger for different aims. List data structure is known as useful structure to manage dynamic data. Therefore we use list data structure to maintain the list of scanning sensor and the ranking in semi-automatic arrangement mode.

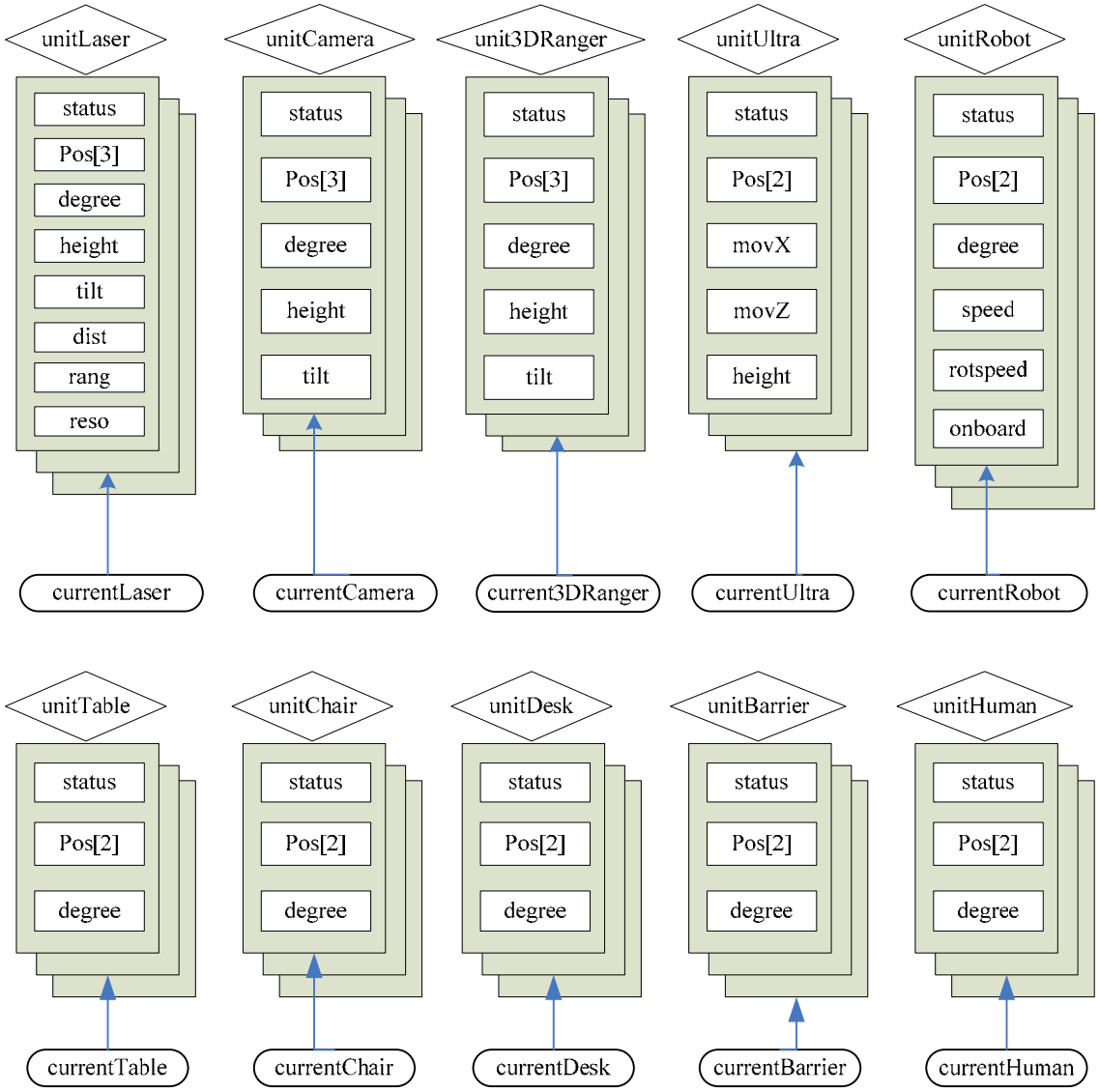


Fig.3. 18 Data Structure of Objects in Sensor Arranger

i) Normal List

The typical list is a list of elements that have common data structure. Usually there are both head pointer and rear pointer to point to the first and last element separately. In Sensor Arranger, semi-automatic arrangement asks for dynamic data structure to save scanning data for sensors. Therefore list type is a suitable form for this application. From Fig.3.19 we can see the arrangement results which are all saved in one normal list. When the sensor scan starts, the list is created to save sensor data. Sensor data accumulates continuously and listed into the data list at the same time. After the complete of sensor scanning, the sensor data can be compared easily using list. At the end, the list can be destroyed and corresponding system source will be recycled.

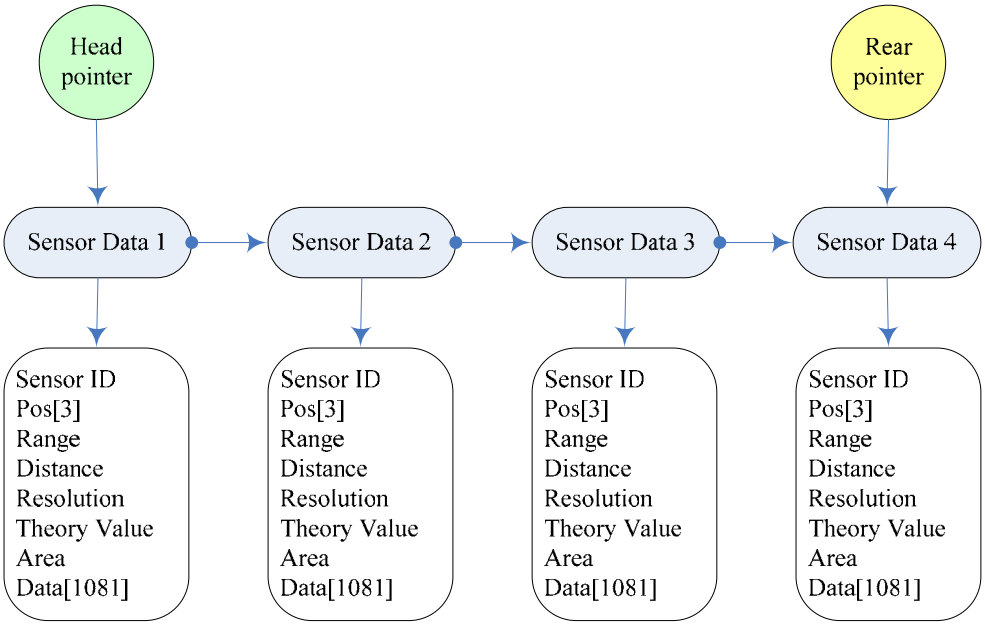


Fig.3. 19 Normal List of Sensor Data in Sensor Arranger

ii) Loop List

Loop List is used in scanning of laser range finder in Sensor Arranger. Because of the algorithm we used is to read z-buffer value around the laser range finder to simulate its principal, when doing scanning with multiple laser range finders, the loop list is used to manage the scanning sequence of all the scanning sensors. During the data processing of sensor scanning, one loop of the program can detect only one laser beam to acquire the distance. In order to give all laser range finder equal opportunity to scan, in another word, to make all sensors work simultaneously, the continuous change of different sensor for scanning is realized by using loop list data structure. The concept figure can be seen in Fig.3.20.

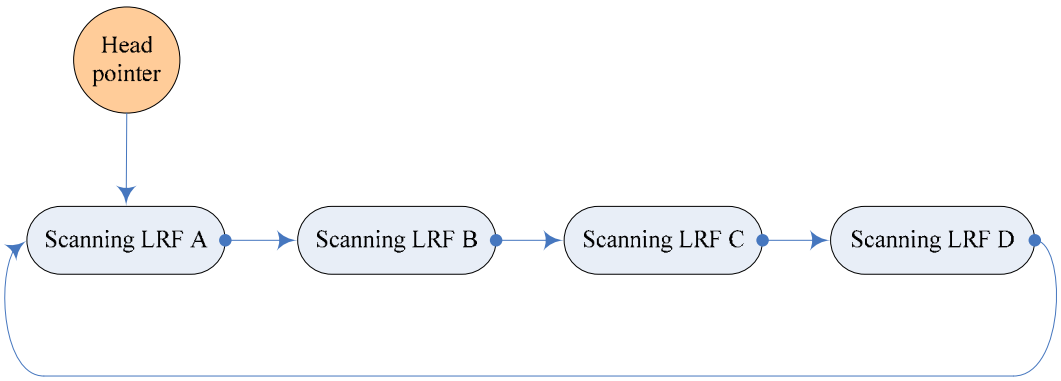


Fig.3. 20 Loop List of Scanning Laser Range Finder in Sensor Arranger

3.4.6 View Mode

In Sensor Arranger, there are three main view modes. One is top view which has been explained in the part of environment designer, the other two are omni-direction view mode and central surrounding view mode. Here we introduce the realization of omni-direction view mode and central surrounding view mode.

i) Omni-direction View Mode

Omni-direction View Mode is the default view mode in Sensor Arranger. It is different from First-person View Mode in that the movement of camera is not restricted to the floor like human usually does. Because the observation of sensor arrangement needs near and far view, while the view angle can be various, the restriction of the camera's height like in First-person View Mode is not proper. Therefore one more free degree is given to the Omni-direction View Mode and thus user can reach any point in the 3D virtual space and observation from any different angle.

ii) Central Surrounding View Mode

Central Surrounding View Mode is useful when user wants to observe the created environment installed with distributed sensors from many different angles. Therefore the Central Surrounding View Mode is supported in Sensor Arrangement. The concept can be seen in Fig.3.21. The green arrows show the rotation of camera to change the view angle while the blue arrows show the tilt of camera to change the height of the camera. The center of the space is always the focus of the camera and the distance of camera to the center can also be changed.

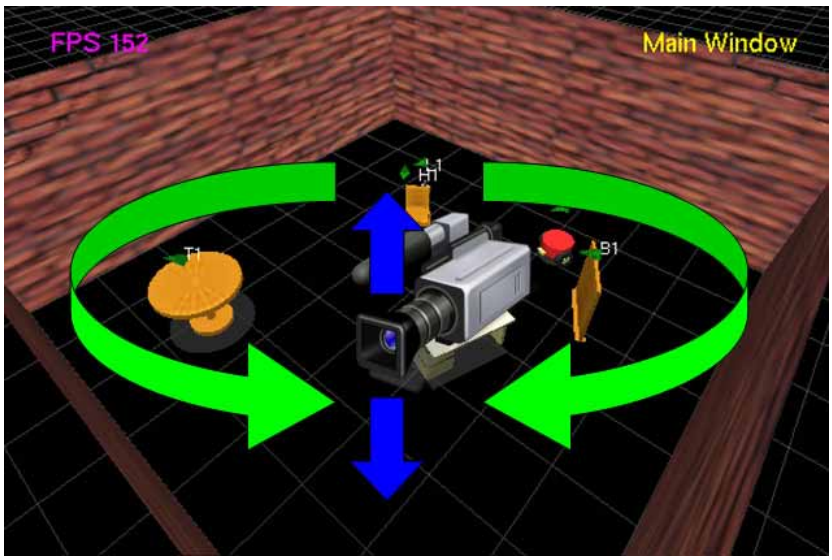


Fig.3. 21 Concept of Central Surrounding View Mode

3.4.7 File Exchange

Because the created sensor arrangement is used in Simulation Runner, all data is loaded in other program through file exchange. There must be some rules obeyed mutually to communicate in a unified format. The extension file name is ".3dm" and its structure is shown in Fig.3.22. The rule of recording 3D object is similar to the environment elements in Environment Designer. Here the narration is omitted.

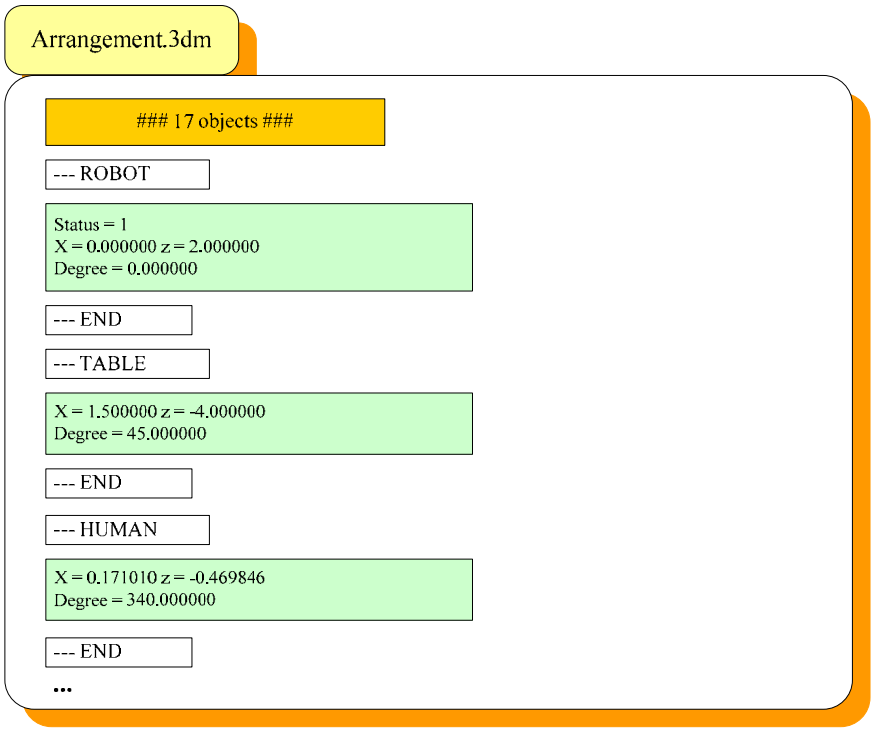


Fig.3. 22 File Format of Sensor Arranger

3.4.8 Sensor Model Tests

In this section, some tests are used to validate the feasibility of sensor model. The test of laser range finder and 3D Ranger are tested in Sensor Arranger and their sensor data is collected, then it is compared with the real sensor data collected from sensors in real experiment environment.

i) Test of Laser Range Finder Model

In this test, we arrange the objects and sensors the same way, both in programs and real world. Then we activate the sensors and acquire data separately. To make more attempts, we use two different laser range finders in the real experiment. Then we plot the map of obstacles separately and compare the

difference. The chosen range is 4 meters, covering degree is 180 degree. The static scenes can be seen in Fig.3.23. In Fig.3.23, the left scene is the photo taken in our laboratory when the experiment is processing. Two different sensors are used here to do the test. The left one is actually the latest version whose photo can be seen in Fig.3.10. It has maximum effective distance as 30 meters, but here we truncate its capability to scan the close 3 meter only. The right one is old version laser range finder URG-04LX created by Hokuyo Corp. Except the maximum effective distance, these two laser sensors have different resolution but the same range angle.

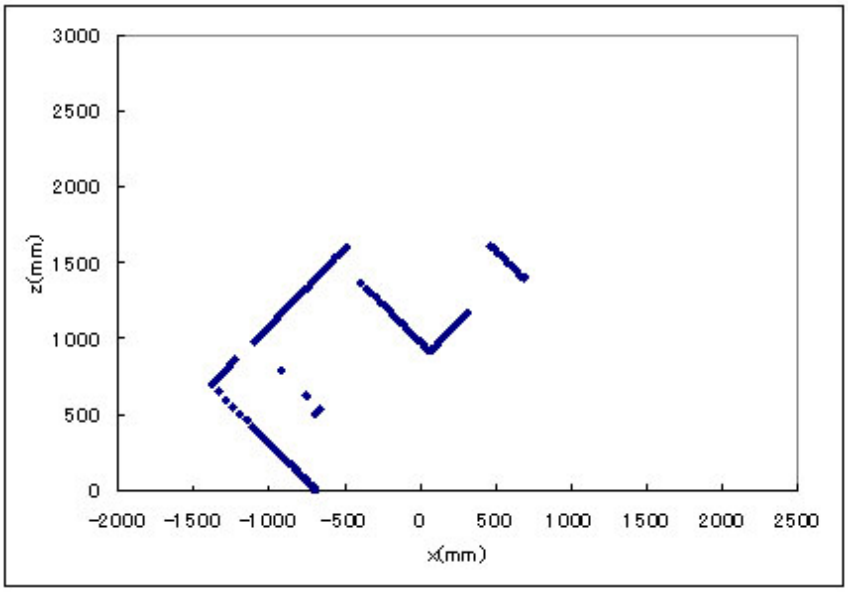
During the experiment, we use four barriers to isolate the influence of other obstacles or walls. In the middle of barriers is a normal desk in size of 1.2 meter long, 0.8 meter wide. The height of scanning plate does not affect very much because the desk we used has almost same contour from the bottom to the top. However because the position of laser range finder can be easily adjusted in Sensor Arranger, we adjusted the height of scanning beam at the same height with actual sensor at about 0.15 meter. The rotation according to the blanket coordination system is set at the same 45 degree for scanning.

(a) (b)

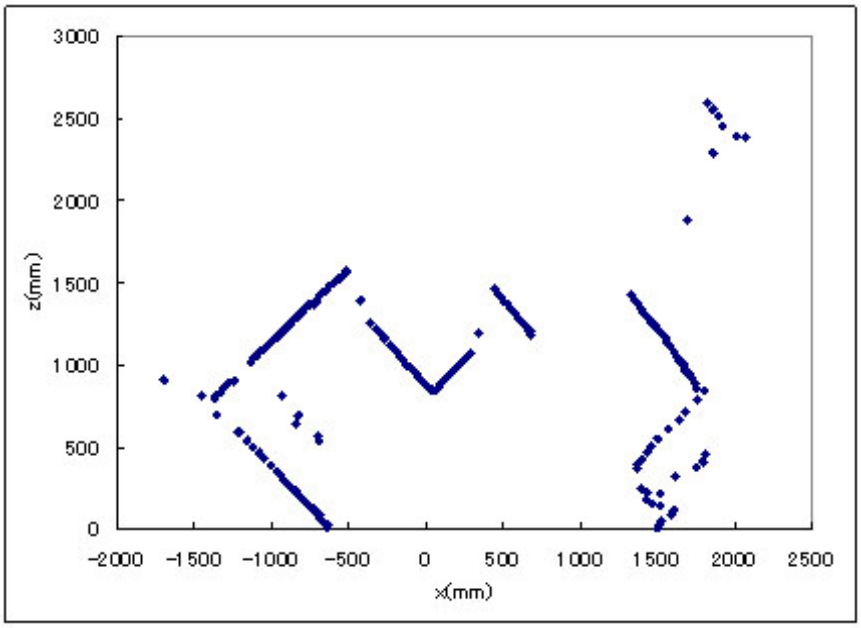
Fig.3. 23 Scene of Experiment in Real World and Simulator

ii) Comparison of sensor data

We plot obstacle maps in Fig.3.24, Fig.3.25 and compare the virtual left sensor with the real one, same in right sensors. As the comparison of left laser range finders, we can find that the biggest difference is on the right half of the map. The map of the real one shows a shape like barrier and something like noises. The reason is on that the virtual sensor has strict limit as 3 meters for effective scanning range while the real one actually outperforms the virtual one at the aspect of effective range. Especially because the left real sensor is a new type, its effective range can reach about 10 meters, and that explains the occurrence of noises. As to the barrier shape, its reason is because there are some facilities near this scene and they reflect the laser beam.



(a)

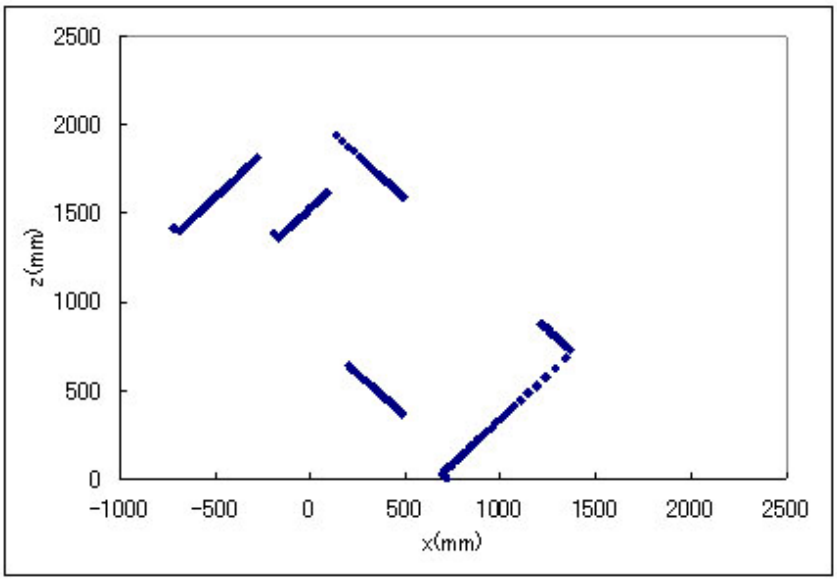


(b)

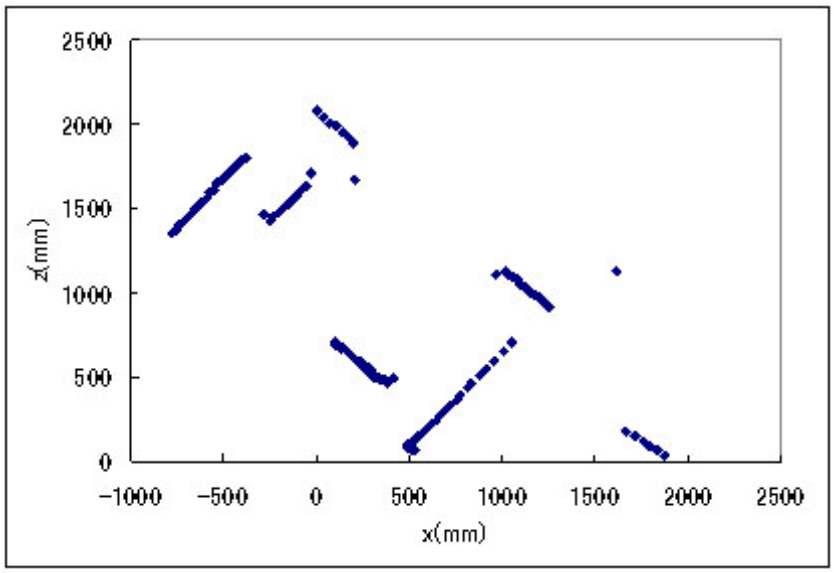
Fig.3. 24 Comparison of the Left Sensors

(a) Virtual obstacle map of the left sensor;

(b) Real obstacle map of the left sensor



(a)



(b)

Fig.3. 25 Comparison of the Right Sensors

(a) Virtual obstacle maps of the right sensor; (b) Real obstacle maps of the right sensor

As the comparison of right laser range finders, we can find that the result is similar. The only difference is that there are fewer noises in the map of real sensor because the used sensor is an old type and its effective range does not outperform 3 meters so much.

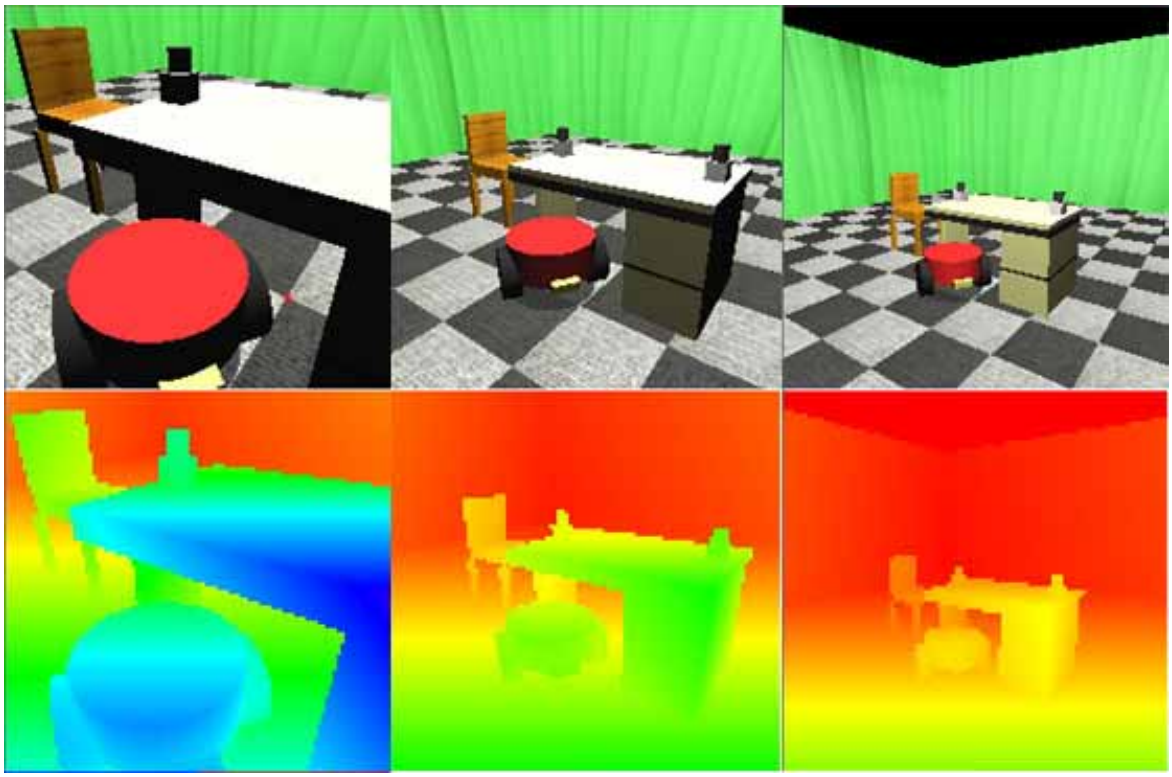
After all, if we look at the scale, we find that the simulation result is almost the same to the real sensor data. It proves our laser range finder model is effective.

iii) Test and Comparison of 3D Ranger Model

To test the capability of virtual 3D ranger in our programs, we firstly executed self test in the virtual environment. We move the ranger from near to far to see if the depth image changes with movement.

Fig.3.26 is the scene we used in self test. We can see that while moving far away gradually from the objects, colors which represent the distance change accordingly. This proves that our virtual 3D ranger works well.

In the second experiment, we executed shots of similar scene in real world and simulator to compare the capability of virtual sensor with real one. Figure 3.27 is the captured photo, real distance image and simulation figures. The left upper one is distance image from Swiss Ranger and its neighbor is the photo. The left lower one is the distance image in the simulator and its neighbor is the normal view. As simple comparison, we can see that the color is a little different in two distance images because of the difference in coloring system. Besides, the limit of effective range in real sensor makes the background full of noise.



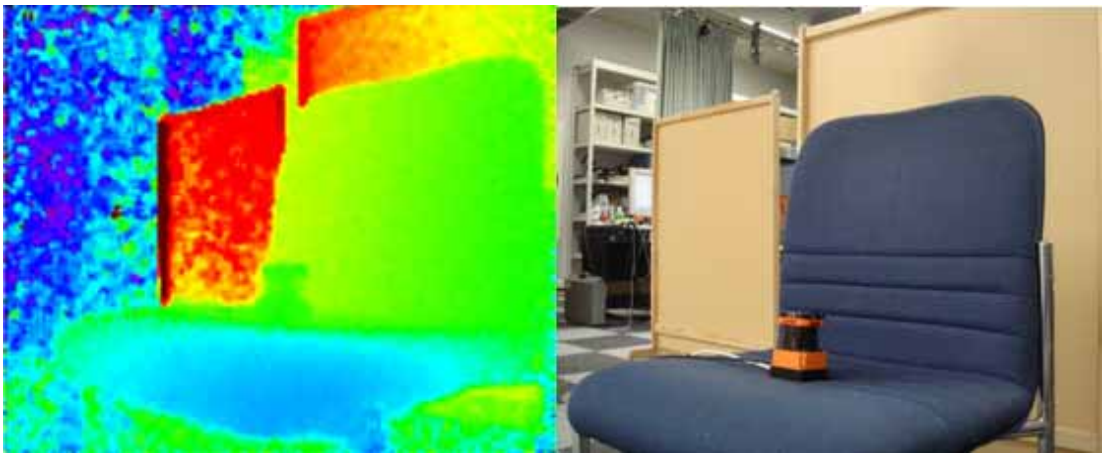
(a) (b) (c)

Fig.3. 26 3D Ranger in Sensor Arranger

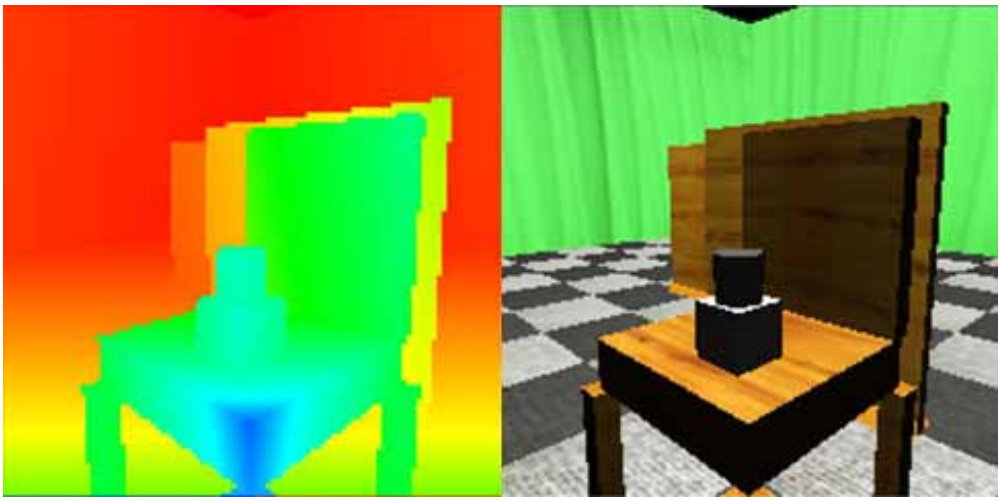
(a) 3D Ranger is 1 meter away from the objects;

(b) 3D Ranger is 2 meters away from the objects;

(c) 3D Ranger is 3 meters away from the objects



(a) (b)



(c) (d)

Fig.3. 27 Comparison of Virtual and Real 3D Ranger

(a) Colored Real 3D Ranger Sensor Data; (b) Applied Scene of Real 3D Ranger;

(c) Colored Virtual 3D Ranger Sensor Data; (d) Applied Scene of Virtual 3D Ranger

Compared with the photo of real experiment scene, the scene in our simulator is a little different from the (b) and (d) in Fig.3.27. But this is not the important factor for our comparison. However the difference of depth image in real experiment and simulation tell us that the sensor model may have no big problems but the spectrum of depth is different with the real sensor. One point is that in depth image of real sensor, the far background shows blue color while in the depth image of simulation the background is totally red. Because the spectrum of real sensor has not value displayed in the menu, we can try to approach its setting values but may not set the absolutely same value. However if we can get the exact value of the spectrum, better simulation of 3D ranger can be realized in the simulator.

3.5 Simulation Runner

As the final stage for simulation, Simulation Runner offers the user enough functions and mechanisms to simulate the control algorithm of mobile robots, or test the effects of distributed sensors. Different with other two tools in our environment simulator, Simulation Runner works as a server to wait for connection by other client programs. Among these programs, robot control program is an important example. However, not only robot relative clients can access the server to do simulation, other clients of sensor program can also take use of the server to observe the created environment or collect necessary sensor data. User may feel confused of the name of our tools. Environment Simulator has three tools. One of them is the Simulation Runner. The Simulation Runner works as the final evaluation stage for distributed sensors as well as offering place for the control of mobile robot. In order to differentiate with other two tools, the three tools are named separately with three main elements in Intelligent Space: environment, sensor and actuator. The main specification is listed in Table.3.3.

Table.3. 3 Specification of Simulation Runner

Windows Arrangement	One Window including all information
User Interface	Mouse and Keyboard Input
View Engineer	1) First-person view; 2) Central surrounding view; 3) Bird view
Actuators and Characters	Mobile robot; Human model
Items and Menu System	Four menus: 1) Facility Menu to adjust parameters of Actuator; 2) Sensor Menu to control the use of sensor; 3) Experiment Menu to establish control strategy; 4) Function Menu to make system change.
Simulation Relative	Switch; Time Display
File Manipulation	Load environment file and sensor arrangement file
Outward Interface	Multi-thread TCP communication with client programs

3.5.1 Graphics User Interface

The GUI in Simulation Runner is introduced according to their graphs from Fig.3.28~Fig.3.32.

Fig.3. 28 Main Menu in Simulation Runner

Fig.3. 29 Facility Sub-menu in Simulation Runner

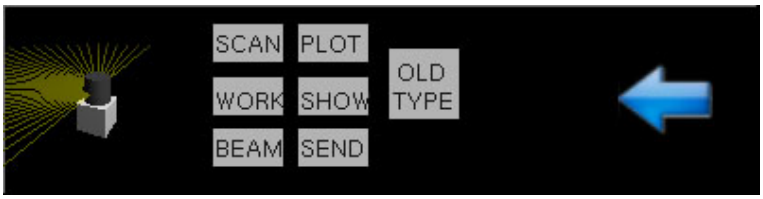


Fig.3. 30 Sensor Sub-menu in Simulation Runner

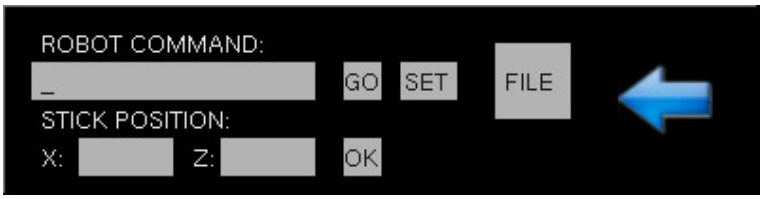


Fig.3. 31 Experiment Sub-menu in Simulation Runner

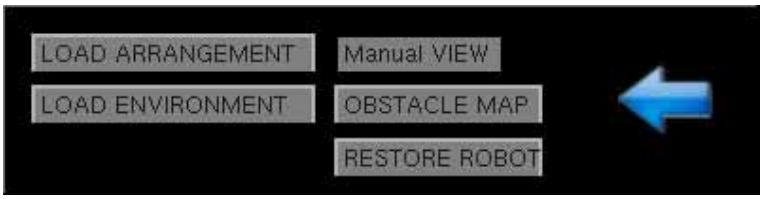


Fig.3. 32 Function Sub-menu in Simulation Runner

As the figure shows, the main menu in Simulation Runner is consisted of four sub-menus. They are “FACILITY”, “SENSOR”, “EXPERIMENT” and “FUNCTION” separately. “FACILITY” includes most operation on actuators like putting on onboard sensor, controlling the random movement of human models or automatic moving robots. “SENSOR” includes the control of laser range finders like the scanning switch, the display of sensor data etc. “EXPERIMENT” includes the control of robot for simple commands, the input bar is used to receive user’s command and then the inner mechanism will execute all commands. The command system will be introduced in latter part. “FUNCTION” sub-menu includes functions like load environment file or arrangement file from the other two tools. Here the view mode changed exchanged and obstacle map can be turned on. The information about obstacle map will be discussed in the next chapter.

3.5.2 Viewer Mode

The specific view mode in Simulation Runner is Robot Surrounding View Mode. Besides, the other two view modes, Bird-View and First-person View have been introduced in previous sections. Actually Robot Surrounding View Mode is quite similar to that of Central Surrounding View Mode. What is different is that here in simulator the observe focus in on the mobile robot, therefore the surrounding target is on the robot as well.

3.5.3 Multi-thread TCP Communication

In our Simulation Runner, we use TCP protocol to communicate with outside program. Actually, not only for robot control program, but also for any asks to acquire the sensor data in simulation and control command of robot. By using advanced network technology, the Simulation Runner becomes a comprehensive platform, which can be utilized by anyone authorized through network. Here the realization of TCP communication part in simulator is explained in detail.

i) Communication in Simulation Runner

There are two main methods to do simulation of mobile robot in simulators. One is to rewrite all control programs into appropriate format of the target simulator [22], like in Microsoft Robotics Studio and OpenHRP as discussed in Chapter 2. Another way is to use the current control programs without big changes. Using network communication, after redirect the communication port, the robot control part and sensor reading part can be used as it was [26]. This method also solves the problem caused by using different platform between simulator and control programs. To some extent, Microsoft Robotics Studio is a little limited in use because they want all users working on windows platform. Recently, many sensor makers only offer programmable sensor driver under UNIX/Linux platform, so it will be necessary to consider the users in that case.

In our Simulation Runner, what should be solved are two main tasks. One is to receive robot command from outside control program, and the other is to send sensor data when asked to do so. These two tasks should be independent with each other like in real world. Therefore, the simultaneous communication necessity asks for multi-thread process in simulator. The concept figure of the communication part in the Simulation Runner can be seen in Fig.3.33. In this figure, five clients are communicating with the simulator at the same time. Robot control clients have mutual communication with the simulator because they read robot status from simulator and give control command to simulator after understanding the current status. Meanwhile, sensor data can be read from outside programs as well. Like in the figure, the sensor data of onboard laser range finder and distributed laser range finder is ceaselessly flowing into the clients.

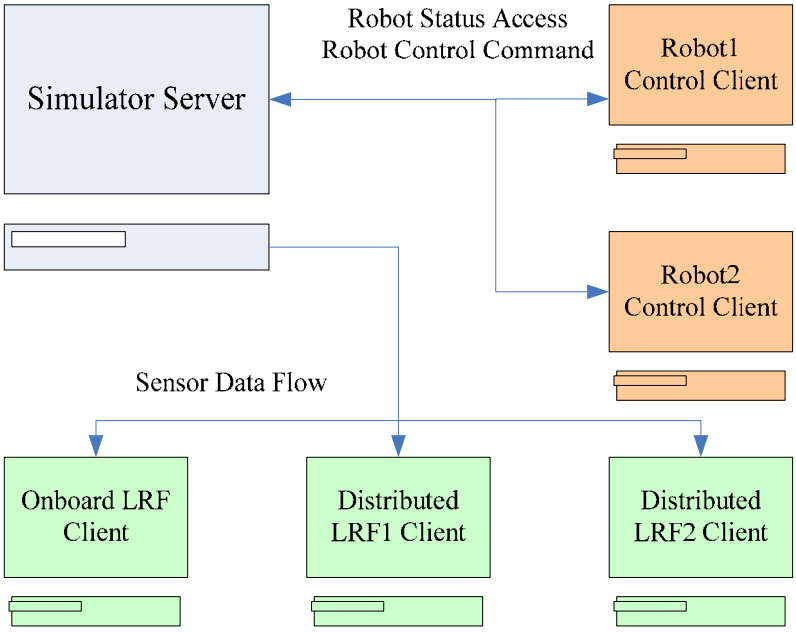


Fig.3. 33 Concept of Communication with Outside Programs in Simulator

ii) TCP Network Communication

The Transmission Control Protocol (TCP) [40] is one of the core protocols in the Internet protocol series. TCP is the central part of the Internet protocol series so that the whole series are often referred to as "TCP/IP" protocol. Whereas IP protocol handles lower-level transmissions from computer to computer as a message makes its way across the Internet, TCP operates at a higher level, concerned only with the two end systems, for example an Internet browser and a server. In particular, TCP provides reliable, ordered delivery of a stream of bytes from one program on one computer to another program on another computer. Besides the Web, other common applications of TCP include e-mail and file transfer. Among its management tasks, TCP controls message size, the rate at which messages are exchanged, and network traffic congestion.

TCP is a reliable stream delivery service that guarantees delivery of a data stream sent from one host to another without duplication or losing data. Since packet transfer is not reliable, a technique known as positive acknowledgment with retransmission is used to guarantee reliability of packet transfers. This fundamental technique requires the receiver to respond with an acknowledgment message as it receives the data. The sender keeps a record of each packet it sends, and waits for acknowledgment before sending the next packet. The sender also keeps a timer from when the packet was sent, and retransmits a packet if the timer expires. The timer is needed in case a packet gets lost or corrupt.

iii) Multi-thread Technology [41]

On a single processor, multithreading generally occurs by time-division multiplexing (as in multitasking): the processor switches between different threads. This context switching generally happens frequently enough that the user perceives the threads or tasks to be running at the same time. On a multiprocessor or multi-core system, the threads or tasks actually do run at the same time, with each processor or core running a particular thread or task. Many modern operating systems directly support both time-sliced and multiprocessor threading with a process scheduler. The operating system kernel allows programmers to manipulate threads via the system call interface.

Threads are distinguished from traditional multitasking operating system processes in that processes: are typically independent, carry considerable state information, have separate address spaces, and interact only through system-provided inter-process communication mechanisms.

Multithreading is a popular programming and execution model that allows multiple threads to exist within the context of a single process, sharing the process' resources but able to execute independently. The threaded programming model provides developers with a useful abstraction of concurrent execution. Operating systems schedule threads in one of two ways. Preemptive multithreading is generally considered the superior approach, as it allows the operating system to determine when a context switch should occur. Cooperative multithreading, on the other hand, relies on the threads themselves to relinquish control once they are at a stopping point.

iv) Combination of Multi-thread and TCP Communication

In our simulator, the simulator program works as server for outside programs which are the clients mean while. So the server/client model is being used in simulator’s communication with outside program. Because the transmission of robot control command and that of sensor data is independent with each other and can not avoid being asked simultaneously, multi-thread programming is used. The flow chart can be seen in Fig.3.34.

The concept is explained as following:

1) Server is always listening to outside programs. Because the rendering of simulator should work at the same time, the server part itself is a thread created when the program starts.
2) Once a client asks for service, the server asks for its service content. This is to differentiate service targets and also decline service to irrespective clients. The current available services are robot command, named as “ROBOT” and reading of sensor data, named as “LASER”.
3) After inquiring client’s aim, check the current status of relative objects. If the current object is being used, then decline the asking. For example, there is a client who is controlling the robot No.1. Now another client asks to issue commands to the same robot No.1, this request is declines because the current resource in simulator is being used and not available. It is the same situation in real

world, because one object can only be used by one controller at one time.

4) If the check is no problem, server creates a new thread for this new service. After that, server continues to listen for other requests.
5) The connected client starts work in simulator like issuing commands or reading sensor data.
6) The simulator responds to the client like moving the target robot as asked or sending specific sensor data to the client.
7) The work finishes, and the current thread will be closed.
8) One service is over and the program returns to the first step in which server waits for requests from clients.

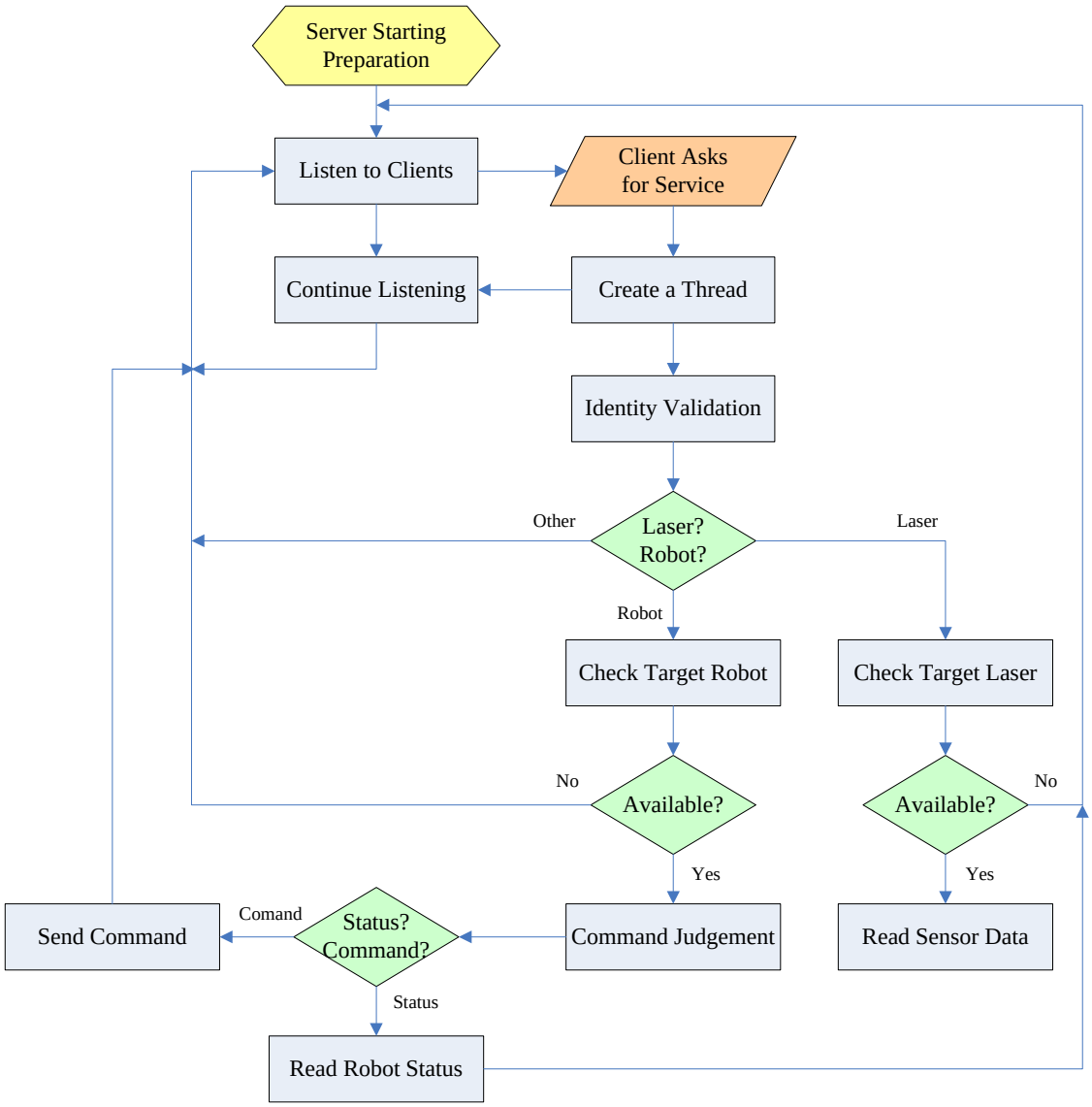


Fig.3. 34 Flow Chart of Multi-thread TCP Communication in Simulator

In a word, the simulator works as a server to offer service to outside programs. The simulator indoor world is like an unknown environment in which several actuators and sensors are waiting for being applied to work. The outside user can choose the way to communicate with simulator. There is no limitation for using robots or sensors, or limitation of users, only if the source is available, they can be requested for applications.

3.5.4 File Operation

Here, file operation not only includes exchange with sensor arranger to extract arrangement of sensors and objects, or with environment designer to gain indoor environment layout, but also works with recording of sensor data during simulation. Main records of simulation data are listed in the Table.3.4.

Table.3. 4 Record File in Simulation Runner

Record Name	Content
Robot Status	Speed, rotation speed, position, acceleration, rotation acceleration
Sensor Data	Scanning data of laser range finder
Simulation Information	Control Interval, elapsed time etc.

3.5.5 Command System

The Command System is used in simulator to give robot simple command to execute. Different with simulation executed by outside program, these commands can be helpful to ask robot do some simple actions like moving around etc. The main commands can be seen in the Table.3.5.

Table.3. 5 Command System in Simulation Runner

Command Name	Action
FORWARD	OK command. Make robot go forward by certain meter.
BACKWARD	OK command. Make robot go backward by certain meter.
LEFT	OK command. Make robot turn left by certain degree.
RIGHT	OK command. Make robot turn right by certain degree.
SPEED	SET command. Make robot move at constant speed.
ROTSPEED	SET command. Make robot rotate at constant rotate speed.
STOP	SET command. Make robot stop.

Because the robot moves complying with kinematics, going forward does not only mean a movement in constant speed. Similarly, turning around does not mean that the robot turns around at constant rotate speed. Acceleration and rotation acceleration have been embedded into the movement of robot. Therefore stop command does not mean that the robot will stop suddenly because the speed can not become zero at once. The detail of implementation of kinematics in simulator will be discussed in the chapter about simulation.

3.6 Summary

In this chapter, the structure of simulator has been introduced in detail. The three tools: Environment Designer, Sensor Arranger and Simulation Runner do not work independently. Inversely, they cooperate with each other to make our simulator a powerful simulation platform for research and many other aims in Intelligent Space.

CHAPTER 4

Automatic Arrangement Mode of Sensor Arranger

Sensor Arrangement is one of the main objectives of our environment simulator. Beside of manual operation as stated in the previous chapter, amateur users are looking forward to more automatic functions. Because the arrangement of laser range finder is an important theme for Intelligent Space research, automatic laser range finder arrangement mode is proposed here.

4.1 Sensor Arrangement Overview

Sensor arrangement has been an old theme in robotics field. Bayesian Decision Analysis [42] has been used to decide effective sensor arrangement in indoor environment. However, Bayesian Decision Analysis can only be used in partly known environments and the limitation is only single sensor arrangement is considered. Many computation approaches [43] have also been developed to do this work, but there are also a lot of limitations like that the algorithm can be very complicated and the calculation takes in too many exceptions. Hierarchically distributed perception methods [44] has been proposed to do the search of potential optimal placement in random way to save the computation time.

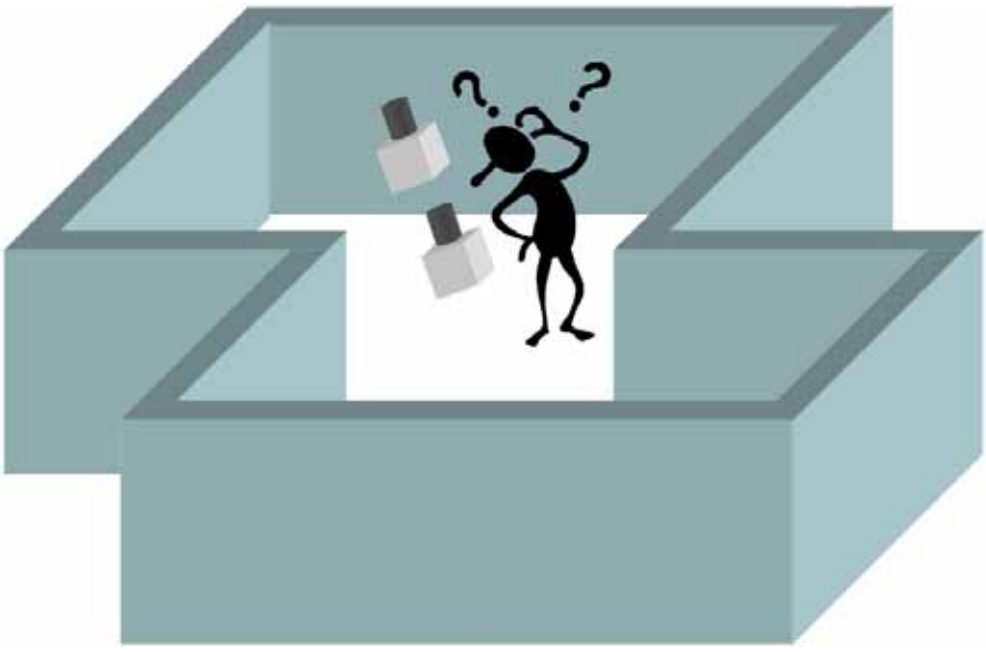


Fig.4. 1 Sensor Arrangement Puzzles

In the research of Intelligent Space, the relative sensor arrangement is mainly around the arrangement of laser range finder. The arrangement has the following factors to consider.

a) 2D horizontal arrangement.
b) Unknown indoor environment
c) There may have various facilities in the indoor environment.
d) Multiple sensors.
e) Different sensor may have different specification.
f) As the sensors for mobile robot tracking, the continuous of scanning zone can also be a factor.
g) Different needs of user, some people do not want put the sensor on the road to disturb other facilities. However, some people may not care about this.

As discussed in the previous chapter, our model of laser range finder was proved accurate and effective. Therefore we propose our detection approach to make automatic arrangement in our environment simulator. To cope with the above tasks, despite of a) and d) because our sensor model can acquire accurate sensor data easily, there are still three main tasks to solve. To apply our approach in unknown indoor environment Limit Acquire Algorithm and Insider Judgment Algorithm are proposed in the following sections. To valuation the fusion of multiple sensors, evaluation factors are abstracted and the evaluation equation is proposed to make proper decision. In the end, in order to improve the efficiency of the execution of automatic arrangement, Fast Detection Algorithm is proposed with reference to hierarchically distributed perception methods to do random detection so that the arrange time can be saved to some extent. As to the f) item, it seems difficult for computer to abstract this task, and we want to listen to the user's needs and help him do such work. The g) factor tells us that the placement of laser range finder can be divided into two main ways, arranging along the wall or arranging free in the indoor environment.

The automatic arrangement function will be explained in detail in the following sections and at last, the execution result and some statistics result will be discussed.

4.2 Graphics User Interface

Firstly the GUI for automatic arrangement is introduced. The GUI panel for arrangement can be seen in the Fig.4.2. There are three main arrangement strategies, several parameters for use to set and some other useful buttons.

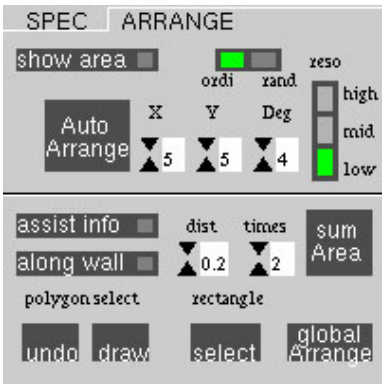


Fig.4. 2 GUI Panel for Automatic Arrangement Mode

As the figure shows, this panel is consisted of three check box, six buttons, five level tuners and two mode selectors. “show area” is a check box to choose the display of laser beam. Because the radial style can be hard to distinguish if multiple sensors with high resolution are scanning together, the “show area” mode only shows the contour of laser beam so that user can distinguish their corresponding sensor clearly. “Auto Arrange” button is to start the execution of automatic arrangement. During the execution, the background color will change to give user a warning about what is happening. The right three level selection tuners are used to adjust relative parameters in arrangement. From left side, they are “X” means the grid number in x direction for the “ordinary arrange mode”, “Y” means the grid number in y direction and “Deg” is the division of direction for scanning. They are all parameters for the ordinary mode while the mode can be changed into random arrange mode by clicking the mode selectors on top of the three tuners. “reso” means the resolution level for scanning and as default we use the lowest level of resolution for arrangement to make the speed fast. Till here, all the GUIs on the half upper of the “ARRANGE” page were introduced. In the half bottom page, “assist info” offers user the assistant lines to see how the arrangement works. However “along wall” is another important arranging strategy that makes the scanning along the wall side. And usually this strategy is very efficient. “dist” is the dist to wall and “times” is scanning times on one wall. While we can get the total effective area of scanning after the automatic arrangement, user may want to calculate that area after his own arrangements, at that time “sum Area” button can be useful. The lowest three buttons are used to select specific zone to make arrangement and “global Arrange” button makes the zone back to the whole indoor environment.

4.3 Execution Flow Chart

As the big picture for automatic arrangement function, its execution flow chart can be seen in Fig.4.3.

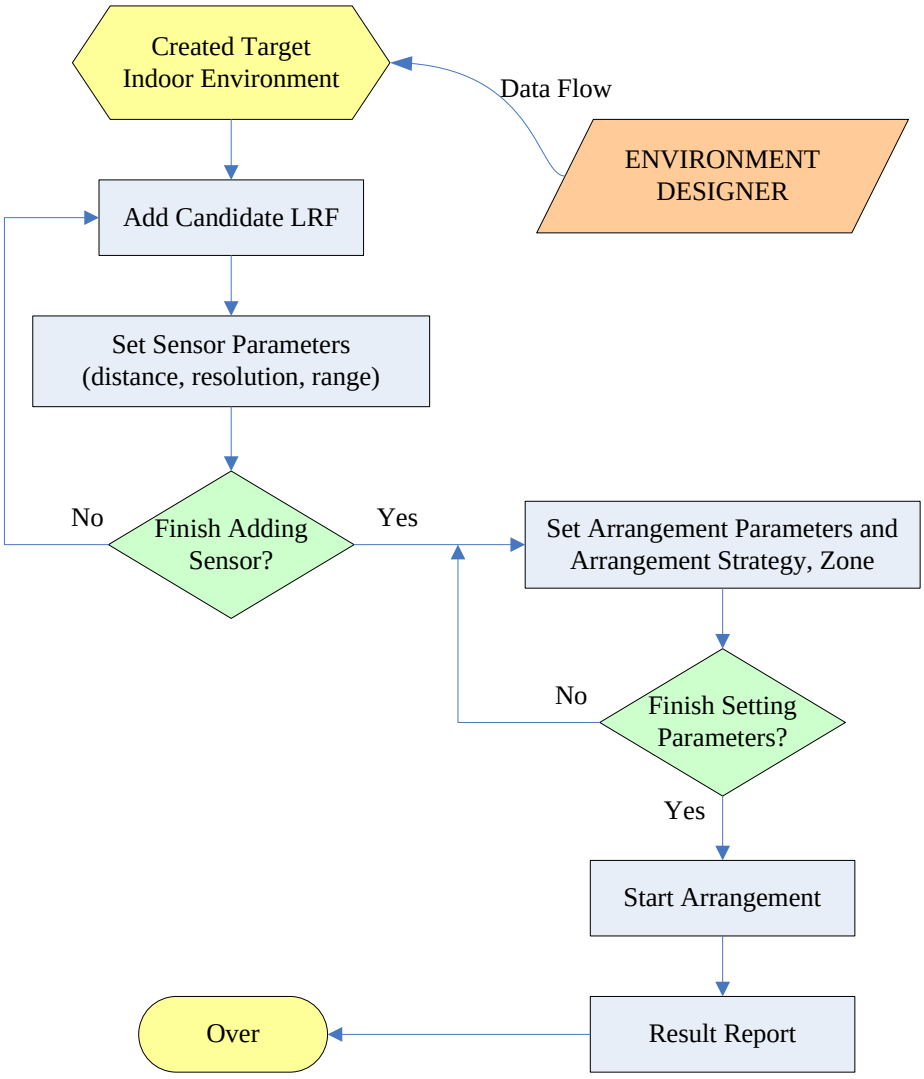


Fig.4. 3 Flow Chart of Automatic Arrangement

Firstly, the arrangement target environment is created by Environment Designer. After that, use item-window to add multiple laser range finders and set their parameters separately. Then start setting arrange parameters and begin the execution of automatic arrangement. Hence, wait for the result to come and validate the arrangement result. If the time mode is set to highest, the result may not be contented with, therefore the above flow can be repeated again to find better arrangement result.

4.4 Algorithms

The algorithms discussed here are all used in the process of automatic arrangement. Firstly, the Limit Acquire Algorithm applies to define the square region for further exploration. In succession, depending on the time level selection, only if the user selects the highest time level, the Hierarchical Random

Algorithm will be applied to select hierarchical random methods to do sample selection. Otherwise the Ordinary Grid Algorithm applies. After the potential sample position and direction were decided, this sample position will be judged by Insider Judgment Algorithm to judge whether this position is valid which means the position is inside the indoor environment. If it does, the scanning starts and one scanning result will be collected into the candidate list. Else, the sampling restarts to get another sample position. Beside of these two strategies, along wall arrangement usually shows better arrangement result. The flow chart of the process of automatic arrangement can be seen in Fig.4.4.

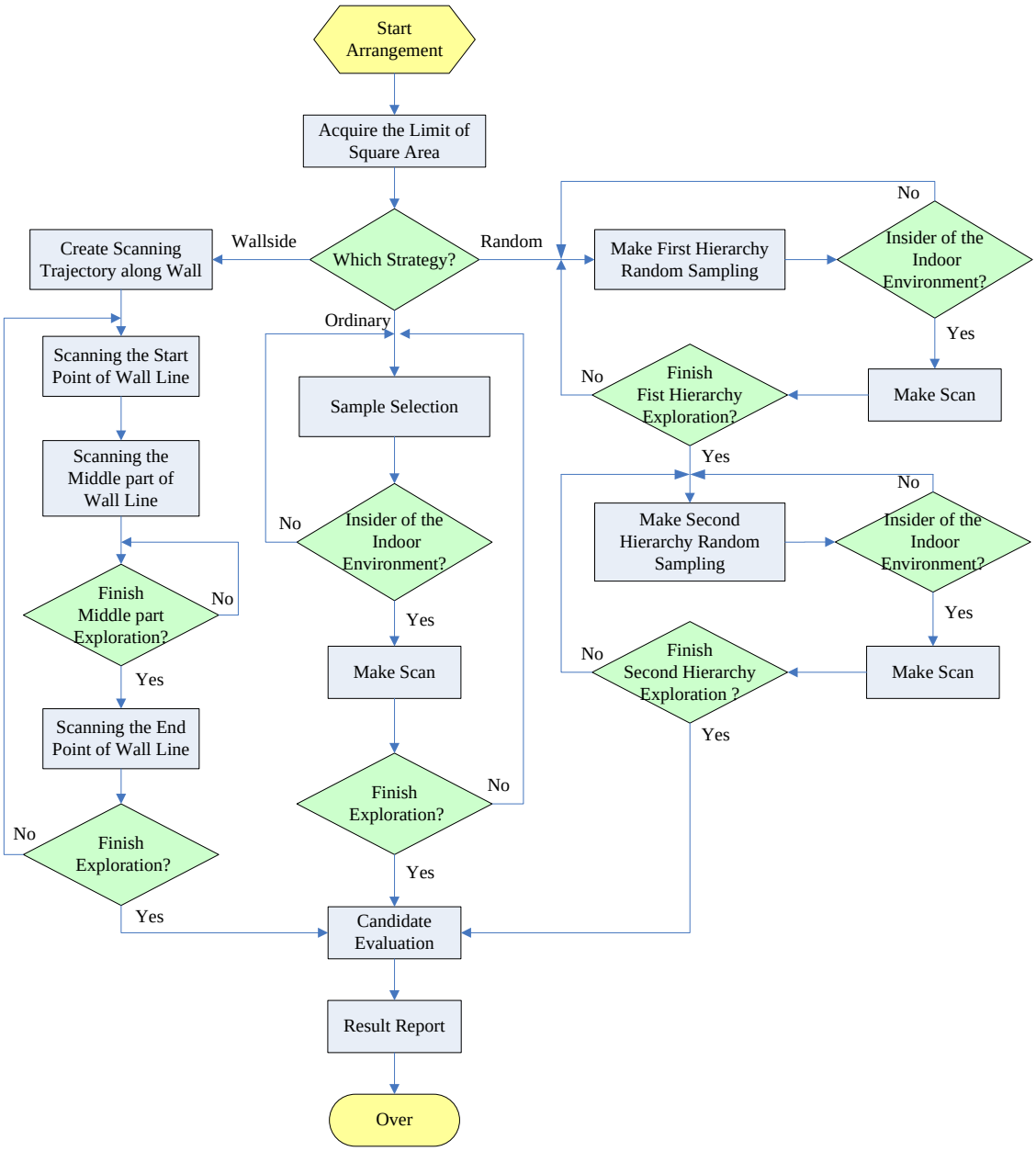


Fig.4. 4 Flow Chart of Arrangement Algorithms

4.4.1 Limit Acquire Algorithm

Limit Acquire Algorithm is used to get the maximum length and width of the indoor environment. In order to implement the following detection, we firstly define the external connected square area of the indoor environment. Afterwards, we select sample position from this square area and judge whether the position is inside the indoor environment or not. The concept can be seen in Fig.4.5.

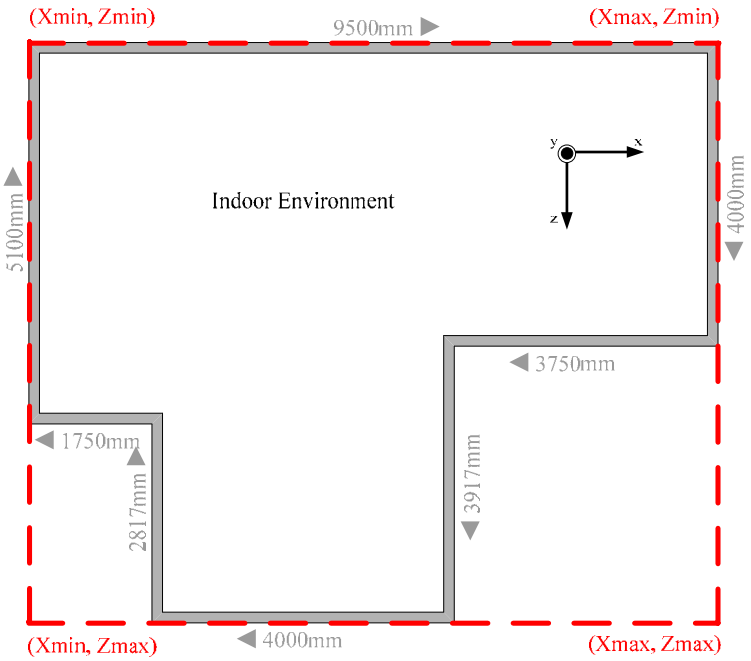


Fig.4. 5 Concept of Limit Acquire Algorithm

The algorithm is to rank all vertexes of the wall lines which form the indoor environment. The record of indoor environment's position data will be explained in the next chapter, in the section of Obstacle Map. After ranking, the minimum value and maximum value in X direction and Z direction are available to do the further algorithms.

4.4.2 Ordinary Grid Algorithm

The Ordinary Grid Algorithm applies when user chooses normal time level except the highest level. In order to make full use of the calculation power of computer, the way to select sample positions to place laser range finder has important meaning relative to the final result.

As the limit coordinates of external connected square area have been collected by the above algorithm, we choose candidate position to do scanning according to the size of the indoor environment and the effective distance of the used laser range finder. The following equations can be used to decide

the grid size of potential placing position for the sensor.

$$m = \frac{x_{\max} - x_{\min}}{dist} \cdot seg - 1 \tag{4.1}$$

$$n = \frac{z_{\max} - z_{\min}}{dist} \cdot seg - 1 \tag{4.2}$$

m is the explore times at the x axis. It can be set as “X” tuner in the GUI panel.

n is the explore times at the z axis. It can be set as “Y” tuner in the GUI panel.

$dist$ is the maximum scanning distance of the current laser range finder.

seg is the coefficient to divide one segmentation of the indoor environment. In the simulator 5 is used at the default coefficient.

From the above equations, we can get the grid size by dividing the size of indoor environment’s square area with the maximum distance of laser range finder. The coefficient is used to make further division of the big grid to do sub exploration because we can not assure that the exploration is enough without more division in one maximum distance of the laser range finder. The distance of laser range finder can be different for different sensors. The typical value can be 4 meters for old type laser range finder and 30 meter for new type one. To calculate the separate explore times at x axis and z axis, we deduce the number of grids by one to get the times to do the exploration. In Fig.4.6 the concept of grid in Ordinary Grid Algorithm can be seen. In the figure, red broken lines divide the space into several big grids. Because the size of space may not be the integer times of sensor’s distance, margin may happen. Green lines divide one big grid into further smaller grids. Here the divide coefficient is four as discussed above. The silver circle represents the explore position of laser range finder as the figure shows.

After the position has been decided, the orientation candidates of sensor should be decided as well. In our algorithm, the following equations apply.

$$k = \frac{360}{seg_{rot}} - 1 \tag{4.3}$$

k is the explore times at one position with different orientation. It can be set in the GUI panel.

seg_{rot} is the segmentation of orientation for the sensor. The default value is 8 in simulator.

In the above equation, the times to scan with different orientation can be calculated. As tradeoff between explore time and effectiveness, for middle time level setting, the default value is 8 and the value is 16 for low time level.

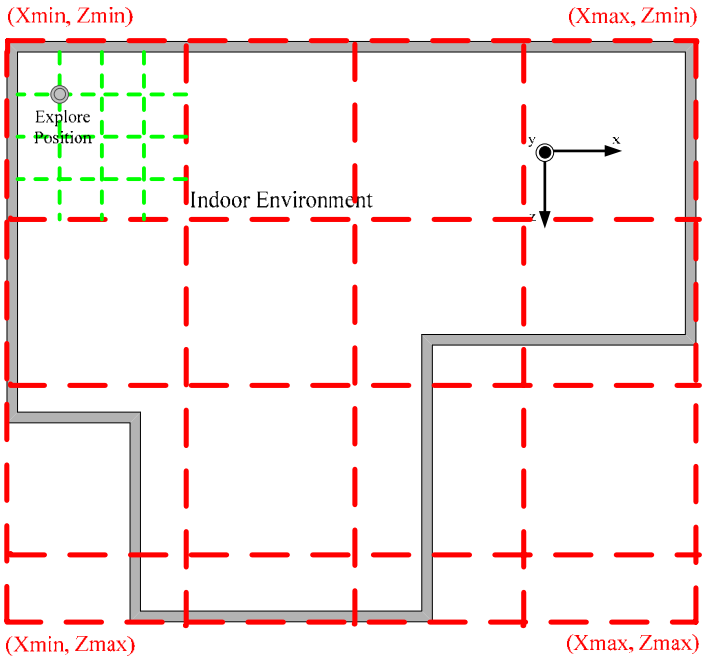


Fig.4. 6 Concept of Grid in Ordinary Grid Algorithm

4.4.3 Hierarchical Random Algorithm

As discussed above, the user can choose the time to do automatic arrangement. The arrange strategy can be chosen as random mode which applies Hierarchical Random Algorithm.

This algorithm is mainly realized by using hierarchy random methods to decide the potential position. Because the limited time and complexity of environment, the normal sample selection methods does not give very good result sometimes. Furthermore, because the exploration is restricted to grids of the space, some better position may be ignored because of the improper setting of the grid size. Therefore random selection algorithm was considered to get more arbitrary position in certain place. Anyway, because the explore times of random algorithm can not be infinite as well. As the tradeoff between time and result, the explore times is set at relatively low value to Ordinary Grid Algorithm. What is important is that if the times of exploration for both algorithms can be very big, the result will converge to the same one theoretically.

As to the flow of this algorithm, firstly, the first hierarchy exploration detection chooses several positions in the space to get a possible best zone. As the second step, the second hierarchy exploration chooses the best result of the first detection as the start point. Then the second random detection executes within certain scope of the start point to do further exploration. The thought of this algorithm is based on belief that global best candidate has large possibility to be the local best candidate at certain area. The demerit, obviously, can be that the final result is not always the global best one by applying

this algorithm with too small explore times. The sample exploration of Hierarchical Random Algorithm can be referred to in Fig.4.7. In the figure, green points represent the explored position and the center position of the green broken circle is the possible best position. Orange points represent the explored position within the bound of the first best position and the candidate near the center is detected to be the best position at last. As we can imagine, the candidate position can be outside of the indoor environment during the exploration, the next algorithm will be explained to judge whether the candidate point is inside the indoor environment or not.

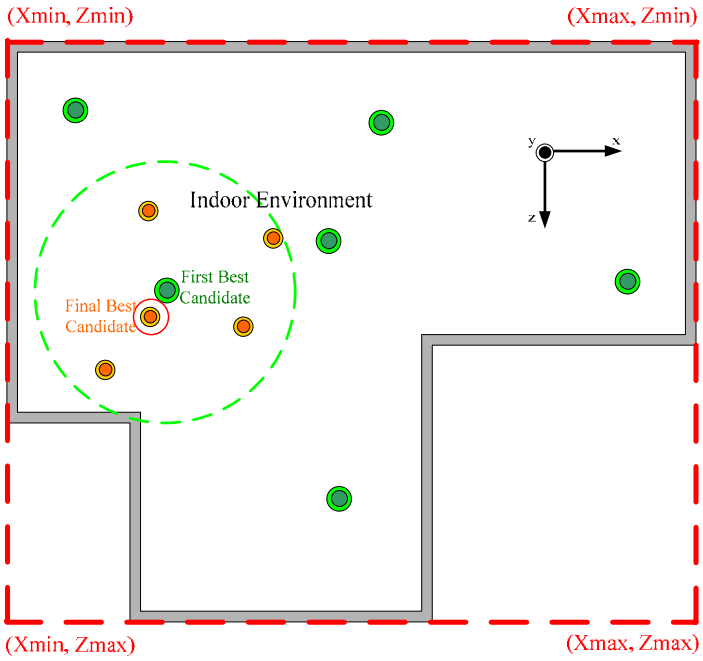


Fig.4. 7 Sample Exploration of Hierarchical Random Algorithm

4.4.4 Insider Judge Algorithm

Inside Judgment Algorithm is used to judge whether one position in the limit of outside square area is truly inside the closed indoor space or not. This is important for two reasons. First, outside position detection wastes time because the detected result is invalid for indoor scanning. Second, outside position detection can mislead the final optimal result. Consider an area prefer user may get a solution that offers him a outside laser range finder scanning against walls like in Fig.4.8. In this scene, the best arrangement is set at acquiring largest detection area using the current laser range finder. Because of the "L" shape structure of the indoor environment, the inside position has restricted detection area compared to the outside position. The laser range finder in the picture has 4 meters distance as its effective detect distance and the size of indoor environment is not very large relatively. Therefore, no

matter that Ordinary Grid Algorithm is used or Hierarchical Random Algorithm is used, the outside position usually has larger detect area than the inside one and the final result is misled by the outside candidate points absolutely.

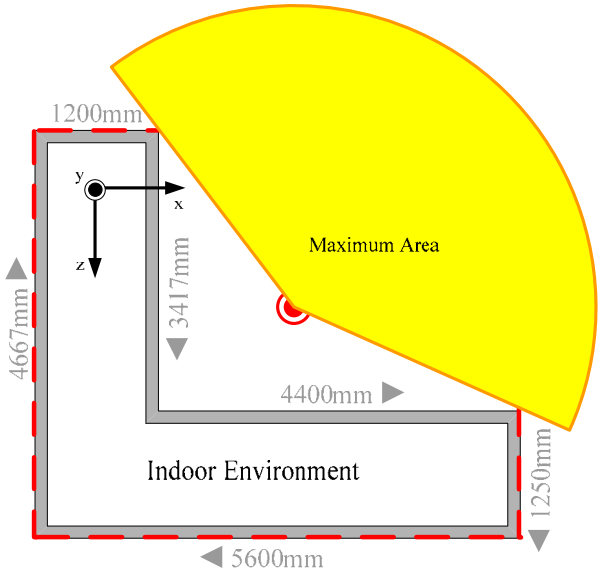


Fig.4. 8 Sample of Misleading Detection Result without Insider Detection

In order to judge whether the candidate position selected by the previous algorithm is inside or not, the following rules apply to judge the status of candidate position.

1) Get one candidate point in the square area by using the previous algorithm.
2) Find a known insider point in the indoor environment.
3) Connect the candidate point to the known inside point, link up a line segment.
4) Judge the candidate point according to the following rule: if the connected line segment crosses even number of borders, it is an inside point; otherwise it is an outside point.

This rule can be proved by topological theory and its concept can be seen in Fig.4.9. In the figure, two outside points have one and three crosses with the borders on the line segment linked to the known inner point while two inside points have none and two crosses with the borders on the line segment linked to the known inner point. Although the shape of the indoor environment can be very complicated, if only the environment is closed this rule works.

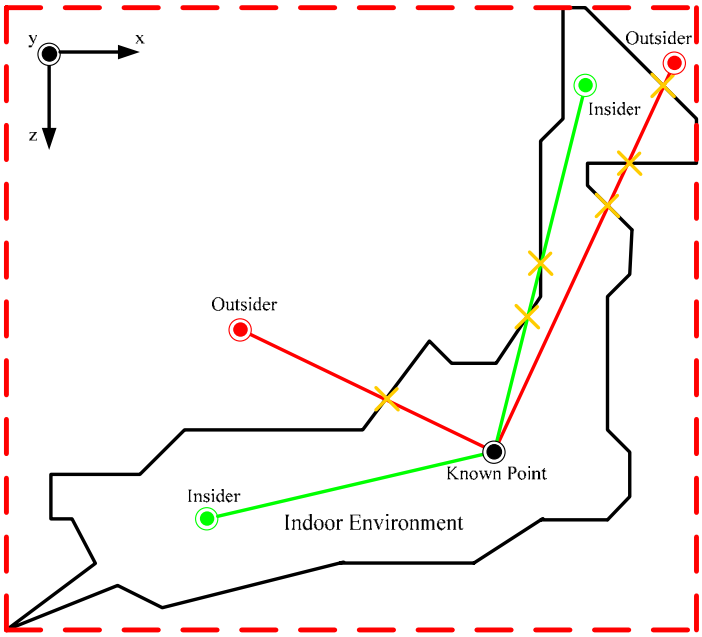


Fig.4. 9 Cross Judgement of Insider Judge Algorithm

As the judgment of whether two line segments cross with each other, the following algorithm can be applied.

1) Substitute one point of Line Segment A into the equation of Line B to get the function result.
2) Substitute the other point of Line Segment A into the equation of Line B to get the function result. If the two results have same signs, they are on the same side of the Line B. Else, they are on different sides of Line B and Line Segment A may cross Line Segment B.
3) Conversely, substitute one point of Line Segment B into the equation of Line A to get the function result.
4) Substitute the other point of Line Segment B into the equation of Line A to get the function result. If the two results have same signs, they are on the same side of the Line A. Else, they are on different sides of Line A and Line Segment B may cross Line Segment A.
5) If the two points of Line Segment A are on different sides of Line B and the two points of Line Segment B are on different sides of Line A, Line Segment A crosses Line Segment B, otherwise, they do not cross each other.

4.4.5 Along Wall Algorithm

Beside of ordinary exploration or random methods to explore the inner potential position of placing laser range finder, there is necessity to consider the placement along the wall only. There are two

reasons for this consideration. First, if the sensor is put in the middle of the indoor environment, despites its big scanning area, itself becomes obstacle to actuators or human beings. Second, the arrangement along the wall usually shows good scanning effect and it can be said a common sense to put laser range finder against the wall for many people.

I have done some investigation about the common way for various people to arrange laser range finder. The Fig.4.10 is an arrangement scene I used to ask for answers.

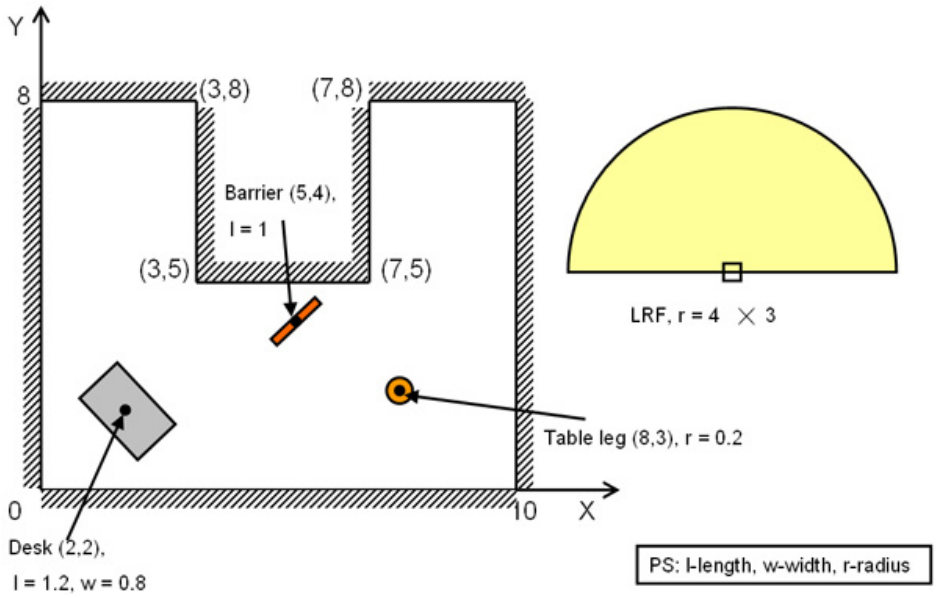


Fig.4. 10 Questionnaire for Arrangement from Various People

After analyzing the result from ten various targets to make the arrangement have biggest scanning area but smallest intersection between sensors, I find that 80% people firstly consider the arrangement against the wall. In Fig.4.10, the most common answer is to put the sensor along the left wall, the right wall or the bottom wall. Because in these ten targets, there are professionals with much experience in sensor arrangement, I think the algorithm of arranging laser range finder along the wall is a good choice for most users.

In order to do the scanning along the vector of the wall, the Along Wall Algorithm is as follows.

1) Calculate the perpendicular direction of the vector of the wall's line.
2) Use the known insider point to judge the inner normal direction. Take advantage of Insider Judge Algorithm.
3) Along the inner normal direction, move "dist" length to the inside of the wall line.
4) Create a loop list to save all the new created vector lines and their corresponding inner normal direction for further use.

Because the sensor can not be really embedded into the wall, there must be some distance from the wall to the arranging position of the sensor. Therefore we define “dist” parameter for user to choose as the distance of sensor to the wall. As about the orientation of sensor, usually in the case of 180 degree as the range of laser range finder, the perpendicular orientation of the wall line vector should be the optimal choice. But at the end of the wall line vector, concavity environment may ask for more candidates of orientation. In other case like that the range of laser range finder is 240 degree or 270 degree which is the current available range, inner exploration strategy like Ordinary Grid Algorithm or Hierarchical Random Algorithm can be better choice. The work of the above algorithm can be seen in Fig.4.11.

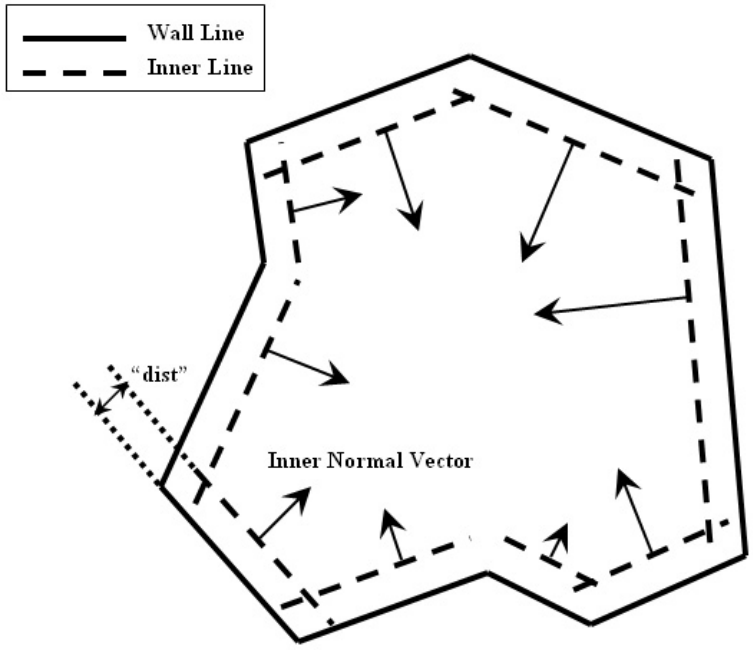


Fig.4. 11 Concept of Along Wall Algorithm

4.4.6 GUI Selection Zone Algorithm

There are many situations that user does not want the scanning result of the whole environment only, they may also want the arrangement in specific place limited by a polygon consisted by several simple lines. In order to support our potential users well, the arrangement target zone selection GUI has been developed and that can be seen in Fig.4.2. The bottom four buttons are used for zone selection.

In many cases, user may want to select a rectangle zone, therefore “select” below the tag “rectangle” is for this use. People can use mouse to do this work. First click the left button of mouse to select the start point of the rectangle, do not release the button and drag the rectangle to choose the expected place. If the created rectangle is not satisfying, user can click the “select” button again to recreate. The

view mode of selection zone is bird-view with specific height to localize the mouse position.

However, polygon can be used to cope with all situations because polygon can be used to make all kinds of shape. But there is a problem for further exploration in the situation of polygon. As we use the Insider Judge Algorithm to judge whether the test position is inside of a polygon or not. One inner point must be known before exploration. In the case of indoor environment, the origin works as the inner point while in the case of polygon selection it seems difficult to get the exact inner point using only end point positions of a polygon [45]. To cope with this problem, the center of gravity has been calculated to be the default inner point and user can see its position after decide the shape of polygon. If the center of gravity is out of the polygon, we recommend user to select a new district. The creation of polygon can be realized by simply clicking several points and use “stop” button below the tag “polygon select” to finish the polygon. The points of polygon are saved in a loop list because of its dynamic mechanism.

4.5 Evaluation Factors

The evaluation of scanning result decides the ranking of all candidate position and orientation. The valuation of sensor arrangement is an important part of our automatic arrangement mode. Although experts and veterans can valuate the arrangement by only watching or checking sensor data relying on their abundant experience, we believe that ordinary users look forward to more intelligent assistance. Therefore we propose following evaluation factors in the next sections.

4.5.1 Scanning Area for Singe Sensor

Here the sensor is referred as laser range finder and detection area is the most important evaluation factor to laser range finder or other range sensors like 3D ranger. Because of their limited detect distance and range angle, their detect area is limited as well. In most cases, user wants maximizing the capability of range sensor by maximizing the detect area.

In order to calculate the scanning area of laser range finder, we use slicing methods to accumulate the value. The concept can be seen in Fig.4.12. In the figure, we can see the half circle laser beam area can be approximately measured by many slices with different detected distances. Videlicet, According to the resolution of the sensor, the half circle is divided into many pieces. Because of the limitation of resolution, some details of the obstacle can not be caught absolutely accurately by the sensor. For example, about 90 degree of the laser beam, the distance of one beam and the next one connects the line segment which is not very well defined the contour of the desk. Anyway, different with this figure, actual resolution is far better than that in the figure and has good enough accuracy for the evaluation of laser range finder.

The following formula is used to calculate the scanning area of laser range finder.

$$Area = \sum_{i=0}^{k} \frac{reso \cdot \pi \cdot dist_i^{\,2}}{360} \tag{4.4}$$

$$k = \frac{180}{reso} \tag{4.5}$$

$Area$ is the scanning area of laser range finder.

$reso$ is the current resolution of laser range finder, the unit is degree.

$dist_i$ is the i th laser beam detected distance value

k is the maximum number of laser beam.

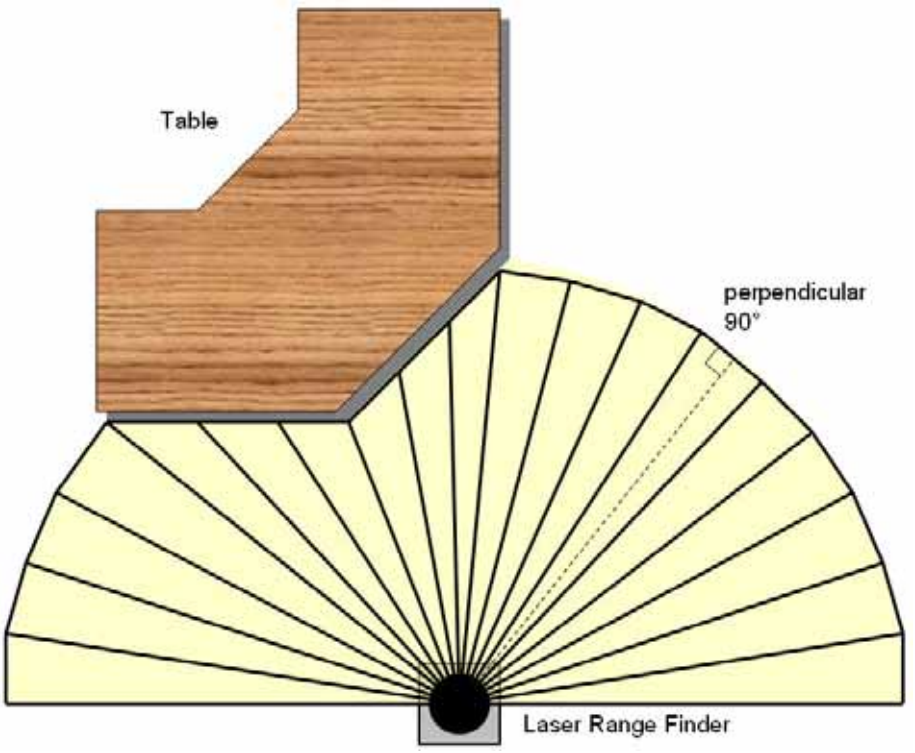


Fig.4. 12 Scanning Area Calculation of Single Laser Range Finder (Top View)

4.5.2 Total Effect Area

While single sensor detect area can be calculated simply by applying the above equation, in the situation of multiple sensors, the intersection becomes a problem for us to make full use of the capability of the sensor. Usually in our arrangement we consider the distance of sensor as its accurate working distance. Therefore, the intersection of multiple sensors means the waste of overall distributed sensors' capability. The intersection problem is not as easy as we have expected. In the case of two semi-spheres, the intersection is so hard to calculate because of their discretional radius and orientation. If there are more than two sensors in complicated environment, it is nightmare to do this work.

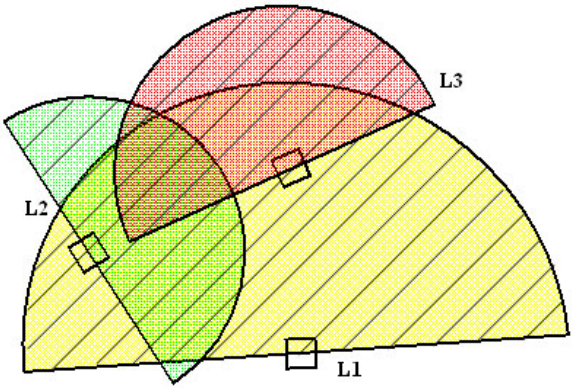


Fig.4. 13 Puzzle of Multiple Sensor Area Calculation

From the Fig.4.13 we can see that even three sensors without any obstacles inside their scanning area bring us big problems to calculate the intersection between different sensors. However, we solved this problem without considering the intersection but get the contour of all sensors' area instead. This area is the truly total area of multiple sensors, so we call this area as "Total Effective Area".

The calculation of total effective area is as follows.

1) Get Limit Points of the whole scenario, use Limit Acquire Algorithm.
2) Create grid maps of the external rectangle area. The grid number in our program is 100*100.
3) Generate the insider point for every sensor. We set it as one point close but in front of the sensor's central position. The distance is 1 percent of sensor's effective distance, therefore in most cases, it can be the inside point.
4) Judge every grid using Insider Judge Algorithm to record whether one grid is belonging to any sensor.
5) Make "OR" logic operation to filtering all grids, aggregate the covered grids.
6) Use the following equation to calculate the total effective area.

$$Area_{TotalEffective} = \frac{N_{Effective}}{P \cdot Q}(X_{\max} - X_{\min})(Z_{\max} - Z_{\min}) \tag{4.6}$$

$Area_{TotalEffective}$ is the total effect area of multiple sensor.

$N_{Effective}$ is the number of effect grids, in another word, the covered grids.

P and Q are the grid number in x direction and z direction. Our default setting is 100.

$X_{\max}$, $X_{\min}$ are the maximum and minimum x position calculated by Limit Acquire Algorithm.

$Z_{\max}$, $Z_{\min}$ are the maximum and minimum z position calculated by Limit Acquire Algorithm.

4.5.3 Coverage

After the calculation of Total Effective Area for multiple sensors, user should know the coverage of current sensor arrangement against the current indoor environment. “Coverage” here means the covering percent using current sensor arrangement.

Before its calculation, the area of indoor environment should be solved first. We use the following algorithm to solve this problem.

1) Get Limit Points of the whole scenario, use Limit Acquire Algorithm.
2) Create grid maps of the external rectangle area. The grid number in our program is 100*100.
3) Judge every grid using Insider Judge Algorithm to record whether one grid is belonging to any sensor. The known point is the origin.
5) Make “OR” logic operation to filtering all grids, aggregate the covered grids.
6) Use the similar equation to (4.6) to calculate the total effective area.

The calculation of environment area is similar to that of sensors’ covering area, but simpler. After this calculation, we can calculate the coverage using the following equation.

$$C = \frac{Area_{TotalEffective}}{Area_{Environment}} \times 100\% \tag{4.7}$$

C is the Coverage we discussed.

$Area_{TotalEffective}$ is the Total Effective Area of sensors which can be acquired using equation (4.6).

$Area_{Environment}$ is the area of whole close indoor environment.

Fig.4.14 shows the calculation of environment area. In our program, the world coordination is set as x and z direction for the horizontal plane. The environment in the figure is a not typical but possible indoor environment. In order to calculate its area, its limit point is taken first. Then the grid map is created to separate the space into many little pieces as the integral usually does. By using Insider Judge Algorithm, every grid can be judged as the insider grid or outside grid of the environment polygon. After counting the number of covered grids, the area is possible to calculate using equation (4.6). Here we can see better precision can be available if only we enlarge the grid number of gird map. However, we do not want to give heavy burden to computer for this calculation and according to our statistics, the grid number at about 10 thousand (100*100) is appropriate to be a tradeoff of precision and performance.

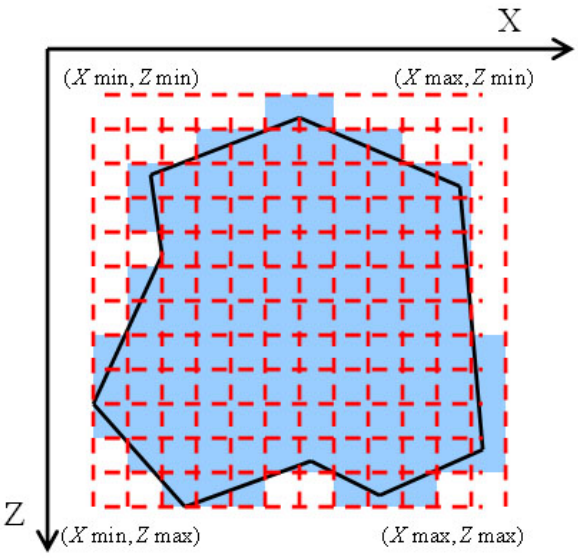


Fig.4. 14 Calculation of Environment Area

4.6 Execution and Statistics

Execution here is a sample of executing arrangement, because we have parameters for user to set, there can be many possible result of automatic arrangement. Here we discuss the influence of parameters and compare the three different arrange strategies. At the end of this section, some statistics during the execution and trials are listed for reference.

4.6.1 Execution and Trials

The target arranging scenarios are the arrangement problems we have to make arrangement. It can bee seen in Fig.4.15.

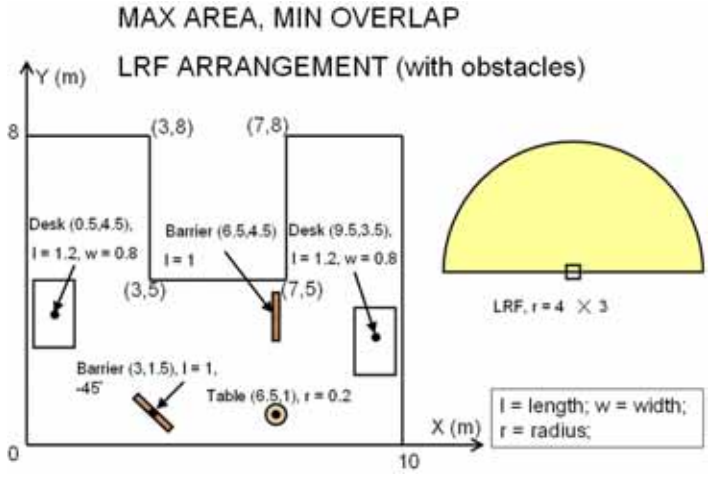
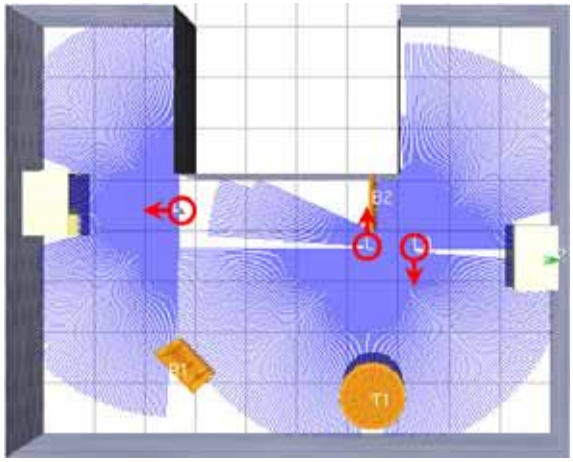
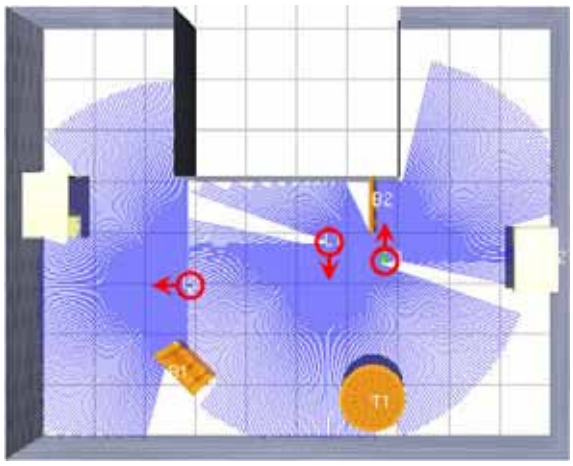


Fig.4. 15 Scene of Arrange Problems

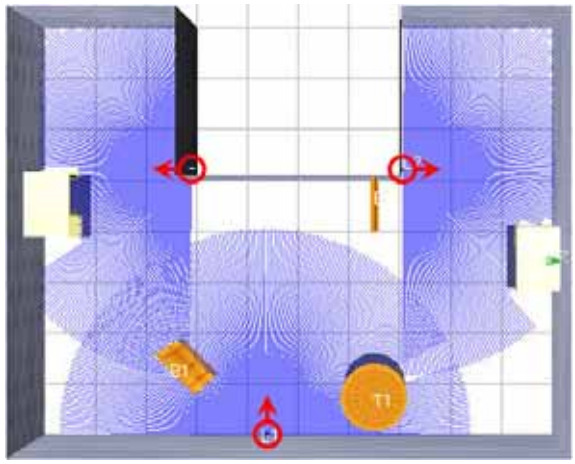
We can see that there are many facilities in this scene and they will apply different kind of influence to the arrangement of laser range finders. Three laser range finders with effective distance as 4 meters, range as 180 degree will be arranged in this scenario. The execution result can be seen in Fig.4.16.



(a)



(b)



(c)

Fig.4. 16 Execution of Auto Arrangement

(a) Ordinary Mode Arrange Result; (b) Random Mode Arrange Result;

(c) Along Wall Mode Arrange Result;

4.6.2 Result

From Table.4.1, we can find that the automatic arrangement mode in Sensor Arranger works well with complicated indoor environments. Whether there are facilities or obstacles in the environment, the above algorithms can work properly. Although Ordinary Grid Algorithm shows best performance, it spent longest time to get the best result. Meanwhile the Along Wall Algorithm shows not bad result and it is the fastest algorithm among three. However, it can not work well if there are many facilities along the wall to make the laser range finders scan nothing at all. Besides, in the case that sensor's range is more than 180 degree, this algorithm will certainly be exceeded by other two algorithms because the part more than 180 degree of its scanning range will be wasted. As to the random algorithm, different with ordinary algorithm whose exploring time increases with the increase of exploring area, will outperform other two algorithms in wider area. Therefore all the three strategies have their advantages and disadvantages in our conclusion.

Table.4. 1 Data of Execution Result

Strategy	Parameters	Coverage	Elapsed Time	Average FPS
Ordinary	X = 11, Y = 11, Deg = 8	80.8%	8 minute 40 sec	330
Random	R1 = 20, R2 = 5, Deg = 8	73.9%	3 minute 50 sec	320
Along Wall	Dist = 0.05m, times = 21	80.6%	1 minute 50 sec	320

4.7 Summary

In this chapter, the function of automatic arrangement has been explained in detail. From the closest part to user, the GUI to the deepest part in the program, the algorithms, most important points have been narrated. At the end, the execution of automatic arrangement and some statistics data were given.

CHAPTER 5

Simulations and Experiments

In this chapter, the realization of simulation stage will be introduced through discussing main steps in it. After that, three simulations with respective aims were executed and will be discussed here. Their relevant real experiments will be included as well and the comparison and evaluation will be given. Beside of the three main simulations, test of new ideas of using laser range finder with the simulator and other applications will be introduced at the end.

5.1 Simulation Runner Structure

Simulation Runner is the stage for all simulation including the simulation of mobile robots and distributed sensors. The structure of Simulation Runner is consisted of three main sections for the simulation of mobile robots: communication protocol to connect outside control programs, kinematics of mobile robot and the essential collision detection.

5.1.1 Simulator TCP Protocol

We defined our own TCP network protocol for simulator named Simulator TCP Protocol. This protocol includes robot version and sensor version. In once network communication, our Simulation Runner works as the server for any service like the interface to receive command to control robot movement, the interface to transfer sensor data to outside program. Due to the utilized multi-thread style programming, multiple services can be offered simultaneously. It means that all available actuators and sensors can be accessed by outside client through this Simulator TCP Protocol.

5.1.2 Kinematics

In our Simulation Runner, the kinematics for mobile robot is different with actual robot control. Because in simulator the control of mobile robot's two main parameters: speed and rotation speed can be accurately controlled. Therefore no specific pattern of control is applied here and it meets the expectation that what is wanted in simulation is theoretical behavior of all objects include actuators and sensors. The mobile robot model comes from Pioneer 2DX [46], indoor type 2 wheels mobile robot. The kinematics of mobile robot is discussed in the following equations. The speed of mobile robot in robot coordination system is υ, the rotate speed is ω. The current coordination of mobile robot in the world coordination system is (x, y).

$$u = \begin{bmatrix} \upsilon \\ \omega \end{bmatrix} \tag{5.1}$$

By using the above speed and rotation speed in robot coordination system, the speed and rotation speed in world coordination system can be derived as the following.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} u \tag{5.2}$$

To apply the acceleration in mobile robot, the acceleration can be derived by the following equation. a is the acceleration of mobile robot in simulator, set at the same value with real Pioneer 2DX, a_r is the rotation acceleration of mobile robot, set at the same value with real robot.

$$\dot{\upsilon} = a \tag{5.3}$$

$$\dot{\omega} = a_r \tag{5.4}$$

The Kinematics model of 2 wheeled mobile robots can be seen in the Fig.5.1. (X_W , Y_W) is the world coordination system and (X_R , Y_R) is the robot coordination system.

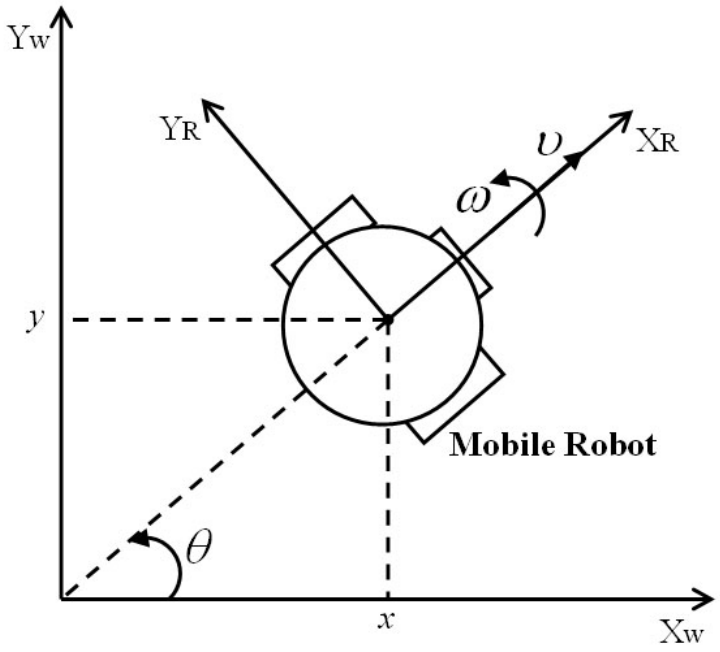


Fig.5. 1 Kinematics Model of Mobile Robot

5.1.3 Collision Detection

The collision detection is an essential part of robot simulator. Actually it is not only for robot simulation, but also for human models and all kinds of actuators' action. In order to give the 3D models the most basic physical attributions, collision detection is needed to make them avoid binding together.

In our Simulation Runner, collision detection is realized by using obstacle map which is often used in robot's path plan [47]. Obstacle map in our simulator is the map of model contours of all objects in the virtual space. Because what is going to be done is mobile robot's simulation, 3D obstacle map has very high cost in calculation. Therefore 2D obstacle map is used in our program. However the height of this 2D map is on the height of top of mobile robot in the simulator. The merit of using 2D obstacle map is that its detection overhead is low so that the very frequent of this diction can be executed fast enough to nor hurt the whole performance very much. Because of the continuity of the object in height, the 2D obstacle map does not cause big miss in collision detection.

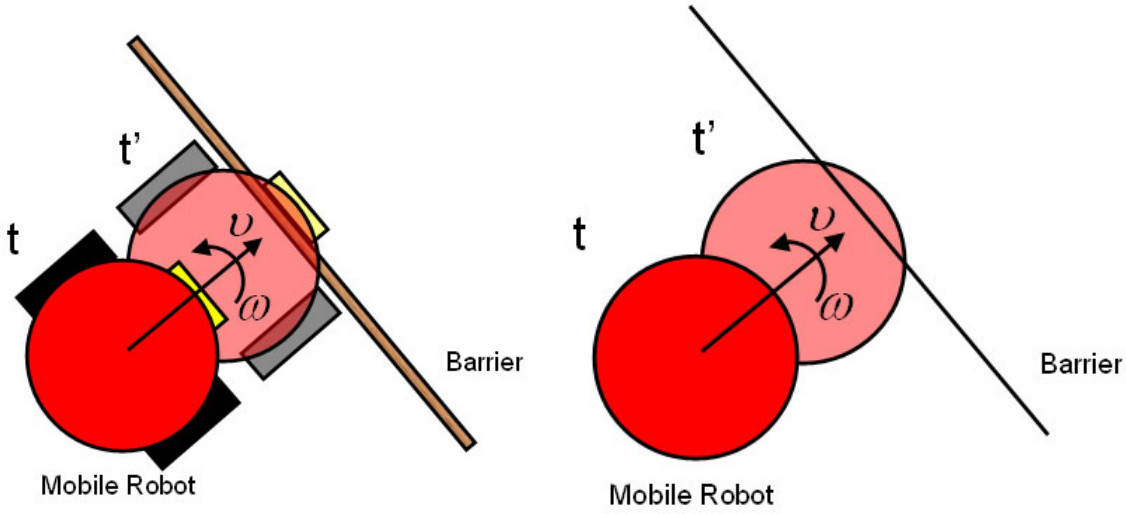


Fig.5. 2 Bumping of Mobile Robot represented by Simple Shapes

In Fig.5.2 the bumping of mobile robot into barrier can be seen. At the next step of mobile robot's movement, the collision detection gives out warning because the abstracted circle which represents mobile robot is crossing the line which represents the barrier. Because the objects in simulator should not fuse with other objects, the next step of mobile robot's movement will be cancelled and the warning that currently collision happens will be given to the user. In the same way, human model's collision with other obstacles or moving objects can be detected.

5.2 Robot Obstacle Avoidance

Among mobile robot's applications, obstacle avoidance is one of the most important one. Many people may consider it exists without doubt. However the realization of obstacle avoidance is not easy to realize. During once avoidance there are several steps for the robot and its onboard sensor [48] to work out. In our research, the obstacle avoidance algorithm we used is the mixture of Path Planning Algorithm, Dynamic Window Algorithm and Extended Kalman Filter Algorithm [49].

Before the simulation some work must be done to connect the control program to our Simulation Runner. The reload of Aria Class for Pioneer 2DX Mobile Robot is the preparation for the further simulation in our simulator.

5.2.1 Aria Class Reload

The Origin Aria Class is the class that has been used during the control of Pioneer 2DX Mobile Robot. In order to change the communication port from real robot control relative functions to our simulator, the corresponding changes are necessary to do with all the functions used in the control program. To offer convenience to the further research, making a simulation version Aria Class should be helpful. Therefore the user can use the origin program with the only change of the include head file from <Aria.h> to <Sim_Aria.h>. The functions used in the program of obstacle avoidance are listed in Table.5.2.

Table.5. 1 Usual APIs of Aria Class

Origin Aria Class	Functions
ArRobot	Aria(); init(); shutdown(); addRangeDevice(); setDeviceConnection(); blockingConnect(); comInt(); runAsync(); isLeftMotorStalled(); isRightMotorStalled(); getVel(); getRotVel(); unlock(); stopRunning(); disconnect(); lock(); setVel(); setRotVel();
ArPose	ArPose();
ArSonarDevice	ArSonarDevice();
ArSerialConnection	ArSerialConnection(); open();
ArSignalHandler	ArSignalHandler(); unhandle();
ArCommands	ArCommands();
ArUtil	ArUtil();

From the Table.5.1 we can see that almost all important relative functions are in the class of “ArRobot”, therefore it is the main target for us to revise for communicating with our Simulation Runner. In Fig.5.3 the communication from real robot port to simulator port can be seen. Outside program communicates with the simulator through TCP protocol and simulator protocol. The API to check the collision status of real robot is now converted to check the mobile robot in our simulator. The loading and setting of speed or rotation speed works in the same way. By so, the outside program can work in the same way as it usually does without changes of any codes in the program. As to the other APIs of Aria Class, because the theoretical situation of simulator, many APIs like locking the robot to set parameters are simplified in the situation of virtual mobile robot.

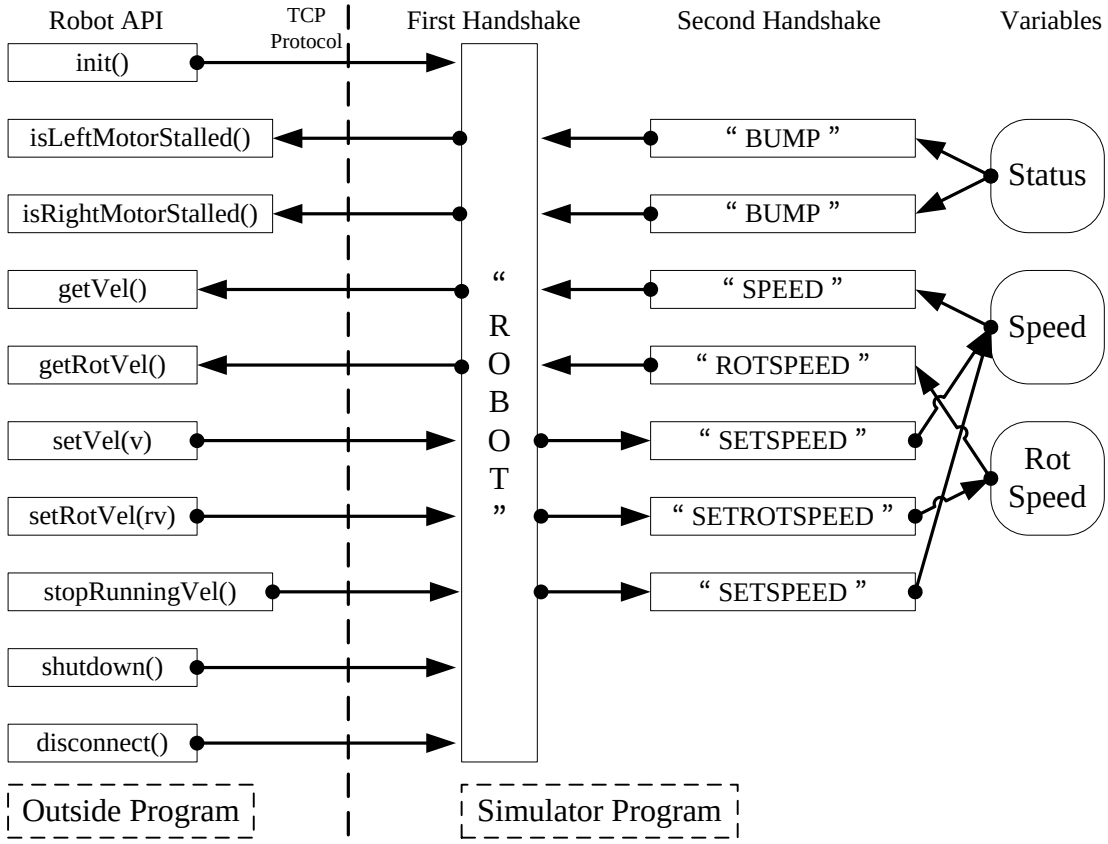


Fig.5. 3 Port Conversation from Real Robot to Simulator

5.2.2 Simulation and Experiment

As a typical indoor environment, the scene of this robot simulation and experiment is in out laboratory. In the center of the laboratory, there is a zone about 8 meters long and 8 meters wide called ZPS zone. In this zone, the position of robot of people with ultrasonic tag can be detected by the ultrasonic positioning system called ZPS [50]. The layout of this experiment can be seen in the Fig.5.4.

In the central zone of our Intelligent Space, the robot started at (0.5, 2) and the target is at (0.5, -2). In front of its moving road, there is one barrier on the way. In normal situation without obstacle avoidance the robot will go forward and bump into the barrier. In order to test our obstacle avoidance algorithm, this is a good scene. Beside of the front barrier, the left barrier which is perpendicular to the front one is used to make the robot decide to go left after starts. Because the front barrier is on the middle of the road, if there is no left barrier to isolate the right road, the robot can not make definitely the same path plan according to the offset at the start point. And there are other two barriers on the left of the road of robot. They are used to isolate the influence of peripheral objects in the laboratory.

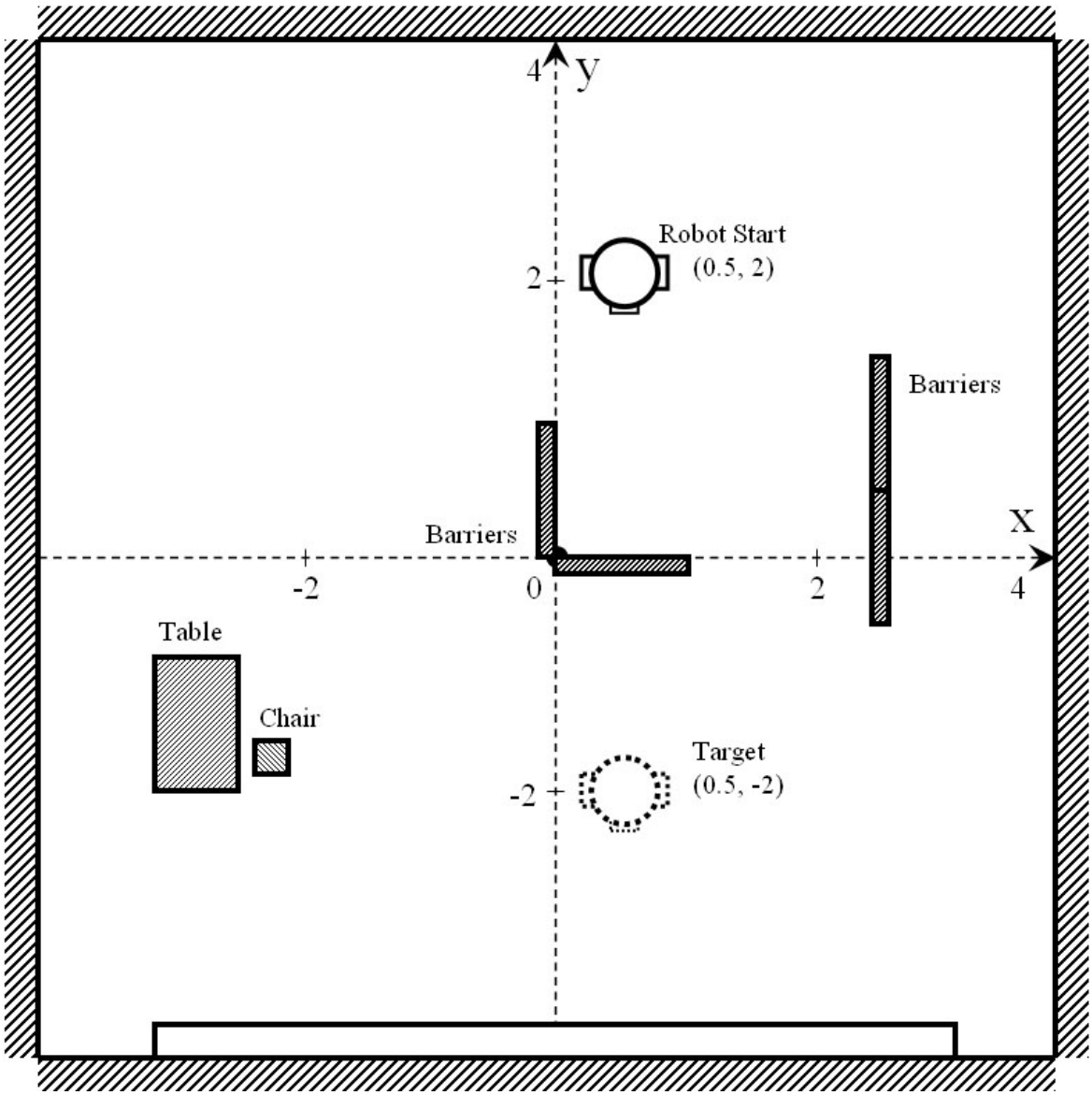


Fig.5. 4 Layout of Robot Experiment

The scene in simulator was created as close as the actual. The photo of the experiment scene and the screenshot of that in simulator can be seen in Fig.5.5 and Fig.5.6 separately. As Fig.5.6 shows, two distributed laser range finders (red arrowheads) are used in simulator to track the movement of robots at the same time. All facilities and objects are arranged in the same layout. Robots are both at the start point and ready for experiment. The real experiment can be seen in Fig.5.7 in 9 steps. From start to the end, the robot moved round the front barrier and found its way to the target. According to the setting in the program, the robot thinks it has reached the target by calculating its own position according to the encoder data. If its position is inside the circle with 0.4 meter radius around the target, it stops and the program is over. In Fig.5.8 is the simulation step figure, the last three pictures are in different view angle with the previous six. But the trajectory of robot can be validated by observing its position with the black-white blanket.

Fig.5. 5 Real Scene of Robot Experiment

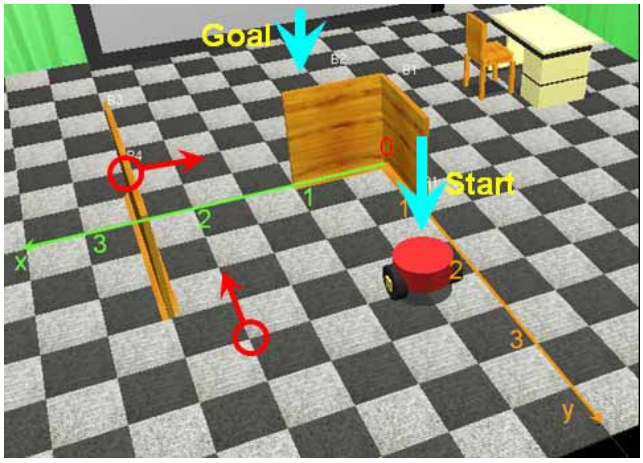


Fig.5. 6 Simulation Scene of Robot Experiment

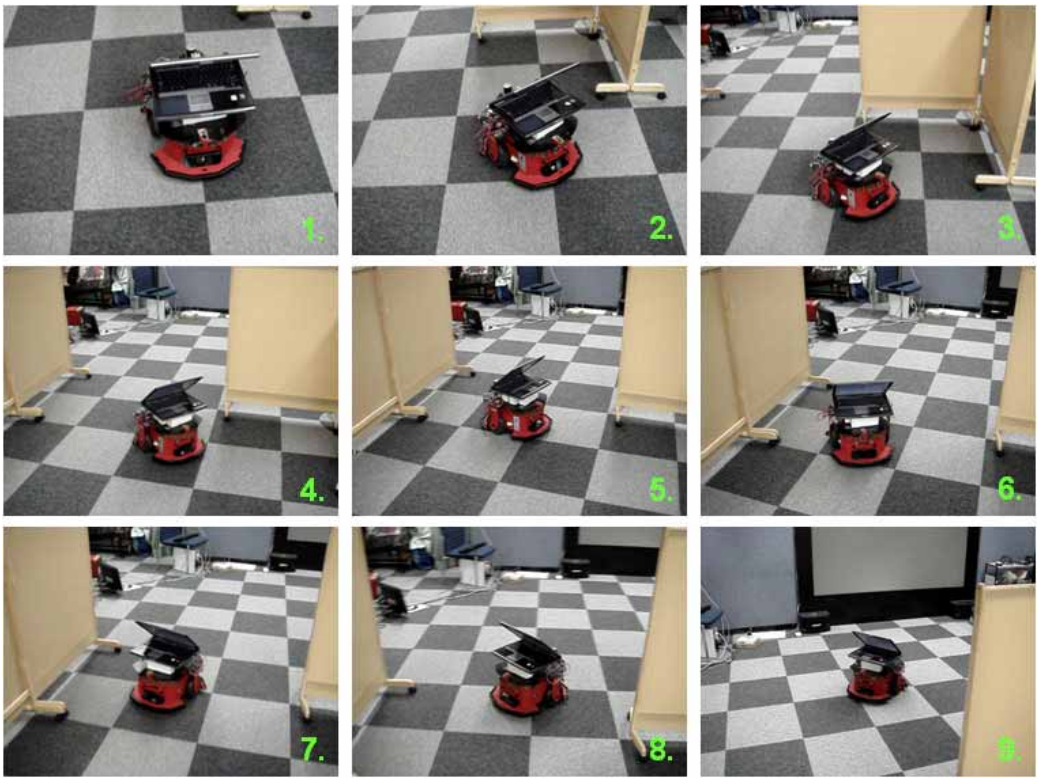


Fig.5. 7 Real Robot Experiment Step Figure

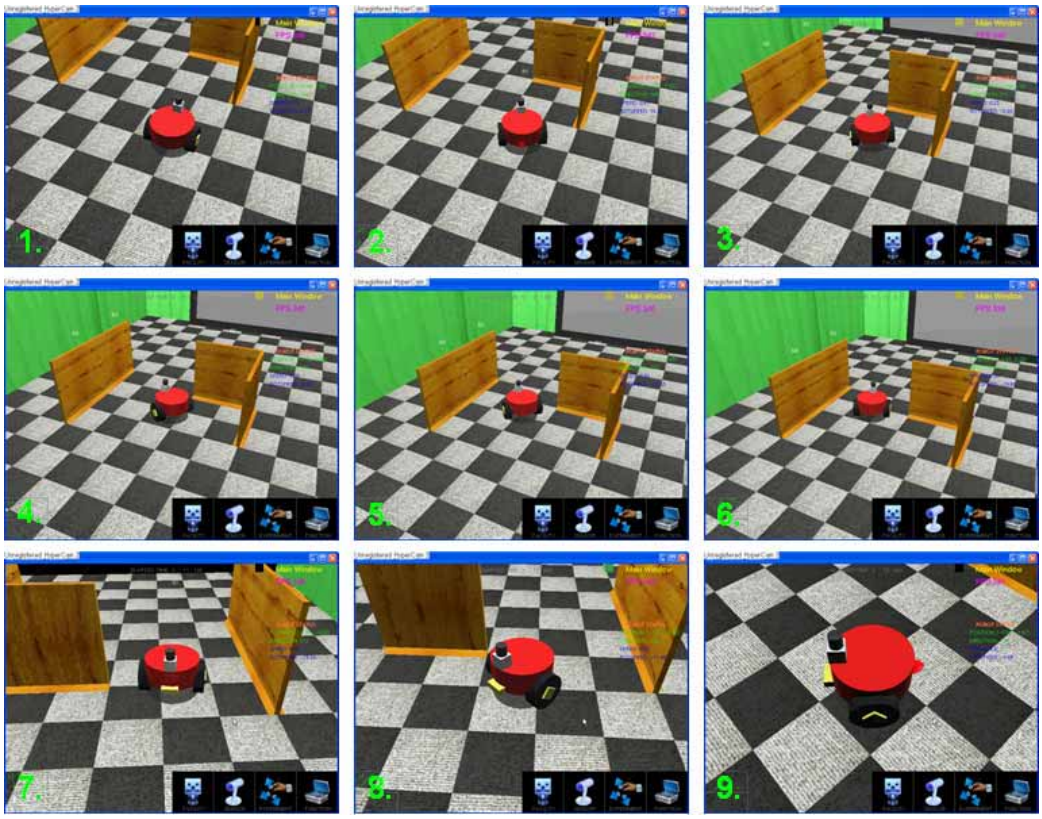
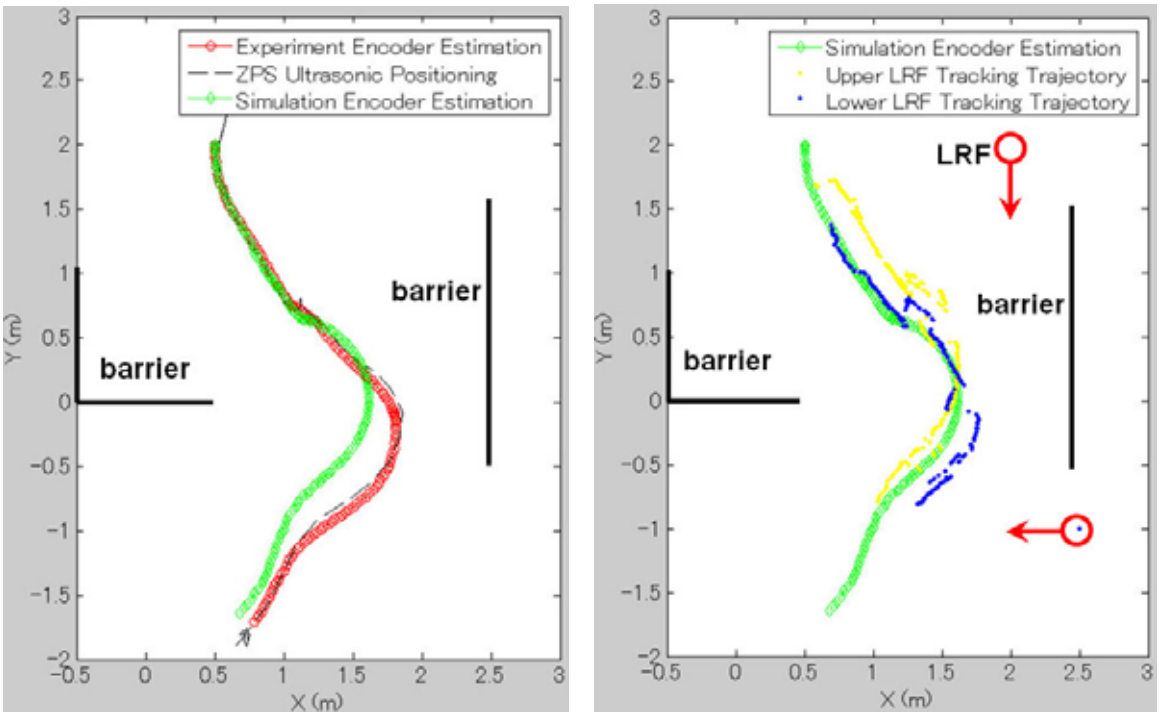


Fig.5. 8 Simulation Step Figure

Fig.5.9 (a) shows the trajectory of real mobile robot and simulated robot. As the result of experiment and simulation, both mobile robots reached the goal and avoided bumping into the obstacles. In this sense, our simulation is proved correct according to the algorithm. However, we can see that the movement of mobile robot in real experiment is different with that in simulation. In the latter part of the movement, the mobile robots took different path to reach the goal. About the reasons, three factors are expected to have influence in this difference. First, in real world, sensor noise is an important factor while in simulation we have pure sensor data; secondly, physical attributes like friction is also not included in our simulator; the third, the original posture of mobile robot in real experiment is very hard to adjust very accurately.

Fig.5.9 (b) shows the tracking result of two distributed laser range finders and the trajectory of mobile robot in simulation. The positions and orientations of two distributed laser range finders are shown as the red arrows. Taking the simulated trajectory of mobile robot as benchmark, we can see the tracking results of these two sensors have big bias. One expected reason is that tracking error will enlarge when the tracking target moves not linearly. Another reason is the modeling of mobile robot in the tracking program does not match the actual situation very well.



(a) (b)

Fig.5. 9 Experiment and Simulation Trajectories

(a) Trajectory of Real Robot and Simulated Robot;

(b) Trajectory of Simulated Robot and Tracking Result.

5.3 Distributed Laser Range Finder Tracking System

Here the experiment in Hashimoto Lab. will be discussed with the result of simulation of tracking system using distributed laser range finder in our environment simulator. In usual experiments, tracking system needs many humans or autonomous robots to held experiments, but actually it is not easy for us to gather so many people to meet the need of experiment in many cases. Furthermore, we have only two mobile robots in our laboratory, therefore it is impossible to do tracking experiment with more than two mobile robots. Thus we create human model in our environment simulator to meet needs of many tracking experiments.

5.3.1 Human Model

In the simulation of tracking program of distributed sensors, the attendance of human model is necessary because out tested program is designed for tracking human beings in the real world. In the design of the tracking program, human's feet have been modeled in order to improve the tracking ability. In this tracking system consisted of distributed laser range finders, at the same height of sensor's scanning horizontal plane the shape of human's feet are modeled as two circles [51].

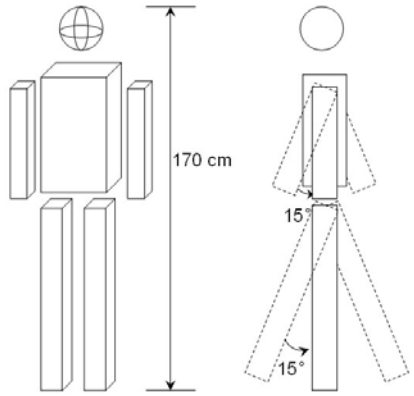


Fig.5. 10 Human Model in Simulator

The human model can be seen in Fig.5.10. The human model is consisted with very simple geometric objects. The head is using ball, arms, legs and body are all using cube to represent the main part of human body. The height of the human model is set at 170 cm which is a normal height for amateur males. In order to realize the action of swinging arms and legs, automatic swing to 15 degree at most is used in our simulator. Besides, the movement of human model can be referred to by the following formula. As we have mentioned in the previous section, the collision detection of human model is also applied to avoid bumping into other objects during his movement.

5.3.2 Distributed Laser Range Finder Tracking

Tracking of mobile robots, human beings is an important part in Intelligent Space. There are many devices can be used to track active targets. Among these devices, laser range finder wins researches' eyes by its high accuracy and frequency of scanning. Different with other sensors like camera, the use of range sensor is easier and the sensor data itself usually does not need too much processing while to camera, image processing is firstly a difficult task. As discussed in the previous part, we are using distributed laser range finder to track human beings and mobile robots in our Intelligent Space. As the environment simulator, this tracking task should be available in simulation so that it can be helpful for tracking algorithm improving or recreating.

The human model was discussed in the last section and we did several real experiments with real people's attendance to make comparison with the simulation result.

The experiment scene is designed as Fig.5.11 shows. In the middle of our Intelligent Space, several barriers and facilities were used to enclose a small space for tracking experiment of laser range finder. In order to make the tracking closer to daily environment, we put two chairs into the enclosed environment. In such a small place, two laser range finders with effective distance at 4 meters and range as 180 degree are enough to do the work. Therefore we apply this scenario into our Sensor Arranger to make the semi-automatic arrangement and use the result to do the experiment.

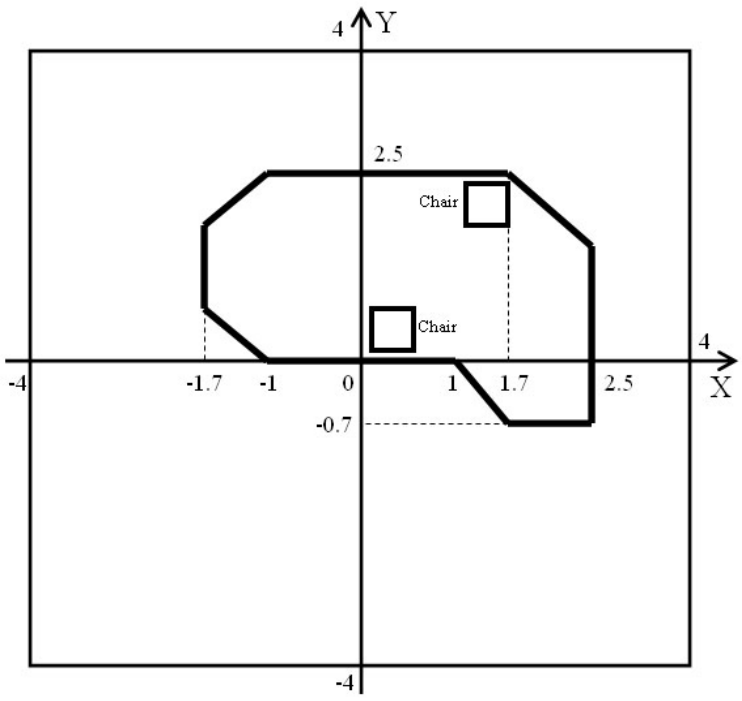


Fig.5. 11 Environment for Tracking Experiment

Fig.5. 12 Real Scene for Tracking Experiment

The real scene of experiment can be seen in Fig.5.12. Because we have not enough barriers to enclose the place, some other facilities are used to make up the insufficiency. The best arrangement result can be seen in Fig.5.13, the coverage is 97.5% when two laser range finders with effective distance as 3 meters are used for this arrangement. The coordination system is translated from x-y to x-z in our program.

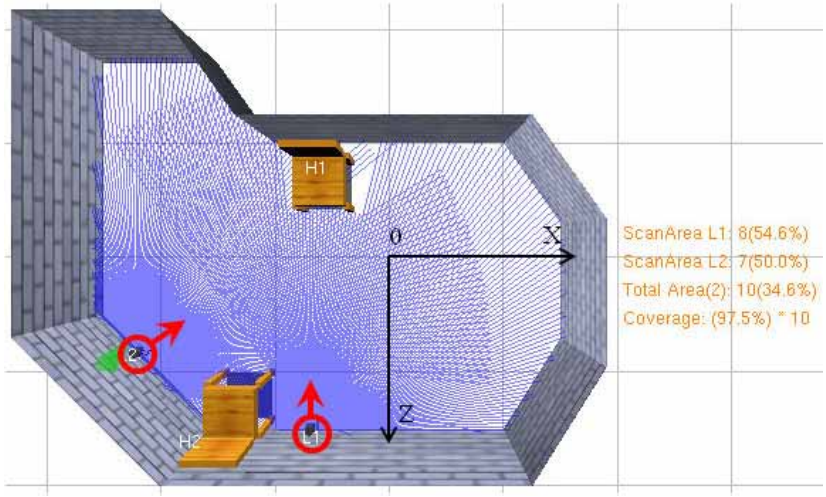


Fig.5. 13 Arrangement Result for Tracking Experiment

The parameters and result of this arrangement is listed in Table.5.2.

Table.5. 2 Parameters and Result of Arrangement for Tracking Experiment

Mode	Parameters	Final Sensor Position	Coverage	Elapsed Time	FPS
Along Wall	dist = 0.05m times = 11	(-0.67, 1.45), 180 degree (-2.13, 0.80), 135 degree	97.4%	1 minute 10 sec	320

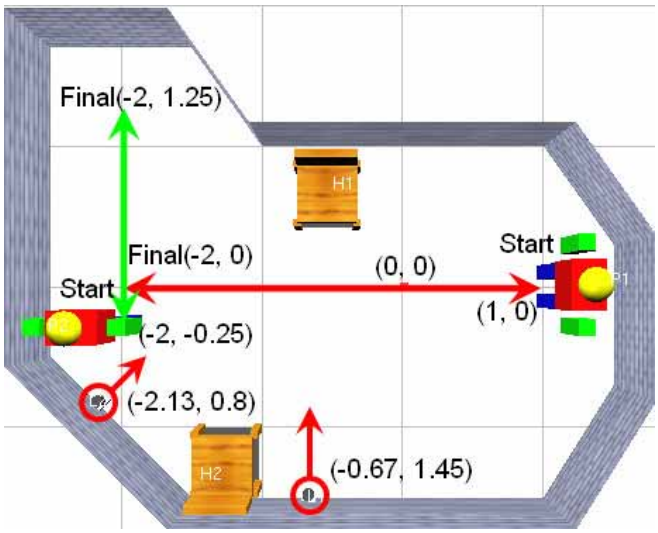


Fig.5. 14 Paths of Two Persons Walking Experiment

As the Fig.5.14 shows, two persons are walking along their own path. The left person is walking vertically while the right person is walking horizontally. They may meet with each other in this scenario. The comparison of experiment and simulation can be seen in Fig.5.15 which is the comparison of the trajectories of the left person tracking by the right sensor. The green little triangles represent the trajectory of the left person tracking by the right laser range finder in experiment. As the figure shows, the left person seems that have walked both vertically and horizontally, but actually he only walked vertically. The red little circles which represent the trajectory of the left person tracking by the right laser range finder in simulation shows the similar result. It is obvious that the tracking of sensor mixed the trajectories of different people. As to the reason, we found that the used algorithm can not distinguish the true target when two different people meet together. In both the experiment and simulation, this situation took place many times so that the mixture of tracking result happened.

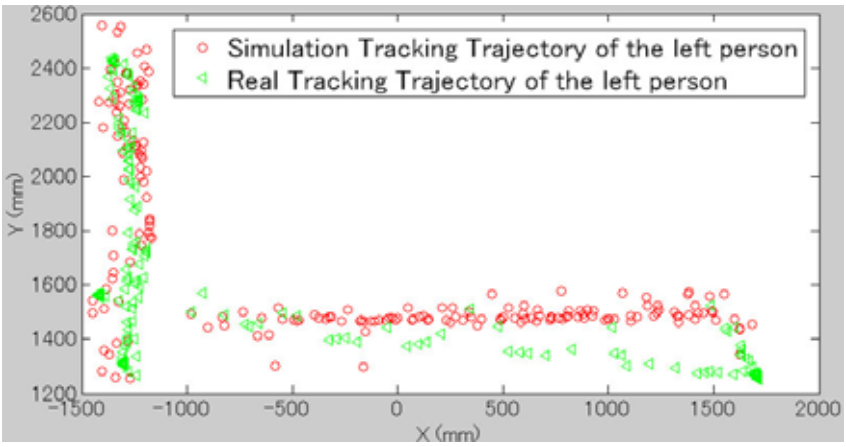


Fig.5. 15 Comparison of Experiment and Simulation Result

5.4 Other Laser Range Finder Test

Here some other tests of laser range finder are introduced. Because the corresponding algorithm of processing the sensor data is still not available, only the simulation environments are shown in the following sections. However, by offering these simulation environments, the further development of the corresponding algorithm will be faster and more convenient. Here we introduce the use of our environment simulator to try new algorithms like road detection using inclined laser range finder or test new idea of sensor's usage like 3D scanning using 2D range sensor.

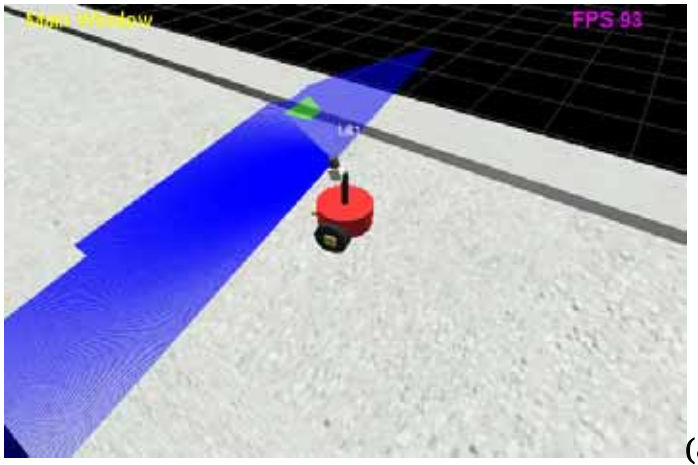
5.4.1 Road detection

There are many methods to detect road when robot moves outside. Among these methods, clever use of laser range finder seems a shortcut to accomplish this task. In the situation of normal road, the road side is a landmark to recognize for robots. By using laser range finder, accurate sensor data can assure the effectiveness of recognition while other methods using camera are very fragile to lighting. The concept can be seen in Fig.5.16. The back side of scanning scene shows the beam-type sensor light while the front side of scanning scene shows the contour-type of sensor light.

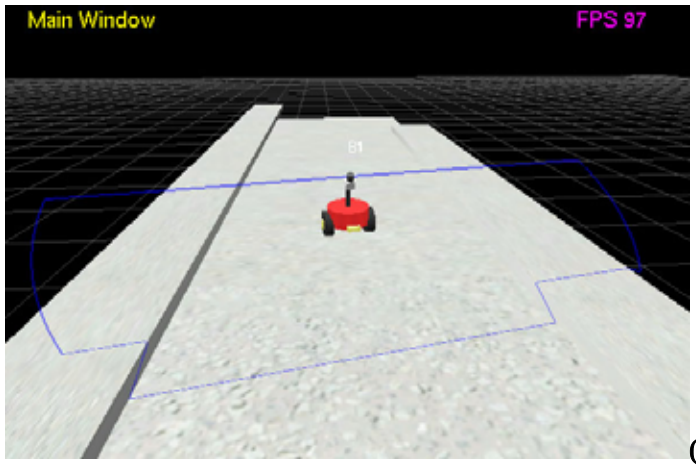
Table.5. 3 Simulation Settings of Road Detection in Simulation Runner

Item	Specification
Road side's height	0.15 meter
Laser range finder's height	0.5 meter
Laser range finder's distance	3 meter (maximum)
Laser range finder's range	-90~90 degree front around sensor
Laser range finder's resolution	0.25 degree
Laser range finder's tilt degree	30 degree

From the pictures, we can see the scanning sensor data has obvious pattern when scanning across the road sides. Fig.5.17 shows the processed obstacle map by using the sensor data. This pattern is expected to make the recognition of road easier. Range sensor usually gives people the most obvious characteristics of target while camera gives most information but none is easy to use. For the further research of road detection, our simulator can be a useful tool by offering experiment environment and accurate sensor model.



(a)



(b)

Fig.5. 16 Road Detection in Simulation Runner

(a) Road Detection with Tilted Laser Range Finder (back side);

(b) Road Detection with Tilted Laser Range Finder (front side).

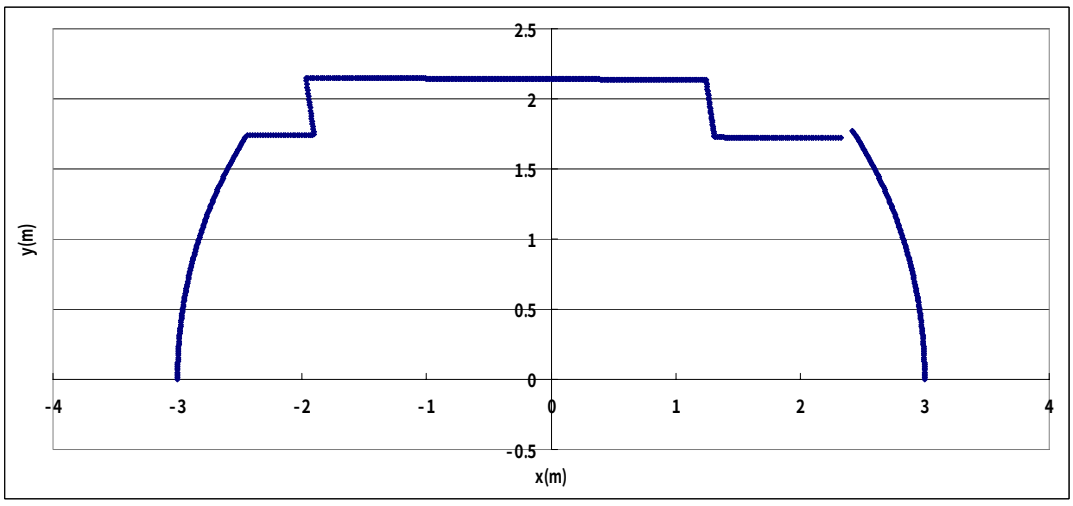


Fig.5. 17 Sensor Data of Road Detection in Simulation Runner

5.4.2 3D Scanning

These years, because of the convenient use of laser range finder, many researchers have proposed their methods to realize 3D scanning by using one laser range finder. One motivation of developing our simulator is to design or test new concepts and inspiration by using existed sensors. Laser range finder Box is a new developed research product in Hashimoto Lab. It is a wireless and no-cable unit made by laser range finder. It has embedded wireless communication unit and battery so that it can work totally without physical connection to other devices.

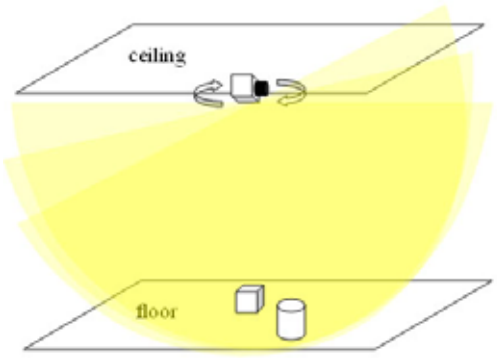


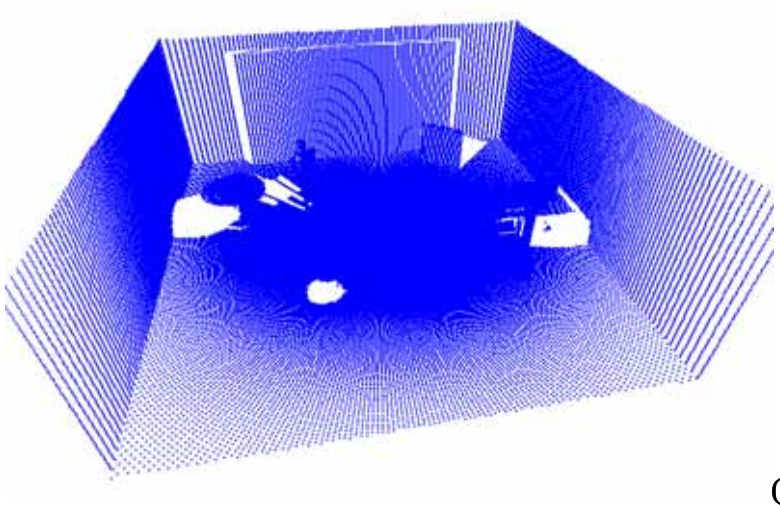
Fig.5. 18 Concept of 3D Rotating Scanning

As one of my resolutions to solve the 3D scanning problem, I considered a new method to use laser range finder box. In my proposition, the laser range finder box is installed on the ceiling. It does scanning while rotating continuously to get all distance information in the range. By using the rotation data and its corresponding sensor data, the 3D distance figure can be reconstructed. This concept can be seen in Fig.5.18. The laser range finder lies down and the scanning space is a hemisphere with the center of laser ranger finder's position. Table.5.4 shows the specification of device and experiment parameters in experiment. The simulation scene and recreated 3D image can be seen in Fig.5.19.

Table.5. 4 Specification of 3D Scanning with Laser Range Finder Box

Parameters	Specification
Laser range finder's height	4 meters high hanged on the ceiling
Rotation resolution	1 degree
Laser range finder's distance	7 meters (maximum)
Laser range finder's range	-90~90 degree front around sensor
Laser range finder's resolution	1 degree
Laser range finder's tilt degree	90 degree

(a)



(b)

Fig.5. 19 Laser Range Finder Box 3D Scanning Simulation
(a) Laser Range Finder Box 3D Scanning Scene in Simulator;
(b) Recreated 3D Image.

5.5 Summary

In this chapter, the detail of simulation part has been explained. After that several themes of simulation has been executed and their corresponding experiments were done and compared with the result of simulator. Most experiment results show that our environment simulator is useful in evaluating algorithms. Other applications that can be realized by using our simulator were introduced at last.

CHAPTER 6

Conclusion and Future Work

In this chapter, the previous chapters are summarized. After that, the conclusion of this thesis is given. At last, some future work of this research is discussed.

6.1 Summary of the Above Chapters

Chapter 1

First, the research background of Intelligent Space was introduced. After that, the purpose of developing our own environment simulator was given and at last was the structure of the whole thesis.

Chapter 2

There were three main parts. First, the relative researches of Intelligent Space were surveyed. Second, many robot simulators and ubiquitous environment simulators were surveyed. The last part is the comparison of other simulators, after that the necessary functions and requirements in detail of our environment simulator were given.

Chapter 3

Chapter 3 is the comprehensive explanation of the implementation of our simulator. The explanation was divided into three parts: Environment Designer, Sensor Arranger and Simulation Runner which are the three tools building up the simulator.

Chapter 4

As one of the main purpose to develop our simulator, not only manual sensor arrangement can be realized, but also the automatic arrangement mode was created to help normal users without much experience in sensor arrangement. The implementation of automatic arrangement of laser range finders was discussed.

Chapter 5

As one of the main purpose to develop our simulator, Experiments and corresponding simulations were compared to prove the usefulness of simulating actuators and distributed sensors in our simulator. At last, other applications available by using our simulator were introduced.

6.2 Conclusion

As the purpose of this research, an environment simulator with distributed sensors was expected to replace of many experiments during the development of new algorithms of sensors in researches of Intelligent Space. Eventually this simulator was realized in the form of three tools, Environment Designer, Sensor Arranger and Simulation Runner. The four functions discussed in Chapter 2 were realized.

During the development of this simulator, we found that the process of realizing simulation for real devices is not as easy as we thought before. Although some problems are easy to understand for us human beings, it can be difficult for the computer to understand.

As a support simulation platform for the further researches in Intelligent Space, many suggestions have been collected in the process of development. However, we found it is really hard to acquire most users' satisfaction if the developer only heard the advices for people in certain field.

6.3 Future Work

Although the purpose of developing our own environment simulator was basically achieved, there are still many places to improve. The first future work is to improve the computation performance of sensor models as well as the convenience of GUI. As we found in comparison of simulation with real experiments, lacking of real attributes like noises and friction makes our simulator hard to replace many experiments related with influence of environment. Therefore, introducing more experiment factors into the simulator is the second future work. At last, this simulator is expected to be utilized in multiple platforms.

APPENDIX

Framework of Programs

Environment Designer

[main]

```
void drawTop();
void drawNormal();
void drawControl();
void drawChoice();
void drawItem();
void drawTopScaleBar();

void enISpace(float hWall, int selected = 0);
void enFloor(int kind, int size, int selected = 0);
void enWall(float hWall, int kind, int size, int selected = 0);
void enInfra(float h, int kind, int size, int selected = 0);

void mouseFuncChoice(int button, int state, int x, int y);
void mouseFuncControl(int button, int state, int x, int y);
void mouseFuncFigure(int button, int state, int x, int y);
void mouseFuncTopScaleBar(int button, int state, int x, int y);
void mouseFuncItem(int button, int state, int x, int y);
void mouseFuncNormal(int button, int state, int x, int y);

int selectHitsChoice(GLuint hits,GLuint *buf);
int selectHitsControl(GLuint hits,GLuint *buf);
int selectHitsItem(GLuint hits,GLuint *buf);
int selectHitsNormal(GLuint hits, GLuint *buf);
void pickUpChoice(int x, int y);
void pickUpControl(int x, int y);
void pickUpItem(int x, int y);
void pickUpNormal(int x, int y);
```

[object]

```
void myGround(double height, GLuint tex, int type = 0, int hit = 0);
void myGround2(double height, GLuint tex, int color, int hit = 0);
void myWall(double height, GLuint tex, int color = 0, int type = 3, int hit = 0);
void myLaser(double width = 0.2);
void zLaserFan(GLdouble d, GLdouble r);
void zPan(GLdouble r, GLdouble h);
void zCameraCube(GLdouble l, GLdouble w, GLdouble h, GLuint tex);
void myCamera();
void myInfra(int row, int col, double h, int hit = 0);
void zCube(GLdouble l, GLdouble w, GLdouble h);
void zUltraSensor();
void myRobot(int type);
void zTablePan(GLdouble r, GLdouble h, GLuint tex);
void myTable(double radius, double height);
```

```
void myChair();
void basicCube(GLdouble ox, GLdouble oy, GLdouble oz, GLdouble length, GLdouble width, GLdouble height);
void zWedge(double r, double l, GLuint tex);
void zBarrierCube(GLdouble l, GLdouble w, GLdouble h, GLuint tex);
void myBarrier();
void myOrigin();
void zDirection();
void zDiamond(float r);
void zRim(float l);
void zFps(float x, float y, float z);
void myScreen();
void zCameraRange(double dist, double aX, double aY);
void myPillar(double h, double r = 0.5);
void myBoxes();

GLuint LoadTextureRAW( const char * filename, int wrap, int w, int h);
void drawBitmapString(float x, float y, float z, char *string, int size = 12);

void zhuan2(double x, double z, int* _x1, int* _x2, int* _z1, int* _z2,
            double rx, double rz, double lenX = 4, double lenZ = 4);
void FPS();
```

Sensor Arranger

[main]

```
void controlLaser();
void controlCamera();
void controlUltra();
void controlRobot();
void controlTable();
void controlChair();
void controlBarrier();
void controlRobot2();
void control3DRanger();
void controlDesk();

void putLaser(int z, bool no_laser = 0);
void putCamera(int z);
void putUltra(int z);
void putRobot(int z);
void putTable(int z);
void putChair(int z);
void putBarrier(int z);
void putRobot2(int z);
void put3DRanger(int z);
void putDesk(int z);

void throwObject(int z);

int selectHitsController(GLuint hits,GLuint *buf);
void pickUpController(int x, int y);
void drawController();
int selectHitsInfo(GLuint hits,GLuint *buf);
void pickUpInfo(int x, int y);
```

```
void drawInfo();
int selectHitsMain(GLuint hits,GLuint *buf);
void pickUpMain(int x, int y);
void drawMain();

void resizeViewWin(int w, int h);

GLvoid buildList();

void shadowArea(int type, double w, double l, double ox, double oz, double theta, float h);
void shadowOn(double ox, double oz, int type, double width, double length, double theta);
float shadowTest(double ox, double oz, int type);

void saveArrangement();
int loadArrangement();
int loadEnvironment();

void addLineObstacle(int type = 0);
void addCubeObstacle(int type);
void addCylinderObstacle(int type = 0);
void drawObsMap();
void updateObsList();

int sensorScan();
void zElements(int z);
```

Simulation Runner

[main]

```
void drawMain();
void drawItemInfo();
void drawItem();
void drawItemFunction();
void drawItemStrategy();
void drawItemSensor();
void drawItemFacility();

void drawObsMap();
void showXYZ();

void renderLaser(int z, bool no_laser = 0); void renderUltra(int z);
void renderCamera(int z);
void renderRobot(int z);
void renderRobot2(int z);
void renderTable(int z);
void renderChair(int z);
void renderBarrier(int z);
void render3DRanger(int z);
void renderDesk(int z);

UINT WINAPI waitReceiveThread(LPVOID pParam);
UINT WINAPI collectSensorDataThread(LPVOID pParam);
void ServerThread();
void SensorThread();
```

```
void ServerDown();
void SensorDown();
int sendData();

void mouseFuncItemWin(int button, int state, int x, int y);
void mouseFuncMain(int button, int state, int x, int y);
void mouseMotionMain(int x, int y);

void controlRobot();
void controlStick();
void sensorScan();
void showSensorData(float w, float h);
void calculateArea();

void zScript(char* words);
void zElements(int z);

void pickUpItemFunction(int x, int y);
void pickUpItemStrategy(int x, int y);
void pickUpItemSensor(int x, int y);
void pickUpItemFacility(int x, int y);
int selectHitsItemFunction(GLuint hits, GLuint *buf);
int selectHitsItemStrategy(GLuint hits, GLuint *buf);
int selectHitsItemSensor(GLuint hits, GLuint *buf);
int selectHitsItemFacility(GLuint hits, GLuint *buf);

int loadEnvironment();
int loadArrangement();

void addLineObstacle(int type = 0);
void addCubeObstacle(int type);
void addCylinderObstacle(int type = 0);
int bumpDetection(double xPos, double zPos, double robotRadius = 0.3);

GLvoid buildList();
```

REFERENCE

[1] H. Hashimoto, "Intelligent Interactive Space - Integration of IT and Robotics," *IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO'05), p.TAR-1-1, 2005,* 2005.

[2] E. Aarts, "Ambient intelligentce: a multimedia perspective," *Multimedia, IEEE,* vol. 11, pp. 12-19, 2004.

[3] C. H. Busso, S.; Chi-Wei Chu; Soon-il Kwon; Sung Lee; Georgiou, P.G; Cohen, I.; Narayanan, S.;, "Smart room: participant and speaker localization and identification," *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on,* vol. 2, pp. ii/1117 - ii/1120, 2005.

[4] J. B. P. T. G. K. Gay, "Designing for context: usability in a ubiquitous environment," *ACM Conference on Universal Usability. Proceedings on the 2000 conference on Universal Usability; Arlington, Virginia, United States,* pp. 80 - 84, 2000.

[5] Y. M. Chris R. Baker, Jana Van Gruenen, Adam Wolisz, Jan Rabaey, and John Wawrzynek, "ZUMA: A Platform for Smart-Home Environments.," *IET Intelligent Environments. Athens, Greece. July, 2006,* 2006.

[6] L. Rudolph, "Project Oxygen: Pervasive, Human-Centric Computing - An Initial Experience," *Lecture Notes in Computer Science,* vol. 2068, 2001.

[7] C. D. K. R. O. G. D. A. C. G. A. I. A. E. B. M. E. M. T. E. S. a. W. Newstetter, "The Aware Home: A Living Laboratory for Ubiquitous Computing Research," *Lecture Notes in Computer Science, Springer Berlin / Heidelberg,* vol. 1670, 1999.

[8] S. S. Sugano, Y., "Robot Design and Environment Design - Waseda Robot-House Project," *SICE-ICASE, 2006. International Joint Conference,* pp. I-31 - I-34, 2006.

[9] Y. N. T. H. T. S. a. S. Hirai, "Sensorized Environment for Self-communication Based on Observation of Daily Human Behavior," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS' 2000),* pp. 1364-1372, 2000.

[10] T. S. Mihoko Niitsuma, Hideki Hashimoto, "Enhancement of Spatial Memory Using Human-Object Relations," *SICE Annual Conference 2008.08, Chofu, Japan,* pp. 3509-3513, 2008.

[11] Q. W. Yoshihisa Toshima, Noriaki Ando, Hideki Hashimoto, "Occlusion Avoidance of Information Display System in Intelligent Space," *SICE-ICASE, 2006. International Joint Conference, Busan, Korea,* pp. 2663-2667, 2006.

[12] N. S. Ando, T. Kitagaki, K. Kotoku, T. Woo-keun Yoon, "RT-middleware: distributed component middleware for RT (robot technology)," *Intelligent Robots and Systems,*

2005. (IROS 2005). 2005 IEEE/RSJ International Conference, pp. 3933-3938, 2005.

[13] M. L. T. Oggier, R. Kaufmann, M. Schweizer, M. Richter, P. Metzler, G. Lang, F. Lustenberger, N. Blanc, "An all-solid-state optical range camera for 3D-real-time imaging with sub-centimeter depth-resolution (SwissRanger)," *SPIE,* vol. 5249, pp. 634-545, 2003.

[14] H. Hashimoto, "Intelligent Space - How to Make Spaces Intelligent by using DIND? -," *IEEE International Conference on Systems, Man and Cybernetics (SMC' 02), 2002.10, Tunisia,* 2002.

[15] L. P. C. Alvise Bonivento, Alberto Sangiovanni-Vincentelli, "Platform-based design of wireless sensor networks for industrial applications," *Hot topic - design, verification, deployment and test of WSN sysems, The conference on Design, Automation, and Test in Europe,* pp. 1103-1107, 2006.

[16] A. C. Vetro, C. Huifang Sun, "Video transcoding architectures and techniques: an overview," *Signal Processing Magazine, IEEE,* vol. 20, pp. 18-29, 2003.

[17] A. P. Y. B. L. H. S. Vetro, "Reduced spatio-temporal transcoding using an intra refresh technique," *Circuits and Systems, 2002, ISCAS 2002. IEEE International Symposium,* vol. 4, pp. IV-723-IV-726, 2002.

[18] T. B. M. George A. Spanos, "Performance Study of a Selective Encryption Scheme for the Security of Networked, Real-Time Video," *icccn, pp.0002, Fourth International Conference on Computer Communications and Networks (ICCCN' 95),* 1995.

[19] J. W. Ken Steele, Eugene Weinstein, "The oxygen H21 handheld," *ACM SIGARCH Computer Architecture News, June 2002,* vol. 30, pp. 3-4, 2002.

[20] M. G. R. Xu, L. Chow, Y.T. Dakin. J.P., "Optical in-fibre grating high pressure sensor," *Electronics Letters, Feb. 1993,* vol. 29, pp. 398-399, 1993.

[21] P. M. Osborne, R. McEvoy, N. Hashtrudi-Zadd, K., "A force-feedback joystick for control and robotics education - affordable software for educational institutions," *Control Systems Magazine, IEEE,* vol. 24, pp. 74-77, 2004.

[22] H. H. Fumio Kanehiro, Shuuji Kajita, "OpenHRP: Open Architecture Humanoid Robotics Platform," *The International Journal of Robotics Research,* vol. 23, pp. 155-165, 2004.

[23] T. S. Noriaki Ando, Tetsuo Kotoku, "A software Platform for Component Based RT-System Development: OpenRTM-Aist," *Simulation, Modeling, and Programming for Autonomous Robots,* vol. 5325/2008, 2008.

[24] S. B. Gershwin, "Assembly/Disassembly Systems: An Efficient Decomposition Algorithm for Tree-Structured Networks," *IIE Transactions,* vol. 23, pp. 302-314, 1991.

[25] P. M. L. Rance Cleaveland, Scott A. Smolka and Oleg Sokolsky, "The concurrency

Factory: A development environment for concurrent systems," *Computer Aided Verification,* vol. 1102, 1996.

[26] R. T. V. Brian P. Gerkey, Andrew Howard, "The Player/Stage Project: Tools for Mutli-Robot and Distributed Sensor Systems," *International Conference on Advanced Robotics (ICAR 2003), Coimbra, Portugal, June 30 - July 3, 2003,* pp. 317-323, 2003.

[27] N. B. Louis Hugues, "Simbad: An Autonomous Robot Simulation Package for Education and Research," *Applied Adaptive Behavior, From Animals to Animates 9,* vol. 4095, 2006.

[28] H. S. Eric Colon, Yvan Baudoin, "MoRoS3D, a multi mobile robot 3D simulator," *Proceedings of the 9th International Conference on Climbing and Walking Robots, Brussels, Belgium - September, 2006,* 2006.

[29] S. Vinoski, "CORBA: integrating diverse applications within distributedheterogeneous enviornments," *Communications Magazine, IEEE,* vol. 35, pp. 46-55, 1997.

[30] O. Michel, "Webots: Professional Mobile Robot Simulation," *International Journal of Advanced Robotics Systems,* vol. 1, 2004.

[31] R. Smith, "www.ode.org."

[32] V. V. John J. Barton, "UBIWISE, A Ubiquitous Wireless Infrastructure Simulation Environment," *Mobile and Media Systems Laboratory, HP Laboratories Palo Alto,* 2002.

[33] M. K. Eleanor O'Neill, David Lewis, Tony O'Donnell, Declan O'Sullivan, Dirk Pesch, "A Testbed for Evaluating Human Interaction with Ubiquitous Computing Environments," *Knowledge and Data Engineering Group, Department of Computer Science, Trinity College Dublin*.

[34] J. K. Minsu Jang, Meekyoung Lee, Joo-Chan Sohn, "Ubiquitous Robot Simulation Framework and Its Applications," *IROS 2005,* p. 1, 2005.

[35] A. S. Andrea Raggi, Renato Zaccaria, "AE-SIM: Simulating Intelligent Robots in Intelligent Environments," *International Symposium on Intelligent Control, Taipei, Taiwan, September 2-4, 2004,* 2004.

[36] M. G. N. Kristian Lindgren, "Cooperation and community structure in artificial ecosystems," *Artificial Life,* vol. 1, pp. 15-37, 1994.

[37] N. S.V.Rao, "Multiple sensor fusion under unknown distributions," *Journal of the Franklin Institute,* vol. 336, pp. 285-299, 1999.

[38] H. Zhang, "Two-Dimensional Optimal Sensor Placement," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 25, 1995.

[39] W. C. W. a. I. H. McLaren, "Time-of-Flight Mass Spectrometer with Improved Resolution," *Review of Scientific Instruments,* vol. 26, 1955.

[40] J. B. G. Postel, Larry L.; Rom, Raphael, "Transmission Control Protocol Specification," *Computer Hardware Non-radio Communications,* p. 183, 1976.

[41] D. M. T. S. J. E. H. M. Levy, "Simultaneous multithreading: maximizing on-chip parallelism," *Proceedings of the 22nd annual International Symposium on Computer Architecture,* pp. 392-403, 1995.

[42] S. Kristensen, "Sensor Planning with Bayesian Decision Theory," *In Reasoning with Uncertainty in Robotics,* 1995.

[43] S. W. Sedas-Gersey, "Algorithms for Automatic Sensor Placement to Acquire Complete and Accurate Information," 1993.

[44] X. Z. Sukhan Lee, "Sensor Planning with hierachically distributed perception net," *In Proceedings of the 1994 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, Las Vegas,* pp. 591-598, 1994.

[45] J. C. T. F.Feito, A. Urena, "Orientation, simplicity, and inclusion test for planar polygons," *Computers & Graphics,* vol. 19, pp. 595-600, 1995.

[46] R. Jaffrey, "Pioneer I software Manual.," *ActiveMedia,* 1996.

[47] J.-C. Latombe, "Robot Motion Planning."

[48] H. H. Drazen Brscic, "Comparison of Robot Localization Methods Using Distributed and Onboard Laser Range Finders," *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2008),* pp. 746-751, 2008.

[49] T. S. Drazen Brscic, Hideki Hashimoto, "Implementation of Mobile Robot Control in Intelligent Space," *SICE-ICASE International Joint Conference 2006.10, Busan, Korea,* pp. 1228-1233, 2006.

[50] H. H. Qinhe WANG, "An Ultrasonic Multi-target Allocation System Using CDMA Signals," *日本機会学会ロボティクス・メカトロニクス講演会 2007, 2007.05, 秋田,* pp. 2A2-K04, 2007.

[51] H. H. Drazen Brscic, "Tracking of Humans Inside Intelligent Space using Static and Mobile Sensors," *Proceedings of the 33th Annual Conference of the IEEE Industrial Electronics Society (IECON' 07), 2007.11, Taipei, Taiwan,* pp. 10-15, 2007.

PUBLICATION

[1] Barna Resko, Kouhei Kawaji, Shaohua Zheng, Hideki Hashimoto, "Retina Inspired Edge Detection for Robot Vision", 日本機械学会ロボティクス・メカトロニクス講演会 2007, p.2A2-K05, 2007.05, 秋田

[2] Drazen Brscic, 佐々木 毅, 周 淼磊, 横井 一樹, 鄭 韶華, 橋本 秀紀, “実環境での走行を目指した自律移動ロボットの研究開発”, 第８回計測自動制御学会システムインテレーション部門（SI 部門）講演会 2007, p. 3C3-2, 2007.12, 広島

[3] Shaohua Zheng, Takeshi Sasaki, Mihoko Niitsuma, Hideki Hashimoto, "Intelligent Space Simulator for Evaluation of Sensor Arrangements -Modeling of Sensors and Building of Environment by using Computer Graphics-", 日本機械学会ロボティクス・メカトロニクス講演会 2008, pp.2A1-F21(1)-2A1-F21(2), 2008.06, 長野

[4] Qinhe WANG, Miaolei ZHOU, Shaohua ZHENG, Hideki HASHIMOTO, "A Simulation of CDMA Based Ultrasonic Localization System in a Large Scale", 日本機械学会ロボティクス・メカトロニクス講演会 2008, pp.2A1-F23(1)-2A1-F23(4), 2008.06, 長野

[5] Shaohua Zheng, Mihoko Niitsuma, Takeshi Sasaki, Hideki Hashimto, “Intelligent Space Architecture Developing Tools”, The 5th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI 2008), p. FB1-5, 2008.11, Seoul, Korea

ACKNOWLEDGEMENT

First of all, I should thank Professor Hashimoto Hideki during the two-year master course. It is his free and open mind that I could select the research theme myself and I really learnt very much. Every time when I need instruction, he will supervise me in time and give me courage. Sincerely thank you!

And then I want to say thanks to Dr. Mihoko Niitsuma, the post doctor of Hashimoto Lab. From her, I received very many good advices. She is sometimes serious but at any time she is a good senior.

Dr. Takeshi Sasaki, who is very good at mathematics, reads extensively and knows almost everything. For many times I asked him for help in programming and mathematics problem, thank you very much.

Dr. Drazen Brscic, the graduated doctor student from Croatia, he is a really nice person and I admire his research work during his doctor course very much. Thank you for you instruction in research.

Mr. Wang Qinhe, graduated from the same university with me, is my best friend and a person who treats me like brother. Thank you for your great help, both in daily life and the two years study.

Also I want to thank Mr Zhou Miaolei, researcher of Hashimto Lab. For the previous two years. He is also my good friend, respected senior researcher.

Mr. Yoshihisa Toshima, Mr. Kohei Kawaji, Mr. Chantanakajornfung Preeda, senior students to me, thank you for your kindness and passion.

I want to say thanks to Mr. Kazuki Yokoi, the same grade student with me. We have had same classes for times and talked about a lot about history and culture. You are a good Japanese teacher to me.

And Mr. Laszlo Attila Jeni, Mr. Adam Balazs Csapo, Mr. Peshala Gehan Jayasekara, as we are all foreign students, you and I have shared a lot in life and research. Thank you for your kindness.

To Mr. Leon Palafox and Mr. Hajime Tamura, you two are new master students to me. Thank you for your active appearance in the lab.

To Miss Huang Xiang Qi and Miss Xu Xin, thank you very much as you are also Chinese and gave me much encouragement.

The current Secretary, Miss Hiroko Ozaki and the previous Miss Chizu Kosaka have devoted a lot to support the research life of Hashimoto Lab. I would like to say I appreciate your work very much.

Finally, I want to say thank you very much to my parents, my sister in China, my previous classmates, roommates, good friends, net friends, tennis club friends, and anyone who helped me and did me favor, your kindness is the power for me to go forward!

Thank you to everybody who has read this acknowledgement and hope we all can remember the people who ever helped us and hope we all happy everyday.