INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA

UMI®
800-521-0600

ORB: OBJECT RECOGNITION FOR REAL-TIME AUTONOMOUS MOBILE ROBOT NAVIGATION

Don T. Bui

Department of Electrical Engineering
McGill University

1997

A Thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements of the degree of
Masters of Engineering

Abstract

An object recognition system called ORB[1] is proposed and implemented for use on a mobile robot. ORB utilizes the QUADRIS sensor platform developed at the Centre for Intelligent Machines (CIM) at McGill University, which is composed of two BIRIS[2] laser rangefinders. ORB performs a series of sensory and perceptual tasks in conjunction with a mobile robot control architecture called SPOTT. ORB's main task is to sense the mobile robot's surroundings and provide laser range data, in the form of line segments, for SPOTT's map database. In an office environment, ORB also identifies and labels the *structural objects* (i.e., walls, doors) in this map. While navigating through an office space, the mobile robot may be required to search for certain objects in the area. In these scenarios, ORB is used to recognize the *movable objects* (i.e., chairs, tables and desks).

ORB is able to perform its tasks in a fast and efficient manner by using simple models to represent the *structural* and *movable* objects in its database. ORB's recognition procedures only require sparse sets of range scans to identify the aforementioned objects. The *structural object* models are built from prior knowledge of the office environment. For example, the doorway model would consist of the known doorway widths found on the experimental office floor. ORB has been tested extensively in the CIM environment, but it can also be applied to any office space provided the structural dimensions are known a priori. ORB's models for the *movable objects* are idealized descriptions with the object's surfaces represented by planes. The physical dimensions of the *movable object* models are defined by *Architectural Standards*, as office furniture are built to conform to these standards.

[1] A system for **O**bject **R**ecognition and map **B**uilding using the QUADRIS sensor platform on a mobile robot.
[2] Official trademark of the National Research Council of Canada.

Résumé

Un système de reconnaissance d'objets appelé ORB est proposé et réalisé sur un robot mobile. ORB utilise la plate-forme de capteur QUADRIS développée au Centre pour des machines intelligentes (CMI) de l'université McGill. Cette plate-forme est composée de deux capteurs télémétrique BIRIS (marque de commerce officielle du Centre nationale de recherche du Canada). ORB accomplit une série de tâches perceptives et sensorielles en conjonction avec un système de contrôle de robots mobiles appelé SPOTT. La tâche principale de ORB consiste à mesurer l'entourage du robot mobile et à procurer des données télémétriques, sous forme de segments de droites, pour la banque de données cartographique de SPOTT. Dans une environnement de bureau, ORB identifie et étiquette les principaux éléments structuraux (ex.: les murs et les portes) dans cette carte. Tout en navigant dans cet environnement de bureau, une des tâches du robot mobile pourrait être de chercher certains objets. Dans un tel scénario, ORS is utilisé pour reconnaître les différents *objects meubles* (ex.: chaises, tables et bureaux).

ORB effectue ses tâches d'une manière rapide et efficace en représentant par des modèles simples les éléments *structuraux* et *meubles* dans sa banque de données. Les modèles des élments structuraux sont construits à partir d'information préalable sur l'environnement de bureau. Par exemple, le modèle du cadre de porte pourrait consister en des dimensions connus de cadre trouvés sur le plancher du bureau. ORB a été testé intensivement dans l'environnement du CMI, mais il peut aussi être appliqué à n'importe lequel espace de bureau à condition que les dimensions des éléments structuraux soient connues a priori. Les modèles de ORB pour les *éléments meubles* consistent en une représentation idéalisée par plans. Les dimensions physiques des modèles des *meubles* sont conformes aux standards architecturaux comtemporains puisque ceux-ci guident le design des meubles rencontrés couramment.

Dedication

This thesis is dedicated to

My parents

Thuy and Dai

for all their love and support for
which I will always be grateful,

and

my siblings

Dan and Helen

for always being there and for being my inspiration.

Acknowledgements

I would like to thank the following individuals, without whom this thesis could never have been completed:

Professor Martin D. Levine

for his guidance and contributions to this work, as well as his patience and understanding throughout the completion of this thesis;

Marc Bolduc and **John Zelek**

for their tireless work in helping me with my experiments, for their advice, insight and dedication;

Thierry Baron, **Paul MacKenzie**, **Romney Ng**, **Gilbert Soucy**

for their technical assistance;

Jan Binder and **Mouse**

for their system administrative assistance;

Ornella Cavaliere, **Dorothy Chase**, and **Kathleen Vandernoot**

for their motherly day-to-day administrative help;

Oliver Astley, **Michael Glaum** and **Matthew Roy**

for their proofreading skills;

and to

Professor Gregory Dudek and everyone else at the **Centre for Intelligent Machines** for all their support.

I would like to express my gratitude for the funding provided for this research by the *National Centres of Excellence Program* through IRIS (*Institute for Robotics and Intelligent Systems*). In particular, this research is part of the *Dynamic Reasoning, Navigation and Sensing for Mobile Robots* project under the ISDE (*Integrated Systems in Dynamic Environments*) theme.

TABLE OF CONTENTS

Abstract ii

Résumé iii

Dedication iv

Acknowledgements v

LIST OF FIGURES vii

LIST OF TABLES x

CHAPTER 1. Introduction 1

1. The Issues 3

1.1. Why Use QUADRIS? 3

1.2. The Tasks 5

1.3. Performance 6

2. The ORB System 7

3. Contributions of Thesis 12

4. Thesis Overview 13

CHAPTER 2. Background 14

1. Sensors for Mobile Robots 14

1.1. Active Range Sensing 15

1.2. Passive Range Sensing 16

1.3. Proximity Sensors 17

1.4. Issues in Data Acquisition for Mobile Robot Sensors 17

2. The QUADRIS Platform 18

3. Map Building 18

3.1. Map Types 19

3.2. Map Building Issues 20
3.3. Updating the Map 21
4. Structural Object Recognition 21
4.1. Related Work 21
5. Movable Object Recognition 22
5.1. Related Work 23
5.2. Function-Based Recognition 24

CHAPTER 3. Map Building and Structural Object Recognition 26
1. Map-Building 26
1.1. ORB-Built Maps 28
1.2. SPOTT 30
2. The Structural Objects 33
2.1. Walls 33
2.2. Map-Building Gaze Control 42
2.3. Doors 48

CHAPTER 4. Recognition of the Movable Objects 55
1. Object Models 55
1.1. Chair Model 57
1.2. Table and Desk Models 58
2. Scanning the Movable Objects 59
2.1. Assumptions 60
3. Chair Recognition 62
3.1. Occlusion 62
3.2. Scanning the Backrest 62
3.3. Scanning the Seat 70
3.4. Producing a Horizontal Planar Surface 70
3.5. Chair Model Matching 73
4. Table/Desk Recognition 75
4.1. Assumptions 75
4.2. Centering the Table/Desk 76
4.3. Producing the Horizontal Plane 77
4.4. Completion 79
4.5. Table and Desk Model Matching 79

5. Summary 79

CHAPTER 5. Implementation 81
1. The ORB System 82
1.1. System Overview 82
1.2. Communications 85
2. ORB's Display 88
3. The BIRIS Laser Rangefinder 91
3.1. Properties of the BIRIS Rangefinder 92
3.2. BIRIS Range Data 94

CHAPTER 6. Experiments 98
1. Computing Costs 99
2. Structural Objects 102
2.1. Walls 102
2.2. Doors 105
2.3. Mapping 110
3. Movable Objects 112
3.1. Chairs 112
3.2. Desks 115
3.3. Tables 118
4. Discussion 121

CHAPTER 7. Conclusions 124
1. Contributions 125
2. Discussion 126
3. Future Work 127
4. Summary 129

REFERENCES 130

LIST OF FIGURES

1.1	QUADRIS: A Mobile Robot Sensor Platform	4
1.2	Overview of ORB System	8
1.3	Mobile Robot Scanning Hallway	9
1.4	ORB Scanning a Chair	9
1.5	Tasks Performed by ORB	11
2.1	The Robot and its Sensors	19
2.2	BIRIS Scan of a Wall from 130 cm	19
3.1	Outline of ORB's Map-Building Process	27
3.2	CAD Map of the CIM Floor	28
3.3	A Raw Map Generated by ORB	29
3.4	Line Segmentation of BIRIS Range Data	31
3.5	Example of a SPOTT Map	32
3.6	Properties of a Wall in a Hallway	34
3.7	The Scan Hallway Initialization Process	36
3.8	The Wall Model Updating Process: Step 1 and 2	39
3.9	The Wall Model Updating Process: Step 3	40
3.10	Opening Detection	41
3.11	Pan/Tilt Movements of the QUADRIS Rangefinders	42
3.12	Outline of How ORB Scans a Hallway	45
3.13	ORB's Scanning Sequence in a Hallway: BIRIS_1 Panning Ahead	46
3.14	ORB's Scanning Sequence in a Hallway: BIRIS_2 Panning Ahead	47
3.15	A Doorway Scanning Sequence	50

3.16	Outline of How ORB Scans an Opening	51
3.17	The Doorway and Hallway Widths on the CIM Floor	52
3.18	A Doorway Opening	53
4.1	The Movable Objects	56
4.2	General Object Model	57
4.3	Chair Model	58
4.4	Table Model	59
4.5	Desk Model	60
4.6	Flow Graph of the Movable Object Scanning Procedure	61
4.7	Outline of Chair Scanning Routine	63
4.8	Chair Orientations	64
4.9	Flow Graph of Backrest Scanning Procedure	65
4.10	Locating the Backrest	66
4.11	Outline of Centering Process	67
4.12	Centering the Chair Backrest in View	68
4.13	Backrest Scan Lines	69
4.14	Outline of Seat Scanning Routine	71
4.15	Locating the Seat	72
4.16	Centering the Chair Seat in View	73
4.17	Scan Lines of a Seat	74
4.18	Range Scans Fit to Chair Model	74
4.19	Outline of Table/Desk Scanning Routine	76
4.20	Locating a Desktop	77
4.21	Scanning a Desk	78
4.22	Range Scans Fit to Desk Model	80
5.1	Inputs and Outputs of the ORB System	83
5.2	Block Diagram of the ORB System	84
5.3	ORB Sends Data to SPOTT	87
5.4	ORB Requests Data From SPOTT	87

5.5	ORB's Graphical User Interface	89
5.6	ORB's GUI Parameter Settings	90
5.7	QUADRIS: A Mobile Robot Sensor Platform	91
5.8	BIRIS Sensor Theory	93
5.9	BIRIS Lines From Close and Far	93
5.10	BIRIS Laser Lines With and Without Filter	94
5.11	BIRIS Scan of a Wall from 130 cm	95
5.12	BIRIS Scan Corrupted by Exterior Light	95
5.13	Corrupted BIRIS Video Images	96
5.14	BIRIS Video Image Verification	97
6.1	Wall Initialization Procedure	103
6.2	Wall Update Procedure	104
6.3	A Closed Door Scanning Sequence	107
6.4	A Partially Open Door Scanning Sequence	108
6.5	An Open Door Scanning Sequence	109
6.6	ORB's Partial Map of a Room	111
6.7	Scanning the Chair Backrest and Seat	112
6.8	The Chair Model and Chair Range Scans	113
6.9	Straight Ahead View of Chair	114
6.10	Top-Down View of Chair	114
6.11	Scanning the Desktop	115
6.12	The Desk Model and Desk Range Data	116
6.13	Top-Down View of the Desk Model and Desk Range Data	117
6.14	Straight Ahead View of the Desk Model and Desk Range Data	117
6.15	Scanning a Tabletop	118
6.16	The Table Model and Table Range Scans	119
6.17	Top-Down View of the Table Model and Table Range Data	119
6.18	Straight Ahead View of the Table Model and Table Range Data	120

LIST OF TABLES

6.1 Computation Times to Complete Processing of a Range Scan. 100

CHAPTER 1

Introduction

An autonomous mobile robot can be used in any number of potential applications. Sample applications include office automation (e.g., mail delivery in an office building or food delivery in a hospital), security surveillance of an office or warehouse environment, and exploration of remote or hazardous areas. In each application there will be specific tasks that the robot will be required to carry out. First of all, the robot must be able to navigate in its environment in order to carry out its assignments. The navigation process will require that the robot perform certain tasks to aid in the path planning. For instance, since localization is imperative for any navigation system, an example of a navigational task as described above would be landmark detection. This is important for certain localization algorithms which rely on positioning with respect to known landmarks in the environment[**6**]. The robot may then be asked to perform further tasks specific to the application once it has successfully navigated to its goal location. These tasks may include searching for an object or landmark, manipulating an object, or just compiling information about the scene (i.e., mapping).

The tasks described above illustrate situations in which a mobile robot must be able to recognize objects in its surroundings. We present an object recognition system called **ORB**[1] for mobile robot navigational tasks. ORB performs the following **sensory and perceptual tasks**:

(i) ORB scans and retrieves range data from an indoor environment to build a map for navigational purposes.

[1] A system for Object Recognition and map Building using the QUADRIS platform on a mobile robot.

(ii) ORB recognizes and labels *structural objects*[2] found in an office environment, like walls and doorways. It determines if the doors found are *open*, *closed*, or *partially open*.

(iii) ORB recognizes *movable objects*[3] like chairs, tables and desks.

ORB uses a mobile robot sensor platform called QUADRIS[4], which is comprised of two laser rangefinders, to carry out the sensory tasks outlined above. The landmarks recognized by ORB, as described in tasks (ii) and (iii), can be categorized into the following three groups:

structural objects: These are landmarks that are part of any office floor foundation: for example, walls and doorways. The doorways have an additional label associated with them concerning the status of the door, whether it is *open*, *partially open* or *closed*.

movable objects: These are larger scale objects found in an office environment that can be moved around, such as furniture items like chairs, tables and desks.

parametric geons: These are specific shapes, such as blocks and cylinders placed in the environment[5]. Work is currently being done at CIM to recognize these objects using QUADRIS, but is not within the scope of this thesis. Further information can be found in [**47**].

In order to recognize the three types of objects, we need to build a database of models describing each of them. The models for the *structural objects* are built from prior knowledge of the environment. It is assumed that access to the structural dimensions, such as doorway widths, are available a priori. The doorway model for instance, would be defined as an opening of a certain width that is found along a wall. That width would have to fall within the range of known doorway widths in that environment. The *movable object* models are built using information provided by The American Institute of Architects' *Architectural Standards* [**1**]. Office furniture items like tables, desks and chairs are built to conform to these standards. Therefore, standard features such as tabletop heights and chair seat heights can be used to build the model database. For example, our chair model consists of a horizontal plane representing the chair seat at a standard height, as defined by *Architecture*

[2]These are part of the permanent structures of any office building.
[3]These are furniture items which can be moved around in the environment.
[4]QUADRIS was built at the Centre for Intelligent Machines (CIM) at McGill University.
[5]Biederman[**5**] theorized that all objects can be composed of geon parts.

Standards[1]. There is also a vertical plane representing the chair backrest. More detailed descriptions of the object models are found in Chapter 4.

1. The Issues

As mentioned above, ORB is a recognition system for use on a mobile robot using QUADRIS as the sensor platform. The tasks are to (i) build a map of an unknown environment and/or update a previously defined map (ii) recognize and label landmarks on the map, such as walls and doors (iii) recognize common office furniture items like chairs and desks. The motive for creating ORB is to exploit the flexibility of QUADRIS in performing these tasks in a fast and efficient manner.

1.1. Why Use QUADRIS?

In building a map of the environment, there is a choice for the type of sensor to use on a mobile robot. The predominantly used sensors for mapping are ultrasonic time-of-flight sensors as well as laser rangefinders. Laser range data are generally more accurate and less prone to noise, but the laser rangefinders themselves are bulkier and need more power to operate. Sonar sensors are generally available on mobile robots, are less costly in terms of hardware and will provide a complete scan in every direction around the robot. But sonar data are plagued by low resolution as well as problems such as specularity and multiple reflections[**27**]. These inherent problems make it very difficult to use sonar data for object recognition and cause problems in mapping as well[**22**]. The difficulty of recognizing open doorways using sonar data is documented in [**14**]. Better resolution and more reliable data are needed for recognizing and labelling the objects outlined above.

There is also the issue of the computational time needed for the recognition procedure. Typical image-based recognition techniques are generally computationally expensive[**4**]. However, the BIRIS[6] laser rangefinder acquires range information quickly[7] and accurately with little computational overhead[**24**].

This research uses a mobile robot sensor platform called QUADRIS[8] which was developed at the Centre of Intelligent Machines (CIM) at McGill University. QUADRIS is a range sensing system comprised of two independently controlled laser rangefinders (BIRIS), which have two degrees of freedom each, a pan and a tilt movement. BIRIS was developed at the National Research Council of Canada (NRC)[**8**][**9**][**40**][**54**].The laser rangefinders can

[6]Official trademark of the National Research Council of Canada. Name comes from "Bi-iris", or two-eyes.
[7]Range data are acquired at 10 frames/second on our sensor.
[8]Name derived from the fact that two BIRIS sensors are used on this platform.

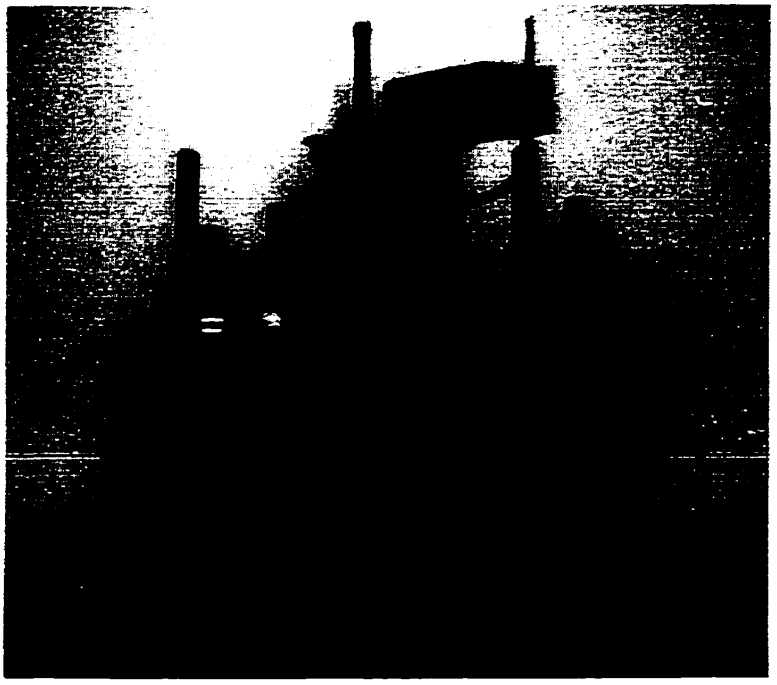


FIGURE 1.1. **QUADRIS: A Mobile Robot Sensor Platform.** QUADRIS is composed of two BIRIS sensors mounted on pan-tilt units.

be used together in a stereo mode, or independently in a "divergent stereo" mode, by pointing them in opposite directions. Having this flexibility facilitates covering as much of the robot's surroundings as possible during map building. The BIRIS sensor projects a laser line into the scene, from which it produces an array of range data. The QUADRIS platform will be discussed in further detail in Chapter 5 and can be seen in Figure 1.1.

In QUADRIS, ORB has two BIRIS laser rangefinders at its disposal. The BIRIS sensors provide range data up to 5 metres, with an accuracy of 2 cm up to a distance of about 2 metres, which is the normal operating range when working in an office environment. Having the two laser sensors on separate pan/tilt units which can be controlled independently provides ORB with great flexibility for mapping its environment. For instance, if we know the robot is in a hallway, then the points of interest will be towards the walls on each side of the robot. The two BIRIS sensors would then be pointed to each side of the robot, perpendicular to its line of motion. A scanning pattern for a hallway would constitute two or three range readings to the sides for each scan to the front of the robot, which generally will be free of obstacles. This is one way to utilize the flexibility of QUADRIS to sense the environment more efficiently.

1.2. The Tasks

The path planning for the mobile robot is done using a "task driven behavioral navigation" system developed at CIM, called SPOTT[**68**]. The mobile robot **navigational tasks** are carried out by SPOTT, and are based on the verbs "*GO*" and "*FIND*". *GO* commands are directives which specify the spatial coordinates to which the robot is to navigate to. The *FIND* command requires a description of the object the robot is required to locate, but the spatial location is not known[**68**]. In conjunction with these navigational tasks, ORB performs a series of **sensory and perceptual tasks** as outlined previously.

1.2.1. *"GO" Commands*

In an office environment, *GO* commands will generally have the robot move from one room to another, or from a hallway into a room. In any case, some passageway into these rooms must be clear for these tasks to be completed. These passageways are doorways, which can either be open or closed. In a dynamic environment, doors can be opened or closed at any given moment so they cannot be mapped a priori. These doorway states need to be updated during the navigation process, and ORB provides this functionality. While navigating down a hallway, ORB will detect the doorways and determine whether the doors are *open*, *closed*, or even *partially open*.

1.2.2. *"FIND" Commands*

The *FIND* command is a task that can be of the following variation: "GO to Room 122 and FIND a CHAIR". So in the process of navigating to a goal location on the office floor, we add the additional task of finding a chair, or some other object. The act of locating the chair is not within the scope of this thesis, but is part of ongoing research done at CIM. The act of scanning and recognizing the chair is done by ORB, which as of now assumes that the chair is in view of one of the BIRIS sensors when the command is given to recognize it. The assumption is that the "attention problem" has been solved and the object's location is known. ORB then determines what the object is.

1.2.3. *ORB Sensory and Perceptual Tasks*

The sensory and perceptual tasks performed by ORB aid in the completion of the *"GO"* and *"FIND"* commands, as described above. There are three main tasks, as was outlined previously. Task (i) is for ORB to provide range data for SPOTT, which uses its behavioral architecture to perform dynamic path planning. If SPOTT has a pre-defined map of the environment, any objects detected by ORB that are not already in the map

are simply added by SPOTT to its map database. Continually updating the map is crucial for navigating in a dynamic environment, for it is impossible to have a complete map of everything on the floor a priori. At best, a partial map of the permanent structures is all that will be available at the start (e.g., an architectural CAD map). While updating and building this map, ORB performs task (ii) in which it recognizes the *structural objects*, walls and doors. ORB sends SPOTT the locations of these landmarks so they can be labelled in the map. Task (iii) is part of a set of task commands available for the mobile robot that are issued by SPOTT. An example of one such task is: "Find a Chair". In this case, once an object that is suspected of being a chair has been located[9], ORB is used to scan and determine if the object is indeed a chair.

1.3. Performance

SPOTT is a "real-time" mobile robot control architecture in that it responds to changes that occur in the environment. It will be shown that ORB's sensor updates to SPOTT are fast enough to allow the mobile robot to move at human walking pace (approximately 1 meter/sec.), which is SPOTT's target speed. In order to have the recognition process done at reasonable speeds, the models for both the *structural* and *movable* objects are kept very simple. For the *movable* objects, the robot will need to remain stationary[10] for ORB to scan the object.

ORB's map is a 2D representation of the environment. Hence, even though QUADRIS could provide a 3D description of the surroundings, it is sufficient that the *structural object* models be 2D. These objects are vertical planar structures, so their 3D representations can be projected down to a 2D plane. Any horizontal slice through a wall or door will look the same no matter where it is taken.

The features used to design the *structural* object models are those which can be detected by a horizontal laser scan line. The width of a doorway is one such feature and can be detected via a series of *horizontal* scans, which generally number 4 or 5, spanning the width of the doorway. It is not necessary to take a whole range image of the entire door just to find its width. Walls can be modelled in 2D as well since they are vertical planar structures. The *wall model* consists of the distance of the wall to the sensor and is updated as the robot moves along the hallway. The distance of the wall to the sensor can be found from one laser line scan.

[9]The attention problem, or process of locating the object in the scene, is not addressed in this work. It is assumed the object has been localized before the recognition process has started.

[10]On the order of 20 seconds or less. This amount of time consists of the communication with and movement of the PTUs, as well as the processing of the data.

The *movable* objects require a more sophisticated model, but it is still a simple one to compute. An idealized model is proposed, in which each of the objects are represented by a series of vertical and horizontal planes. Having the objects modelled by planar surfaces simplifies the matching process since the laser scan lines can be easily fit to planes. These planes can be approximated without having to take a dense range scan of the object. For instance, is was found that a set of only 5 or 6 line scans is enough to model the plane of a chair's seat. These models will be discussed further in Chapter 4.

2. The ORB System

The ORB system consists of several parts, as outlined in Figure 1.2. It communicates with two modules: SPOTT and the QUADRIS platform. The *task commands* come from SPOTT. In the case where SPOTT is performing a *GO* command, its task for ORB will be either (1) *Scan Hallway* or (2) *Scan Room*, depending on where the robot is at the time. If it is carrying out a *FIND* command, then it would send ORB one of the following *Scan Object* tasks: (1) *Scan Chair*, (2) *Scan Table* or (3) *Scan Desk*. Again, in this latter case, ORB assumes that the object location is known so the attention problem is not addressed. A user can also invoke these tasks via a graphical user interface (GUI).

The aforementioned tasks determine what actions ORB will take. The first course of action is how to position the QUADRIS sensor heads. For each task or scenario, a different scanning pattern is needed. For example, if the robot is navigating down a hallway and the *Scan Hallway* task has been issued, then ORB will have the two laser rangefinders point to the sides of the robot, scanning the walls and doors. ORB will not scan behind the robot, and will only occasionally scan to the front since the assumption is that the hallway will not be cluttered. A depiction of this scenario can be found in Figure 1.3. A *Scan Room* task, on the other hand, would need a different kind of sensor control. It is assumed that obstacles in a room are likely to be present all around the robot, so the sensors will scan equally to the sides as to the front. If the task is to scan a *movable object* like a chair, then the scanning pattern of the laser rangefinder will reflect the model of the chair. ORB will be looking for planar surfaces at standard dimensions as defined by *Architectural Standards* [1]. An illustration of such a situation can be found in Figure 1.4.

Once the rangefinders have been properly positioned, a range scan is taken and ORB receives an array of range data from QUADRIS along the laser line. ORB then processes the range data to obtain a line segment representation of the data. Now, depending on what the task is, the appropriate recognition module is called. The *movable* object recognition

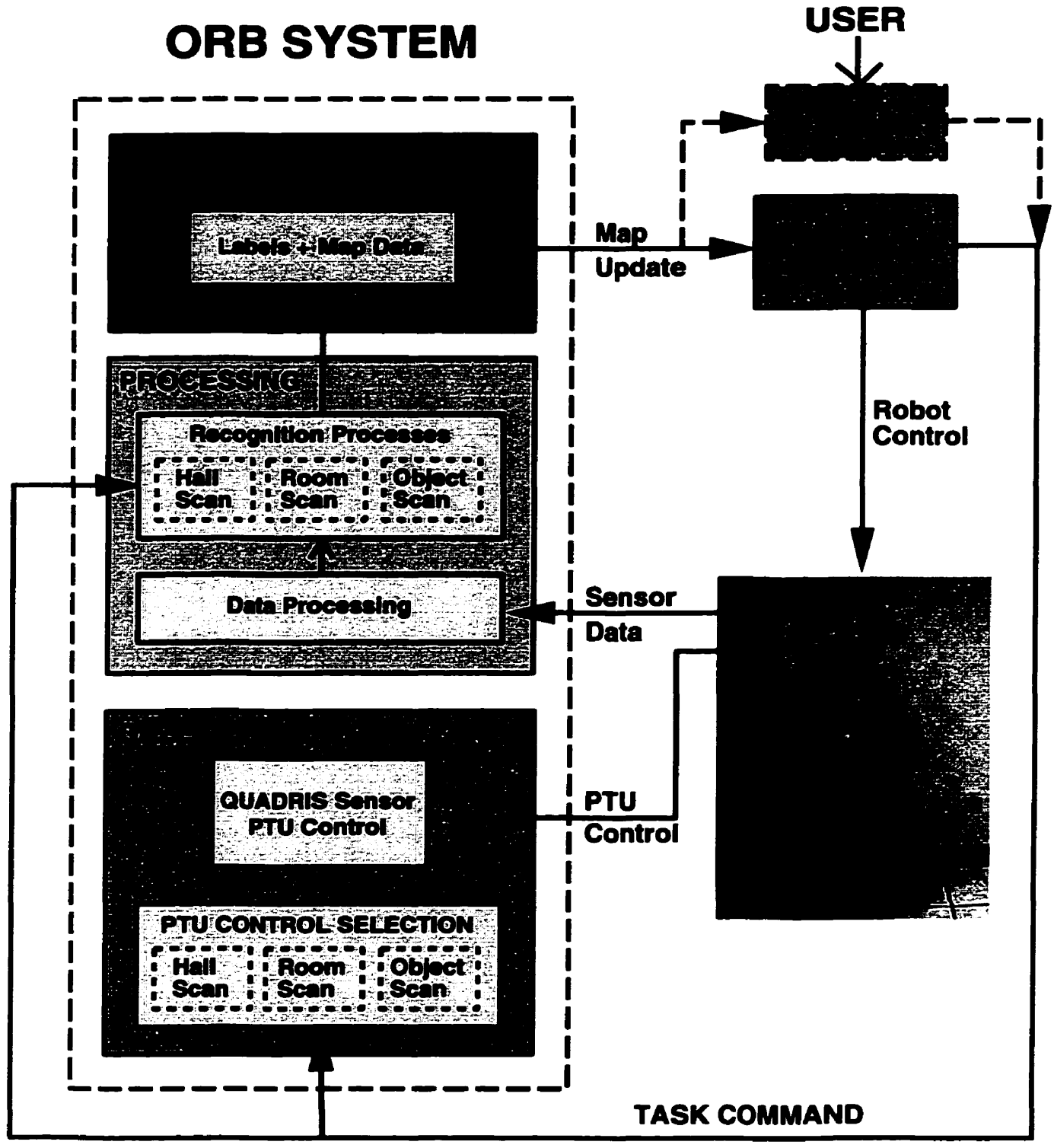


FIGURE 1.2. **Overview of ORB System.** The ORB system and how it communicates with SPOTT and the QUADRIS platform is shown. SPOTT sends a task command to ORB, which then determines how it should position the rangefinders before taking a scan, as well as what recognition process it must perform. It processes the data, obtaining line segments before calling the appropriate recognition module. The **Hall Scan** and **Room Scan** recognition processes produce labels for each line segment. The labels and line segments are then sent back to SPOTT. In the case of the **Object Scan** task, several scans must be taken before the recognition process is complete, and only then is a label returned to SPOTT.

FIGURE 1.3. **Mobile Robot Scanning Hallway.** As the mobile robot travels down a CIM floor hallway, ORB is in *Scan Hallway* mode in which the QUADRIS rangefinders are focused on the walls and doors to the side of the hallway. Range data is retrieved along the laser line.

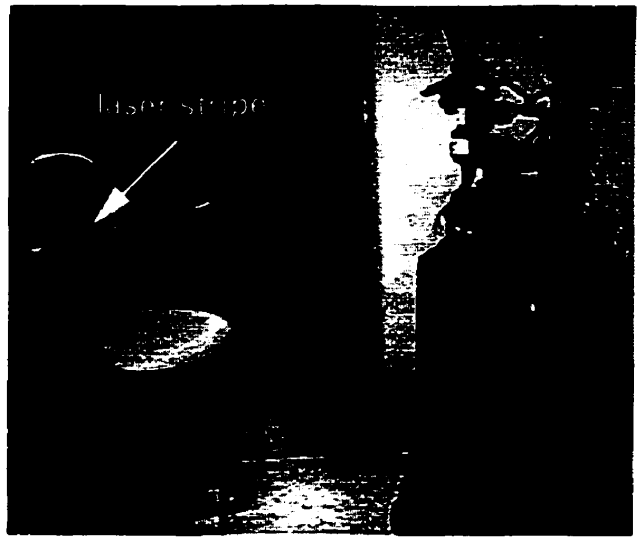


FIGURE 1.4. **ORB Scanning a Chair.** ORB is in *Scan Chair* mode and in the process of scanning the chair backrest. Knowing that the backrest will be at a standard height for chairs, it looks for a vertical plane at that location. Once it determines that a vertical plane is indeed present, it continues by scanning the chair seat looking for a horizontal plane at the standard height.

process, or *Scan Object* module, is called when the task is to scan a chair, table, desk or other *movable object.* In this case, several scans at different vertical heights are needed before the recognition process is complete. When a hallway is being scanned, the *structural* object recognition process, or *Scan Hallway* module is called. These modules produce labels for each object or landmark that has been successfully recognized. These labels are attached

to the corresponding line segments indicating their position in the map. The *Scan Room* task does not attempt to recognize any objects. It simply returns line segments in order to build a map of the area. Since these objects are not being modelled, the default label given to each line segment is "*unknown object*". This is also the label given any object not fitting any *structural* object descriptions while scanning a hallway. The line segments and their corresponding labels are sent to SPOTT, which then computes the appropriate path for the robot[**68**]. The task is performed until a new task command is delivered by SPOTT or the user. A description of the various ORB tasks can be found in Figure 1.5.

TASKS	Scan Hallway	Scan Room	Scan Object
Purpose	Scan hallway to model walls and doors. Determine if door found is open, closed, or partially open.	Scan room and build map of surroundings. No recognition is done.	Recognize movable objects given the task "Find Object", with the object being a chair, table or desk.
Labels	● Walls. ● Doors – open, closed, or partially open. ● Unknown Object.	● Unknown Object.	● Chair. ● Table. ● Desk.
Gaze Control – PAN	Point sensors towards sides of robot to focus on walls and doors. Scans to the front are less frequent than to the sides as hallways are generally unblocked.	Scan equally to the front as to the sides since obstacles in a room are equally likely to appear in all directions.	Pan angle will be determined by the location of the object in the scene.
Gaze Control – TILT	Tilt angle is fixed at pre–determined height.	Tilt angle is fixed at pre–determined height.	Tilt positions are a series of scans depending on the object to be recognized. Sensor will scan for planar features at standard heights as defined by the object model.

FIGURE 1.5. **Tasks Performed by ORB.** These are the various functions performed by ORB. They can be categorized into three groups: Scan Hallway, Scan Room and Scan Object. Each task produces a different set of labels, and requires a different scanning pattern for the laser rangefinders.

3. Contributions of Thesis

This work introduces an object recognition system called ORB for use on a mobile robot. ORB functions mainly as a sensing system that utilizes the QUADRIS platform to scan the surroundings and provide laser range data for the navigation system (i.e., SPOTT). ORB performs additional sensory and perceptual tasks that aid SPOTT in completing its navigational responsibilities. These include the recognition of the *structural* and *movable* objects. Those tasks contain the research direction and contributions of this work, which are outlined as follows:

- This work proposes the use of simple 2D models to represent the *structural objects* (see Chapter 3). ORB identifies the *structural objects* in an office space by exploiting a priori knowledge of the environment's structural dimensions. The objective is to recognize and label walls and doorways in a map database as the robot travels through the hallways of an office area. ORB introduces a novel approach to recognizing these landmarks in a fast and efficient manner, requiring only a minimal number of horizontal laser scan lines.
- This thesis proposes a unique method of modelling and identifying the *movable objects* (see Chapter 4). ORB presents an idealized model for the *movable objects* whereby they are represented by a series of horizontal and vertical planes. ORB uses planar surfaces to model these objects in order to simplify the recognition process. ORB only requires a minimal number of horizontal laser range scans (e.g., 5 or 6) to reconstruct each of these object's surfaces.

The introduction of these simple models allow ORB to complete its recognition processes quickly and efficiently, which is important when working in a dynamic environment.

4. Thesis Overview

The thesis is organized as follows:

- **Chapter 1: Introduction** provides a summary of this work and outlines the contributions of this research.
- **Chapter 2: Background** is a survey of the topics presented in this thesis. Includes a review of mobile robotic sensors, map building issues, and a discussion of the recognition of objects targeted in this work.
- **Chapter 3: Map Building and Structural Object Recognition** demonstrates how ORB provides laser range data for SPOTT's map database. The data itself, as well as ORB's focus of attention while scanning an office environment, are presented. Finally, ORB's recognition of the *structural objects* is discussed.
- **Chapter 4: Recognition of the Movable Objects** illustrates ORB's *movable* object models and discusses how ORB scans and identifies these objects.
- **Chapter 5: Implementation** describes the experimental setup and some of the engineering issues involved in running ORB.
- **Chapter 6: Experiments** discusses the experimental results and what to expect from ORB is terms of performance and computing times.
- **Chapter 7: Conclusions** provides a summary and review of the topics covered in this thesis, as well as some suggestions for future work.

CHAPTER 2

Background

ORB is an object recognition system for use on a mobile robot that operates in conjunction with a path planning navigational system called SPOTT. In order to navigate properly, a mobile robot needs reliable sensors to survey its surroundings, since obstacle detection is clearly vital for successful path planning. There is a plethora of sensors to choose from, depending on the application and the task at hand. A survey will be presented of the sensors available for mobile robots, as well as the particular ones used on our mobile platform.

During the course of navigation, a map of the environment may either be explicitly predefined for the robot (i.e., a CAD map) or it can be dynamically built. A map is needed for an autonomous mobile robot to place sensed environmental structures and to position itself with respect to these structures. ORB is capable of performing in either scenario, though the former case is stressed. When a CAD map of an office floorplan is available, SPOTT uses ORB to: (i) label the existing *structural objects* in the map, and (ii) add objects not already defined in the map. These objects are labelled as "unknown" and are also referred to as "obstacles". The map-building issues will be discussed later in this chapter.

During the course of navigating through an office environment, the robot may be required to locate specified objects, such as the *movable objects* or *geon* objects[**47**]. ORB is summoned by either SPOTT or the user to scan and recognize a *movable object* once it has been localized in the scene. The different areas of object recognition used by ORB will be discussed at the end of the chapter.

1. Sensors for Mobile Robots

A mobile robot needs to be able to sense its environment in order to perform fundamental tasks. The choice of one sensor over another is dependent on the application or the task at hand. There are also other constraints, including speed and reliability, as well as physical constraints, cost, available technology, and basic preference [**20**].

Range sensing basically falls into two categories, *passive* and *active* [2]. In *passive* range sensing, the range data are acquired from the scene through existing energy in the environment such as reflected light. A *passive* range sensing device would use a video camera for instance, to gather images for depth computations. The problem with this type of setup is the processing time required [27] which makes real-time navigation difficult. *Active* range sensing consists of projecting an external energy source onto the scene, for instance, sonar waves or laser light. We will briefly discuss the available technologies for mobile robot range sensors.

1.1. Active Range Sensing

A good overview on the different rangefinding techniques and technologies can be found in [36] by Jarvis and [27] by Everett. A brief summary will be provided here. There are two main classes of *active* range sensors, *triangulation* based systems and *time-of-flight* devices [2][11].

In a *triangulation system*, a laser light stripe or spot is projected onto the scene, and a camera is used to sense the location of the pattern in the scene. By using the known geometry of the imaging system, such as the baseline distance and angle between the camera and laser, the distance of these illuminated points in the scene to the baseline of the sensor can be calculated. A range map can be produced by scanning the laser light across the entire scene, which will produce a detailed range description of the observed scene. The key is for the laser line to be visible to the camera, which presents a problem inherent to any triangulation system, called the "shadow" problem. This occurs when a region of the scene is not visible to either the laser or the camera, due to occlusion or cavities in the object. Triangulation systems are generally used for providing dense range scans for object recognition systems since they are very accurate at short distances.

The other class of *active* range sensors are the *time-of-flight* (TOF) sensors. An energy signal is sent onto the scene, and range is computed by using the time it takes for the signal to reflect and return to the sensor. The energy signals are generally ultrasonic waves or laser light. The "shadow" problem, encountered with the triangulation technique, is eliminated since the laser beam/ultrasonic waves are sent and received along the same path. Ultrasonic TOF sensors are the most widely used range sensor for mobile robots since they are safe, inexpensive, simple to use and widely available. However, they suffer from the following drawbacks: 1) they provide low resolution and 2) they are prone to various

physical phenomena such as multiple and specular reflections, which result in erroneous measurements [27][43].

The problem with *active* range sensors is that they rely on the reflected signal, whether it be sonar waves or laser light. Therefore, if the surface is highly reflective (e.g., a mirror), the returned signal may not reach the sensor receiver. This is the *specular reflection* problem. This problem may be alleviated by painting the surfaces appropriately, but this cannot always be done and it is preferable not to modify the environment. On the other hand, if the surface is highly absorbent (e.g., a black surface that does not reflect laser light), then little or no signal is returned and range data cannot be obtained. In order for a laser sensor to combat this problem, high powered lasers must be used to gain better signals. This results in a system that is potentially harmful to the human eye as well as expensive in terms of cost. Another problem is that the returned signal may be reflected by other surfaces along its path. This is referred to as the *multiple reflection* problem, and also results in erroneous range data. See [36] for a review of the different categories of TOF range sensors, and [26][27] for an exhaustive survey of commercially available sensors.

1.2. Passive Range Sensing

The other rangefinding techniques are generally image based. Some of these passive techniques include texture gradient analysis [3], occlusion effects [55] and focusing methods [38]. Other techniques, which rely on motion or multiple views, include stereo disparity [45][46], depth from camera motion [50], retinal flow and multiple view reconstruction [4][36]. Then there are the "contrived lighting" approaches, in which the scene is illuminated with controlled lighting and the pattern of the projection is used to determine the surface geometry of the scene. Some of these techniques include Moire fringe analysis [34], striped and grid lighting and patterned lighting. For more information on this topic see [4][36][27].

The main problem with these approaches is that the computational requirements are generally quite high. For a mobile robot platform, speed is of the essence, and at this point in time it is not yet practical to use passive range sensing techniques for navigational purposes. Image based techniques usually rely on extracting features (e.g., edges) from video images to determine range. This makes them sensitive to images with low contrast, as these features will be difficult to detect. The reliability of these methods is dependent on how accurately certain features can be detected.

Stereo disparity is a passive ranging technique that is based on biological vision systems [46]. When an object is viewed from two different positions along a plane normal to the

direction of vision, the images appear offset from one another. This shift in the images (i.e., *disparity*) is inversely proportional to the distance of the object to the sensor. For each pixel, a *correspondence* is computed between features from the two or more images. Then, knowing the geometry of the sensor system, range can be computed by triangulation. This method is computationally intensive due to the matching process, or the *correspondence* problem. However, see Ezzati[**28**] for a recent method with potential for real-time implementation.

1.3. Proximity Sensors

A last group of sensors, called *proximity* and *tactile* sensors, are used to simply detect the presence of objects close to the sensor as opposed to determining range. *Proximity* sensors are used to cover a short region, anywhere from a few centimetres to a meter, and can also be used to determine range, though not very accurately. On a mobile robot, they are mainly used for collision avoidance when the longer range sensors fail. *Infrared proximity* sensors are one such type and are used on the experimental platform of this work. The last line of defence against substantial harm to the robot would be a *tactile* sensor system, which detects physical collisions. A *bumper* system which detects when the robot has driven into an obstacle is an example of such a system. For a more thorough examination of *proximity* and *tactile* sensors, see [**26**][**27**].

1.4. Issues in Data Acquisition for Mobile Robot Sensors

As was mentioned previously, there are a number of basic factors which govern what kind of sensor to use on a mobile robot, such as physical size and cost. There are other considerations as well, which are as follows [**26**]:

(i) The *field-of-view* needs to be wide enough, along with adequate range depth for the task at hand.

(ii) The minimum and maximum *range capacities* must be appropriate for the sensor's task.

(iii) The *accuracy and resolution* need to be adequate for the application.

(iv) The sensor must be able to provide data at a rate which is in tune with the robot's working speed. For example, the sensor cannot take 50 seconds to scan and compute one set of range data if the robot is moving at a speed of even 5 cm/sec. The robot would have moved over 2 meters by the time the sensor responded and the data would no longer be relevant.

(v) Data should be concise and easy to interpret.

(vi) The sensor's *power consumption* should be minimal.

(vii) Its *size and weight* should be practical with respect to the robot being used.

Passive approaches to rangefinding are more flexible since an external energy source is not needed. But the computation time needed for these approaches renders them not yet feasible for a real-time navigation system. Faster updates from the robot's sensors are needed to adapt to a changing environment. *Active* approaches are still viable in an indoor office environment [**36**], and is the approach taken with our sensor platform QUADRIS.

2. The QUADRIS Platform

QUADRIS is a range sensing system comprised of two independently controlled laser rangefinders. These laser rangefinders are based on BIRIS[1] which was developed at the National Research Council of Canada (NRC)[**8**][**9**][**40**][**54**]. Each BIRIS sensor is mounted on a two degree-of-freedom pan/tilt unit. It is used in conjunction with a number of other sensors available on our mobile robot, the "Nomad 200"[2] robot, which can be seen in Figure 2.1. The experimental platform consists of sensors that are pre-packaged with the "Nomad 200", namely sonar, infrared proximity sensors and a tactile bumper system and QUADRIS which is mounted on top of the Nomad robot. The BIRIS sensor produces an array of range points, an example of which is shown in Figure 2.2.

The QUADRIS platform and the rest of the experimental setup will be described in more detail in Chapter 5.

3. Map Building

In order for a mobile robot to navigate effectively in a given environment, it must have some kind of map. This map can either be completely pre-defined (i.e., a CAD map) or one which is dynamically built. In either case, in order for the robot to be able to go from some starting position to a goal location, it must know where these points are on the map and where the obstacles lie in order to plan its path. There exist mobile robot control architectures which actually do not depend on maps for navigation, for instance Brooks' reactive control system [**13**]. But in order to ensure successful completion of certain tasks, such as locating a landmark at a specific spatial location, a map is needed.

---

[1] Official trademark of the National Research Council of Canada

[2] The Nomad 200 is manufactured by Nomadic Technologies Inc.

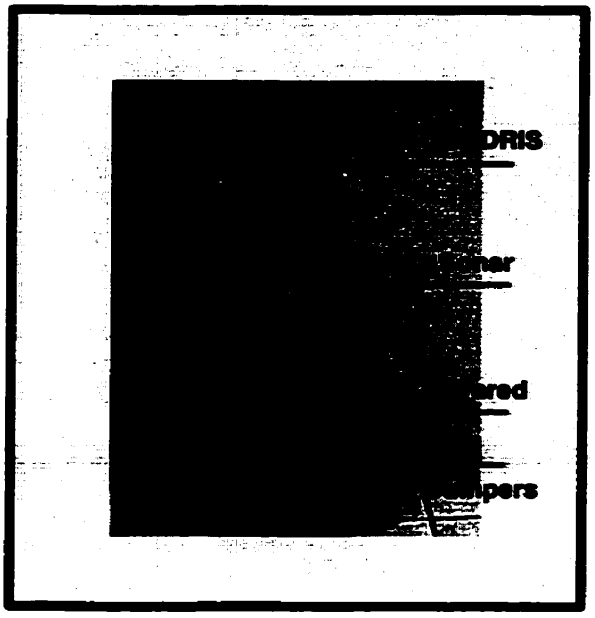
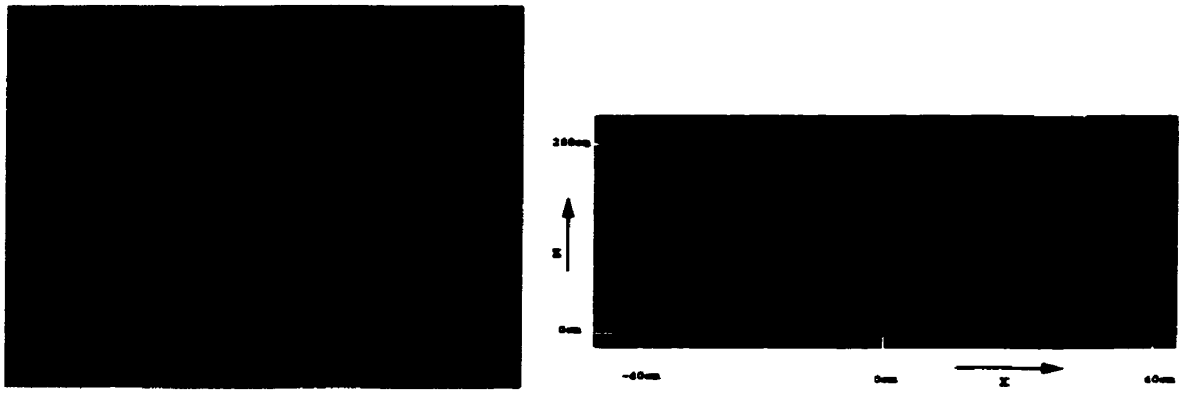


FIGURE 2.1. **The Robot and its Sensors.** The Nomad 200 robot with its sensors: the QUADRIS platform on top, the sonar sensors, infrared proximity sensors and the bumper system.



(a) (b)

FIGURE 2.2. **BIRIS Scan of a Wall from 130 cm.** (a) The filtered image shows the stereo laser lines from a scan of a wall 130 cm away. (b) The range data is plotted with the horizontal axis being parallel to the plane of the lens, and the vertical axis indicating distance of the sensed object normal to the plane of the lens.

3.1. Map Types

So what exactly is a map? A map can be described as a representation of the known world, a collection of information about a given area. There are many different kinds of maps, each serving a particular purpose, and differing in scale and in the type of information presented. For example in road travel, for city-to-city driving, one would use a highway map, and for inner-city driving a detailed city map would be more appropriate.

For mobile robot applications there are generally three kinds of maps used [15]. The first type is a *geometric* map, in which the scene is represented in absolute XY coordinates. This map contains the lowest level information, coming directly from the sensors, and needs to be kept as precise as possible. It is then used to create the other higher level maps, the *topological* and *semantic* maps. A *topological* map is a graphical representation in which a "place" is represented as a node on the graph and the areas connecting these places are the arcs. For example, in an office environment, the various rooms would be the places or nodes on the graph, with the doors and hallways being the connectors or arcs. Last is the *semantic* map, which is used for high level decision making and is a description of the spatial properties of the objects in the map. This could mean attaching a descriptive label to a node on the topological map. An example is Kuipers' *spatial semantic hierarchy* [39] in which semantic descriptions are given to what are termed *distinctive places* in the map.

3.2. Map Building Issues

Maps are used in order to place and locate structures in the world so that navigational tasks can be carried out. The most prevalent problem in building maps from sensor information is the error that is present in these sensors. Therefore, maps will only be accurate to within some margin of error, depending on the sensor uncertainty. As well, robot *localization*, or where the robot thinks it is in the map at any given moment, must also be accurate and updated for the map to have any meaning. The sensor data it receives are mapped with respect to where the robot thinks it is at that given moment. Hence, if its localization is off, this will contribute further to the mapping error. By the same token, localization is dependent on accurate mapping, so any discrepancy in one will directly affect the other. For localization systems that rely on dead-reckoning, meaning the robot's displacement is measured by an on-board odometer, any error in localization will increase unbounded with time. This will result in a mapping error that will persist until the robot is localized once again.

Most of these issues are not the focus of this work, but they have been addressed more directly by others. For instance, sensor uncertainty can reduced by a multi-sensor system using only the most accurate sensor in a given situation, as reported by Chatila and Laumond on the HILARE robot[15]. Features in a map can have their uncertainty represented by a covariance matrix. Each line segment in the map is labelled with a credibility measure that is updated with each scan [41].

3.3. Updating the Map

In building a map, decisions must be made as to how the various sensor information, as well any prior knowledge about the environment, will be combined. For example, when adding line segments to an existing map, the question that must be addressed is how to match a given segment with the elements in the map. If no match is made then obviously the next step is to add it to the map, provided no other processing is required to filter out potential noise or irrelevant data. If a match is found, then some decision must be made regarding whether it is redundant information or not.

Work that has been done in matching laser range data line segments in the process of building a map can be found in [**30**] by Gonzalez et al. A local map of the immediate area is obtained by their radial laser scanner. They update their map by doing a correspondence between this newly acquired local map with a stored global map. The distance between each new line segment endpoint and each global map line segment is computed. A match is found if both endpoint distances fall within a pre-defined threshold. Those line segments which only match one endpoint are fragmented into segments which do and do not match. In the end, all segments for which a match was not found are then added to the global map. Another related work using sonar data is presented in [**21**] by Crowley. The sonar data are first segmented into line segments. Then correspondence is determined by comparing how each newly sensed line segment falls within a "tolerance box" drawn around the stored line segments in the map. A complete correspondence is found if it falls completely within the tolerance box. Other degrees of correspondence are measured depending on how well the line segments fit within these tolerance boxes.

4. Structural Object Recognition

A map of an office environment is mainly composed of certain permanent landmarks, walls and doors. These objects are referred to in this work as the *structural objects*, which are identified and labelled by ORB in SPOTT's map database. A survey of the literature found that there has not been significant research done in recognizing *structural objects* from laser range data. Some of the other approaches to this problem are summarized below.

4.1. Related Work

Work in finding open doorways using video images has been reported by Blaasvaer[**7**] and Sarachik[**58**]. Blaasvaer's method is used for doorway traversal and assumes a priori knowledge of the door positions in the environment. Once positioned in front of a doorway,

it detects the door posts as two straight vertical lines using edge detection, from which it computes the width of the doorway opening. This process is repeated until the robot has passed through the doorway. Sarachik's method utilizes two forward pointing cameras in which depth information is acquired from *stereo*. The door is defined as two vertical edges at the same depth and at a suitable width. An open doorway is separated by at least one other feature of greater depth. The difficulty of detecting doorways using sonar data is shown by Budenske [14]. The problems encountered with sonar are attributed its inherent noise and uncertainty. "Ghost"[3] doors are frequently found, and it is not obvious where the actual doors are on a sonar map, even to the human eye. Multiple reflections that occur near the doorway edges contribute to noise which may block an opening that is actually there.

QUADRIS laser range data are more reliable than sonar and therefore better equipped to track structures such as doorways, as well as determine their state (open/closed). This is useful information for any navigation system where the environment changes dynamically and the status of the various doorways must be continually updated. QUADRIS laser range data is also not as computationally expensive to compute as most video based techniques. On the other hand, QUADRIS data are prone to other problems, as will be discussed in Chapter 5. For example, dark coloured doors, which do not reflect the laser signal well, may not be detected by QUADRIS data.

5. Movable Object Recognition

Object recognition can be defined as the process of identifying a desired object, and if need be, determining its location and orientation in the scene [2]. The "attention problem", or the process of locating the desired object in the scene, is not covered in this work. ORB assumes that the object has been located and is within view of the laser rangefinder when it begins the recognition procedure. There are many different approaches to the object recognition problem, as chronicled by Besl in [4], and the main areas can be categorized as follows. (1) *Data-based recognition*, in which the object models are built from raw sensor data, that are used to compare directly with the input sensor data. (2) *Model-based recognition*, in which the input image is matched to a *predefined* object model that is based on the object's geometrical *features* [18][2]. (3) *Function-based recognition*, an alternative method in which no explicit prior geometric or structural model is used. Objects are categorized by knowledge of their shape and how that relates to their *functions* [62][63]. ORB's approach

[3]These are openings found in the sonar map that appear to be doors, but are not actually there. They are a product of multiple and specular reflections.

to recognizing *movable objects* is most comparable to the *model-based* approach, and also utilizes concepts introduced in the *function-based* area.

The steps involved in designing a model-based system consist of: (i) selecting the type of sensor used for data collection, (ii) selecting the *features* that will provide a *representation* of the collected data and model, (iii) building the *object models*, and (iv) matching the input data to the model[**2**][**18**].

The raw data provided by the sensor, in our case a laser rangefinder, is not of much use unless a suitable *representation* is defined to describe the data and the model. Building a representation generally involves the following steps[**11**]: (i) selection of *features* that will describe the physical properties of the object, (ii) extracting these features, and (iii) combining these features to generate object descriptions or *representations*. Recognition is done by matching these representations with similarly represented objects in a model database.

The recognition process is completed when the input data are matched to an object model. The *matching* problem is the process by which the set of features detected in a given image are matched with those which correspond to the model's description. The features chosen for the model will ultimately determine how the matching process proceeds. For a survey of the various approaches or for more detailed information about the above concepts, see [**2**][**4**][**11**][**18**].

5.1. Related Work

Not much work was found in the literature concerning the recognition of *movable objects*, especially using laser range data. The work that has been done either require multiple types of sensory data[**32**], or a multiple camera setup[**56**].

Grandjean et al.[**32**] proposed a model-based, multi-sensory approach requiring the depiction of a scene by contour images, stereovision 3D line segments, range 3D faces and colour images. Their object models are constructed of subparts that are represented by either a face, segment (edge or line), vertex or cylinder. Their system was tested on a scene with a chair as their target object. Their chair model consisted of faces for the backrest and seat surfaces, along with edge segments to represent the legs. The backrest and seat are detected from dense 3D range data, while the legs are identified in camera images. This method was shown to be effective even in cluttered environments. However, the overhead seems to high considering the number of sensors involved, and the fact that several viewpoints of the scene are needed.

Sandakly et al.[**56**] developed a 3D scene interpretation system for use on a mobile robot, using a sensor platform consisting of three cameras. The data obtained from this sensor system are 3D segments reconstructed with trinocular stereo. The objects in these scenes are polyhedral, and their subparts are modelled by *facets* (i.e., planes). The scenes are reconstructed by merging different viewpoints taken by rotating the robot and/or the cameras. This system was shown to be capable of identifying chairs and tables in a scene. The system searches for groups of 3D segments which form a set of facets. These facets are then matched to each object model in the database. Chairs are modelled by their backrest and seat planes, tables by their tabletop surface.

5.1.1. *Sparse Range Data*

ORB recognizes *movable objects* using a sparse set of laser scan lines (5 or 6 for each planar surface). This is a novel approach to identifying objects of this scale. There has been work done with sparse range data to recognize smaller scale polyhedral objects. Qiang et al.[**52**] introduced a model based technique in which two light-stripe range scans were needed to locate and identify certain polyhedral objects. A database of 3D object models is compiled in which each object is represented by a collection of planar facets. The endpoints of the scanned line segments are matched directly onto the edges of each object model. The process is complete once all the scanned line segments are located consistently on the faces of an object model.

Work done at CIM[4] concerning the recognition of geons also utilize sparse range data[**47**][**67**]. This work is developed for use on a mobile robot. The target geon is first located in a scene based on its colour characteristics via a colour camera. Once localized, the geon is scanned with a set of three vertical and three horizontal laser line scans. This information is enough to differentiate between the geons modelled in the database.

5.2. Function-Based Recognition

A different approach, in which there are no pre-defined geometric or structural models, is the function-based approach. An object is recognized based on knowledge about what is necessary for it to *function* as a member of some object category. The thinking is that a different method is needed in order for a "general-purpose" recognition system to be flexible enough to deal with new objects for which it has no explicit prior model. It is not possible to have a geometric model of every object that a general purpose system might encounter.

[4]The Centre for Intelligent Machines at McGill University.

Stark and Bowyer[**62**] proposed a system which takes a boundary surface description of a 3D object, and using only functional definitions, recognizes whether the object belongs to some object category, for example "bed", and into what subcategory of "bed" it belongs. The knowledge base for defining a bed would include an appropriate sleeping surface, stable support for it and appropriate clearance above the sleeping surface. See [**62**][**63**] for more on function-based recognition.

A discussion of how ORB recognizes the *movable objects* will be presented in Chapter 4. The *structural objects* are described more thoroughly in the following chapter.

CHAPTER 3

Map Building and Structural Object Recognition

In navigating through the hallways of an office space, there are several structures that a mobile robot will continually encounter. These are the *structural objects*: walls and doors. In the process of building a map of an office environment, or updating an existing partial map, these landmarks are sought out and labelled by ORB. Assuming that prior knowledge of the environment is available, models of the doorways are specified a priori. The most prominent feature of this model is the doorway width. Each doorway is also labelled with the state in which it was found (*open*, *closed* or *partially open*). Mobile robot navigation and control is undertaken by SPOTT, which maintains the map database.

An outline of how ORB scans an area to build a map and recognize the *structural objects* is shown in Figure 3.1. This also serves as a summary of the topics discussed in this chapter. An overview of ORB's map-building and how it supports SPOTT in maintaining the map database begins the chapter. This is followed by a discussion of how the *structural objects* are modelled and recognized. The gaze control issue is also discussed throughout. ORB positions the rangefinders based on the task command issued by SPOTT. A specific scanning routine is employed to deal with each scenario.

1. Map-Building

As the mobile robot travels through its environment performing various navigational tasks, it generates a map in order to remember the locations of the sensed structures. When a CAD map of the office space is available, it is the foundation from which a *complete* representation of the environment is built. The CAD map only contains the permanent structures in the area. In any dynamic environment, a complete map must contain all other objects that could not be mapped a priori (e.g., boxes, people, chairs, etc.). SPOTT maintains a map database containing the CAD map as well as additions provided by the sensors on-board the mobile robot. SPOTT must fuse data from the various sensors (sonar,

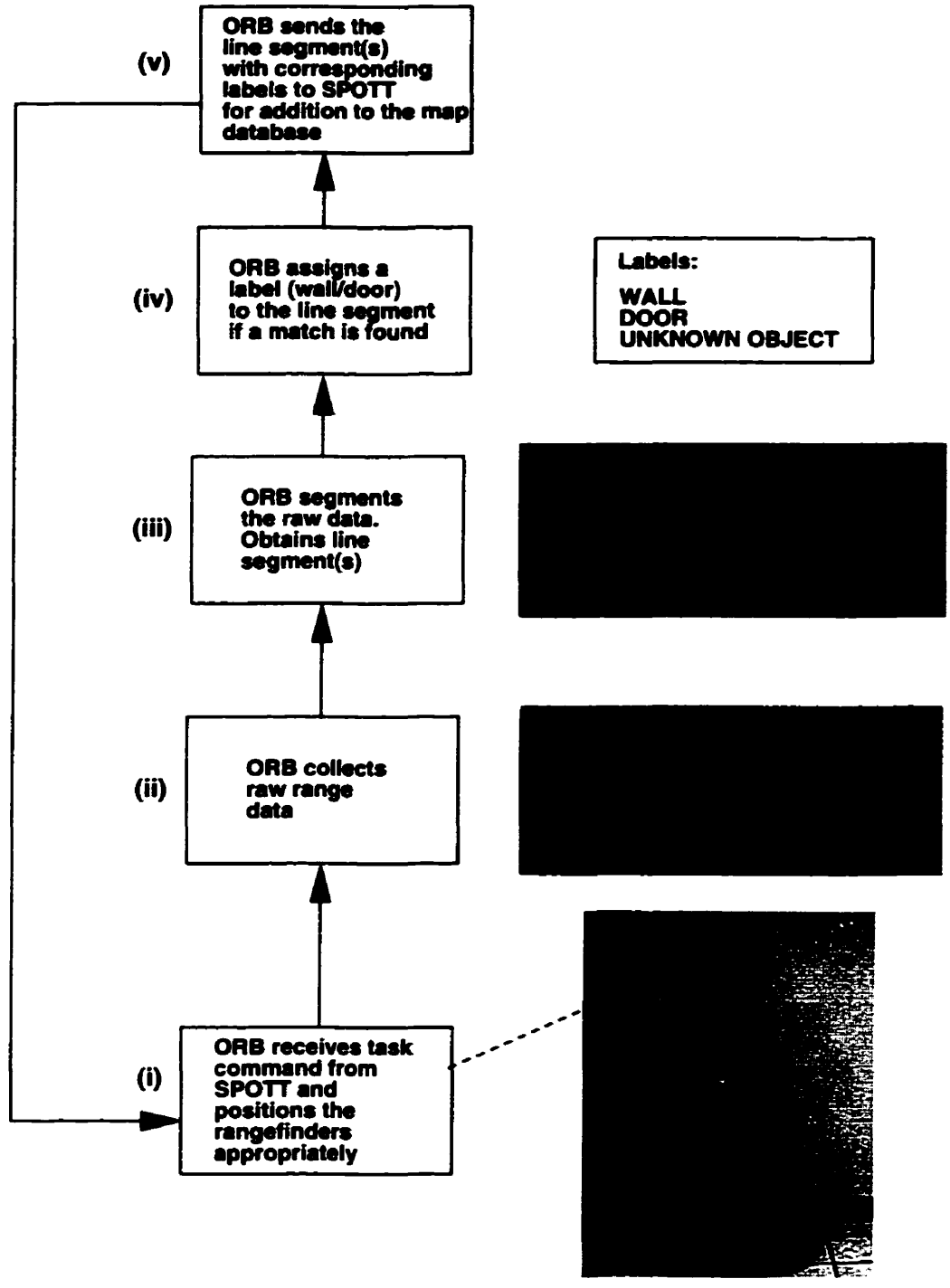


FIGURE 3.1. **Outline of ORB's Map-Building Process.** This is an outline of the steps taken by ORB when scanning an office area for map-building and recognition of structural objects. (i) Based on the task command sent by SPOTT, ORB positions the rangefinders appropriately. ORB's focus of attention differs for each given scenario (e.g., hallway or room). (ii) ORB takes a range scan once the rangefinders are properly positioned. (iii) ORB computes a line segment representation of this data. Range data is processed and handled in the form of line segments. (iv) Once the appropriate recognition process is completed, a label (wall/door/unknown object) is assigned to the line segment(s). (v) The line segment(s) and label(s) are sent to SPOTT to be incorporated in the map database. ORB continues this process until a new task command is received.

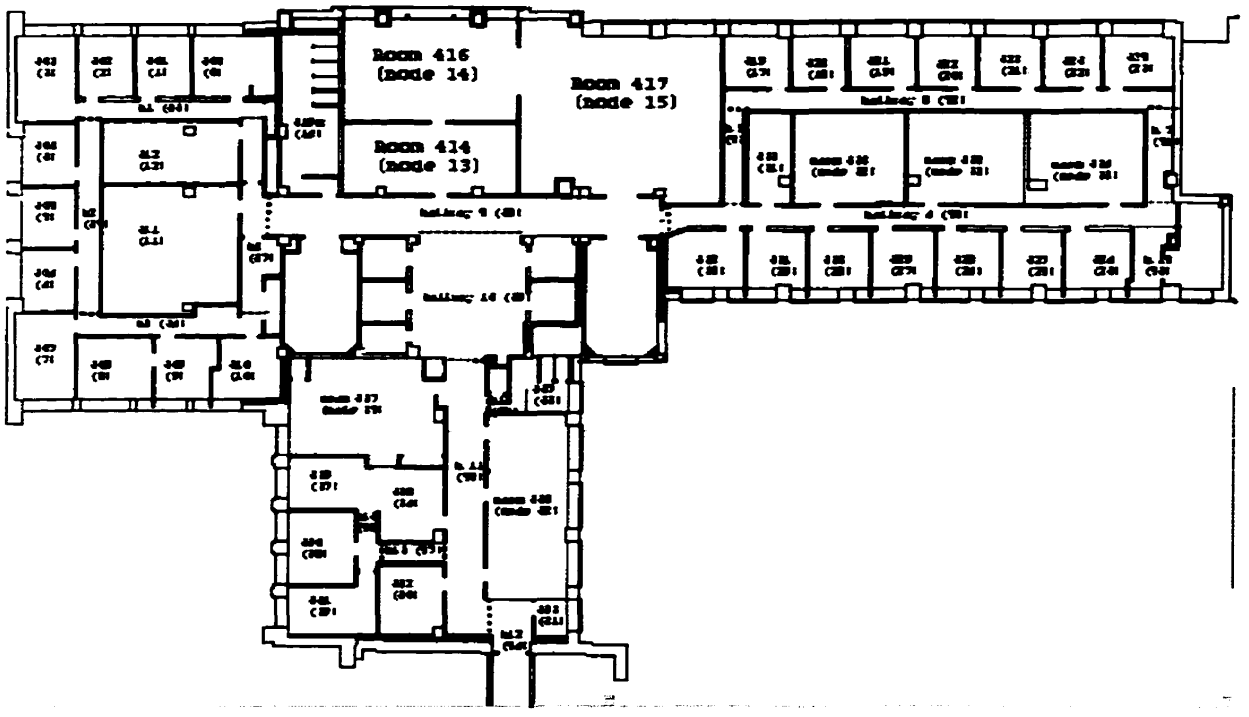


FIGURE 3.2. **CAD Map of the CIM Floor.** This is the CAD map of the CIM office space, which is on the 4th floor of the McConnell Engineering building at McGill University.

QUADRIS, infrared, bumpers) into one map. ORB provides SPOTT with laser range data, acquired via QUADRIS, in the form of *line segments*. These line segments are sent to SPOTT with labels defining what ORB has interpreted these line segments to represent (e.g., wall, open door, unknown object).

1.1. ORB-Built Maps

This work is concerned with an indoor office space with the following two scenarios possible: (i) the mobile robot is sent into a completely unknown area and a map is built from scratch, or (ii) given a pre-defined map of the permanent structures, the map is updated as new obstacles[1] are encountered. For example, when the CIM office space is used as the testing environment, a CAD map of this floorplan is available and used in SPOTT's map database. The CAD map of the CIM floor is shown in Figure 3.2.

An office space, however, is a dynamic environment so a map of this floor cannot remain static. A CAD map alone will not be complete as there are objects that cannot be mapped

[1]These are any objects that are not found in the pre-defined map. A CAD map will only be composed of the permanent landmarks on the floor (i.e., walls, doorways, stairwells, etc.). Any other obstacles must be added to the map.

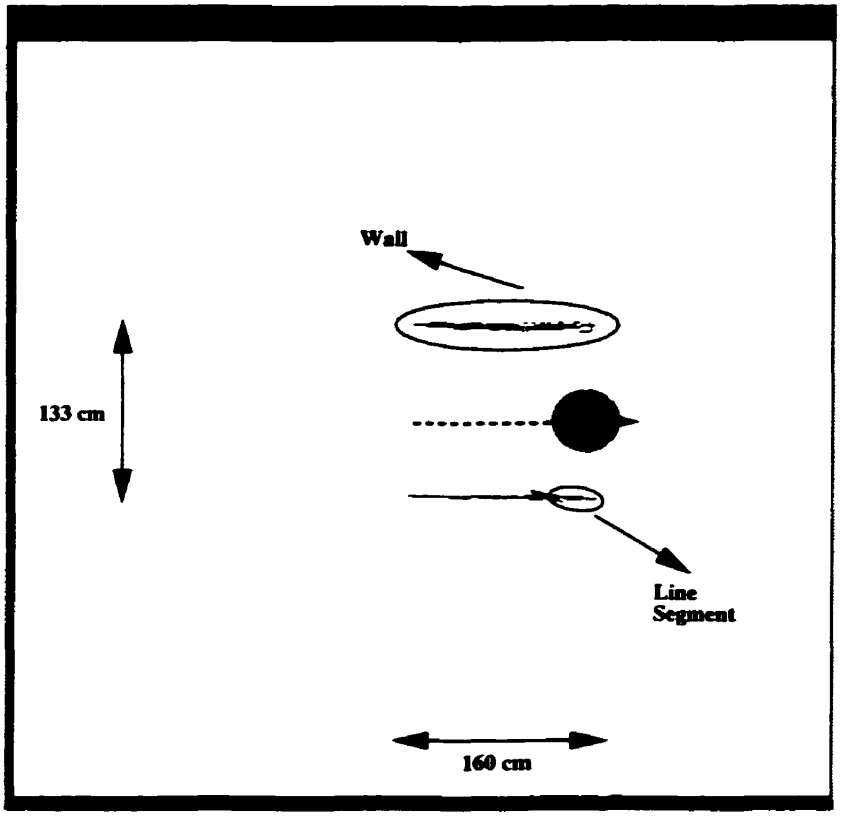


FIGURE 3.3. **A Raw Map Generated by ORB.** This is an ORB-built partial map of a hallway. The line segment indicated is a reading along the laser line of one BIRIS range scan. The data represents the accumulation of QUADRIS range readings taken of the walls as the robot navigated down the hallway. The mobile robot and its path down the center of the hallway is shown.

a priori. These include any movable furniture items (chairs and desks), as well as doors which can be opened or closed. Therefore, the map database maintained by SPOTT needs to be continually updated by sensors on the robot.

SPOTT's map is a 2D representation of the floor, as the world is projected onto a 2D plane. The mapping information passed by ORB to SPOTT is a set of 2D line segments. As well as obtaining new data for SPOTT's map, ORB also *labels* landmarks (i.e., walls and doors) that it recognizes. These landmarks are the *structural objects* and will be discussed more fully in section 2 of this chapter. A sample map generated by ORB in a hallway is seen in Figure 3.3. The map consists of a series of line segments obtained from the QUADRIS sensor platform. The segmentation of the raw data is discussed in the following section.

1.1.1. *Line Segmentation*

The raw range data acquired from the QUADRIS laser rangefinders are an array of 640 readings along the horizontal laser line projected onto the scene. To facilitate the handling of this data, these range points are fit to line segments, which lie on the 2D plane parallel to the floor. The segmentation is based on Ramer's method [**53**].

This procedure involves approximating a polygon fit to a plane curve. The fit criterion is based on a maximum allowed Euclidean distance from the curve to the approximating polygon. The only parameter required is a *threshold* value for this maximal distance. The procedure begins by assuming that a single line segment fits the curve, with the first and last points of the curve being the endpoints. If the point on the curve furthest from the approximating line segment exceeds the *threshold*, then this point becomes the third vertex on the approximating polygon. This process is then recursively repeated on the two newly created line segments. New vertices are added to the approximating polygon until the distance threshold is satisfied for every point on the curve. There are other approaches which also offer fast algorithms, such as one proposed by Wall and Danielsson[**66**], but due to the relative simplicity of most cases dealt with here[2], Ramer's procedure was found to perform most efficiently. Several examples of line segmentation are shown in Figure 3.4.

1.2. SPOTT

SPOTT is a robot control architecture that is implemented as a "real-time and parallel system of concurrently executing and co-operating modules" [**69**]. It is a real-time control system that can dynamically adapt to a changing environment "in order to successfully execute and complete a set of navigational tasks for an autonomous mobile robot" [**68**]. SPOTT keeps a map database containing an a priori defined CAD map, a graph abstraction of the CAD map, as well as a map of newly detected features. These newly sensed features are generated from any of the on-board sensors: sonar, QUADRIS, infrared proximity or bumper system. Localization is done by SPOTT using an existing sonar localization technique[**43**].

The importance of finding correspondences between the stored map and updated sensor data stems from the fact that SPOTT gathers labels sent by ORB and allocates them to the appropriate locations on the map. ORB sends the line segment endpoint coordinates of each label and SPOTT does a simple search through its map looking for a match. This search is based on computing the Euclidean distance between the ORB line segment endpoints and

[2]Most of the objects scanned (e.g., walls and doors) return range arrays that are segmented quickly using Ramer's method since they are generally single line segments.

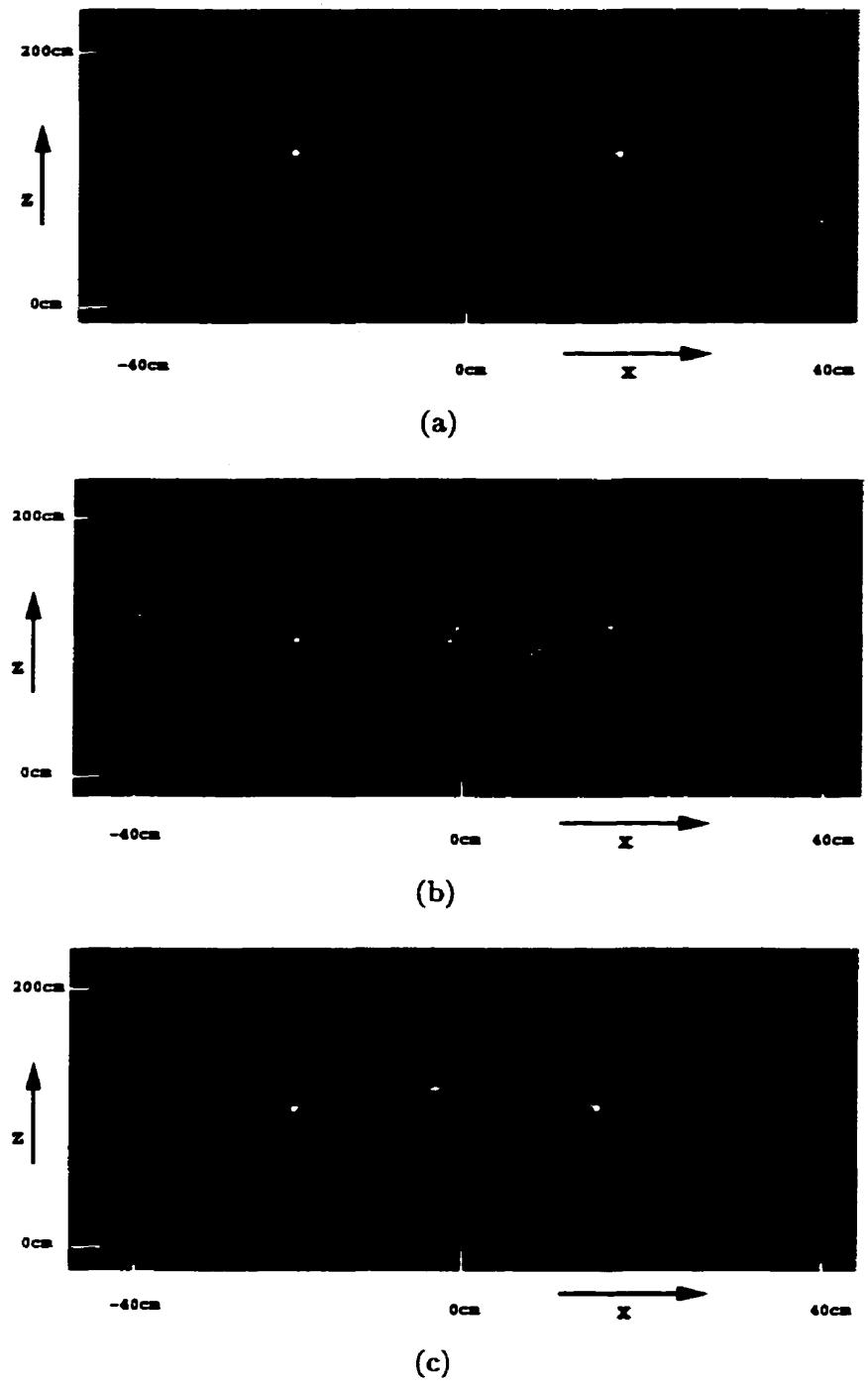


FIGURE 3.4. **Line Segmentation of BIRIS Range Data.** Various examples of segmentation of the BIRIS range data are shown. The raw data is the lower of the two plots, with the plot above being the resulting approximation using Ramer's segmentation method. (a) This is a simple scan of a wall that yields one line segment. The threshold for a wall segment is 3 cm, which was not exceeded in the array. (b) A step is shown, which results in three line segments. The step is 10 cm which exceeded the 3 cm threshold. (c) This is a scan of a corner of a room. The result is two line segments with the middle point, the furthest point from the original approximating line segment, being the third vertex in the approximating polygon.

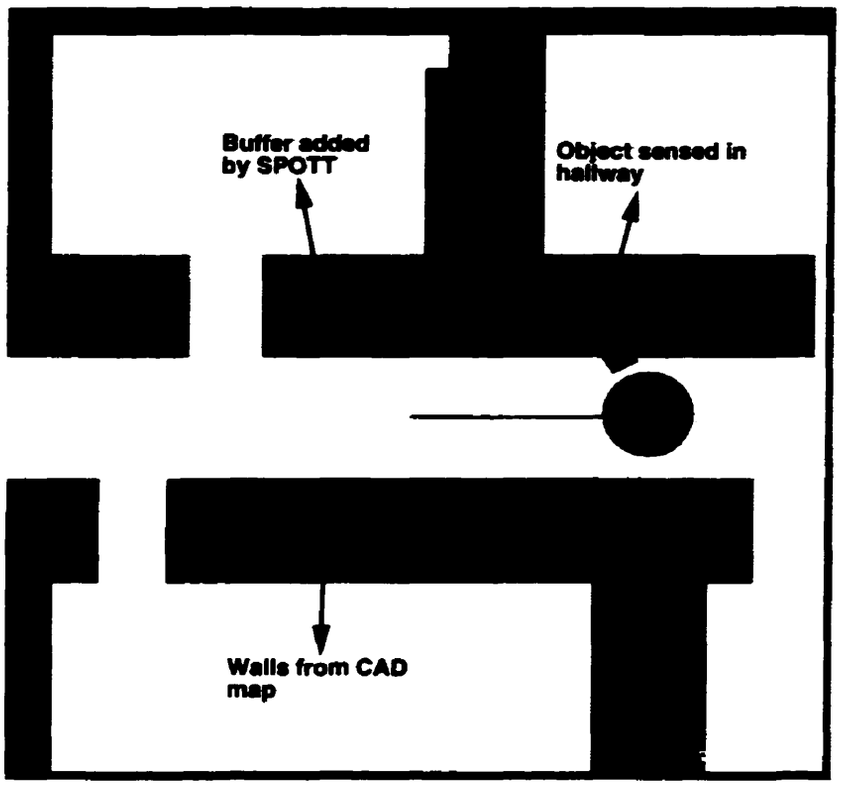


FIGURE 3.5. **Example of a SPOTT Map.** This is a map generated by SPOTT. The walls are from the CAD map of the floor. The buffer added by SPOTT around all objects is for added protection against collisions.

the data in SPOTT's map[**69**]. The distance threshold for accepting a match is 10 cm. If a match is found, then the appropriate location on the map is designated with ORB's label. If the line segment is not matched, then the label and line segment are simply added to the map. SPOTT builds its map by fusing data from the various sensors with the stored CAD map via the aforementioned comparison. The data are accepted in the order that they arrive on a "first come first served" basis.

An example of a map generated by SPOTT is shown in Figure 3.5. The CAD map is used as a starting point with additional updates from the on-board sensors. SPOTT adds a border around the elements of the map as an added buffer against collisions.

2. The Structural Objects

During the course of compiling information for the map database, ORB also recognizes and labels the *structural objects*: walls and doors. These objects are found while performing a *Scan Hallway* task. As SPOTT navigates the mobile robot around an office environment, it will call on ORB to perform this task. The *structural objects* are not scanned to be fit to 3D models like the *movable objects*[3]. Instead, they are recognized using 2D features that are based on prior knowledge of the environment. In recognizing a door for instance, the main feature is the doorway width. There is no need to acquire a complete $2\frac{1}{2}$D range scan of the entire doorway to determine its width. A series of horizontal line scans spanning the width of the doorway[4] will suffice. Another issue is speed. Since the goal is to recognize doorways as the robot travels down a hallway, it is not practical for the robot to stop at each door to acquire dense range scans.

ORB's process of generating a map is the same regardless of whether SPOTT has a stored (i.e., CAD) map or not, since ORB attempts to build as complete a map as possible with QUADRIS range readings. This allows ORB the flexibility of performing in either scenario. How SPOTT processes ORB's measurements is dependent on the task being performed and the resources it has at its disposal. If the area to be explored is completely unknown and no CAD map is available, SPOTT may use ORB's results on its own to build the map database, or it may decide to fuse ORB data with other sensor data such as sonar. If a CAD map is accessible, such as the one shown in Figure 3.2 of the CIM environment, then SPOTT uses it as the main map in its database and adds sensor data (i.e., QUADRIS, sonar, infrared) as it becomes available. Other than the doorway width information, which is used to build the door model, ORB does not utilize any prior knowledge that SPOTT may have in possession (i.e., a CAD map). ORB detects *structural objects* by matching the range data it obtains from the scene with expected range profiles of a wall or door. A discussion of how *structural objects* are scanned and recognized follows.

2.1. Walls

Walls are the major structural units of hallways, and are the main building blocks of a map of an office area. There is no prestored model for walls based on CAD data or any other a priori known information. ORB's recognition of walls is based on a set of criteria and assumptions which is compared to the data it obtains after receiving a *Scan Hallway* task command from SPOTT. ORB relies on SPOTT to keep track of the robot's position and to

[3]The *movable object* models are discussed in Chapter 4.

[4]This series of scans generally number four or five at 5° increments.

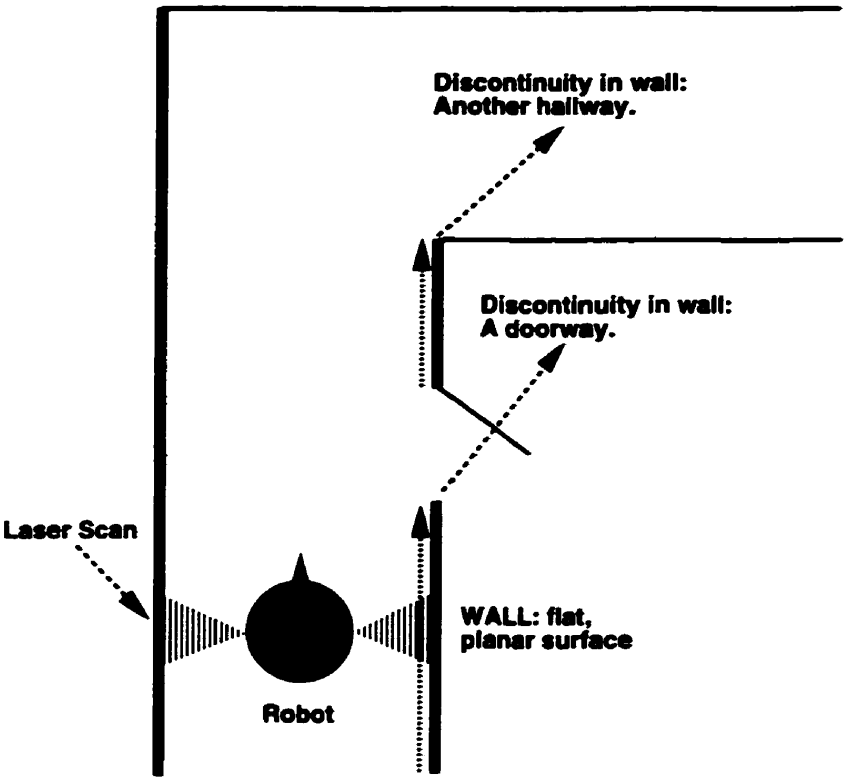


FIGURE 3.6. **Properties of a Wall in a Hallway.** A robot in a hallway scanning walls to either side. A wall is a flat, planar surface whose only discontinuities are doorways and other hallways.

know when the robot is in a hallway. Therefore, when SPOTT issues the *Scan Hallway* task command, ORB assumes that the robot is facing down the length of a hallway, as seen in Figure 3.6. Hence, there will be walls to either side of the robot and ORB searches for them accordingly. ORB's recognition of walls is a learning process, in which it locates a new set of walls each time it receives a new *Scan Hallway* task command. This is done every time the robot has entered a new hallway.

ORB defines a wall as a planar surface that delineates a hallway, with discontinuities occurring only at doorways and at junctions with other hallways. These properties are shown in Figure 3.6.

ORB builds a 2D map of the environment, so walls are projected onto a 2D plane. Walls are detected by horizontal laser line scans which provide the distance of the wall to the robot. The range points along the laser line are segmented into *line segments* for incorporation into SPOTT's map database. A wall returns a *single* line segment along the laser line since it is assumed to be a flat, planar structure. An sample plot of a range scan of a wall was shown in Figure 3.4(a).

2.1.1. *Initialization*

ORB begins a *Scan Hallway* task assuming that:

(i) The robot is in a corridor.

(ii) The robot is situated between two walls.

(iii) The robot is facing down the length of the hallway[5]. The BIRIS rangefinders in their home positions would then point towards the walls to either side of the robot. This initial position was previously shown in Figure 3.6.

Once these conditions are met, the first step in the wall recognition process, the *initialization* stage, is executed. This procedure carries out the following two functions: (i) verifies that scans of the wall return a single line segment of range data, indicating that a flat surface has been found and (ii) notes the initial distance between the wall and the center of the robot. This distance is referred to as *Wall_Distance* and is the basis of ORB's model of the currently scanned wall. *Wall_Distance* defines where the wall is expected to be found with respect to the robot. Subsequent scans of this wall must be found within a *threshold* of this stored distance, and this threshold is called called *Wall_Threshold*[6]. Any scan which results in multiple line segments, or is not found at *Wall_Distance* within the allowable threshold (i.e., *Wall_Threshold*), indicates that the wall is no longer in view.

The *Wall_Distance* value for the wall scanned by BIRIS_1[7] to the right of the robot is called **Wall_Distance1**. **Wall_Distance2** is used for the wall on the left which is scanned by BIRIS_2. The *initialization* procedure locates the walls and initializes the two values, *Wall_Distance1* and *Wall_Distance2*. Since SPOTT will navigate the mobile robot down the center of the hallway[**69**], each wall will remain in view of the same rangefinder for the duration of the robot's trip through this hallway. When the robot enters a new hallway, or rotates such that the walls are no longer in view of the rangefinders as outlined above (e.g., a rotation of 180°), the current task ends and a new *Scan Hallway* task command is sent by SPOTT when the robot is properly positioned again. This spawns a new initialization process in which the *Wall_Distance* values are recomputed. This *initialization* process for scanning a hallway is outlined in Figure 3.7.

---

[5]The positioning of the robot is done by SPOTT. SPOTT sends ORB a *Scan Hallway* command only after properly positioning the robot.

[6]For the experiments in this thesis, the threshold value was 7 cm, which was determined after accounting for factors such as sensor uncertainty.

[7]The BIRIS sensors on the QUADRIS platform will be denoted as follows: BIRIS_1 is on the right side of the robot, BIRIS_2 is on the left.

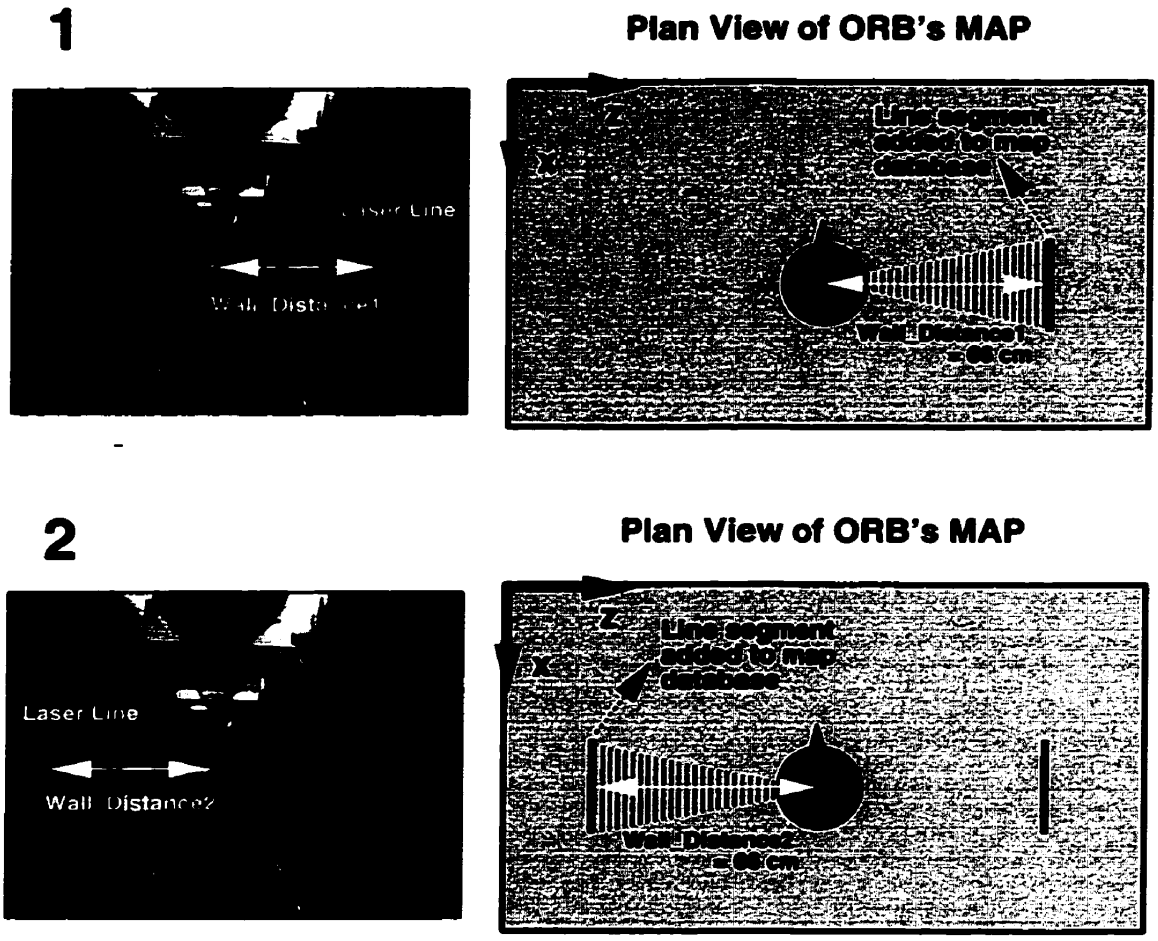


FIGURE 3.7. **The Scan Hallway Initialization Process.** This initialization routine starts the *Scan Hallway* task as ORB locates the walls in the hallway. With the BIRIS rangefinders in their home positions, ORB begins by taking a scan with BIRIS_1, the sensor on the right. The range scan produces a single line segment whose distance, in the Z-axis direction, to the center of the robot is stored as *Wall_Distance1*. A total of three range readings of the wall are averaged to calculate this *Wall_Distance1* value. ORB repeats this process with BIRIS_2, the sensor on the left, and computes *Wall_Distance2* to model the wall on the left.

2.1.2. *Updating Procedure*

As the robot travels down the hallway after the *initialization* procedure, each subsequent scan of a wall is compared with its *wall model*[8]. The comparison involves determining: (i) if the scan produced a single line segment and (ii) if the scan is found at the expected distance (*Wall_Distance*) from the robot, within the allowable *threshold* (*Wall_Threshold*). Each new reading satisfying the wall model is used to update *Wall_Distance1* or *Wall_Distance2*. The *wall model* is continuously updated with either

[8]ORB's *wall model* consists of the *Wall_Distance* values for each wall which were determined during the initialization step. The *wall model* is not an actual model, but rather a description containing information ORB has learnt about the walls in the scanned hallway.

the most recently accepted[9] scan, or with an average of a number of previously (e.g., three) accepted scans.

Considerations for choosing the *threshold* value include the BIRIS sensor's uncertainty. BIRIS is accurate to 2 cm up to a distance of 150 cm, which is the normal operating range when scanning a CIM hallway[10]. Again, it cannot be assumed that the initial *Wall_Distance* values will remain perfectly accurate after the robot has travelled some distance down a hallway. There will be some drift in the robot's path, as it cannot realistically take a perfectly straight trajectory down the middle of a corridor. However, SPOTT's control architecture [**68**] will ensure that the robot does not deviate much from a linear path, so this is not too critical an issue. For these experiments, 7 cm was found to be a suitable *threshold* for wall detection. This value is referred to as *Wall_Threshold*.

An example of the wall model *updating* procedure is shown in figures 3.8 and 3.9. This is a simple example in which the robot's path does not deviate from the center of the corridor and all scans fit the wall model. ORB's scanning pattern in a hallway will be discussed in greater detail in an upcoming section, but is previewed in figures 3.8 and 3.9.

(i) After the initialization stage is complete, BIRIS_1, the rangefinder on the right, is panned ahead 30^o in a search for openings further up along the wall. Since the wall is still in view at this point, a single line segment of range data is retrieved. The distance of this latest scan from BIRIS_1 is termed *New_Range_Scan1*. This reading is compared to the stored *Wall_Distance1* value in the *wall model*, and is found well within the allowable threshold since they match perfectly. Therefore, *Wall_Distance1* is updated with the latest scan of the wall (*New_Range_Scan1*). Another option for updating the *Wall_Distance* values is to average the latest reading with a set of previous acceptable measurements of the wall. A scan of the left wall is then taken by BIRIS_2. BIRIS_2 remains in its initial home position, fixated on the left wall while BIRIS_1 pans ahead. BIRIS_2 obtains a single line segment which is at a distance called *New_Range_Scan2*. This reading is within the allowable threshold of *Wall_Distance2*, so Wall_Distance2 is updated accordingly.

(ii) For the next step, BIRIS_1 is pointed towards the front of the robot to check for obstacles in the hallway. The right wall is no longer scanned so *Wall_Distance1* is not updated. BIRIS_2 repeats the previously described operation by scanning the

[9] An acceptable scan is one that (1) returns a single line segment of range data and (2) is within the allowable threshold of the compared *Wall_Distance* value.

[10] The width of the CIM hallways range from 133 to 240 cm. With the robot generally in the center of the hallway, the sensor will usually be scanning walls at most 120 cm away.

left wall from its home position. The robot has since travelled up the hallway so another section of the wall is now in view. An acceptable range reading is acquired so *Wall_Distance2* is updated.

(iii) The sequence is completed as BIRIS_1 returns to its home position to take a scan of the right wall, updating *Wall_Distance1*. BIRIS_2 then takes a range reading of the left wall to update *Wall_Distance2*. The robot is in motion during this entire routine so new areas of the walls are being scanned at each step.

This process repeats with BIRIS_2 panning ahead and BIRIS_1 remaining in its home position pointing towards the right wall. The rangefinders alternate in this fashion until ORB is given a new task to perform, or an opening is discovered along a wall. While scanning a wall, if there is a jump in range data *greater* than the acceptable distance[11], then an *opening* has been discovered. This opening is either a doorway or another hallway. If the reading finds something *closer* than the acceptable distance, then an *obstacle* has been found in the hallway. These two events indicate that the wall is no longer in view. The discovery of an opening along a wall is depicted in Figure 3.10. The case of the robot encountering an opening (i.e., a doorway) will be discussed in a later section.

To summarize, ORB's **wall detection** consists of the following set of criteria and actions:

(i) The robot must be in a hallway. Mobile robot navigation and positioning are handled by SPOTT.

(ii) A range scan of a wall must return a single line segment of range data after segmentation, an example of which was shown in Figure 3.4(a).

(iii) The scanned distance of the wall to the center of the robot, as shown in Figure 3.8, is compared to the stored distance (*Wall_Distance*) that the wall is expected to be found. *Wall_Distance* was determined during the *initialization* stage. The line segment should lie within the allowable threshold (*Wall_Threshold*) of *Wall_Distance* for this scan to be part of the wall.

[11] This acceptable distance is anywhere within the allowable threshold (*Wall_Threshold*) of *Wall_Distance*.

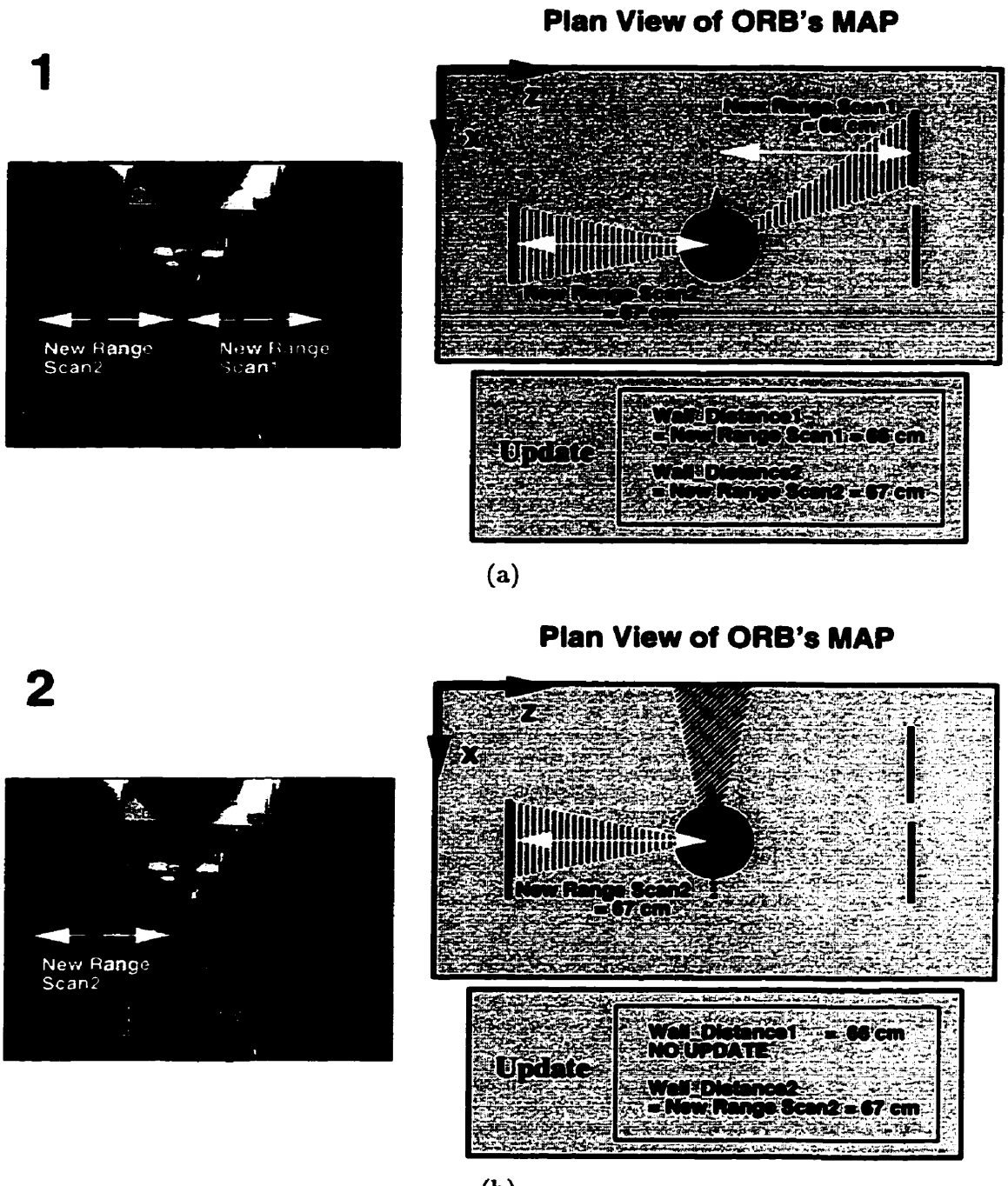


FIGURE 3.8. **The Wall Model Updating Process: Step 1 and 2.** (a) After initialization, BIRIS_1 is moved forward 30° to scan the right wall further ahead along the corridor. This scan produces an acceptable single line segment, as its distance in the Z-axis direction to the center of the robot is within the allowed *threshold* of *Wall_Distance1*. *Wall_Distance1* is updated with this latest scan. ORB then takes a reading from BIRIS_2, which has remained stationary in its home position, to update *Wall_Distance2*. (b) The robot has moved ahead in the hallway and BIRIS_1 is sent to scan the front of the robot. *Wall_Distance1* is not updated as the right wall is not being scanned. ORB then takes a scan from BIRIS_2 of the left wall. A line segment representing the left wall is found and *Wall_Distance2* is updated.

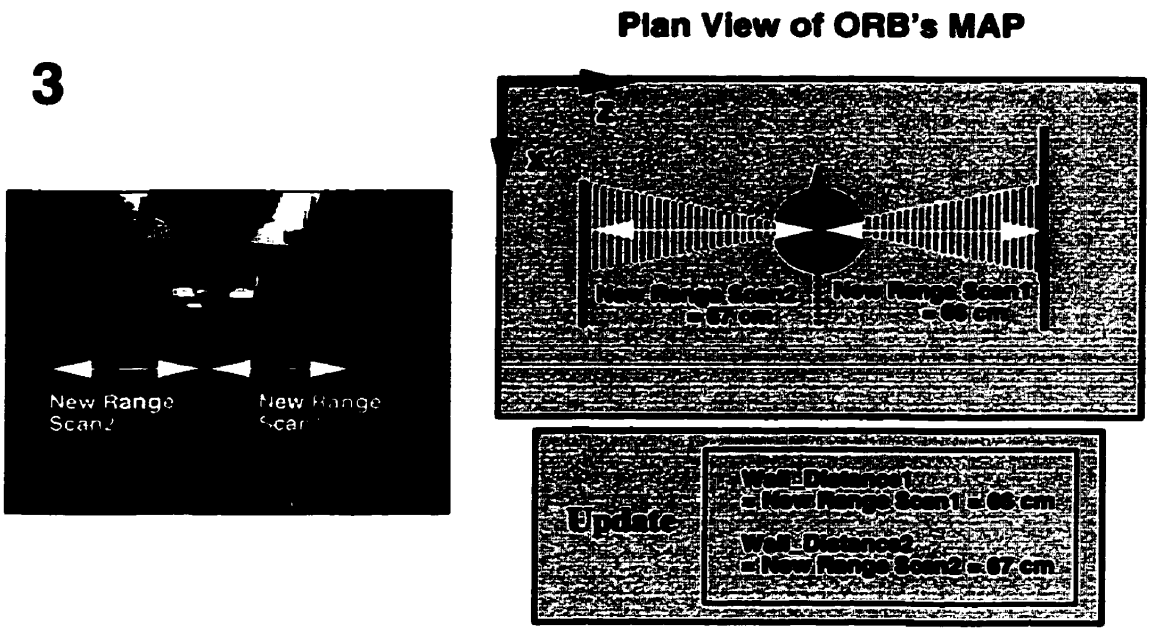


FIGURE 3.9. **The Wall Model Updating Process: Step 3.** BIRIS_1 returns to home position and ORB scans the right wall to update *Wall_Distance1*. ORB then takes a range reading from BIRIS_2 to update *Wall_Distance2*. Scans of both walls yield acceptable results and the corresponding *Wall_Distance* values are updated. The above procedure is repeated with BIRIS_2 panning forward (30° and directly ahead) along the left wall and BIRIS_1 remaining fixed on the right wall in its home position. This sequence continues until completion of the *Scan Hallway* task, or until an opening is found. The action taken for scanning an opening is described in a later section.

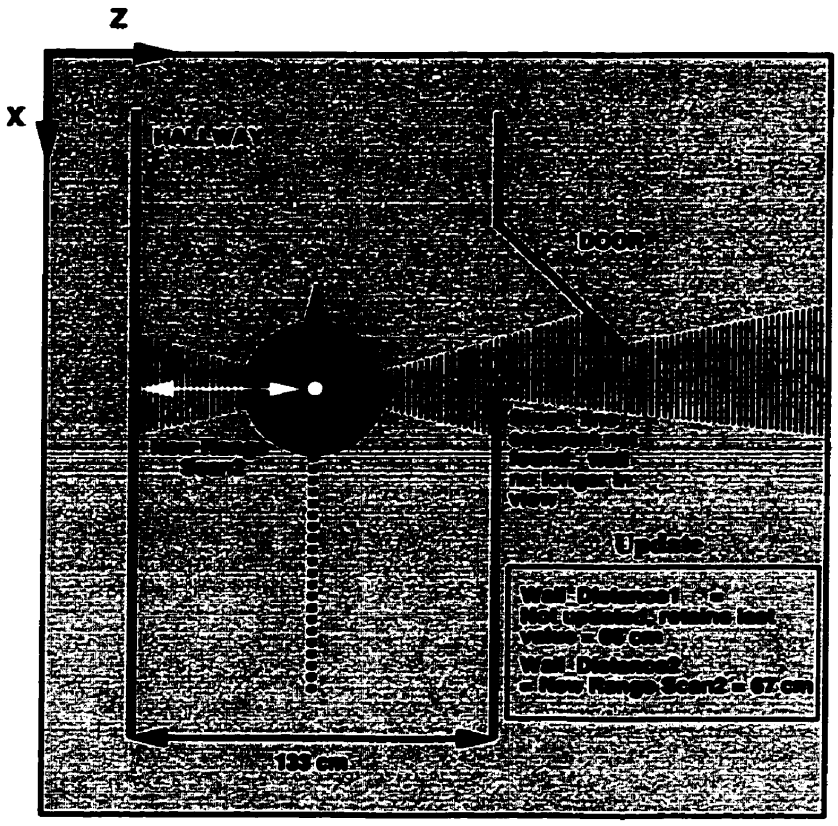


FIGURE 3.10. **Opening Detection.** The left wall is detected and the wall model is updated with this new reading (*New_Range_Scan2*). But the right wall is no longer in view as the range scan of that wall returns multiple line segments well beyond *Wall_Distance1*. Therefore, *Wall_Distance1* retains its previous value and is not updated.

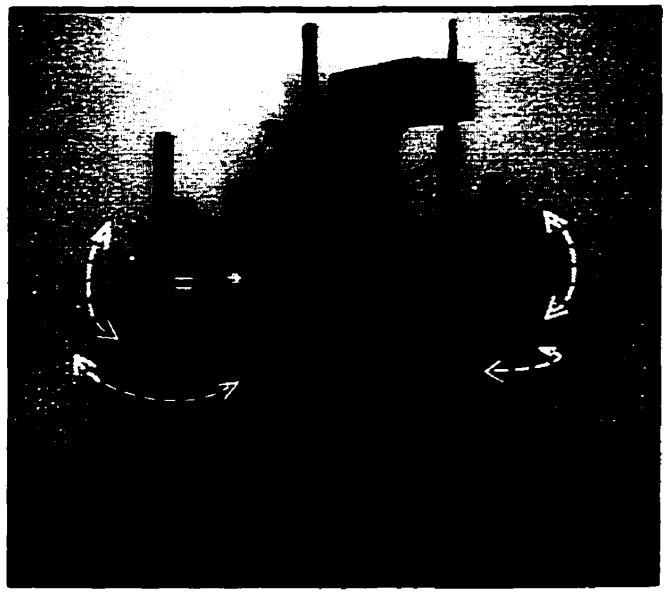


FIGURE 3.11. **Pan/Tilt Movements of the QUADRIS Rangefinders.** The range of motion of the BIRIS laser rangefinders are indicated. The PTUs have two degrees of freedom: along the PAN and TILT axis.

2.2. Map-Building Gaze Control

In building a map of a region, having prior knowledge of the surroundings will provide clues as to what parts of the scene should be focused upon. ORB has two modes of operation when scanning an office space: (i) *Scan Hallway* mode and (ii) *Scan Room* mode. Each has a slightly different scanning pattern for controlling the rangefinders, as each area has different points of interest. The QUADRIS system has two BIRIS laser rangefinders that are movable along a *pan* and a *tilt* axis, as shown in Figure 3.11.

Walls and doors have vertical planar surfaces that do not vary over their height, allowing these structures to be modelled using 2D features. Since a horizontal range profile is used to detect these objects, the *tilt* angles of the PTUs are fixed at a pre-determined position. The rangefinders are usually kept at the home tilt position, with the laser line projected horizontally at a height of 120 cm from the floor. This is the tilt position used for experiments in this thesis, as was shown in Figure 3.9. The surroundings are then scanned by panning the rangefinder in the appropriate direction.

When ORB scans a hallway, its attention is focused mainly on the walls to either side of the robot. A hallway is generally uncluttered and the robot's path will usually be clear, leaving ORB to concentrate on the structures to its sides. Upon entering a room however, obstacles are likely to appear in all directions, so ORB's focus of attention is equally divided

between the areas to the front and to the sides of the robot. It is redundant to scan behind the robot since that area has already been traversed.

2.2.1. *Scanning a Hallway*

ORB's attention to the walls on either side of the robot when travelling down a hallway, is influenced by the honeybee biological vision system[**60**][**61**]. It has been suggested that moving insects infer range from the speed of images on its retina[**16**][**59**]. The faster an object appears, the closer that object will be. Srinivasan [**60**] conducted experiments in which honeybees were trained to travel down a tunnel to reach a feeding site. It was found that the honeybees would tend to travel in a linear trajectory down the middle of the tunnel, as opposed to some random path. Srinivasan concluded that the honeybees maintained an equal distance to each wall by balancing the apparent velocities of the images of the walls in each eye. Their eyes point out laterally at about 180^o [**57**], so each eye has a wall in view. The bee would be situated in the middle of the corridor if the velocity information computed in the left eye was the same as that in the right eye. The honeybee uses optical flow cues [**61**] for measuring these angular velocities.

Other work that has been influenced by this honeybee centering behaviour includes Sandini *et al.*[**57**], who developed a visual navigation system based on this concept. Their sensor platform consists of two cameras pointing out laterally to either side of the robot, analogous to the eyes of a honeybee. When in a hallway, the cameras are focused on the walls. The image velocities as seen by the two cameras are balanced in order to keep the robot centered in the hallway. To determine and compare the image velocities, the average optical flow field is computed over windows of images acquired from each camera. Range is then inferred in terms of image velocity, much like the honeybee. Coombs and Roberts[**19**] also developed a navigational control system that centers their mobile robot in a corridor by also using optical flow cues. Using essentially the same idea as Sandini, two cameras are mounted looking out towards each wall. The *maximum* optical flows computed from each camera are compared. In both works, some form of texture is needed on the walls to compute the flow fields.

ORB retrieves range data directly from its scans of a wall [**24**], instead of having to infer range from optical flow computations. Since a hallway will usually be free of obstacles, ORB will focus on the sides of the hallway to locate the objects of interest (i.e., walls and doors). ORB will also scan to the front of the robot for collision avoidance purposes although the

sonar sensors, infrared proximity and bumper systems will also detect any objects in the robot's path.

ORB's procedure for scanning a hallway is outlined in Figure 3.12 and the scanning sequence is depicted in figures 3.13 and 3.14. Initially, the rangefinders are in their home positions pointing towards the walls on each side. Range readings *alternate* between the two sensors for the duration of the scanning sequence. After three scans are acquired at the home position from each sensor during the *initialization* stage, BIRIS_1 on the right is panned ahead 30^o to scan further up the right wall. This is to detect any openings the robot may be approaching (i.e., a doorway). Then BIRIS_1 is panned directly ahead to search for obstacles in the robot's path before it is returned to the home position. During the entire time BIRIS_1 is scanning forward, BIRIS_2 remains at its home position fixated on the left wall. This sequence is then repeated with BIRIS_2 scanning forward to the left, with BIRIS_1 remaining in its home position on the right.

2.2.2. *Scanning a Room*

The assumption is that in a given room, objects are equally likely to appear in any direction. An office will contain obstacles (chairs, tables, desks, cabinets, boxes etc.) that may be placed almost anywhere around the room, though most likely against the walls. ORB's focus of attention will reflect this. The focus of attention in a room is equally divided among the three directions the rangefinders point[12]. ORB takes one range scan at each position and obtains a line segment (or segments) that is sent to SPOTT for incorporation in the map database. The scanning sequence in a room would be the same as shown in figures 3.13 and 3.14 for hallways, but without the three consecutive scans that concentrate on the sides of the robot. Instead, it would consist of a single range scan taken at each position.

[12]These three directions are the same as shown for the hallway scanning procedure: home position (to the side of the robot), 30^o forward, and straight ahead towards the front of the robot

Gaze Control: Scanning a Hallway

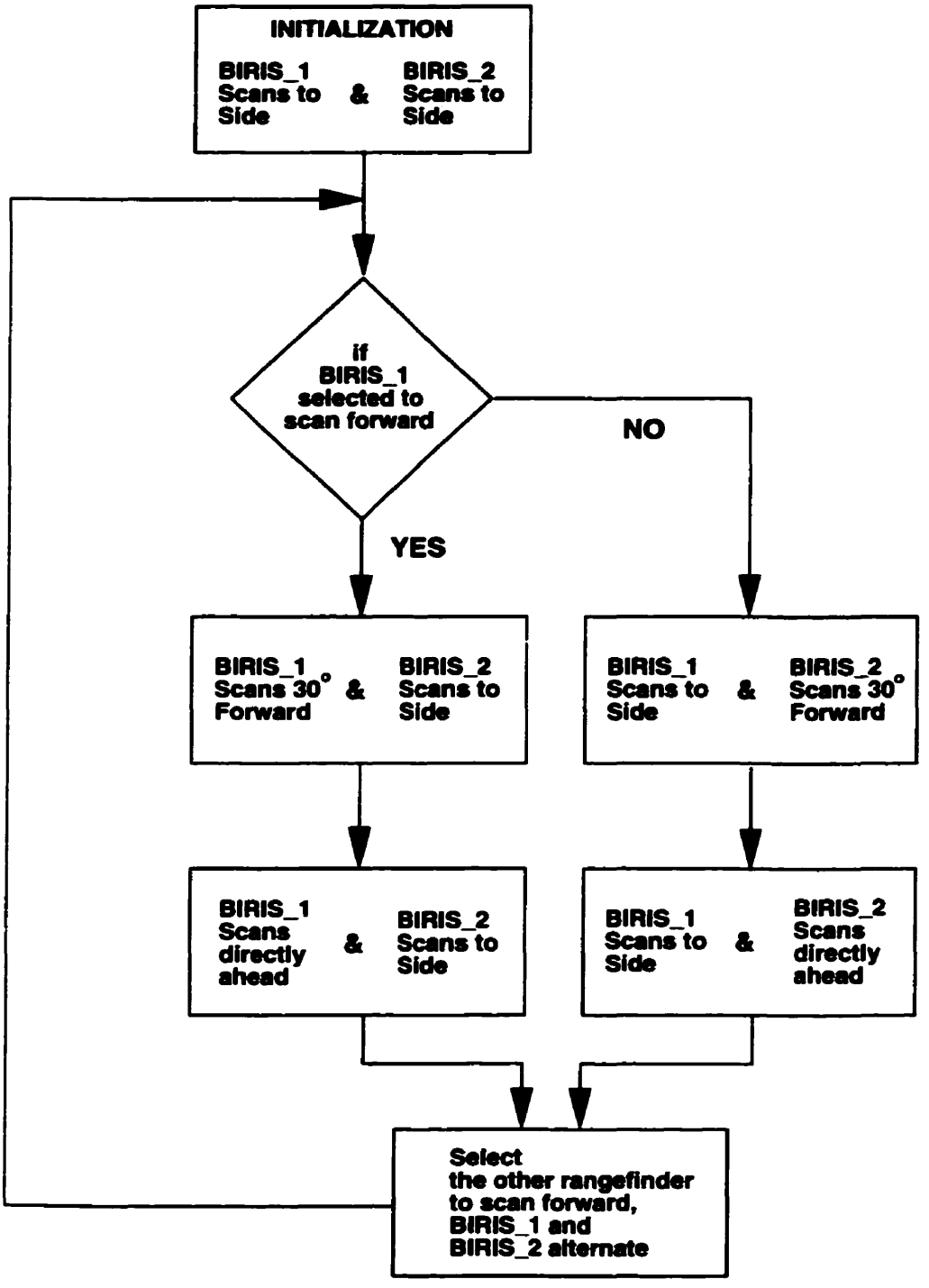
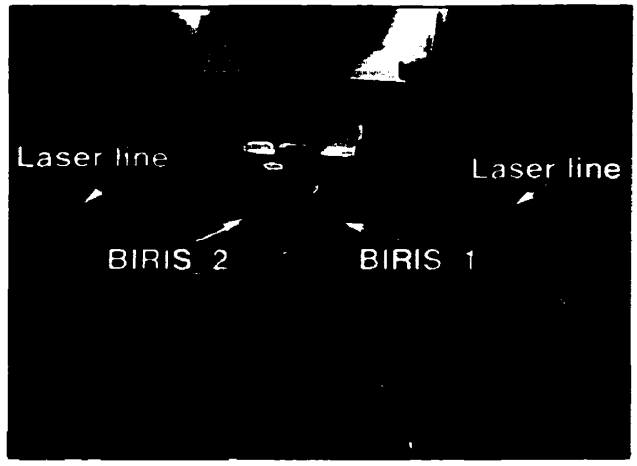
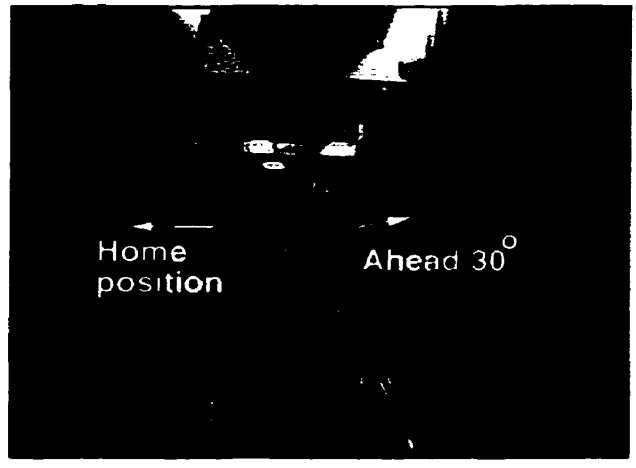


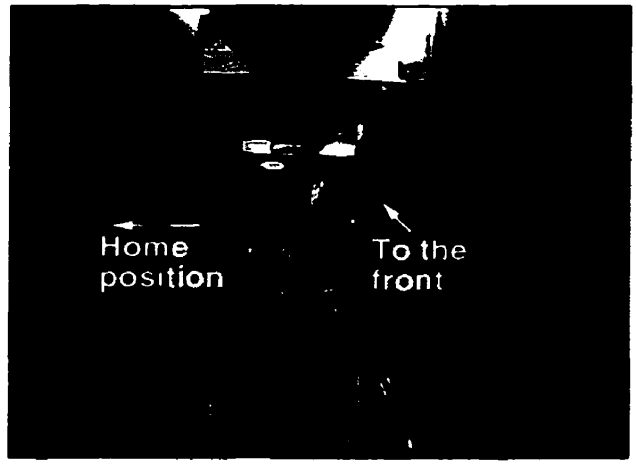
FIGURE 3.12. **Outline of How ORB Scans a Hallway.** A flow chart of how ORB scans a hallway, as depicted in figures 3.13 and 3.14.



(a)

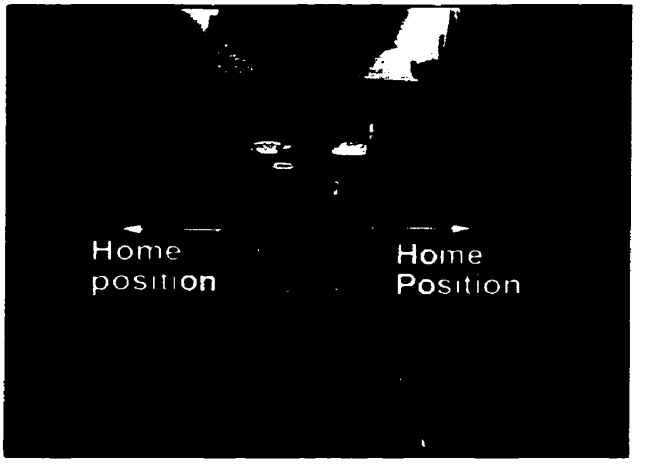


(b)



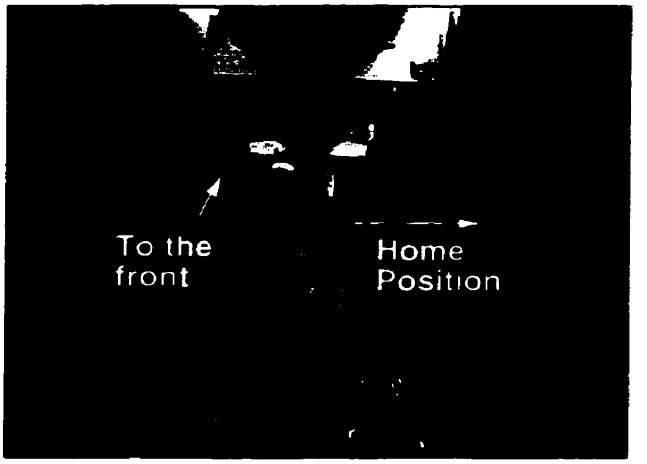
(c)

FIGURE 3.13. **ORB's Scanning Sequence in a Hallway: BIRIS_1 Panning Ahead.** (a) ORB is in *initialization* phase and takes a total of three scans from each sensor in their home position to localize the walls to either side. (b) **BIRIS_1** on the right is panned forward 30° to search for openings ahead in the hallway. **BIRIS_2** remains fixated on the wall to the left, in its home position. (c) **BIRIS_1** then scans the front of the robot for obstacle avoidance.



(a)

(b)



(c)

FIGURE 3.14. **ORB's Scanning Sequence in a Hallway: BIRIS_2 Panning Ahead.** (a) **BIRIS_1** returns to its home position to scan the wall on the right. (b) The scanning sequence is now repeated for **BIRIS_2**, as it is panned forward 30°. (c) **BIRIS_2** scans the front of the robot looking for objects in the robot's path. After it returns to home position, the entire process repeats until: the *Scan Hallway* task is completed or an opening is detected. The process of scanning an opening is discussed later in the chapter.

2.3. Doors

A doorway is a type of *opening* found along a wall. Before a door can be recognized, the wall it is found along must have already been located. This means that a *Wall_Distance* value has been defined for that wall. As previously discussed, the assumption is that walls are flat and planar, with no discontinuities other than doorways and hallways. When a range scan is found beyond the wall[13] past the allowable threshold, this means an *opening*[14] has been detected.

A doorway consists of a door encased in a *doorframe*, as seen in Figure 3.15. When viewed from a hallway, the door is usually inside its frame at some distance from the wall surface. ORB detects this cavity in the doorway as a jump in range data appearing beyond the wall surface (i.e., beyond *Wall_Distance* past the allowable threshold). After discovering this opening, ORB scans horizontally across the breadth of the opening to determine its width. This width provides concrete information as to what the opening is (i.e., doorway or hallway). Doorways have unique widths on a given office floor and are narrower than hallways. Using prior knowledge about the floorplan, the range of doorway widths for the office area is tabulated for the *door model*. The scans through the opening of the doorway are also used to determine the doorway's state (i.e., *open*, *closed*, or *partially open*).

ORB's **door model** is summarized as follows:

(i) The range of doorway widths found on an office floor is tabulated for the door model (e.g., doorway widths range from 74 cm to 94 cm in the CIM office space).

(ii) During a *Scan Hallway* task, each wall in the hallway must be localized prior to any doorway identification. The walls are identified by their *Wall_Distance* values. If an *opening* is found along a wall and its width is determined to be within the range defined above in (i), then a doorway has been detected.

(iii) The state of the doorway (i.e., whether it is open, closed or partially open) is established by its horizontal range profile. The criteria for distinguishing each state are explained in an upcoming section.

[13] The wall is located at *Wall_Distance* from the robot.

[14] The opening is either a doorway or another hallway.

2.3.1. *Doorway Scanning*

A scenario depicting how ORB scans an open doorway is shown in Figure 3.15. The events in this scene are outlined below. A flow chart of the process of scanning an opening is presented in Figure 3.16.

(i) ORB detects an opening to its left, from rangefinder BIRIS_2, via a jump in range data beyond the expected position of the wall *Wall_Distance*. The last point on the wall before the doorway is stored as the starting point of the doorway frame.

(ii) The rangefinder scans horizontally across the opening to locate the boundary of the frame on the other side. The PTU is moved at 5^o increments. A scan is taken and analyzed at each position. The range profile acquired during the traversal of the opening provides information regarding the state of the door (i.e., *open*, *closed* or *partially open*), as will be discussed in later sections.

(iii) The other end of the doorway is discovered as the wall on the other side comes into view. The first point of the wall after the doorway is noted as the end point of the frame. The width of this opening is computed and compared to the range of doorway widths found on the floor.

2.3.2. *Doorway Widths*

The doorway width information is used to create the door model for that particular floor. The range of doorway widths on the floor is used to define what opening widths are to be classified as doorway openings.

The CIM office environment[15] is used for these experiments, a map of which was seen in Figure 3.2. On this floor, doorway widths range from 74 cm, for ordinary office doors, to 94 cm for laboratory doors. Hallway width range from 130 cm to 240 cm. These measurements are labelled on the CAD map in Figure 3.17.

If the opening is found to be a doorway, then it may be in either one of these three states: (i) closed, (ii) open or (iii) partially open.

2.3.3. *Closed Doors*

When seen from the hallway, a closed door appears at some distance from the surrounding wall surface in its doorframe. From inside the room on the other hand, the door is usually flush along the wall with no indentation apparent. These structural descriptions fit most office space environments and form the basis of the closed door model.

[15]The CIM office space is located on the 4th floor of the McConnell Engineering Building at McGill University.

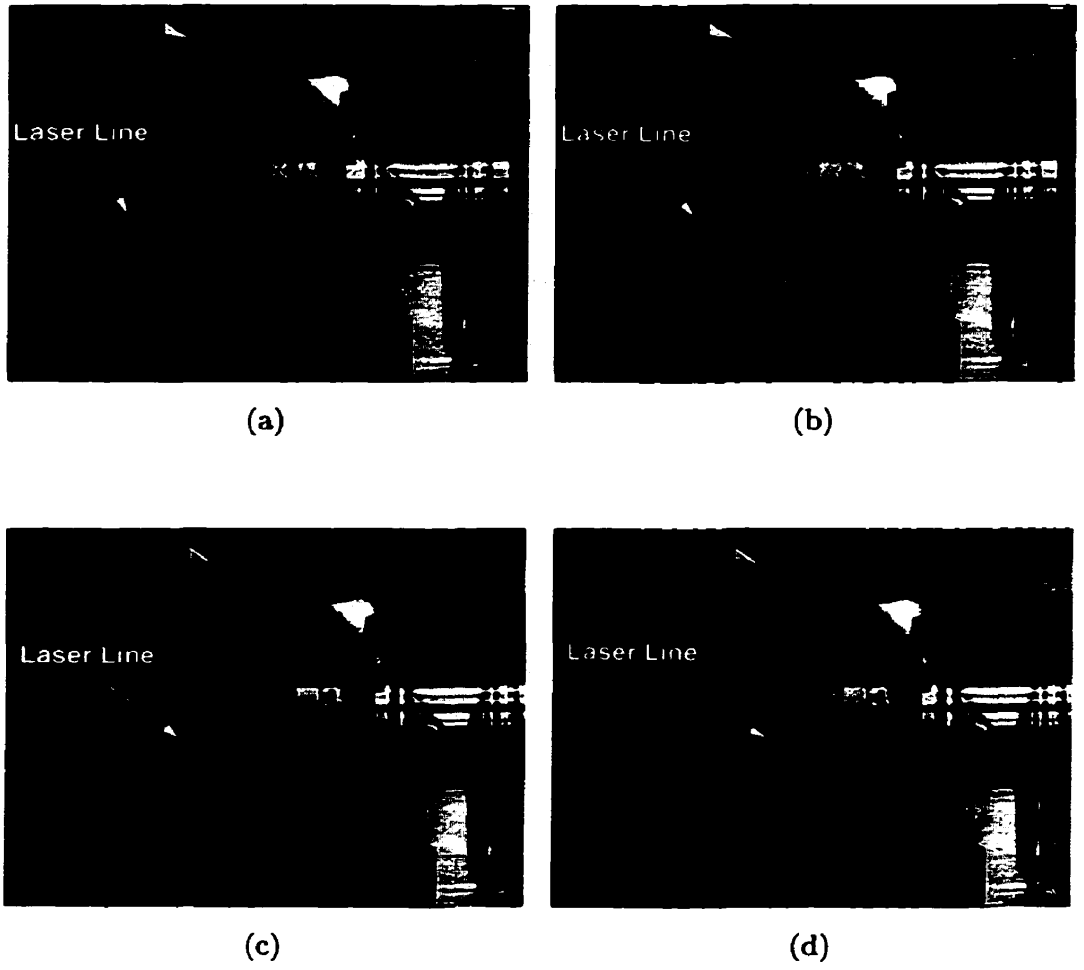


FIGURE 3.15. **A Doorway Scanning Sequence.** (a) ORB detects the doorway opening and notes the position of the beginning of the frame. (b)&(c) ORB pans the sensor across at 5° increments searching for the other end of the doorway frame. ORB also uses these scans to categorize the state of the doorway (i.e., open, closed, or partially open). (d) The other boundary of the doorway frame is detected. The width is computed and compared to the range of widths for doorways on this floor.

In the CIM office space, a closed office door is 10 cm from the wall surface inside the doorframe. The horizontal range profile of a closed door will show a sharp 10 cm step increase in range from *Wall_Distance*. This step remains constant for the duration of the doorway width. This is followed by a return to range data at *Wall_Distance*, indicating that the wall on the other side of the door is in view. Therefore, an opening is classified as a *doorway* if the opening's width falls within the range of doorway widths found in this working area. Then, the doorway is categorized as a *closed door* if the range profile of the opening is as described above. Experimental results are shown in Chapter 6. The doorway model is shown in Figure 3.18.

Scanning an Opening

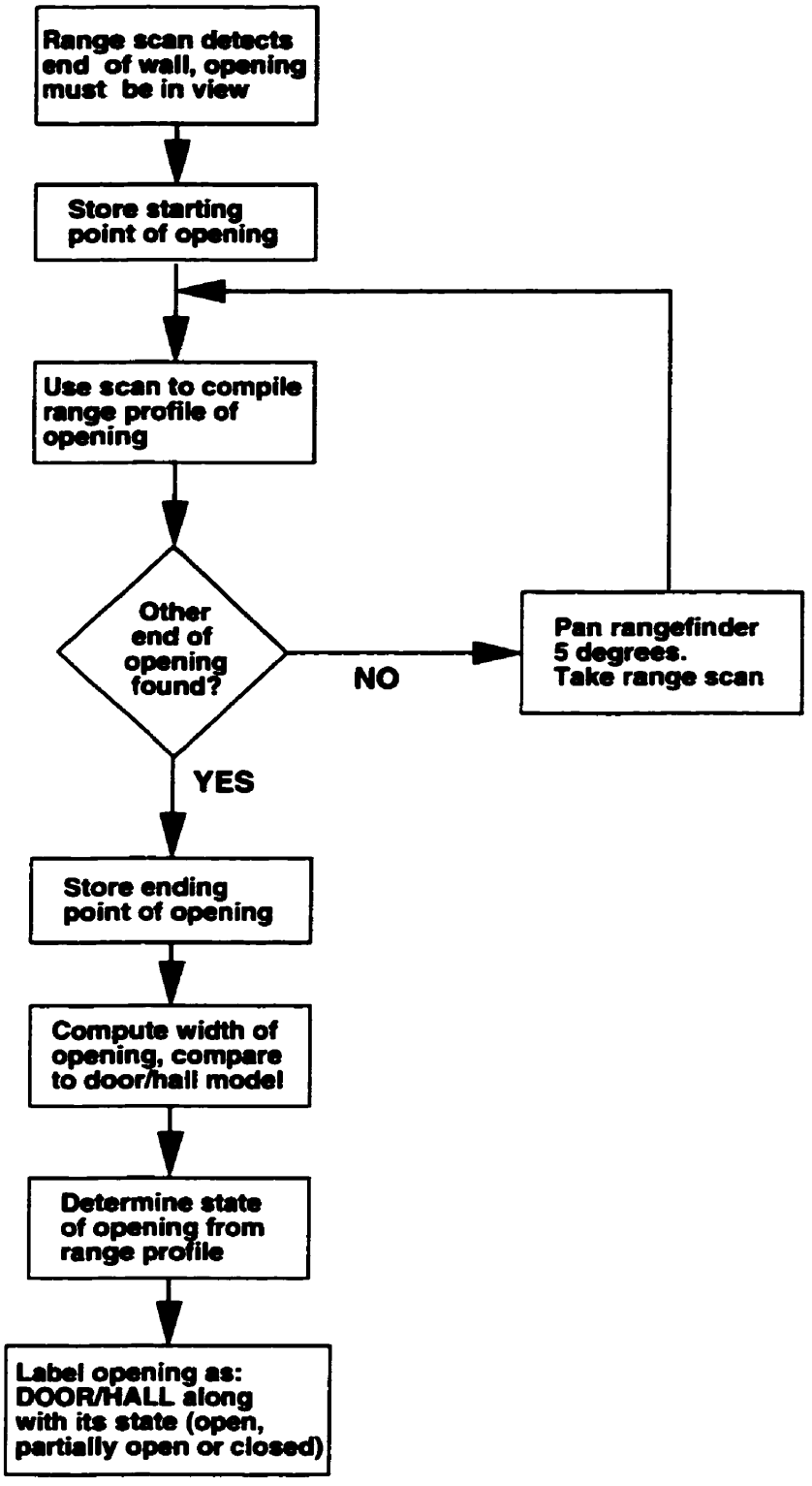


FIGURE 3.16. **Outline of How ORB Scans an Opening.** This is a flow chart of how ORB scans an opening, an example of which was shown in Figure 3.15.

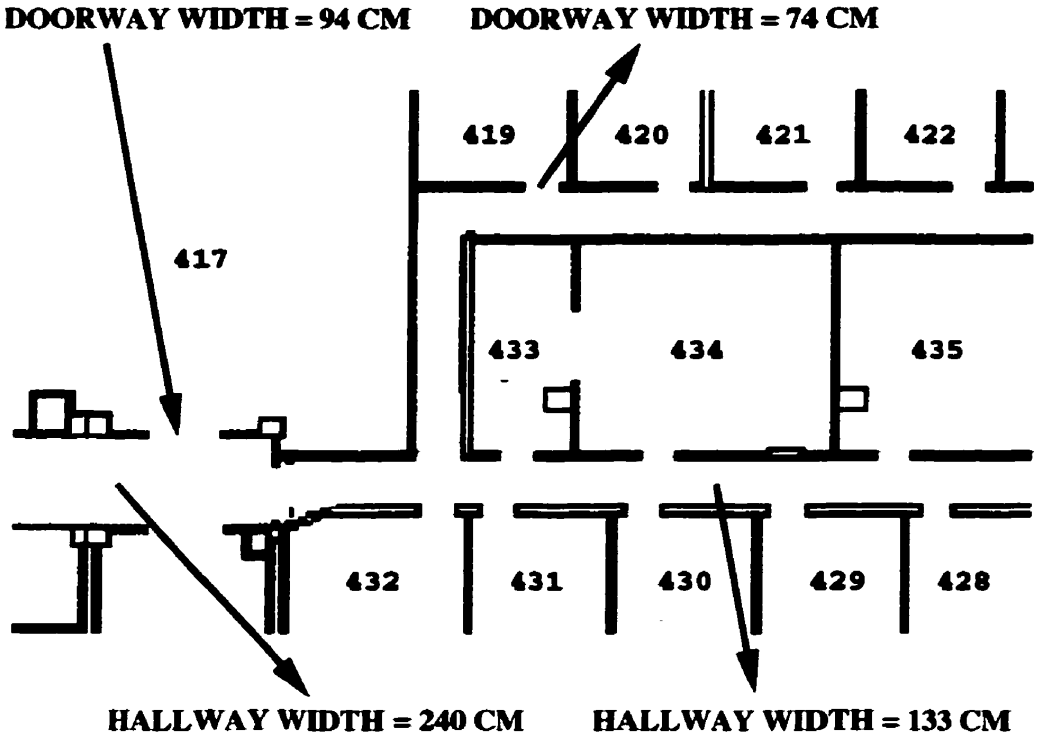


FIGURE 3.17. **The Doorway and Hallway Widths on the CIM Floor.** Some sample door and hallway widths on the CIM floor. Most of the experimentation was done in the 133 cm width hallway where most of the office rooms are.

2.3.4. *Partially Open Doors*

A door that is open but obstructs the path, or view, through the doorway is called a *partially open door*. The boundaries of a partially open doorway are marked by the same features sought from a closed door as described above. The defining signature of this doorway state is its horizontal range profile. A *partially open door* is found further from the wall than a closed door[16], yet within the area that is covered by a door as it opens into a room. A door swings on its hinges in an arc that covers an area with radius equal to the width of the door, as seen in Figure 3.18. Therefore, the furthest point on a partially open door, as seen from the hallway, is found along the edge of this arc. A *partially open door* would then be located at a distance beyond that of a closed door, but not further than the area delineated by arc it traces as it opens. In the CIM office space, that would mean a *partially open door* is found beyond 10 cm from the wall, yet not further than 94 cm. Anything beyond 94 cm, or the width of the widest door on the floor, cannot actually be a part of the door.

[16] A closed door is located a set distance from the wall surface (10 cm in the CIM environment).

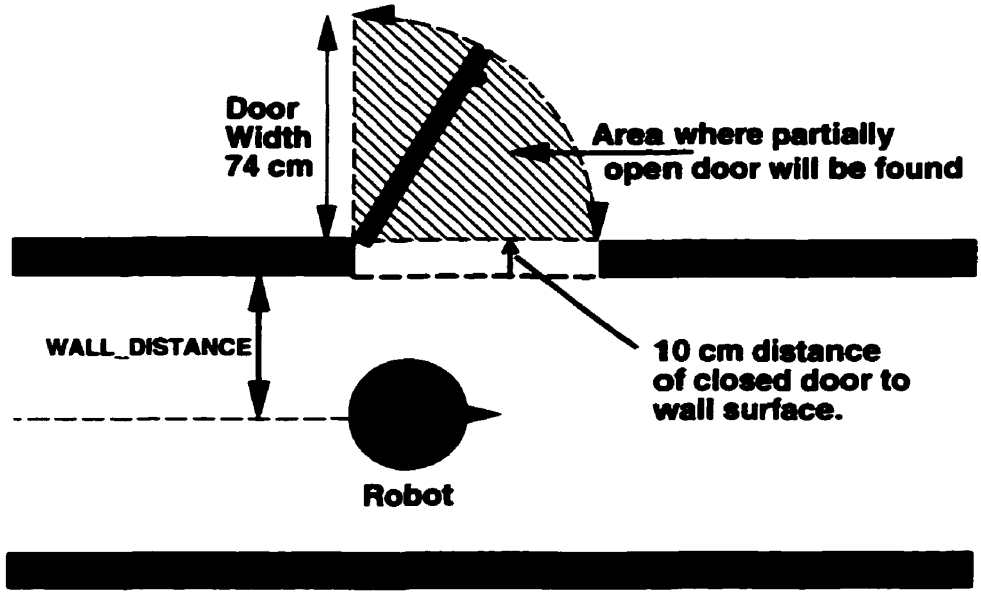


FIGURE 3.18. **A Doorway Opening.** The various components of a typical doorway and how these features are used in ORB's door model. The *doorway width* is used to differentiate doorway openings from all other openings (i.e., hallway openings). The range of doorway widths found on the office floor is recorded a priori for the door model. A *closed door* will lie at some distance inside the doorframe from the surrounding wall. In this case, for a CIM office door, it is 10 cm from the wall as seen from the hallway. An opening door will trace an arc with a radius the length of the door's width. Therefore, a *partially open door* will be found in the outlined area of the semi-circle as shown. An *open door* will leave a completely unobstructed doorway, and any range data obtained in scanning an open doorway will be beyond the areas defined for a closed doorway, or partially open doorway.

2.3.5. *Open Doors*

A completely open door leaves the doorway opening unobstructed, resulting in a clear view of the room from the corridor. The range scans acquired through an *open doorway* will be readings of objects inside the room. The range profile of an *open door* begins with the initial detection of the doorway, whereby the wall ends and a sharp increase in range data is detected. The range data received through the open doorway is fairly noisy since these are range readings of objects somewhere inside the room. The opposite end of the doorway frame is found when the wall on the other side comes into view. This is indicated by a return to range data at *Wall_Distance* from the robot. Therefore, an *open door* is detected if the range readings through the doorway are beyond that of both a closed door and a partially open door. Experimental results showing ORB's scanning and recognizing each of these doorway states will be presented in Chapter 6.

While navigating through the hallways of an office space, the mobile robot may also be required to enter rooms to perform various tasks. One of these tasks is to find and recognize

the *movable objects*. These are basic furniture items found in all office areas, the recognition of which is discussed in the next chapter.

CHAPTER 4

Recognition of the Movable Objects

The "*FIND*" commands, as described in the first chapter, are mobile robot navigational tasks carried out by SPOTT which require the robot to seek out and locate specified objects. The targeted objects include *movable objects* and *parametric geons*[47][1]. The *movable objects* are certain pieces of furniture found in all office spaces, namely: chairs, tables and desks. When searching for a *movable object*, SPOTT calls upon ORB to scan and recognize the object once it has been located in the scene[2]. A sample scenario would be the following: SPOTT is given a "*Find Chair*" task to complete as the robot navigates through an office space. Once an object suspected of being a chair is located, SPOTT calls on ORB to execute its "*Scan Chair*" routine, leaving ORB to determine if the object in question is indeed a chair. The models of the *movable objects* and ORB's recognition procedures are now discussed.

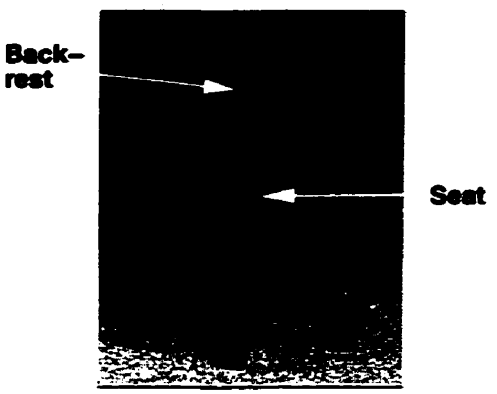
1. Object Models

This work is concerned with an office environment where basic furniture items like chairs, tables and desks are found in every room. These *movable objects* are shown to be composed of planar surfaces. A standard chair has two planar surfaces, the seat and the backrest, as shown in Figure 4.1(a). A table has a single planar surface, the tabletop, as seen in Figure 4.1(b). Desks consist of a planar surface for the desktop with three additional supporting planes underneath, as seen in Figure 4.1(c).

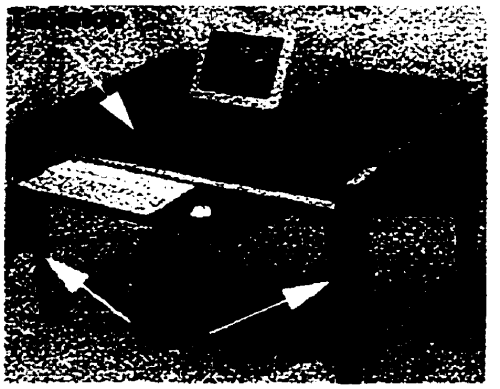
ORB's models for the *movable objects* are idealized descriptions with the object's surfaces represented by planes. These objects are described by the *general object model* shown in Figure 4.2. The *movable objects* in the database are modelled by the rectangular faces

[1]The search for and recognition of parametric geons is not part of this work, but is an ongoing research project at CIM.

[2]ORB assumes that the "attention problem" has been solved and that the object is localized in the scene before the recognition procedure begins.



(a)



(b)

(c)

FIGURE 4.1. **The Movable Objects.** (a) A chair is composed of two planar surfaces: the backrest and the seat. (b) A table consists of a single horizontal plane: the tabletop. The tabletop is supported by four legs, which are not modelled because they are not easily detected. (c) A desk has a single horizontal planar surface for the desktop. The desktop is supported underneath by three vertical planar surfaces.

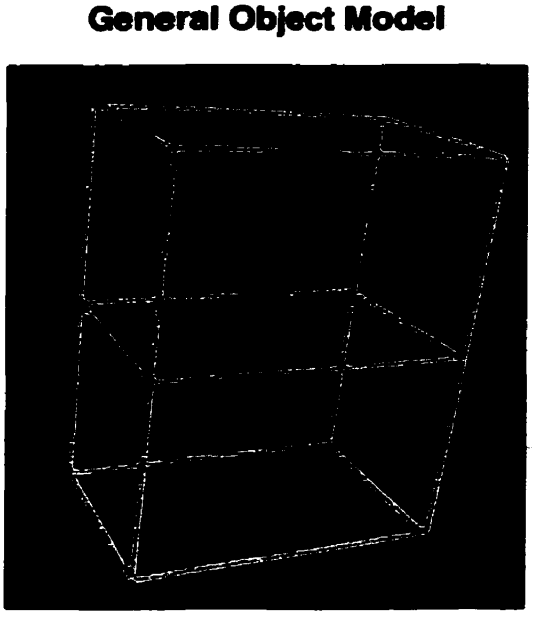


FIGURE 4.2. **General Object Model.** This is the *general object model* used to describe the objects in ORB's movable object database. The planar surfaces of each object are mirrored by rectangular faces in this structure.

in this structure. The physical dimensions of these models are defined by *Architectural Standards*[1], as most office furniture is built to conform to these standards. The objects currently in ORB's *movable object* database are chairs, tables and desks. The models for each are now discussed.

1.1. Chair Model

A standard chair is composed of two major parts: the backrest and the seat. These are the two planar surfaces modelled using the *general object model.* The other parts of a chair, such as the legs and armrests, are not used because they are too narrow to be easily detected by BIRIS range scans. ORB's *general object model* only accounts for planar surfaces.

The chair model is shown in Figure 4.3. The standards outlined by *Architectural Standards*[1] do not vary much for the different types of chairs used in these experiments[3]. The seat for these types of chairs is found at heights of 43 cm to 47 cm off the ground. The backrest is positioned directly above and behind the seat when viewed frontally. The top of the backrest is found between 75 cm to 78 cm off the floor.

[3]Chairs found in the CIM office environment were used for these experiments. Among these are the "secretarial chair", "Ergon secretarial chair", and "Brno chair", whose dimensions are outlined in *Architectural Standards*[1].

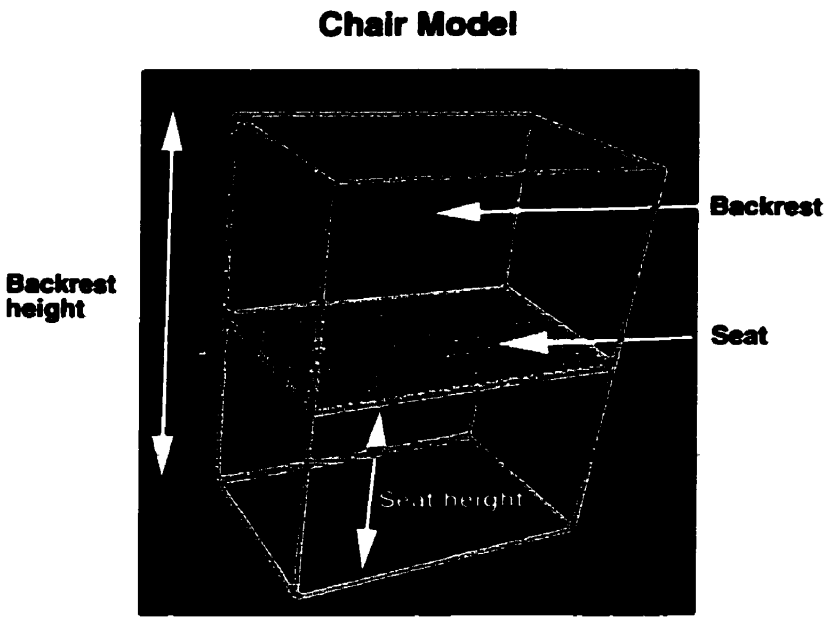


FIGURE 4.3. **Chair Model.** The chair model consists of two planar surfaces that represent the seat and backrest. Office chairs are built to standards that include the seat height and the backrest height. These values are defined in *Architectural Standards*[1] and provide the dimensions of this model.

1.2. Table and Desk Models

Tables and desks are very similar in design. The main difference between the two is that tabletops are usually supported by four legs, whereas desktops have planar supports. A desk's supporting planes house drawers which tables usually do not have. Desks have more of a box-like configuration, whereas tables have only one main component, the tabletop.

1.2.1. *Table Model*

The table model is shown in Figure 4.4. The legs of the table are not part of the model since they are not easily detected by the BIRIS rangefinder. The tabletop is the only planar feature used by ORB. There are many different categories of tables that cover a wide range of heights. Typical "low" tables range in height from 30 cm to 45 cm and are generally for home use. Typical "end" or "side" tables range in height from 45 cm to 72. The majority of the tables found in the CIM office space are called "conference" or "dining" tables which range in height from 67 cm to 72 cm[1]. These are the types of tables most likely to be found in office spaces and form the basis of ORB's table model.

1.2.2. *Desk Model*

The desk model is shown in Figure 4.5. The desk model consists of a desktop with vertical planar supports on three sides of the desk. The side supports would be visible only

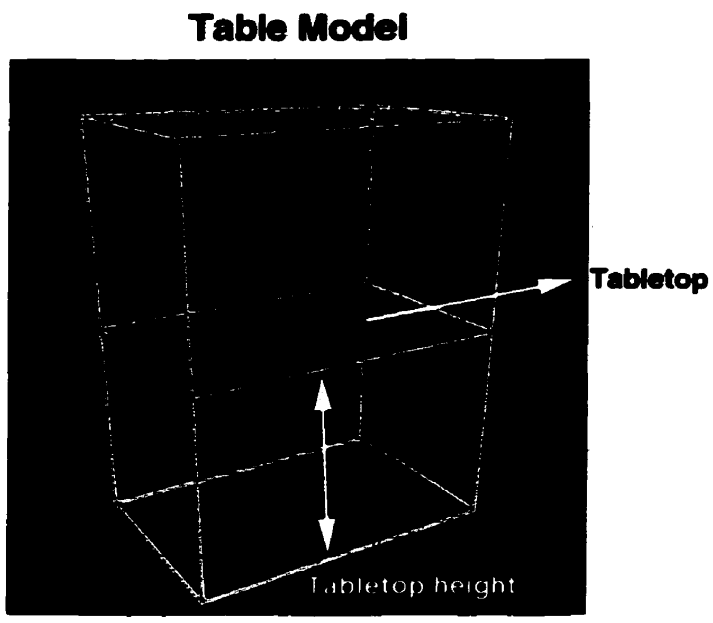


FIGURE 4.4. **Table Model.** The table model consists of a single horizontal planar surface that represents the tabletop. This surface is found at the standard height for "conference" tables, which is between 67 cm and 72 cm[1].

when the desk is approached from the side. Since desks are ordinarily positioned against a wall, the third support is generally inaccessible to the rangefinder. The fourth side is the open space for sitting.

The two basic types of desks found in office environments are "single pedestal" and "double pedestal" desks. The names refer to whether the desk has one or two columns of drawers. The standard height of these desktops range from 65 cm to 72 cm[1]. Note that this is basically the same height range as for tabletops.

2. Scanning the Movable Objects

It has been shown that the *movable objects* may be represented in terms of planar surfaces using the *general object model*. The recognition of a *movable object* requires the detection of these planar features. The following sequence of events occur when recognizing a *movable object*. They are outlined in the flow graph in Figure 4.6.

(i) ORB is executed by SPOTT or by a user via the command-line. The task performed is one of *Scan Chair*, *Scan Table* or *Scan Desk*.

(ii) ORB assumes that the object has already been located in the scene and is in front of the chosen BIRIS rangefinder at the time it begins scanning[4].

[4]ORB must be told which of the two BIRIS rangefinders has the object in view.

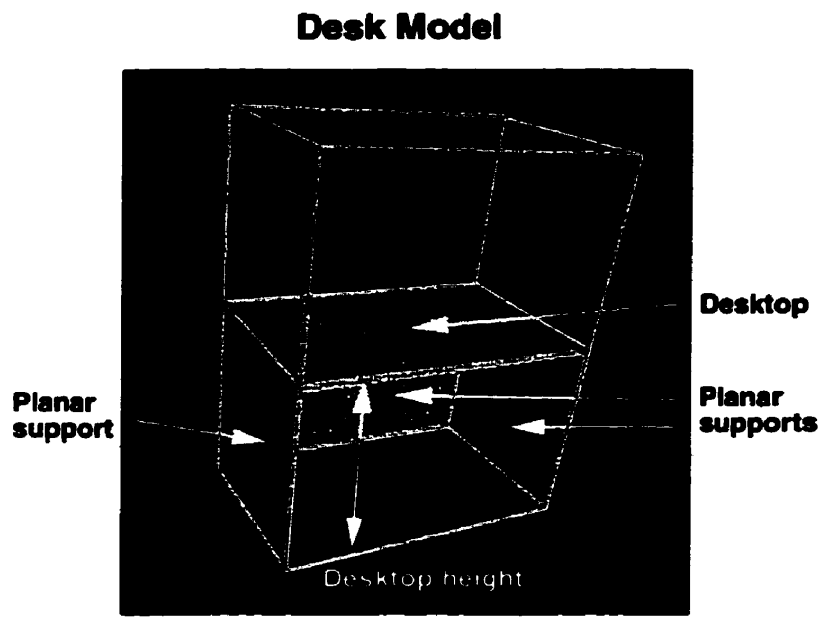


FIGURE 4.5. **Desk Model.** The desk model consists of a planar surface that represents the desktop, as well as three vertical planar supports underneath. The desktop surface is found between 65 cm and 72 cm in height from the ground[1].

(iii) ORB then scans the object to search for its planar features. ORB's focus of attention (i.e., where to position the rangefinder) is determined by the object's model. A scanning routine is employed whereby ORB systematically searches for each planar surface on the object. Knowing the dimensions of the object and where it is positioned with respect to the rangefinder, ORB determines where each surface is located in the scene. Once located, ORB scans over the surface[5], collecting range readings which should combine to form a plane (within an allowable threshold).

(iv) Recognition is complete once ORB has successfully detected the planar features of the object, as described by its model. The range data collected are stored for display and archival purposes. ORB labels the object as a "chair", "table" or "desk" and sends this conclusion to SPOTT and/or the user. A failure to properly recognize the object will result in a message stating that the object is "unknown".

2.1. Assumptions

ORB makes certain assumptions before scanning and recognizing a *movable object*. The general assumptions are outlined below. More specific assumptions for scanning individual *movable objects* will be discussed in later sections pertaining to those objects.

[5]This process entails sweeping the BIRIS laser line over the area.

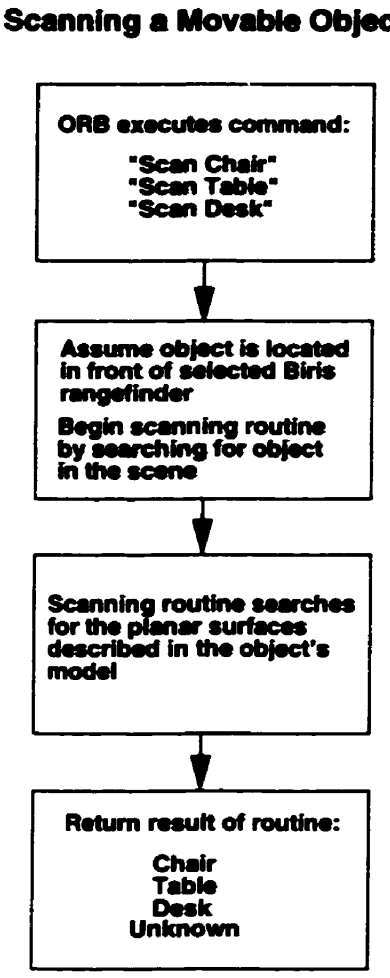


FIGURE 4.6. **Flow Graph of the Movable Object Scanning Procedure.** ORB scans a *movable object* by locating it in view and then searching for their planar surfaces. Each *movable object* requires a specific scanning routine, as they each have unique structures. The scanning routine focuses on the planar surfaces found in the object's model.

(i) ORB assumes that the object has been localized in the scene when ORB receives the *Scan Object* task command. This means that upon receiving this directive, the object must be situated in front of the selected BIRIS sensor. The object does not necessarily have to be in view of the rangefinder, but it must be positioned such that the rangefinder would only have to TILT downwards to find it[6].

(ii) The second assumption is that the approximate distance of the object to the rangefinder is 150 cm[7]. The robot should be positioned as close to this distance

[6]At their home positions, the rangefinders point out horizontally at 120 cm above the floor. Chairs are about 75 cm in height, tables and desks from 65 cm to 72 cm in height, so the rangefinder would have to point down to see these objects.

[7]This distance is ideal since it is within the optimal operating range of the BIRIS sensor.

as possible. If the actual distance is known from sonar or other data, this value is sent to ORB (by SPOTT or by the user via the GUI).

3. Chair Recognition

To model a chair, ORB uses a *model based* description based on the planar structure shown in Figure 4.3. ORB also uses a *functional*[63] description that permits any object that may *function* as a chair[8] to be labelled a chair. As a result, ORB would allow for instances when the chair is oriented sideways to the rangefinder with only the seat surface visible. Even though the *model based* description is not completely matched since only the seat is detected, ORB would still consider the object to be a chair. This is because the object may *function* as a chair since it has a horizontal plane that can be used as a seat.

The chair scanning procedure is outlined in Figure 4.7. ORB attempts to locate the backrest of the chair by aiming the rangefinder towards the upper half of the backrest. After scanning over this surface, ORB turns its attention to the seat. Once the seat is scanned, the procedure is complete and conclusions are made based on the results.

3.1. Occlusion

Occlusion is a problem when the chair is not oriented properly in front of the rangefinder. For instance, if the chair is positioned sideways to the rangefinder, then the backrest's surface is perpendicular to the sensor and cannot be detected (Figure 4.8(a)). Occlusion of the seat occurs when the chair faces away from the rangefinder and the backrest obstructs the view (Figure 4.8(b)). The seat may also be obstructed by armrests. For these reasons, the ideal position of a chair with respect to the rangefinder is a *frontal* position, in which both the seat and backrest are visible (Figure 4.8(c)).

The assumption is that the chair is facing the rangefinder *frontally*, meaning that the seat, and preferably the backrest, are in view. Currently, ORB does not recognize chairs from *behind* if the chair seat is not visible. The key to recognizing chairs is locating the seat at the pre-defined standard height. An ideal view of a chair as seen by the rangefinder was shown in Figure 4.8(c).

3.2. Scanning the Backrest

ORB begins scanning a chair by locating the backrest. An outline of the backrest scanning procedure is shown in Figure 4.9, the details of which are discussed in this section.

[8]This object would have a horizontal planar surface at the same height from the ground as a standard chair seat.

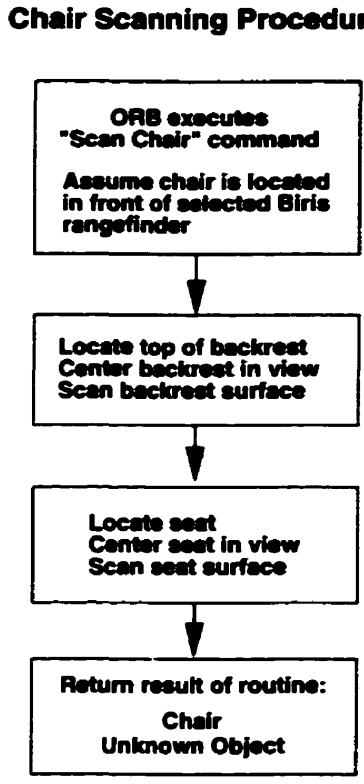


FIGURE 4.7. **Outline of Chair Scanning Routine.** The process of scanning a chair entails the actions outlined above. The chair is assumed to be located in front of the selected BIRIS rangefinder at the time the command is received. The top of the backrest is then located in the scene and the backrest surface is scanned. The next step is to locate the seat and scan its horizontal planar surface. Conclusions are made based on the results of this scanning procedure.

As previously mentioned, ORB assumes that when the command is given to scan a chair, the chair is situated in front of the rangefinder. In its home position, the rangefinder would only need to tilt down to have the chair in view. The average height of chairs found in the CIM office space is 75 cm[9]. Knowing the height of the chair and its distance to the rangefinder, simple geometry is used to determine the TILT angle required to position the rangefinder towards the top of the backrest. This TILT angle is shown in Figure 4.10.

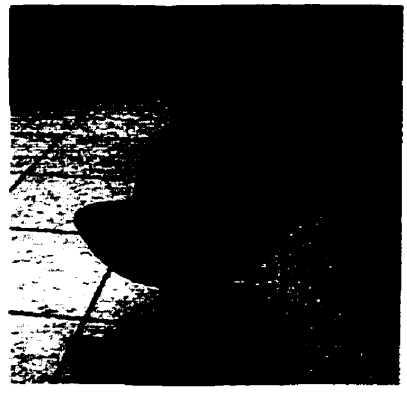
3.2.1. *Centering the Backrest*

Once the rangefinder is properly positioned, the backrest should be *centered* in the rangefinder's field of view. Since the backrest is modelled as a planar surface, scans of a *centered* backrest should return single line segments[10]. Obtaining these single line segments is very important in determining that the backrest's planar surface is being scanned.

[9]This height is from the top of the chair to the floor.

[10]These line segments are obtained after segmentation of the range data using Ramer's method, as described in Chapter 3.

(a)



(b)

(c)

FIGURE 4.8. **Chair Orientations.** (a) When a chair us approached from the side, the surface of the backrest is not visible. The seat can be detected however, which is enough to satisfy ORB's *functional* description of a chair. (b) When a chair is viewed from behind, the seat is obscured by the backrest. The robot must move to another position to allow ORB to properly scan and recognize this chair. (c) This is the ideal view when scanning a chair, with both the seat and backrest in view.

Scanning a Chair Backrest

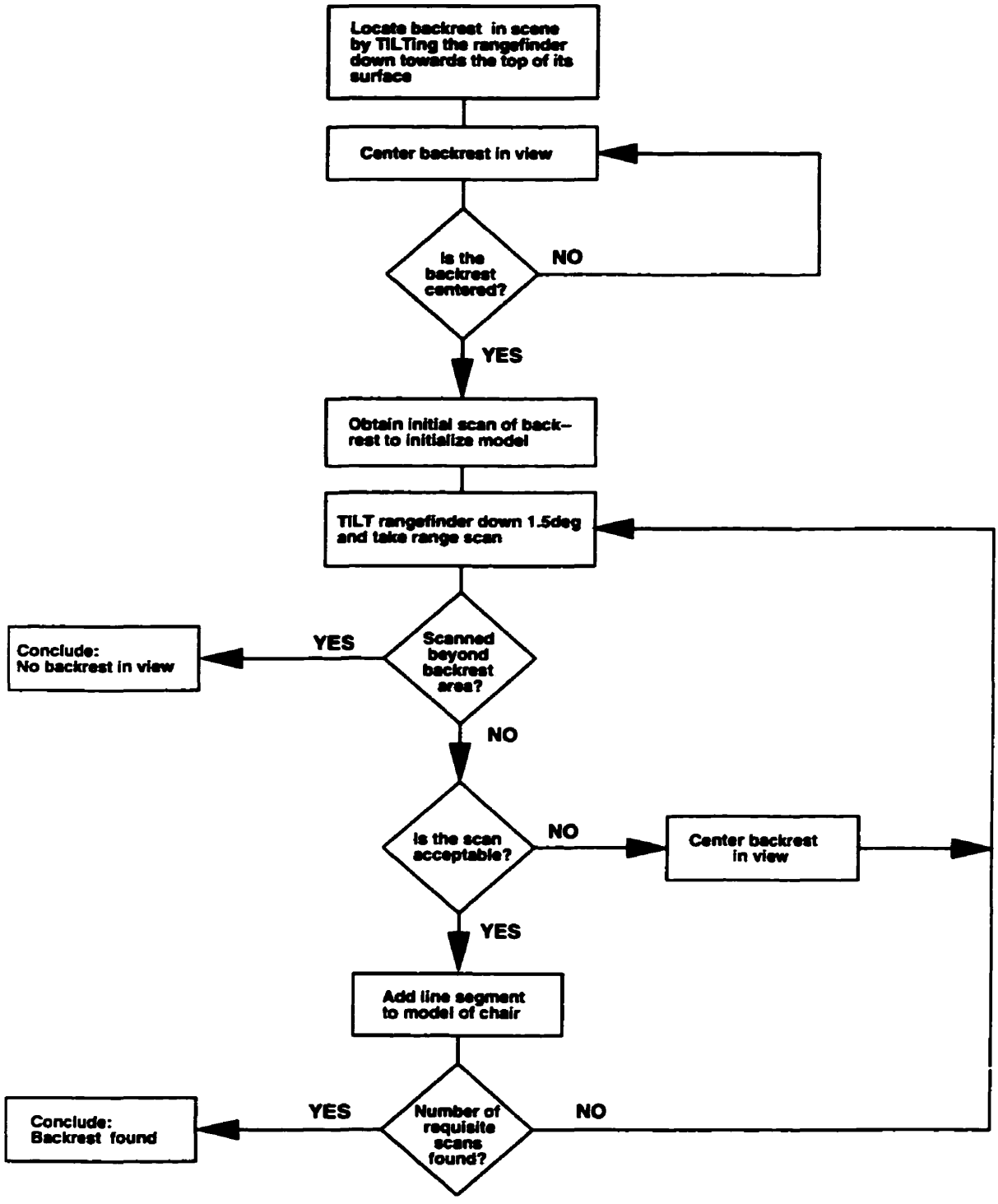


FIGURE 4.9. **Flow Graph of Backrest Scanning Procedure.** These are the steps taken by ORB to scan a chair backrest. The details of each step are discussed in the following sections.

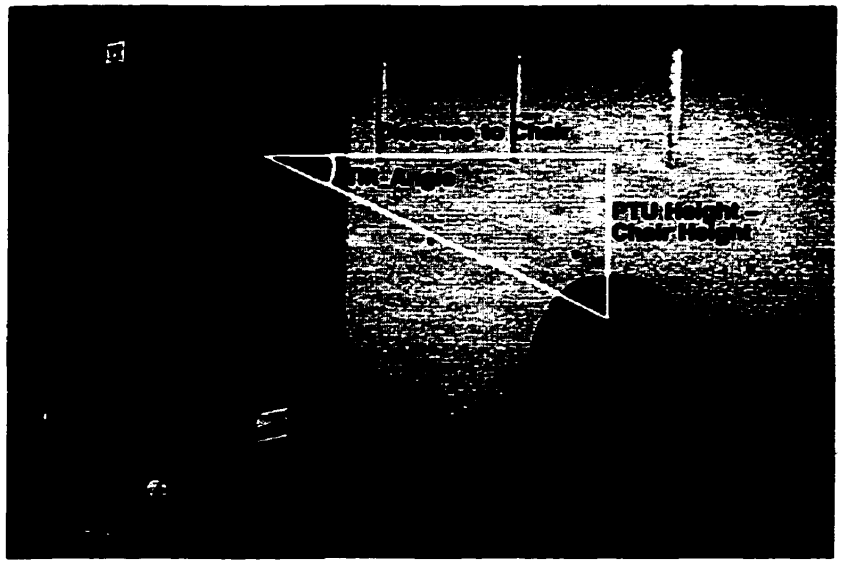


FIGURE 4.10. **Locating the Backrest.** With the chair positioned in front of the rangefinder, the angle needed to tilt the rangefinder down towards the top of the backrest is calculated as follows:

$$Tilt\_Angle = arctan((PTU\_Height - Chair\_Height)/Distance\_to\_Chair)$$

The *centering* procedure consists of a series of PAN movements that focus the rangefinder towards the closest object in view. The rangefinder is panned horizontally at 2^o increments in the direction of the object until it is centered in the field of view. This procedure is used to center any planar surface being scanned, horizontal (e.g., chair seat) or vertical (e.g., chair backrest). This process is outlined in Figure 4.11.

The backrest is centered once an *acceptable*[11] scan is found. This would mean that the chair is the only object in view and the rangefinder is focused solely on the backrest. The range scans taken of a backrest during this centering process are shown in Figure 4.12.

This centering process determines if the backrest is visible. For instance, if the chair is oriented such that the backrest's surface is obscured from view (i.e., sideways to the rangefinder) or if there is no backrest in the scene at all, this centering procedure would fail[12].

If the centering procedure fails, ORB requests that the robot be moved to a location with a better view of the chair. Once this is done, ORB locates the backrest as before and again attempts to center it in view. As previously mentioned, ORB assumes that the *attention problem* has been solved and that a separate module is responsible for locating

[11] An acceptable scan is one which returns a single line segment after segmentation.

[12] This is because there is no predominant planar surface in view which can be focused upon.

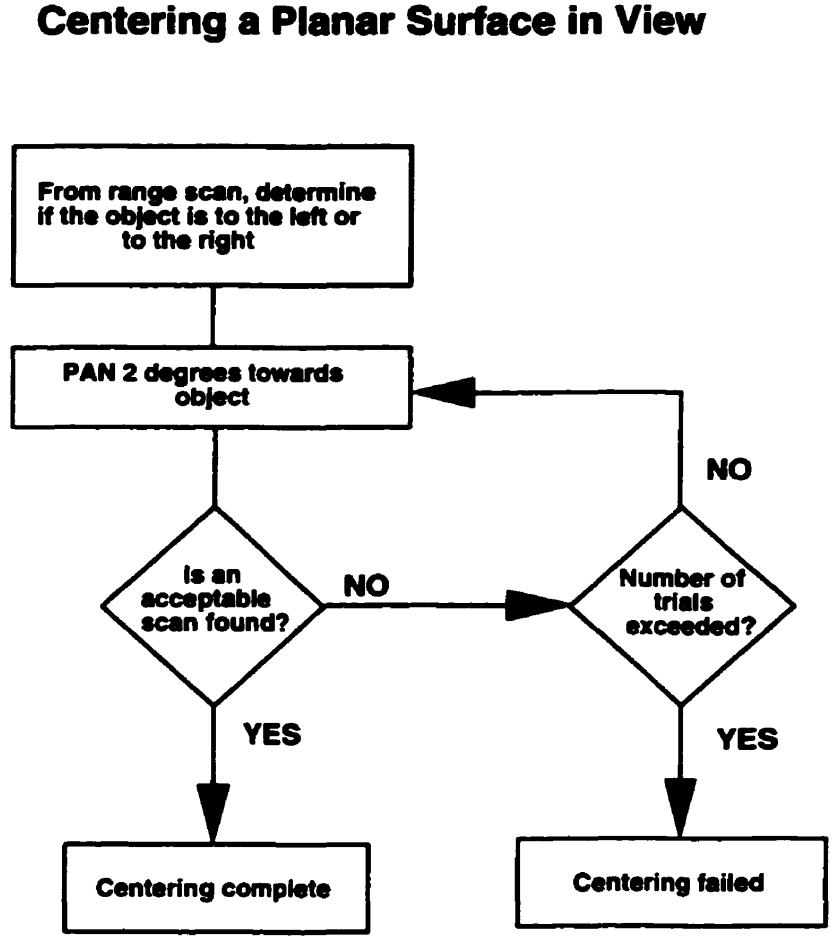


FIGURE 4.11. **Outline of Centering Process.** The first step of the centering process is to determine whether the object is to the left or the right of the rangefinder's view. It is assumed that the object being scanned is the pre-dominant object in view and would provide the closest data points. If not centered, these points would be offset towards one end of the BIRIS image. The rangefinder makes a series of PAN movements at 2° increments towards these set of points until the object is centered in view. The object is centered once an acceptable scan, in other words a scan returning a single line segment of range data, is found. There are a set number of trials (default is 20) allowed before this process is abandoned.

the targeted objects. This version of ORB relies on SPOTT, or the user, to control the positioning of the robot.

3.2.2. *Producing the Vertical Planar Surface*

Once the backrest is centered, ORB uses the first *acceptable* scan of the backrest to initialize its knowledge of the chair. This *initial* scan provides the distance from this part of the backrest to the rangefinder. Subsequent scans of the remaining backrest surface should yield single line segments at the same distance as this *initial* scan. This set of scans form

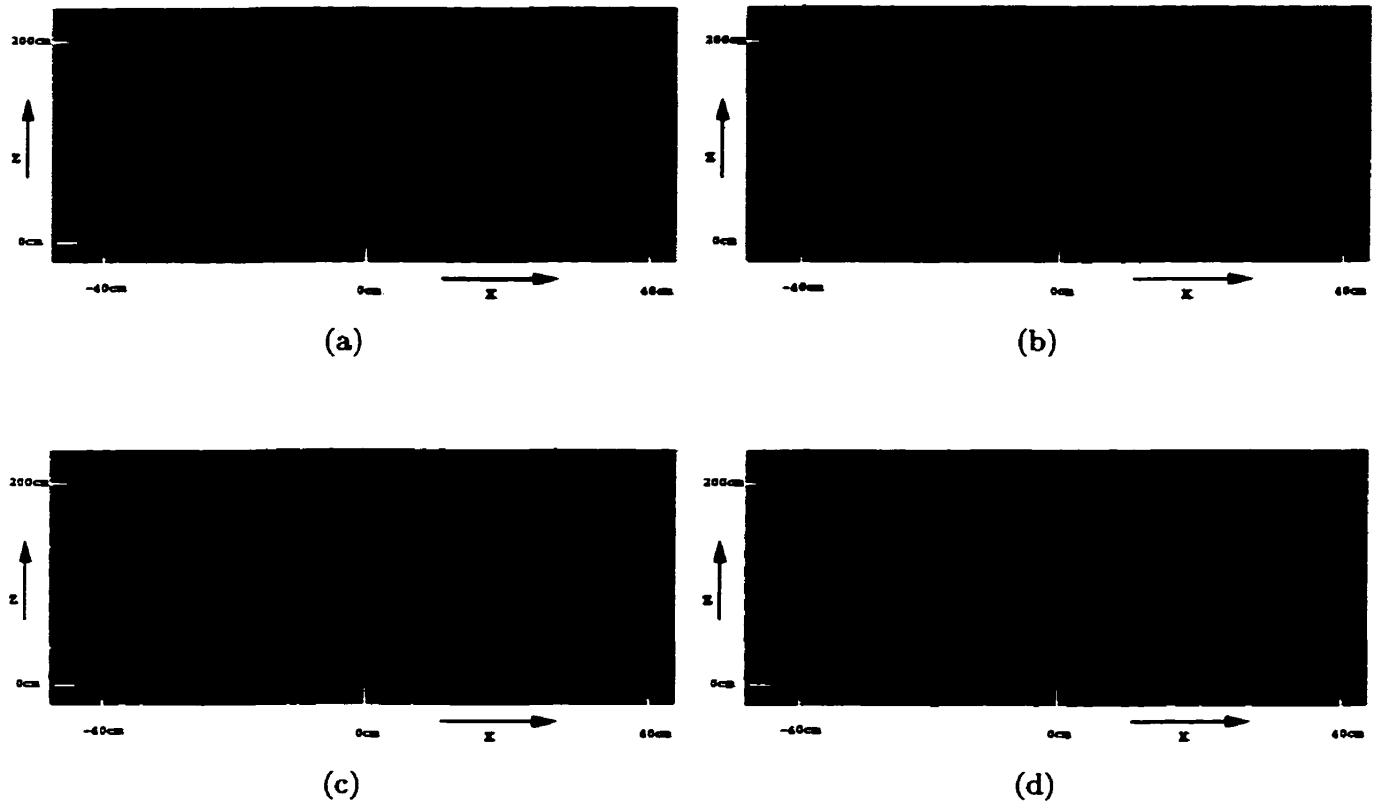


FIGURE 4.12. **Centering the Chair Backrest in View.** These are range scans taken during the centering process of a backrest. (a) Initially, the backrest is found to the right of the rangefinder's view. Since the backrest is not centered, background objects seen to the left cause multiple line segments to be returned in this scan, which are not shown in the plot because their range points are beyond 200 cm. (b) & (c) The rangefinder is moved horizontally with PAN movements at 2° increments towards the backrest surface. (d) The backrest is now centered as a single line segment of range data is returned. The backrest is now the sole object in view.

the vertical planar surface of the chair model, which represents the backrest. A sample set of backrest scans is shown in Figure 4.13.

A *threshold* is allowed when comparing the *initial* scan to each succeeding scan[13]. This value may vary due to experimental conditions and is up to the discretion of the user, but a range of 5 cm to 7 cm is considered appropriate for most cases.

Once the *initial* scan has been noted, ORB scans down the backrest surface at 1.5° increments. At each position, ORB segments the acquired range data to ascertain if an *acceptable* scan has been obtained[14]. If so, the distance of this scan to the rangefinder is compared to the *initial* scan. If this latest scan is located at the same distance as the *initial*

[13]This threshold value accounts for uncertainty in the range readings. The accuracy of BIRIS range data is dependent on the reflectivity of the object's surface, as well as the lighting conditions in the scene.

[14]In other words, ORB determines if this scan produces a single line segment using Ramer's method, as described in the previous chapter.

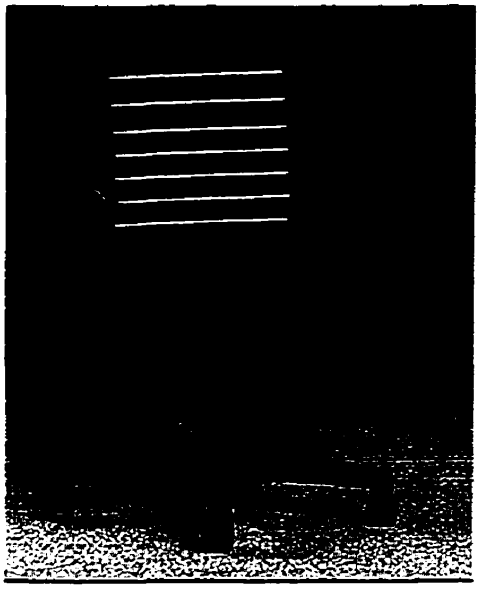


FIGURE 4.13. **Backrest Scan Lines.** A sample set of BIRIS range scans of a chair backrest.

scan, within the allowable threshold, then it is considered to be part of the vertical plane representing the backrest. If an *acceptable* scan was not found, ORB attempts to re-center the backrest before continuing. ORB searches for a series of *acceptable* scans that fit the backrest model[15] before concluding that the backrest has been found.

From *Architectural Standards*[1], the backrest surface area ranges from 30 cm to 35 cm in height. Assuming that the chair is 150 cm from the rangefinder , 1.5^o increments results in scans that are spaced 4 cm from each other. ORB searches for a number of acceptable scans[16] to model a backrest, so that 16 cm to 24 cm in height of this surface is covered. Once this set of scans is obtained, the backrest is noted as being found.

3.2.3. *Completion*

ORB scans down the backrest surface until one of the following scenarios occurs: (i) the required number of *acceptable* scans is obtained, or (ii) the rangefinder has scanned beyond the backrest area without obtaining this requisite number of *acceptable* scans.

ORB scans down no further than 35 cm from the top of the backrest. This is the maximum height of the backrest's surface area[1], so scans below this point would fall on the seat. If the rangefinder has scanned down to this level without detecting a backrest,

[15]This means that this series of scans each produced single line segments that form a vertical plane with the *initial* scan. The number of scans needed is up to the discretion of the user, though six or seven were found to be sufficient for the experiments in this thesis.

[16]This number of scans is chosen by the user, though 5 to 7 scans was found to provide a suitable representation of a backrest, and is the range used in these experiments.

ORB concludes that a backrest is not in view. This could happen for a number of reasons. (i) There is no chair in view because the robot was not positioned properly. (ii) The chair is in view of the rangefinder but is oriented such that the backrest's surface is not visible (i.e., sideways to the rangefinder). (iii) Erroneous BIRIS range data due to factors such as the lighting conditions in the environment, or the material construction of the chair[17] would adversely affect ORB's perception of the scene.

Once this process is complete, ORB notes the presence or absence of the backrest and then continues with a search for the seat.

3.3. Scanning the Seat

The seat is detected as a horizontal planar surface found at the standard height for office chairs. Finding the backrest makes the model complete, but is not absolutely essential due to the possible orientation problems described before. If a horizontal planar surface is found where the seat is supposed to be located, that is sufficient evidence to conclude that this object can at least *function* as a chair. An outline of the seat scanning procedure is shown in Figure 4.14.

The height of chair seats found in the CIM office space range from 42 cm to 47 cm[**1**]. From scans of the backrest, ORB knows the exact distance of the chair to the rangefinder. This allows ORB to accurately locate the seat, using the same principle as that used to search for the backrest. ORB positions the rangefinder towards the back of the seat, close to the backrest, before scanning down over the surface of the seat. This scene is shown in Figure 4.15.

3.3.1. *Centering the Seat*

Once the rangefinder has been properly positioned, the seat should be centered in the rangefinder's view. If not, the seat is centered using the same process used to center the backrest. This procedure is depicted in Figure 4.16.

3.4. Producing a Horizontal Planar Surface

After centering the seat in view, ORB uses the first *acceptable* scan of the seat to update its knowledge of the chair. This *initial* scan determines the actual height of the seat from the floor. It should fall within the pre-defined range for office chairs (42 cm to 47 cm[**1**]). The remaining scans of the seat should produce single line segments[18] at the same height

[17] Dark surfaces do not reflect the laser line well, which could result in erroneous range data. Examples of these situations will be presented in the next chapter.

[18] These line segments are produced by segmenting the BIRIS range data.

Scanning a Chair Seat

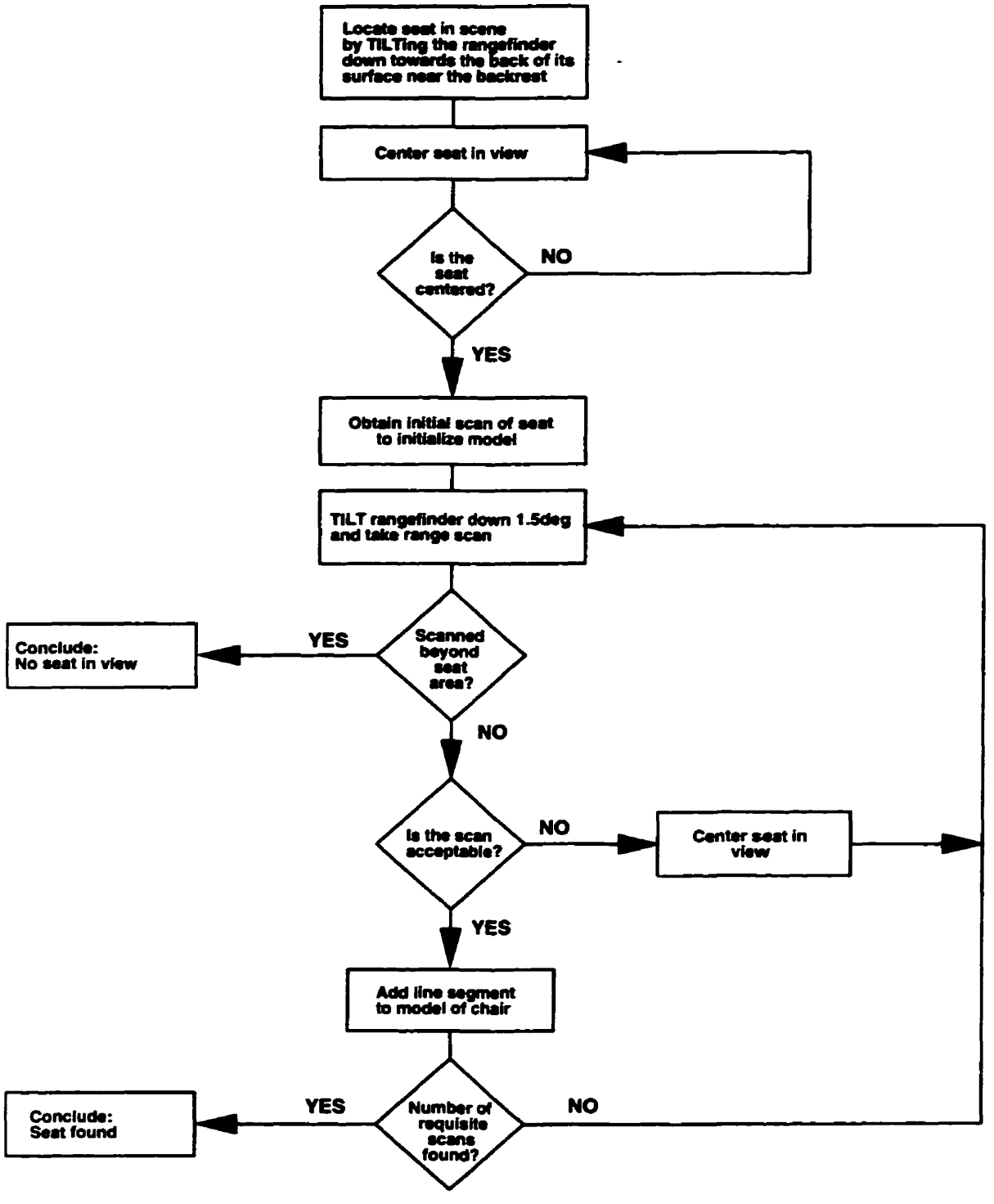


FIGURE 4.14. **Outline of Seat Scanning Routine.** This is an outline of the steps taken by ORB to scan a chair seat. This is basically the same routine used to scan a backrest, only the focus of attention is different.

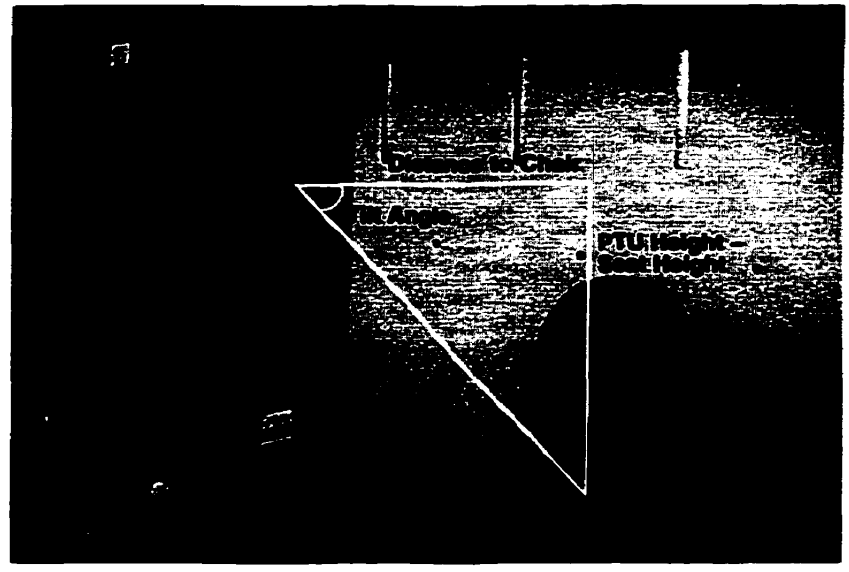


FIGURE 4.15. **Locating the Seat. ORB applies simple geometry to locate the seat, using its height and the chair's distance to the rangefinder. The rangefinder is positioned towards the seat with a tilt angle computed as follows:**

$$Tilt\_Angle = arctan((PTU\_Height - Seat\_Height)/Distance\_to\_Chair)$$

from the floor as the *initial* scan. These scans produce the horizontal plane of the chair model, representing the seat. The same *threshold* used for the backrest is allowed when comparing the *initial* scan of the seat to its subsequent scans (5cm to 7cm is usually the range used).

After the *initial* scan is found, ORB scans down over the seat surface at 1.5^o increments. ORB searches for a number of *acceptable* scans (five to seven), usually the same amount as chosen for the backrest, to form a horizontal plane representing this seat. Assuming the seat is approximately 150 cm away, the scans will be 4 cm apart. ORB will search for scans that will cover anywhere from 16 cm to 24 cm of depth on the seat surface, which is a suitable representation since the average seat is approximately 40 cm deep[1]. Figure 4.17 shows a sample set of seat scans.

3.4.1. *Completion*

ORB ends its search for the seat once: (i) the needed number of *acceptable* scans is found, or (ii) the rangefinder has scanned 45 cm[19] in depth from its initial starting point without detecting the seat surface. If the rangefinder has scanned to the 45 cm limit and the

[19]This is the maximum depth for seats found in the CIM environment.

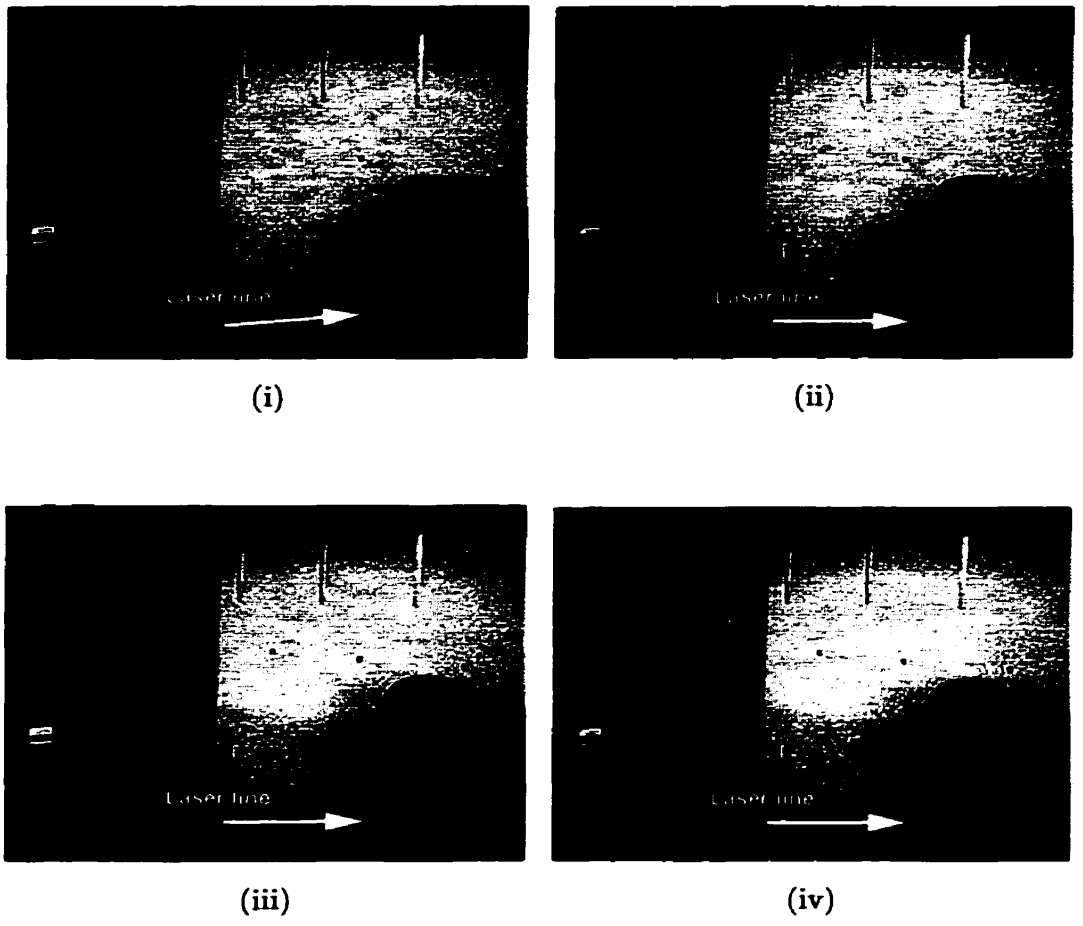


FIGURE 4.16. **Centering the Chair Seat in View.** If the rangefinder is not completely focused on the seat, background objects come into view and multiple line segments are found in the scan. (i) The chair is to the right of the rangefinder's view. (ii) & (iii) The rangefinder is then moved horizontally towards the seat surface (i.e., the closest object in view), at 2° increments until (iv) the seat is the only object visible. This results in a single line segment of range data returned from the BIRIS sensor as the rangefinder is focused solely on the seat.

horizontal plane is still not found, ORB concludes that a seat is not in view and therefore the object in question is "unknown" .

3.5. Chair Model Matching

To summarize, ORB searches for planar range profiles that fit the *model* based description shown in Figure 4.3. The model consists of a vertical planar surface representing the backrest and a horizontal planar surface representing the seat. ORB's range scans of each component should fit to these corresponding planes. These planes are fit within a given threshold to allow for uncertainty in the range readings. A diagram showing how

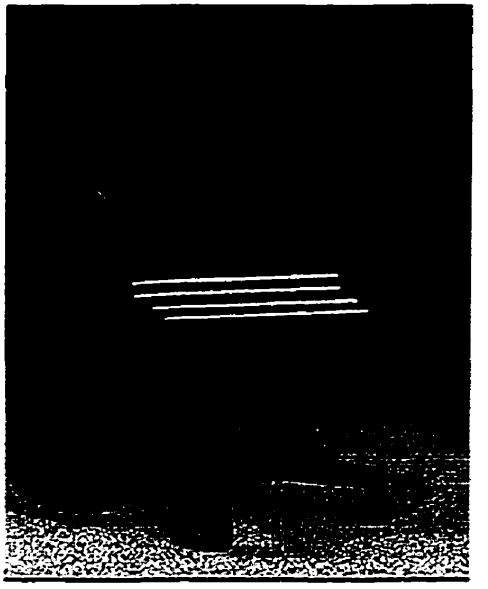


FIGURE 4.17. **Scan Lines of a Seat.** A sample set of BIRIS scans of a chair seat.

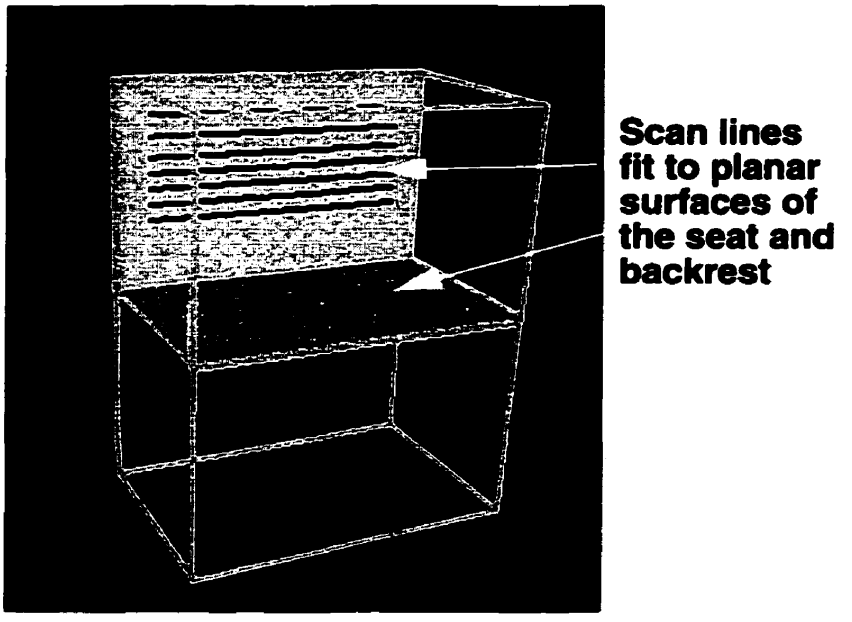


FIGURE 4.18. **Range Scans Fit to Chair Model.** The BIRIS range scans of the backrest and seat are fit to the corresponding planes as shown.

these range scans form the planes in the chair model is shown in Figure 4.18. These planes should be found at dimensions outlined in *Architectural Standards*[1].

An object other than a chair may also satisfy ORB's model based description[20]. But since the experimental area is an office space, satisfying these conditions would strongly indicate that a chair has been found. But even if the object is not a chair, since it fits the *functional* description, it could conceivably be *used* as a chair. This justifies why ORB would label this object so.

4. Table/Desk Recognition

Tables and desks are basically interchangeable in terms of appearance as well as function. ORB attempts to differentiate the two based on the desk's extra vertical supports. However, these features are not always detectable due to occlusion. For example, if the robot approaches a desk from the front[21], the side supports are not visible leaving the desk indistinguishable from a table. This differentiation is not very crucial though, as long as the object is properly recognized as a table/desk. ORB utilizes the same strategy when scanning tables or desks since tabletops and desktops are found in the same height range. This procedure is outlined in Figure 4.19.

ORB searches for a horizontal planar surface within the pre-defined range of tabletop/desktop heights[22]. ORB concludes that the object is a table/desk once this planar feature is found. ORB then scans underneath the horizontal plane (i.e., the tabletop/desktop) to determine if there are any vertical planar supports. If so, the object is considered to be a desk, otherwise it is deemed a table. Again, these labels (table/desk) are basically interchangeable since the differences between the desk and table models are minimal.

The table/desk scanning procedure is very similar to the chair scanning process, only the focus of attention is different. The planar surfaces are detected using the same principles, therefore the table/desk scanning algorithm will be described without delving into details that were previously discussed.

4.1. Assumptions

ORB assumes that the table/desk is situated in front of the selected BIRIS rangefinder[23] when the scanning procedure begins. If an approximate distance of the table/desk to the rangefinder is unavailable, then the default estimate of 150 cm is presumed. Lastly, the table/desk is assumed to be free of clutter and the tabletop/desktop surface is clear enough to be scanned.

---

[20] For example, the object could be a set of boxes of the right configuration and dimensions.
[21] The "front" of the desk is the side with open space for placing a chair.
[22] As discussed before, the heights of the tables and desks used in these experiments range from 65 cm to 72 cm.
[23] Meaning that the rangefinder need only tilt down to have the table/desk in view.

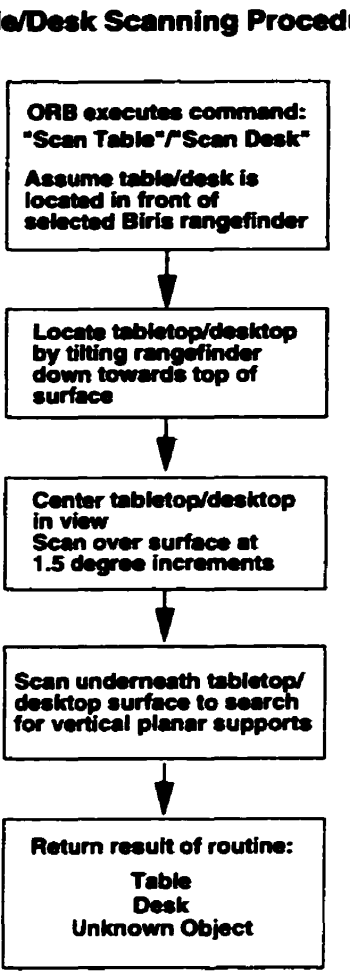


FIGURE 4.19. **Outline of Table/Desk Scanning Routine.** An overview of the steps taken by ORB to scan a table or a desk is outlined above. The details of this process will be further examined in the following sections.

4.2. Centering the Table/Desk

ORB begins by locating the tabletop/desktop in the scene. The tabletop/desktop is found at a certain height from the floor[24] and at a certain distance from the rangefinder[25]. ORB uses simple geometry to compute how far the rangefinder should tilt down to have the tabletop/desktop in view, as shown in Figure 4.20. The rangefinder should be aimed towards the far end of the table/desk, before scanning down over the tabletop/desktop surface towards the robot.

Once the rangefinder is properly positioned, the laser line should be completely projected onto the tabletop/desktop surface. If not, the tabletop/desktop must be *centered* in

[24] 65 cm to 72 cm in the CIM office space.

[25] If not known, this distance is assumed to be 150 cm.

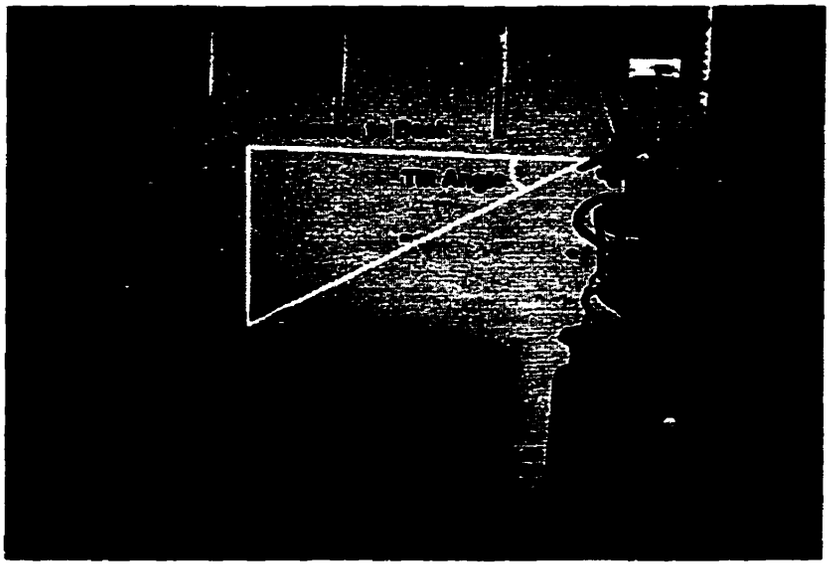


FIGURE 4.20. **Locating a Desktop.** With a desk in front of the rangefinder, the angle needed to tilt the rangefinder towards the desktop is calculated as follows:

$$Tilt\_Angle = arctan((PTU\_Height - Desktop\_Height)/Distance\_to\_Desk)$$

the rangefinder's field of view. ORB uses the same centering procedure as described for scanning a chair's planar surfaces. The tabletop/desktop is centered once an *acceptable* scan is found. The *initial* scan is used to initialize ORB's knowledge of this table/desk.

4.3. Producing the Horizontal Plane

Once the tabletop/desktop is centered, the *initial* scan determines the exact height of the tabletop/desktop. This height should fall within the pre-defined standards for tables/desks (i.e., 65 cm to 72 cm). The subsequent scans should lie along a horizontal plane representing this tabletop/desktop surface.

ORB scans over the tabletop/desktop at 1.5^o increments. Once the requisite number of *acceptable* scans[26] is obtained, it is concluded that a tabletop/desktop has been found. These *acceptable* scans: (i) return single line segments of range data after segmentation and (ii) are found within the same height range as the *initial* scan. The height of each *acceptable* scan is allowed to deviate from the height of the *initial* scan by an allowable *threshold*. This *threshold* value accounts for sensor uncertainty and the fact that different tabletop/desktop surfaces have varying reflective properties that could lead to erroneous readings.

[26]This series generally consists of seven scans that cover a length of 24 cm, with a 4 cm spacing between each scan. The number of acceptable scans needed to represent a tabletop/desktop is at the discretion of the user.

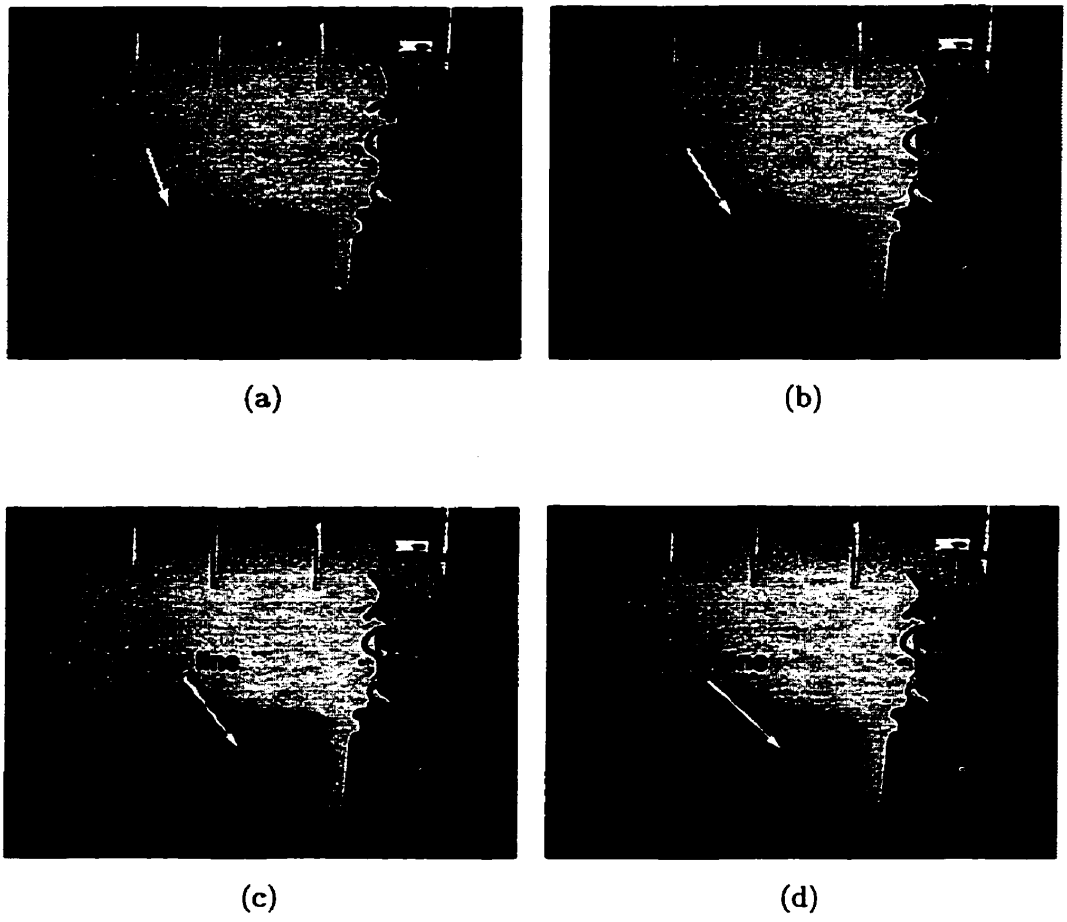


FIGURE 4.21. **Scanning a Desk.** (a) & (b) The top of the desk is scanned as ORB searches for a horizontal plane representing the desktop. (c) & (d) Once the desktop is found, ORB searches for the presence a planar support underneath. Upon detecting this planar support, ORB determines that a desk has been found.

4.3.1. *Search for Desktop Planar Supports*

The only planar feature available on a table is the tabletop surface. However, a desk has additional planar supports underneath its desktop. After establishing the existence of a planar surface representing a tabletop/desktop, ORB scans underneath this plane to search for any vertical planar supports. If one is found, then it is assumed that a desk is being scanned. Otherwise, the results are inconclusive and by default the object is called a table. This distinction is not very important as long as the table/desk is properly recognized. The search for these supporting planes is depicted in Figure 4.21.

4.4. Completion

ORB scans over the tabletop/desktop until: (i) the requisite number of *acceptable* scans are found to represent this surface, or (ii) the rangefinder has scanned down to its lowest extremity (-60 degrees) without acquiring the necessary data. In the latter case, ORB concludes that no table or desk is in view.

If ORB fails to properly recognize a table or desk, potential causes are as follows:

(i) The robot is not positioned properly, so the table/desk is not in view of the rangefinder[27].

(ii) The rangefinder readings are poor. This can be due to the lighting conditions or the reflectivity of the table/desk.

(iii) The tabletop/desktop is filled with clutter which prevented the surface from being properly scanned.

4.5. Table and Desk Model Matching

To recap, ORB searches for a horizontal planar surface at a height standard for tables and desks. The *model* based description of a table was shown in Figure 4.4, the desk model in Figure 4.5. They both consist of a horizontal planar surface representing the tabletop/desktop, with desks having additional planar supports. The range scans that form the tabletop/desktop horizontal planar surface is shown in Figure 4.22.

It is possible that an object that is neither a table nor a desk may be found to fit this model. For instance, just as a set of boxes of the right configuration and dimensions could match ORB's chair model, the same scenario could also occur with the table or desk model. But in the unlikely case that an object is falsely labelled a table/desk because it coincidentally matches the model, this action may be justified by the fact that this object may still *function* as a table/desk. Since the working environment is an office space, it can be assumed that any object satisfying this description will indeed be a table/desk.

5. Summary

This chapter dealt with modelling and recognizing the *movable objects.* These are furniture items found in all office environments, namely tables, chairs and desks. The models for these objects are idealized descriptions in which the main components are represented by planar surfaces. These planar features are detected by sweeping BIRIS range scans over the object's surface, the dimensions of which are known and modelled a priori.

[27]The control and positioning of the mobile robot is undertaken by SPOTT.

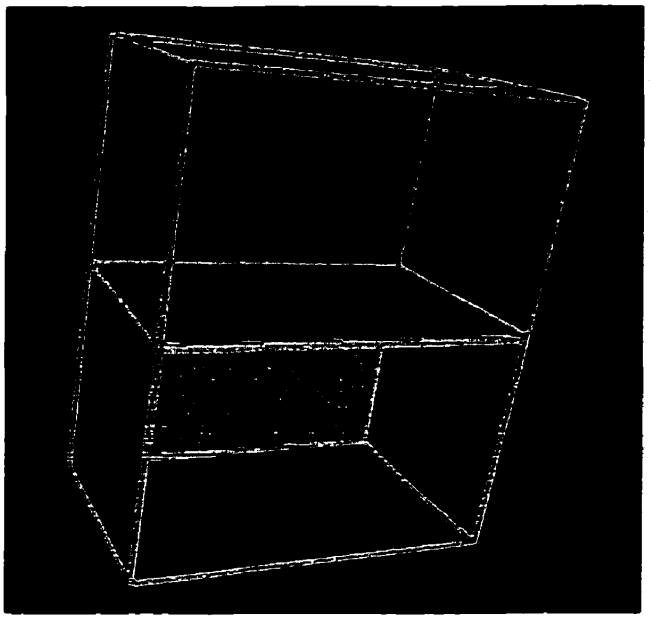


FIGURE 4.22. **Range Scans Fit to Desk Model.** The BIRIS range scans of a tabletop/desktop are fit to the corresponding horizontal plane as shown. This plane is found at the pre-defined standard height: 65 cm to 72 cm.

The robustness of these recognition procedures is dependent on the accuracy of the rangefinder. The lighting conditions in the surroundings and the material construction of the scanned object will affect BIRIS range readings. Therefore, the lighting in the experimental environment must be controlled. Exterior sunlight, or any abnormally bright lighting, must be kept out of the BIRIS rangefinder's view. This type of lighting will corrupt the laser signal, resulting in inaccurate data. Also, objects made of unreflective material, especially of darker colour (black or dark brown) will not be easily recognized. These objects attenuate the laser signal, producing less reliable data. These recognition procedures perform best in a controlled environment where the lighting is not abnormally bright, and the target objects are of light colour and fairly reflective. Distance is another factor, as the BIRIS rangefinder's optimal working range is within 2 metres. The target object should ideally be 150 cm from the robot.

The implementation issues concerning the hardware and software packages used by the ORB system are discussed in the next chapter. This will be followed by the experimental results.

CHAPTER 5

Implementation

ORB is a software package that utilizes the QUADRIS platform to complete sensory and perceptual tasks. ORB positions the BIRIS rangefinders according to scanning patterns that are unique for each object (i.e., chair or desk) and scenario (i.e., scanning a hallway or scanning a room). ORB is also in constant contact with SPOTT during navigational tasks. ORB provides SPOTT with sensor readings and labels for recognized objects. In return, SPOTT provides ORB with required data such as the robot position and orientation.

ORB is executed in one of three modes:

(i) ORB may be spawned by SPOTT[**68**] as a PVM (Parallel Virtual Machine[**29**]) process[1].

(ii) ORB may be run as a command-line executable, with command-line arguments specifying the various run-time options. This mode is generally employed during the testing/debugging stage.

(iii) ORB may also be run via a graphical user interface (GUI), which simplifies its use and offers the most basic functionality.

The main factor affecting ORB's performance is the reliability of the QUADRIS platform, since erroneous range data will affect ORB's perception of the environment. QUADRIS is vulnerable to exterior lighting conditions since certain light sources elude the BIRIS rangefinder's filter, corrupting the signal. The colour and material of the scanned object will also affect range readings. Darker objects, which do not properly reflect the laser line, will cause problems since the returned signal will be attenuated. As well, proper calibration of the BIRIS rangefinder must be performed regularly to ensure optimal results.

[1] This mode of execution will be described in more detail later on.

1. The ORB System

ORB is written in the *C* and *C++* programming languages for use on SGI Indy workstations. ORB is executed in one of three modes as previously outlined. The first option[2] is utilized when SPOTT executes a task requiring ORB. For instance, when navigating through the hallways of an office space, SPOTT uses ORB to label walls and determine what state the doorways are in (i.e., open/closed/partially open). The second and third options[3] are generally employed for testing and debugging purposes, and for demonstrations highlighting ORB's functionality. It is also the mode used to demonstrate the *Scan Object* experiments shown in the following chapter.

The inputs to the ORB system are as follows:

(i) The task command from SPOTT[4] or from the user[5].

(ii) The current robot position and orientation[6].

(iii) The BIRIS range data provided by the library package *libBiris*, which converts the BIRIS video image to range data.

ORB's outputs are as follows:

(i) The positioning of the pan-tilt units (PTUs) of the QUADRIS platform (i.e., the PAN and TILT angles for each PTU).

(ii) The selection of the rangefinder to obtain the current range scan (i.e., BIRIS_1 or BIRIS_2).

(iii) The segmented range data, with corresponding labels if applicable, which are sent to SPOTT for incorporation in the map database.

These inputs and outputs to the ORB system are outlined in Figure 5.1. An overview of the entire system and how ORB communicates with the various software and hardware components will now be discussed.

1.1. System Overview

A block diagram of the ORB system was presented in Chapter 1. A slightly more detailed version is found here in Figure 5.2, with the flow of operations outlined.

---

[2]ORB is run as a PVM process spawned by SPOTT.

[3]ORB is run "stand-alone" as a command-line executable or via a GUI.

[4]SPOTT determines which functionality it requires from ORB and selects the appropriate executable. ORB is compiled such that a separate executable exists for each available task.

[5]Could be selected via the GUI, or the appropriate executable is run from the command line.

[6]The robot position is supplied by SPOTT through *Robodaemon*. Robodaemon is a software development system that allows external software to interact with the mobile robot[**23**].

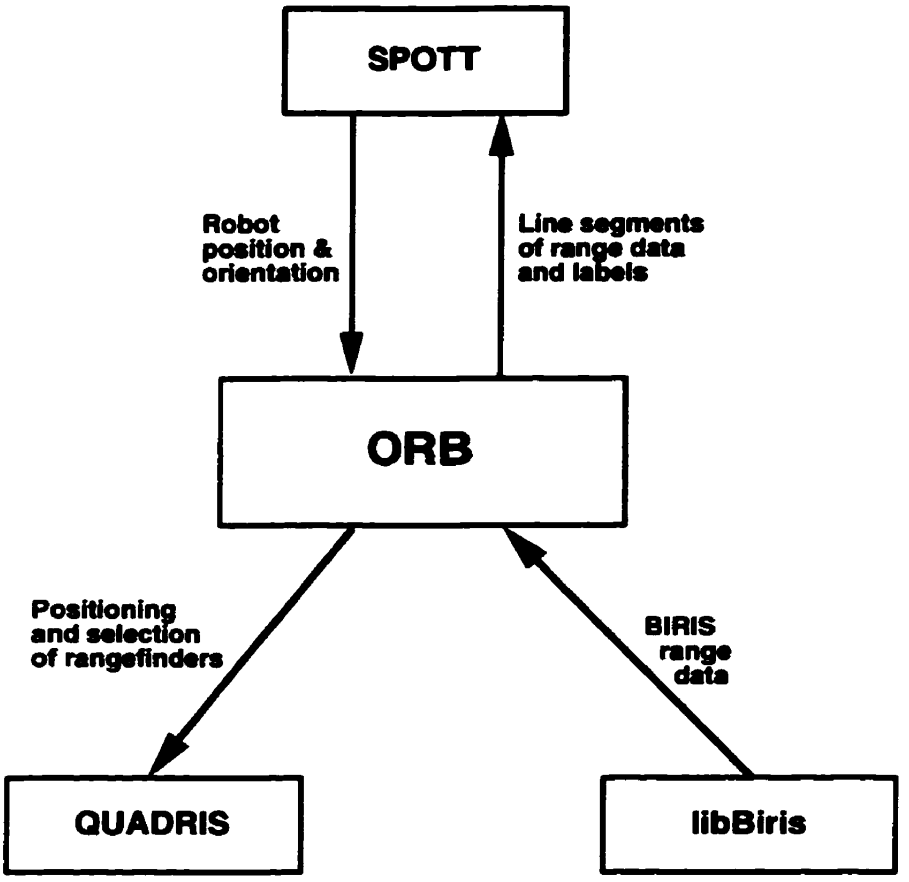


FIGURE 5.1. **Inputs and Outputs of the ORB System.** This is an outline of the data passed between ORB and the hardware and software packages it utilizes. ORB sends positioning commands to control the hardware components of the QUADRIS platform, and receives BIRIS range data from the software library libBiris. ORB is also in constant communication with SPOTT, providing QUADRIS range data updates to the map database, along with labels of recognized objects/structures. SPOTT provides ORB with the robot position and orientation.

When ORB is executed, by either SPOTT or by the user, a task command (e.g., *Scan Hallway*, *Scan Chair*) needs to be specified. The user may select a task via the GUI, or the user can just launch the appropriate executable from the command-line[7]. Similarly, when ORB is spawned as a PVM process, SPOTT selects the executable to perform the required task.

From the task command, ORB determines what recognition routine to perform as well as what PTU scanning pattern (i.e., the positioning and selection of the rangefinders) is needed. Once the rangefinders are properly positioned, ORB obtains range data from one of the rangefinders for processing. ORB uses the current robot position and orientation, obtained from SPOTT, to transform the raw data from the local frame of reference of the

[7]ORB is pre-compiled to perform each available task.

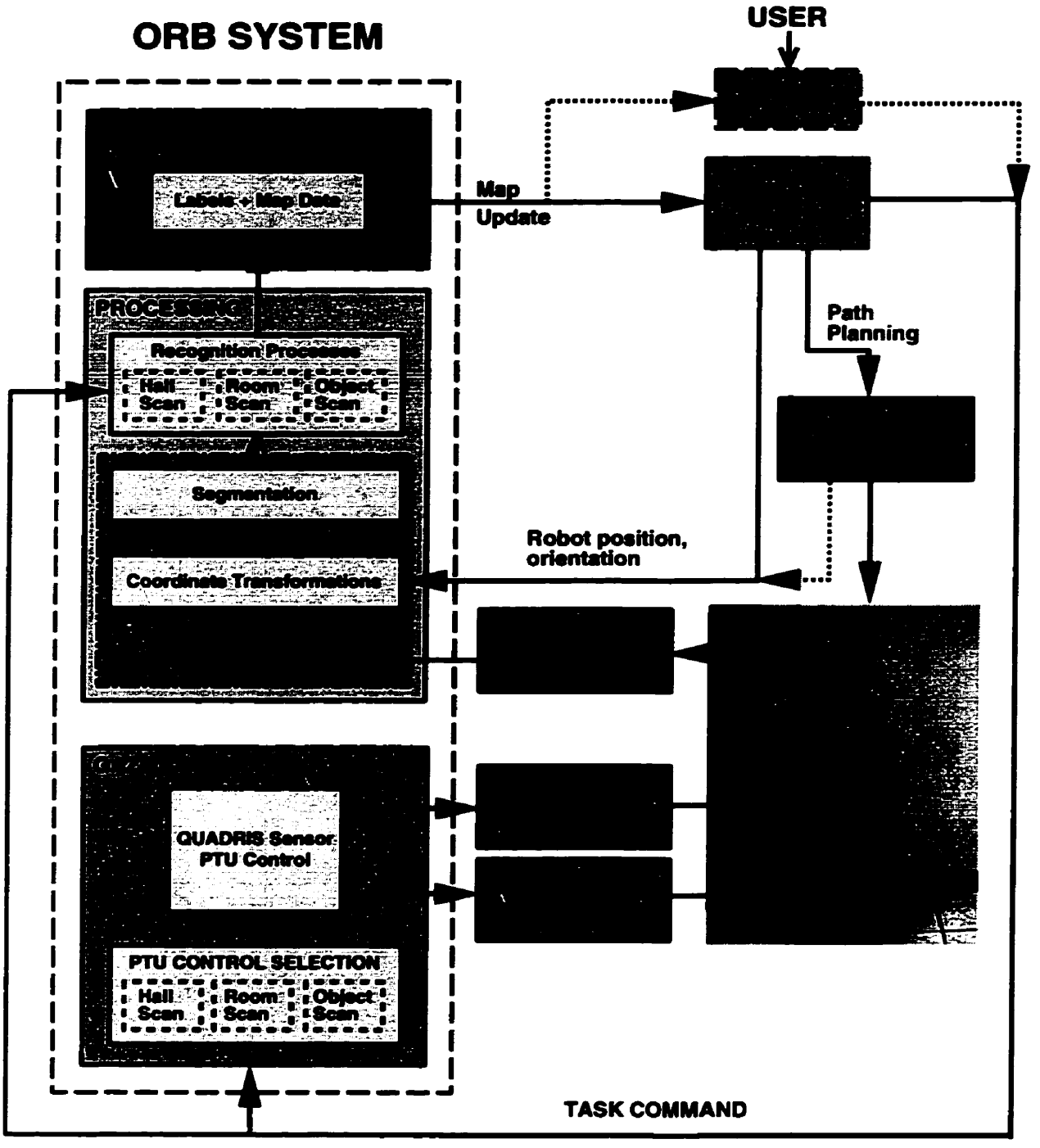


FIGURE 5.2. **Block Diagram of the ORB System.** The different sections of the ORB system are shown, along with the various hardware and software components it utilizes. The flow of operations begins with a task command specified by either SPOTT or by a user. This command determines what recognition process ORB will execute, as well as the PTU scanning pattern required. ORB then selects (using cportd) and positions (using ptud) a rangefinder before obtaining a BIRIS range scan (using libBiris). The data is transformed from the local frame of reference of the rangefinder, to the world coordinates of the map database before being segmented. The line segments are consequently processed by the appropriate recognition module and labels are assigned to each. The line segments and their labels are sent to SPOTT and are displayed on screen for the user.

rangefinder, to the world coordinate system of the map database. The range data is then segmented using Ramer's method[**53**] to obtain a line segment representation. Each line segment is then given a label by the selected recognition process. For a *Scan Object* task (e.g., "Scan Chair"), several scans must be taken before the recognition process is complete. A label (i.e., chair/table/desk) is assigned only after the object has been fully scanned and identified. The *Scan Hallway* and *Scan Room* tasks label each line segment as they are received. ORB sends the line segments, with their labels, to SPOTT for incorporation in the map database. ORB continues to perform the current task until told to stop[8], or until a new task is specified.

1.2. Communications

ORB's performance is highly dependent on its communication with the QUADRIS hardware and with SPOTT. The response time of QUADRIS' hardware components are currently the bottleneck to faster processing by ORB. The limiting factors are the rangefinder selection and the placement of the PTUs (i.e., the speed limitations of the PTUs).

1.2.1. *QUADRIS*

The QUADRIS platform consists of two BIRIS rangefinders mounted on pan-tilt units (PTUs). ORB positions the PTUs and selects the rangefinders via the daemon processes *ptud* and *cportd* respectively[9]. A PTU positioning command specifies the direction of movement (i.e., PAN or TILT) along with the degree of movement (i.e., angle in degrees). The commands are sent to the daemon processes through UNIX socket connections. The daemon processes communicate with a processor (i.e., 486) on board the mobile robot, to control the QUADRIS hardware, via a wireless radio ethernet link. Once the rangefinders are properly placed, ORB obtains a BIRIS range scan from the *libBiris* software library, which converts the BIRIS video image into an array of range data. The BIRIS video frame is obtained from QUADRIS' video transmitter on board the robot. The image is sent to a video receiver, which is connected to the framegrabber of an SGI Indy workstation where the processing is done.

A problem encountered during experimentation in the CIM office environment was the occasional loss of radio transmission with the mobile robot. This occurs when the robot has travelled a long distance from the receiver, usually beyond 40 metres. The video link that transmits the BIRIS images also suffers when there is no direct line of sight between the

[8]SPOTT sends an exit command once ORB's services are complete.

[9]These daemon processes were designed and developed at CIM in the Mobile Robotics Laboratory.

video transmitter and the receiver. The images become corrupted resulting in erroneous data. This problem could be alleviated by placing a network of antennas throughout the floor connected to the receiver. This would allow the receiver to be in direct contact with the transmitter no matter where the robot travels on the floor. However, this would mean altering the environment, which is not always feasible. The experiments for this thesis were done with the receiver located at a central location on the office floor. The video transmissions were satisfactory with this set-up, except at the extremities of the floor where the signal was somewhat degraded. Another hardware consideration is the battery power for the mobile robot, which has a life span of approximately five hours of continuous use. The accuracy of QUADRIS degrades as the battery power weakens, but this is only a problem for navigational tasks of very long duration.

1.2.2. *SPOTT*

SPOTT uses the PVM (Parallel Virtual Machine)[**29**] software tool to distribute its processing across a network of SUN and SGI workstations[**69**]. PVM is a software system that enables a collection of interconnected computers to be used for concurrent or parallel computation in a network environment[**29**]. PVM provides a set of library routines for automatically starting up processes on the PVM network, and allowing these processes to communicate and synchronize with each other. PVM provides high level library function calls which make actual system calls and UNIX socket communications transparent to the user application's code.

ORB is spawned by SPOTT as a process which runs on an SGI Indy workstation in the PVM network. All subsequent communication between ORB and SPOTT is done through PVM resources. ORB is compiled as a set of executables which perform each available task (e.g., *Scan Hallway*, *Scan Room* etc.). SPOTT chooses the appropriate executable based on the navigational task it is required to complete. For example, when navigating through a hallway, SPOTT would call on ORB's *Scan Hallway* executable. On the other hand, if the robot is in a room, the *Scan Room* executable would be selected.

The PVM library provides routines for packing and sending messages between processes running on the PVM network When ORB requires data *from* SPOTT (i.e., the current robot position and orientation), ORB sends a request and waits for SPOTT to reply with the desired information. When sending data *to* SPOTT, for example a line segment of range data for the map database, ORB sends a signal to SPOTT indicating that this data is on the way, before packing and sending the data. SPOTT receives the data in a buffer

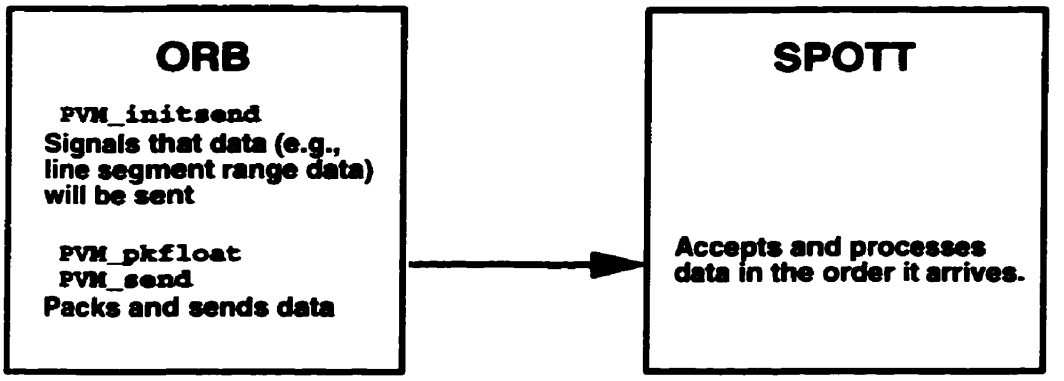


FIGURE 5.3. **ORB Sends Data to SPOTT.** The protocol followed when ORB sends data to SPOTT is outlined above. The actual PVM function calls are also included for clarity. When sending data to SPOTT, ORB initiates the transfer by sending a signal (*PVM_initsend*) indicating that the data is to follow. ORB then "packs" the data (e.g., PVM_pkfloat(data) for floating point values) before "sending" it (i.e., PVM_send(data)).

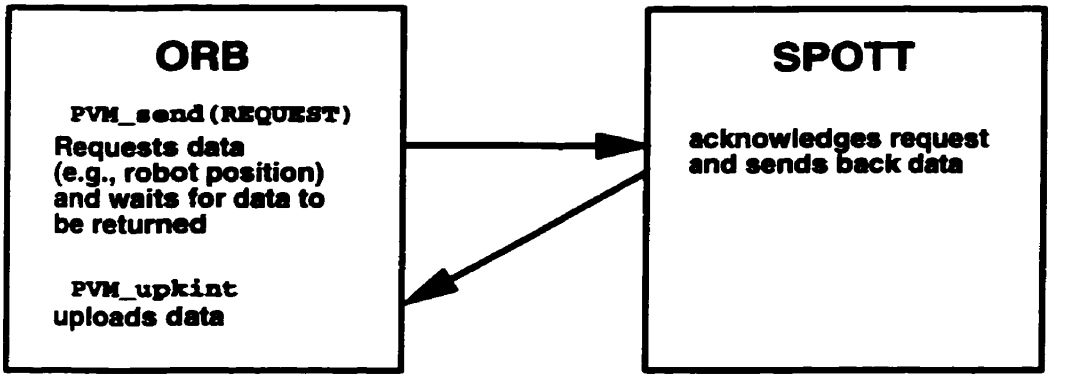


FIGURE 5.4. **ORB Requests Data from SPOTT.** When ORB requires data from SPOTT, for example the current robot position, ORB sends a PVM "request" command (i.e., *PVM_send(QUADRIS_ROBOT_REQUEST)* indicating the type of data needed. ORB then waits for SPOTT to acknowledge and process the request, before uploading the desired data (e.g., *PVM_upkint(data)* to upload an integer value).

and processes the data in a "first come first served" basis. These scenarios are outlined in figures 5.3 and 5.4. Descriptions of the actual PVM function calls for these routines are found in [**29**].

2. ORB's Display

ORB provides a user friendly graphical interface to update the user of its progress. The display includes the latest range reading from QUADRIS, which is continually updated as soon as data is received. If the task is either of *Scan Hallway* or *Scan Room*, a map showing a plan view of the range data acquired thus far is also displayed. For tasks that recognize *movable objects* (i.e., *Scan Chair*), the range data are stored and can be viewed upon completion of the task via the *rim*[10] software package. ORB's display is shown in Figure 5.5. The GUI option is demonstrated so that all of ORB's display windows are presented.

From the GUI, the user selects one of the available recognition procedures. Each selection opens a window that allows the user to modify a set of parameters pertaining to the chosen task. This arrangement is shown in Figure 5.6, with the *Scan Chair* task used as an example. Once the user is content with the settings, the process is initiated by the "START" button.

[10] *Rim* (Range Image Monitor) was developed at CIM by the Artificial Perception Laboratory.

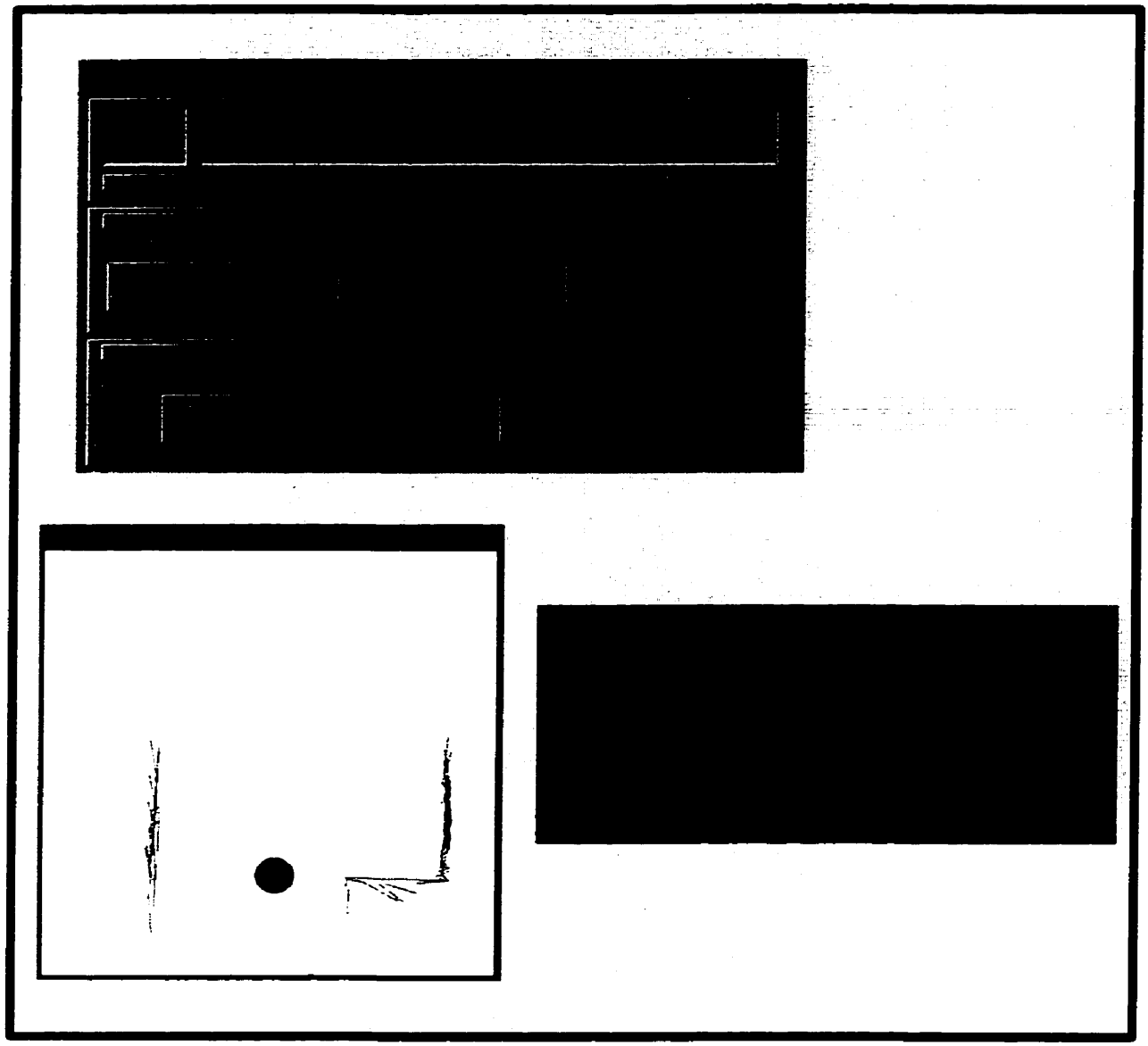


FIGURE 5.5. **ORB's Graphical User Interface.** ORB's display consists of a GUI, a window displaying the current BIRIS range reading, and a plan view of the range data acquired from the environment thus far (only used for the *Scan Hallway* and *Scan Room* tasks). The GUI allows the user to select a task for ORB, as well as vary certain parameters pertaining to that recognition procedure.

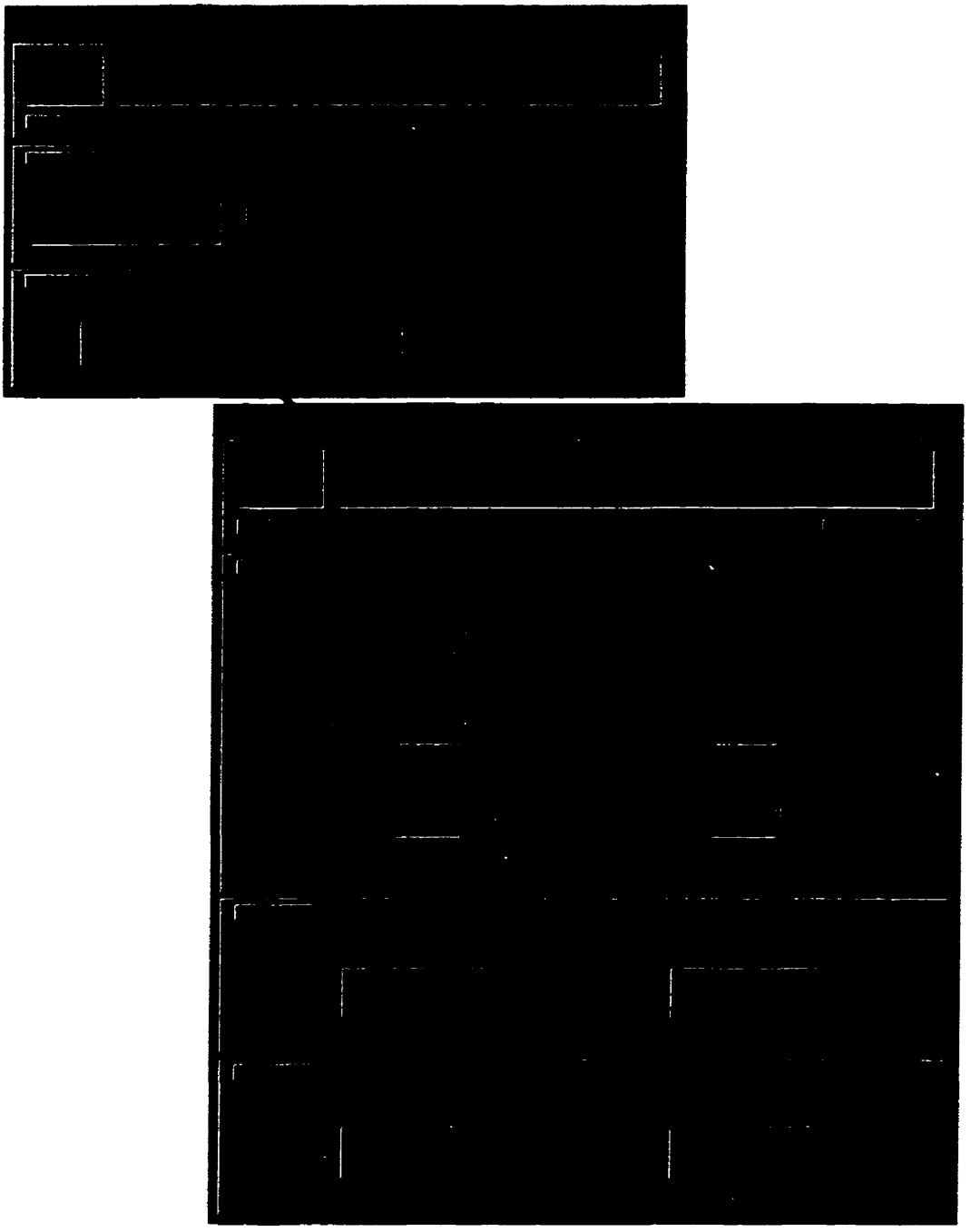


FIGURE 5.6. **ORB's GUI Parameter Settings.** When the user selects a task from the main GUI, in this case *Scan Chair*, a secondary window appears offering the user several options. Parameters may be adjusted to cope with different experimental conditions. The default values are generally suitable, but special cases need to be accounted for. Among these variable parameters are the thresholds for fitting range scans to planes. The thresholds for the seat and backrest are modifiable for a *Scan Chair* task. If the range readings are less reliable than usual (due to the reflectivity of the chair's surface for instance), the thresholds may be increased. Also, the target chair's dimensions may not conform to the known standards, so the seat height and backrest height of a particular chair may be entered. Once all the parameters have been set, the process is launched by the START button. At any point in time, the process may be terminated by the STOP button. After completion of the scanning procedure, the range scans of the chair may be viewed by *rim*, and the chair's model can be viewed using *Inventor*.

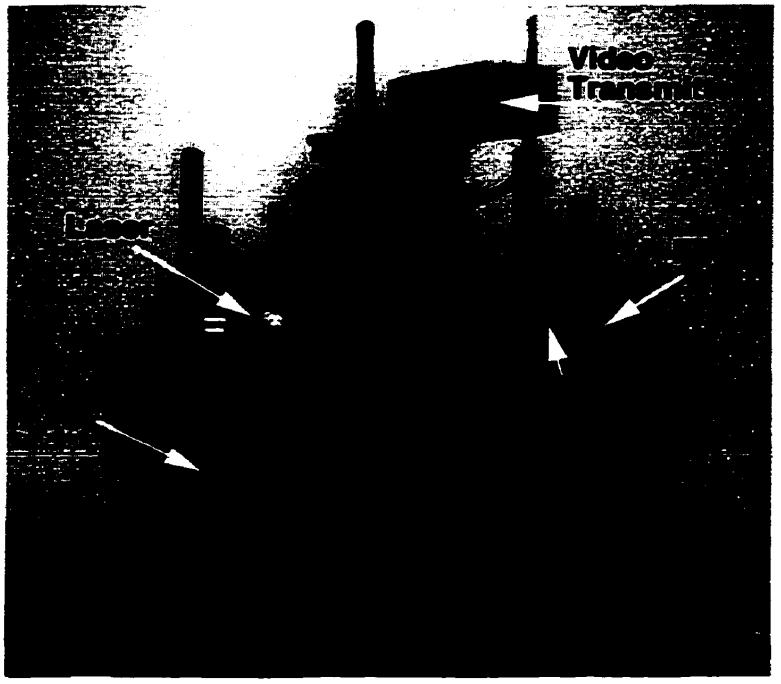


FIGURE 5.7. **QUADRIS: A Mobile Robot Sensor Platform.** QUADRIS is composed of two BIRIS sensors mounted on pan-tilt units. Each BIRIS sensor is composed of a CCD camera, a laser line projector and a bandpass filter.

3. The BIRIS Laser Rangefinder

The robustness of ORB's recognition algorithms is dependent on the reliability of the BIRIS range data. The BIRIS sensor is comprised of a CCD camera, a laser line projector and a bandpass filter, as can be seen in Figure 5.7.

In front of the CCD camera lens is a mask with two apertures. This mask produces two offset images superimposed on one another, with the offset between the two images of an object being proportional to the distance of that object to the sensor. This is analogous to the *stereo disparity* principle in that range is inferred from the *disparity* between two images acquired from the one scene. The optical principle of the BIRIS sensor can be seen in Figure 5.8. A horizontal laser line is projected onto the scene and its reflection on an object will appear as two lines on the CCD sensor (see Figure 5.9). The disparity between these two lines is inversely proportional to the distance of the object to the sensor. For objects that are closer, the laser lines will be further apart from each other than if the object were at a greater distance away. A comparison is shown in Figure 5.9. A vertical laser line is also available, but for use with a mask with four apertures producing *three* laser lines in the image. This arrangement is used for the recognition of *geons* [**47**] and is not

discussed in this thesis. In order to facilitate the processing of the image, a bandpass filter, tuned to the wavelength of the laser light ($\sim$=680nm), is placed in front of the camera lens to obtain an image of the laser lines with the background attenuated. An image of the laser projected onto the scene and the filtered image can be seen in Figure 5.10. An array of range data is produced along the laser line. There are 640 pixels across the width of the CCD array providing 640 range readings along the laser line.

To correlate the disparity between the lines with range, a lookup table is produced. The lookup table consists of a set of coefficients (C_1[x],C_2[x], x=[1,2,...,640] for each column in the array) which relate disparity to range. The calibration procedure to produce the lookup table consists of taking disparity measurements from positions of known distance to an object (e.g., a wall). These series of measurements result in a set of disparity versus range data. The filter and lens cause distortion which increases towards the edge of the lens, so an array of coefficients is needed for each position in the array. A least square minimization and fitting routine is used on the range versus disparity data to compute C_1[x] and C_2[x]. The relation between disparity and range is as follows:

$$disparity[x] = \frac{C_1[x]}{range[x]} + C_2[x] \tag{5.1}$$

with x being the position on the array [1,2,...,640].

A range reading would constitute computing the disparity between the two laser lines in the BIRIS image for each column in the array (640 readings). The corresponding range values are calculated from the above relation using the coefficients (C_1[x],C_2[x]) stored in the lookup table. A more thorough discussion of how the disparity is computed, and hence how range is inferred, can be found in [**24**].

3.1. Properties of the BIRIS Rangefinder

Some properties of the BIRIS rangefinder are now described.

(i) The laser line spans 30 degrees so that is BIRIS's field of view. But since the data towards the border of the camera lens are less reliable, margins are set on the range array which then limit the field of view to the most accurate data, spanning around 20 degrees.

(ii) BIRIS is calibrated to provide range data from 30 cm up to a distance of 5 meters. A typical BIRIS scan of a wall from 130 cm away is shown in Figure 5.11. The filtered stereo image of the laser line is shown next to the resulting range data.

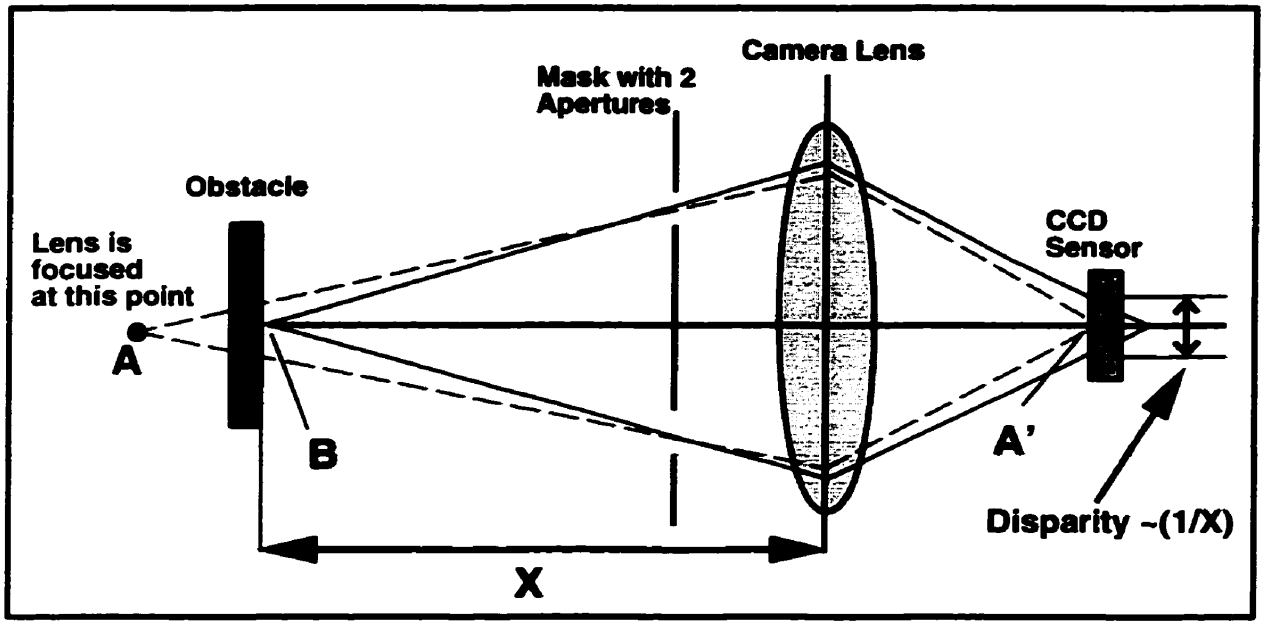
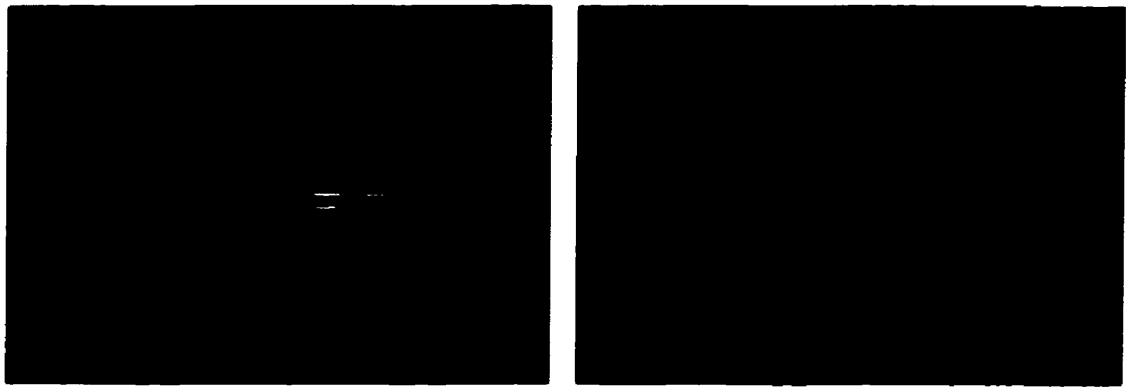


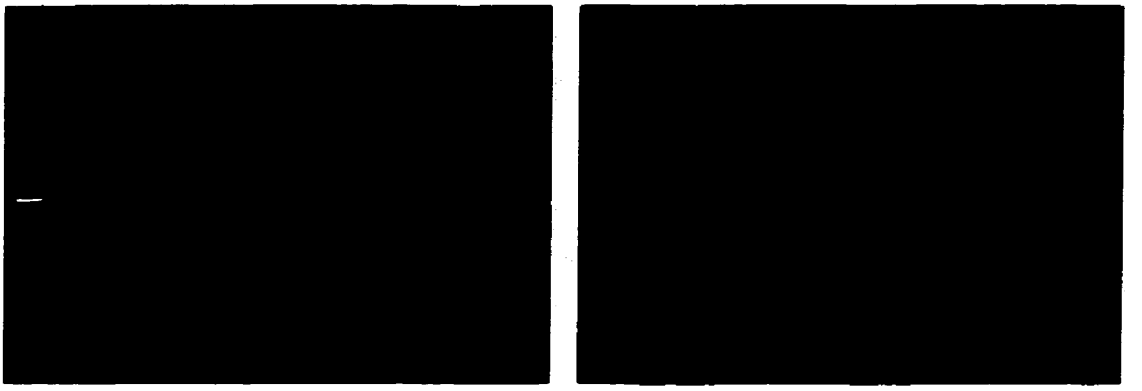
FIGURE 5.8. **BIRIS Sensor Theory.** The BIRIS sensor consists of a two aperture mask placed in front of the lens of a CCD camera. This creates a double image in which the disparity between the two images is a function of its distance to the CCD sensor. The reference plane is set by the camera lens focus, which in our setup is infinity. So a point at this position A will not be affected by the mask, and will be projected onto the CCD at point A'. But another point, B, will be projected onto the CCD at two points, with the disparity between them a function of the distance of point B to the sensor.



(a) (b)

FIGURE 5.9. **BIRIS Lines From Close and Far.** (a) How the laser lines appear through the mask from 30cm. (b) How the laser lines appear from a further distance, 200cm away. The disparity between the lines is inversely proportional to the distance of the object to the sensor.

(iii) BIRIS is accurate to within 2 cm up to a distance of 2 meters, after which there is a gradual degradation in performance. BIRIS accuracy is also dependent on the environment, as surface reflections and exterior lighting conditions are very important. Surfaces that do not reflect well (i.e., darker coloured surfaces) and do not



(a) (b)

FIGURE 5.10. **BIRIS Laser Lines With and Without Filter.** (a) Image taken without the bandpass filter. The laser lines are visible, but their signal will be obstructed by the external lighting conditions. (b) Image with the filter. With the background attenuated, there is no interference with the laser line signal. One drawback is that the filter induces an effect whereby the edges of the image are lost, narrowing the field of view.

return the laser signal will cause problems. As well, exterior light which cannot be filtered out will corrupt the signal and cause erroneous range data. An example of this phenomenon is seen in Figure 5.12. The filtered image clearly shows external light eluding the filter and resulting in corrupted range data.

(iv) BIRIS captures its signal at 10 frames/second, which is more than adequate for most mobile robot tasks.

(v) BIRIS data consists of an array of range readings along the laser line.

(vi) Power for the sensor is provided via the existing on board battery system powering our robot, the "Nomad 200", which consists 12 V power for the camera and PTU's, as well as 6 V power for the laser.

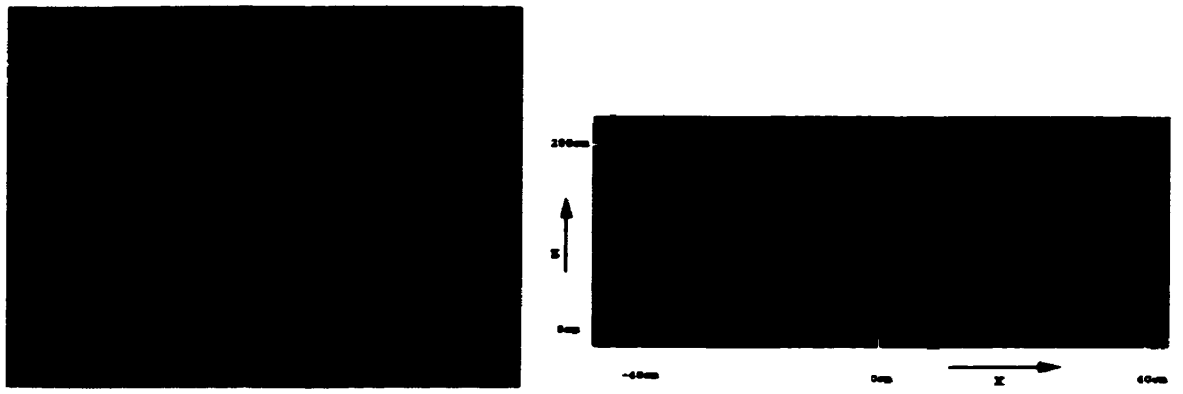
(vii) The QUADRIS platform, which consists of two BIRIS sensors, fits comfortably on the "Nomad 200", which has a diameter of 46 cm.

3.2. BIRIS Range Data

BIRIS range data may be affected by a number of factors. First of all, the lookup table that correlates disparity between the BIRIS laser lines to range needs to be accurate. For this reason, the lookup table must be updated frequently for optimal results[11]. The other factors have to do with the working environment.

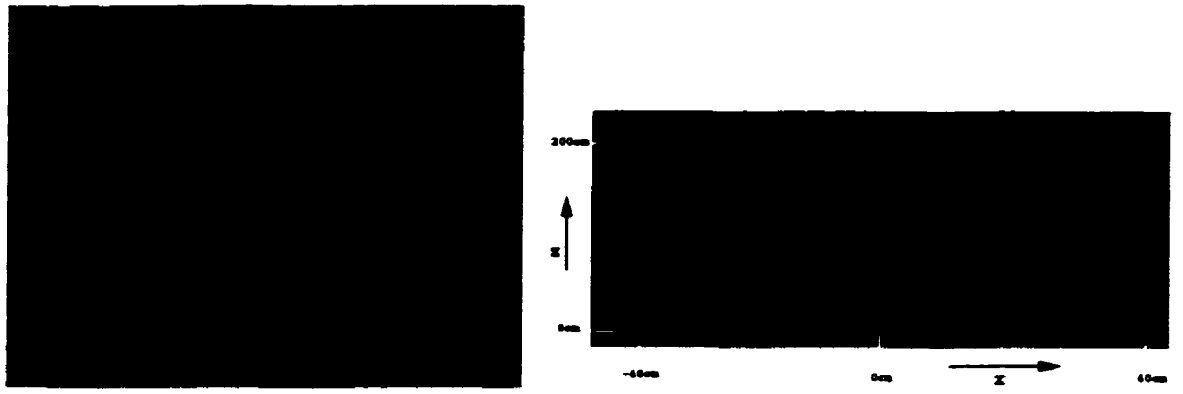
External lighting conditions may cause problems for BIRIS. Direct sunlight or any abnormally bright light may pass through the BIRIS filter corrupting the signal. This

[11]The calibration procedure that produces this lookup table is described in[**24**].



(a) (b)

FIGURE 5.11. **BIRIS Scan of a Wall from 130 cm.** (a) The filtered image shows the stereo laser lines from a scan of a wall 130 cm away. (b) The range data is plotted with the horizontal axis being parallel to the plane of the lens, and the vertical axis indicating distance of the sensed object normal to the plane of the lens.

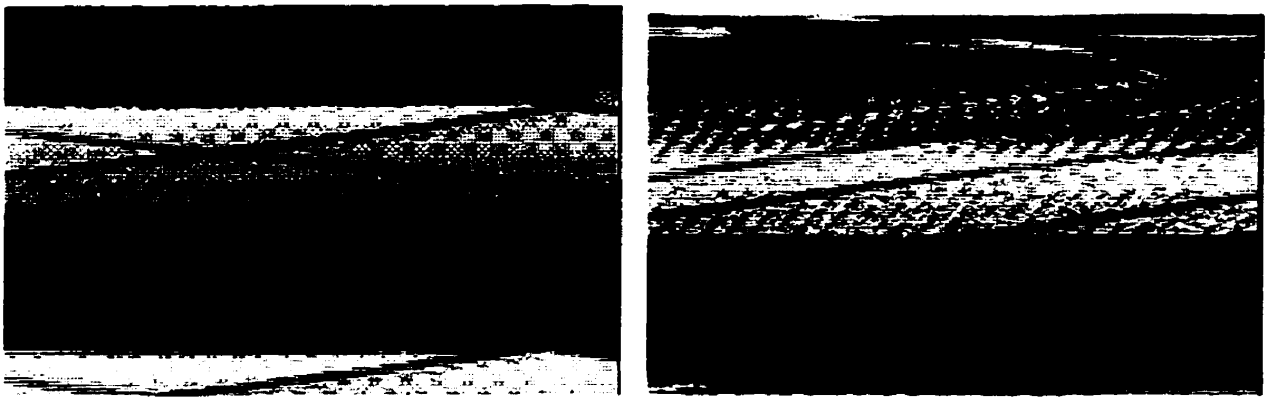


(a) (b)

FIGURE 5.12. **BIRIS Scan Corrupted by Exterior Light.** (a) The filtered image shows how external lighting may pass through the filter, corrupting the signal. This is a scan of a box on a windowsill. The window in the background is covered with horizontal blinds that are slightly open. Sunlight causes BIRIS problems as it cannot be filtered out by our current setup.(b) The resulting noisy range data.

phenomenon was shown in Figure 5.12. Unreflective materials and darker colours will also have an effect on QUADRIS data. Surfaces which do not reflect the laser line well produce a weaker signal, resulting is less accurate data.

There are also occasional problems transmitting the BIRIS video image from the robot to the SGI workstation. When there is no line of sight between the video transmitter and the receiver, the video signal degenerates. Examples of the degraded video image are shown in Figure 5.13.



(a) (b)

FIGURE 5.13. **Corrupted BIRIS Video Images.** These are sample BIRIS video images that were corrupted due to the loss of transmission from the video transmitter on board the robot. The BIRIS laser lines are lost which results in totally erroneous data.

Corrupted BIRIS video images, as shown above, contain much more noise than properly filtered ones, and can be detected and discarded before being processed. Areas of the image which are supposed to contain no signal are monitored. These areas are anywhere above and below the two BIRIS lines in the center of the image. If any noise is found in these areas, the image is discarded. New scans are taken until a suitable image is received (i.e., an image with no noise signal found in the monitored areas). Figure 5.14 shows some sample BIRIS video images with the monitored areas outlined.

However, these problems rarely affect ORB's performance. ORB can effectively filter out any corrupted BIRIS frames as shown above. ORB was designed to utilize QUADRIS as a mobile robot sensor platform that recognizes landmarks in an office space. BIRIS has proven to be quite capable of completing these tasks, as will be shown in the experimental results discussed in the following chapter.

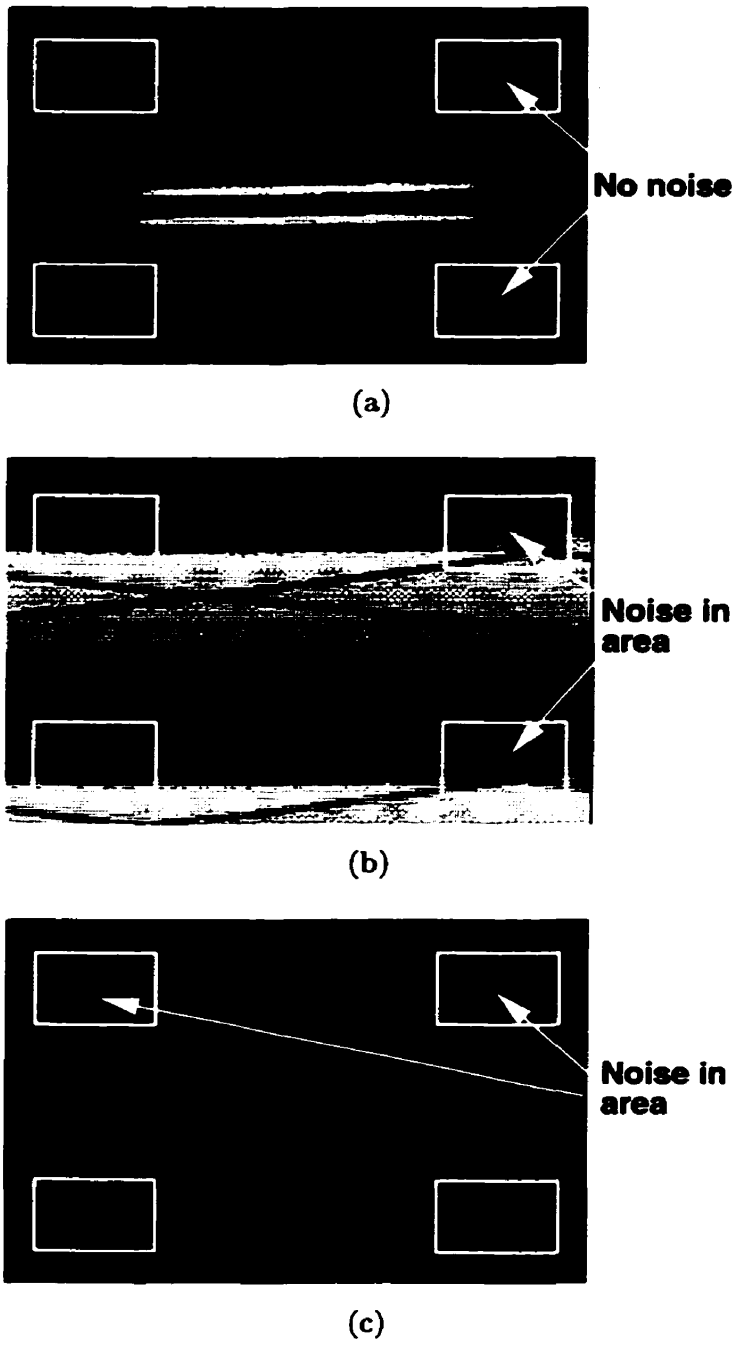


FIGURE 5.14. **BIRIS Video Image Verification.** The areas of a BIRIS video frame which are monitored for noise are four rectangular windows in each corner of the image. (a) In a stable BIRIS video image, the background should be completely filtered leaving only the BIRIS laser lines. If the windowed areas contain no signal, this indicates that the frame is acceptable. (b) If line of sight is lost between the video transmitter and the receiver, transmission problems occur. This is a worst case scenario whereby the BIRIS lines are lost and the whole image is distorted. The monitored areas detect noise where the image should be filtered, so the frame is rejected. (c) Abnormally bright lighting may pass through the filter and corrupt the image. This noise would be found in the monitored windows resulting in a rejection of the frame.

CHAPTER 6

Experiments

The experiments conducted in this work were done in the hallways of the Centre for Intelligent Machines (CIM) at McGill University, as well as in the Mobile Robotics Lab at CIM. Although this work is modelled on the CIM environment, ORB can be applied to any office space, provided certain dimensions of the *structural* and *movable* objects are available.

ORB's processing speed is suitable for the applications at hand (i.e., mobile robot exploration of an office environment [**69**]), but is somewhat limited by communications with the QUADRIS hardware. The movement of the PTU (PTU speed) and the selection of the rangefinder[1] are the two of the more time consuming procedures when performing a scanning routine (i.e., *Scan Hallway*). The rest of ORB's processing entails the computation of range data from the BIRIS video image, and the interpretation of this data (i.e., the recognition procedure).

The following experiments were performed to demonstrate ORB's functionality.

(i) The computing costs for the different components of ORB's system is determined, with the limiting factors to ORB's performance outlined.

(ii) Recognition of the *structural objects* during a *Scan Hallway* procedure is exhibited.

(iii) A sample map produced by ORB during a *Scan Room* routine is presented.

(iv) The last set of experiments concern the recognition of the *movable objects*: chairs, tables and desks.

The experiments were run with ORB as a "stand-alone" unit, as well as in conjunction with SPOTT as a PVM process.

[1] Only data from one QUADRIS sensor may be obtained at a time, so the selected sensor must be specified. This is only an issue when scanning a room or hallway, where ORB switches alternately between the two rangefinders. When scanning a *movable object* (i.e., a chair), the rangefinder is specified at the outset.

1. Computing Costs

The different steps taken by ORB to obtain and process a BIRIS range scan are outlined in this section, with the focus on the computing costs. The *Scan Hallway* task is used here as an example, although the steps taken to process a range scan is the same regardless of the task.

The scanning of a hallway can be broken down into the following actions:

(i) ORB's *Scan Hallway* executable is spawned by either SPOTT or by the user.

(ii) ORB selects a rangefinder based on the scanning pattern used for this task.

(iii) The PTU is moved into place (i.e., the rangefinder is positioned, if need be, according to the scanning pattern.).

(iv) A range scan is taken at this position, which constitutes: (a) querying SPOTT/robodaemon for the current robot position and orientation, (b) grabbing a BIRIS video frame, (c) computing range from disparity of the BIRIS laser lines in the image, (d) transforming the coordinates of this data from the local frame of reference of the rangefinder to the world coordinate frame of SPOTT's map database.

(v) The range data is segmented using Ramer's method.

(vi) ORB's labels each line segment and sends these line segments with their corresponding labels to SPOTT.

If no new command is given, the current task is continued with steps (ii) on through (vi) again. A sample set of times taken to complete each stage is shown in Figure 6.1. These values show the range of time needed to complete each step of a *Scan Hallway* task, based on observations of 20 separate experimental runs. These computation times will vary based on the workload of the workstations running ORB and the daemon processes (*cportd* and *ptud*[2]). It is preferable that the experiments be done on workstations dedicated solely to ORB, and that the workload on the network is light.

The first two steps are the rangefinder selection and movement of the PTU. These processes utilize hardware components that have inherent delays associated with them. The rangefinder selection is done through an alarm switch (VS5004 Video Switcher), which can select between four video signals (QUADRIS uses only two of these ports)[**44**]. The user application (ORB) connects to the alarm switch via the daemon software application *cportd*. This rangefinder selection process can take anywhere from 100 msec to 300 msec to

[2]These daemon processes were designed and developed at CIM in the Mobile Robotics Laboratory, as explained in the previous chapter.

Hallway Scan Steps		Time (msec)	
		Minimum	*Maximum*
Camera Selection		**100**	**300**
PTU Movement		**1057**	**1260**
Range Scan			
	Query Robot Position	**60**	**200**
	QUADRIS Frame Grab	**52**	**75**
	Frame Verification	**0.5**	**0.7**
	Compute Disparity	**153**	**298**
	Compute Range	**1.0**	**1.2**
	Coordinate Transformations	**7.7**	**9.8**
Segmentation		**2.6**	**8.3**
Recognition		**17**	**22**

TABLE 6.1. **Computation Times to Complete Processing of a Range Scan.** This is a table containing the range of times taken to complete the processing of a single range scan during a *Scan Hallway* routine. These results show the maximum and minimum values obtained from 20 separate experiments, and are indicative of what to expect from ORB in terms of performance.

complete. The physical movement of the PTU also requires some time. It takes the PTU approximately 1.2 seconds to move 45 degrees.

Once the rangefinder has been selected and is properly positioned, a range scan is taken. The majority of the range scan computation is for the disparity calculation[**24**], at 150 msec to 250 msec. By comparison, the conversion of disparity to range, and the subsequent coordinate transformations require minimal work. The frame grabbing and frame verification procedures are also insignificant in terms of time taken for completion. The time needed to obtain the robot positioning information varies anywhere from 100 msec to 200 msec, depending on the workstation load. This data is provided by SPOTT via robodaemon, so access to this data is dependent on how quickly robodaemon distributes the information.

Once the range scan is obtained, the range array is segmented using Ramer's method[**53**]. The computational complexity increases with the number of line segments produced, but since most of ORB's scans produce single line segments, the time to complete this process is almost negligible. ORB then categorizes these line segments through

the appropriate recognition procedure. The use of simple models to classify each range scan enables ORB to complete this process relatively quickly.

The time taken to pack and send data to SPOTT is not discussed in detail here, although this information can be found in [**69**]. During a *Scan Hallway* routine, ORB sends line segments of range data along with their accompanying labels to SPOTT for incorporation in the map database. Communication times with SPOTT vary and depend on a number of factors, including the network load and the number of tasks being executed by SPOTT. For the size of data being sent by ORB, the average time taken for packing is 0.169 msec, and the average time to send this packet is 4.101 msec[**69**].

ORB obtains and processes a range scan in approximately 500 msec. With the camera selection contributing another 200 msec, ORB returns a range scan with the corresponding labels every 700 msec or so. So when the rangefinders are stationary (i.e., pointing out towards the sides of the robot), ORB returns range data at a rate of about 1.4 frames per second. For scans which require the rangefinder to be repositioned, an additional 1.2 sec is needed to allow for this movement. Since most of the constraints are hardware related, (i.e., rangefinder selection and PTU movement), ORB's performance cannot be improved without changing the hardware setup. However, ORB's response time is adequate for supplying SPOTT with sensor updates during a navigational task. Since SPOTT aims to travel at the human walking pace of 1 m/sec[**69**], ORB's return rate of approximately 1 frame of range data per second is quite suitable with QUADRIS operating at a range of 5 metres.

2. Structural Objects

Structural objects are located and recognized during the execution of a *Scan Hallway* routine. The *Scan Hallway* task begins with an initialization procedure whereby ORB determines where the walls are located to either side of the robot. As SPOTT navigates the robot down the hallway, the knowledge of each wall is continually monitored and updated. The walls should be found at approximately the same distance from the robot as that found during initialization, within an allowable threshold. Each line segment that is found to fall along a wall is sent to SPOTT for incorporation into the map database, along with a "wall" label. Doorways are the other structural units in a hallway. They are detected when an opening is found along a wall, whose width is determined be within the range specified in the door model for this office floor. Objects found in the hallway that are deemed to be neither part of a wall nor a doorway are labelled "unknown objects" (also referred to as obstacles).

2.1. Walls

The first step in scanning a hallway is to locate where the walls are positioned with respect to the robot. ORB obtains this information by taking readings from each BIRIS sensor, which are pointed at the walls on either side of the robot at initialization. A BIRIS scan of a wall will produce a single line segment of range data after segmentation. These readings provide the distance of each wall to the robot, forming ORB's knowledge of the walls, which is called the "wall model". These initial line segments and their "wall" labels are sent to SPOTT and displayed for the user, as seen in Figure 6.1.

As SPOTT navigates the robot down the hallway, ORB continually updates the map database. Each new reading of a wall is sent to SPOTT, and is also used to update the "wall model". These updated range readings should fall within an allowable threshold of the wall model distances found during the initialization routine. The wall updating procedure is depicted in Figure 6.2.

As soon as a reading does not match the wall model, the wall is assumed to be no longer visible. If the readings are found closer than the expected position of the wall in the hallway, then an "unknown object" or obstacle has been found. If the readings lie further than the wall, then an opening is in view. This opening could either be a doorway or a hallway. ORB's recognition of doorways is now examined.

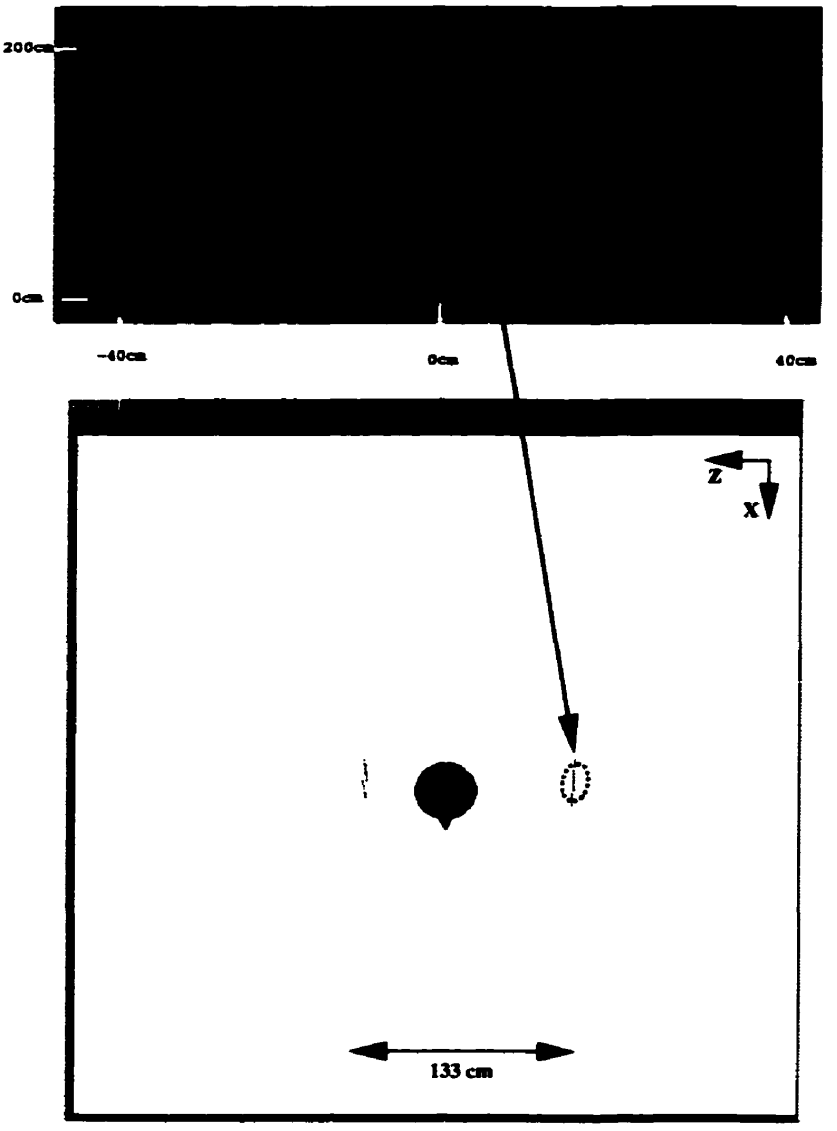


FIGURE 6.1. **Wall Initialization Procedure.** The initialization procedure localizes the walls to either side of the mobile robot. ORB takes a reading from each BIRIS sensor to obtain the distance of each wall to the robot. A BIRIS range scan of a flat wall will produce a single line segment of range data after segmentation. A scan of the right wall taken during initialization is outlined above. This scan is related to the resulting map of the hallway produced after initialization is complete. The distance of the right wall to the robot in the Z direction is 75 cm, with the left wall at 58 cm from the robot. This information constitutes ORB's knowledge of the walls in this hallway, and is referred to as the "wall model".

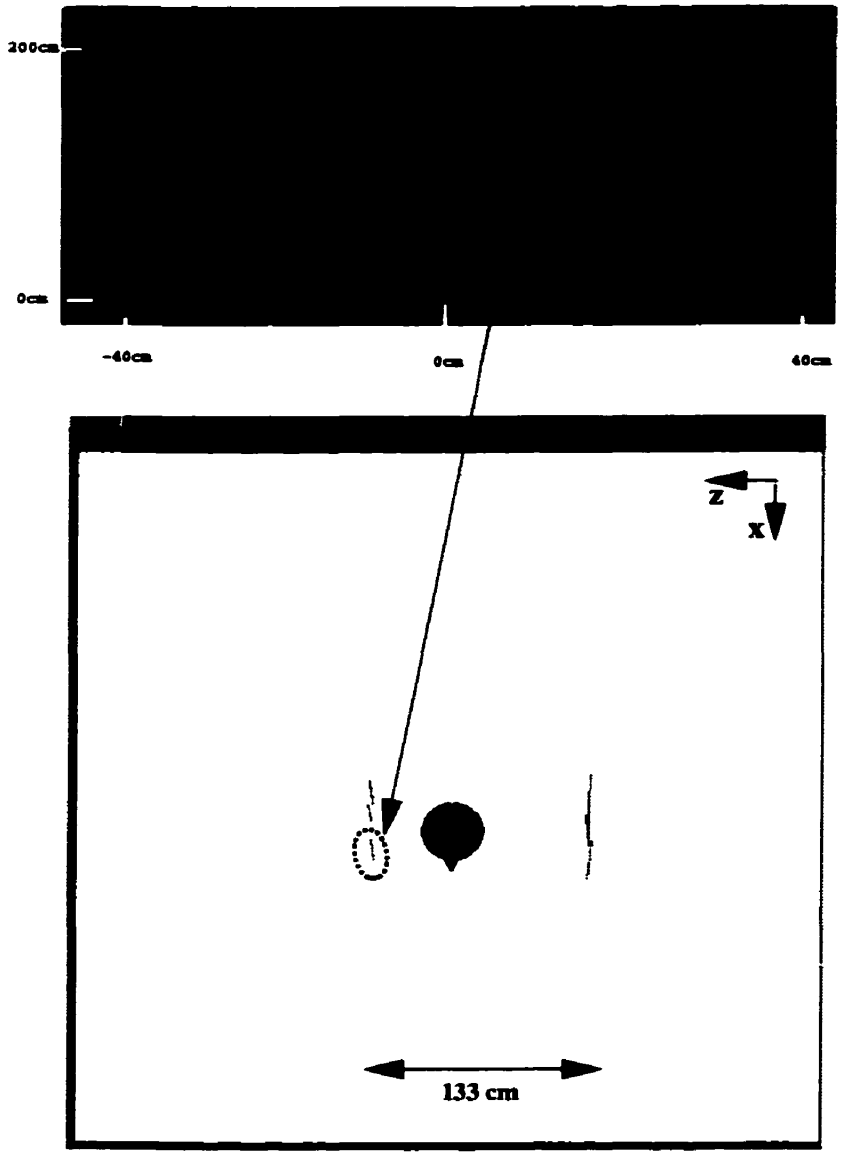


FIGURE 6.2. **Wall Update Procedure.** Once the walls have been localized in the initialization stage, subsequent scans of the walls are compared to the "wall model". As SPOTT navigates the robot down the hallway, ORB continually scans each wall to update the map database. Each new reading of a wall should lie within an allowable threshold of its *wall distance* as stored in the "wall model". The latest reading that satisfies this criteria is used to update the "wall model". The range scan outlined above is one of ORB scanning further up the left wall in a search for openings that may be forthcoming. This scan of the left wall produced a single line segment whose distance from the robot in the Z direction is 56 cm. The reading falls within the allowable threshold (7 cm) of the *Wall_Distance* value of that wall, which is 58 cm, as found in the initialization stage. This range scan is then concluded to be part of the left wall. This wall updating procedure continues until a scan is found to not match the "wall model". In this case, either an "unknown object" has been found in the hallway, or an opening (e.g., a doorway) has been found along a wall.

2.2. Doors

Doorways can only be detected once the walls they are embedded in have been localized. Doors can be found in one of three states, (i) closed, (ii) partially open, or (iii) open. A doorway is basically an opening in a wall, the width of which can be modelled a priori. Doorway widths are generally uniform across an office floor, and can be used to differentiate doorway openings from hallway openings. The doorway widths on the CIM office floor vary from 74 cm, for an office door, to 94 cm for a laboratory doorway. Once an opening has been detected along a wall, ORB scans across it to determine its width. If the width of the opening falls within the range of known doorway widths for the floor, the opening is considered to be a doorway. The boundaries of the doorway used to compute this width are defined as the two points where the surrounding wall ends, and the opening begins. The range profile of the doorway determines what state (i.e., open/partially open/closed) it is in.

2.2.1. *Closed Door*

Closed doors in the CIM office space are 10 cm from the surrounding wall surface in the door frame, as seen from the hallway. When first detected, the range scan should show a sharp 10 cm step increase in range from the wall (located at *Wall_Distance*). This 10 cm step remains constant for the duration of the opening. When the wall on the other side of the doorway comes into view, the range readings return to where the surrounding wall is found (*Wall_Distance*), and the process of scanning the doorway is complete. This sequence of scanning a closed door is depicted in Figure 6.3.

2.2.2. *Partially Open Door*

A partially open door is one that is ajar, obstructing the path through the doorway and is also visible from the hallway. The range profile of a partially open door will show a sharp increase at initial detection, followed by a gradual increase or decrease[3] at a constant slope until the other end of the doorway comes into view. The range data gathered from the door itself must lie beyond that of a closed door (10 cm) but not beyond the limits of a partially open door. These limits are based on the area a door covers as it opens into a room. This area is delimited by an arc with a radius equal to the width of the doorway, as was shown in Chapter 3. The scanning of a partially open door is shown in Figure 6.4.

[3]This will depend on which side the sensor starts scanning the doorway from. If it is approaching from the hinged end, the slope of the door will gradually increase to the other side. If the scanning starts at the doorknob end, the slope of the door will decrease to the other side.

2.2.3. *Open Door*

An open door is not visible from the hallway and leaves an unobstructed path for the mobile robot into the room. Range scans through an open door will result in readings of objects that might be visible inside the room. This usually yields empty readings[4] because anything inside the room will likely lie beyond the working range of BIRIS. The door is initially detected as a jump in range much like the previous two scenarios, except that this increase would be the greatest of the three. If there are any range readings picked up from scanning through an open doorway, they would lie beyond that of the limits of a partially open door. The process of scanning an open door is demonstrated in Figure 6.5.

[4] When the scanned objects are located beyond the working limits of BIRIS (5 metres), zero readings are returned, which are referred to here as empty readings.

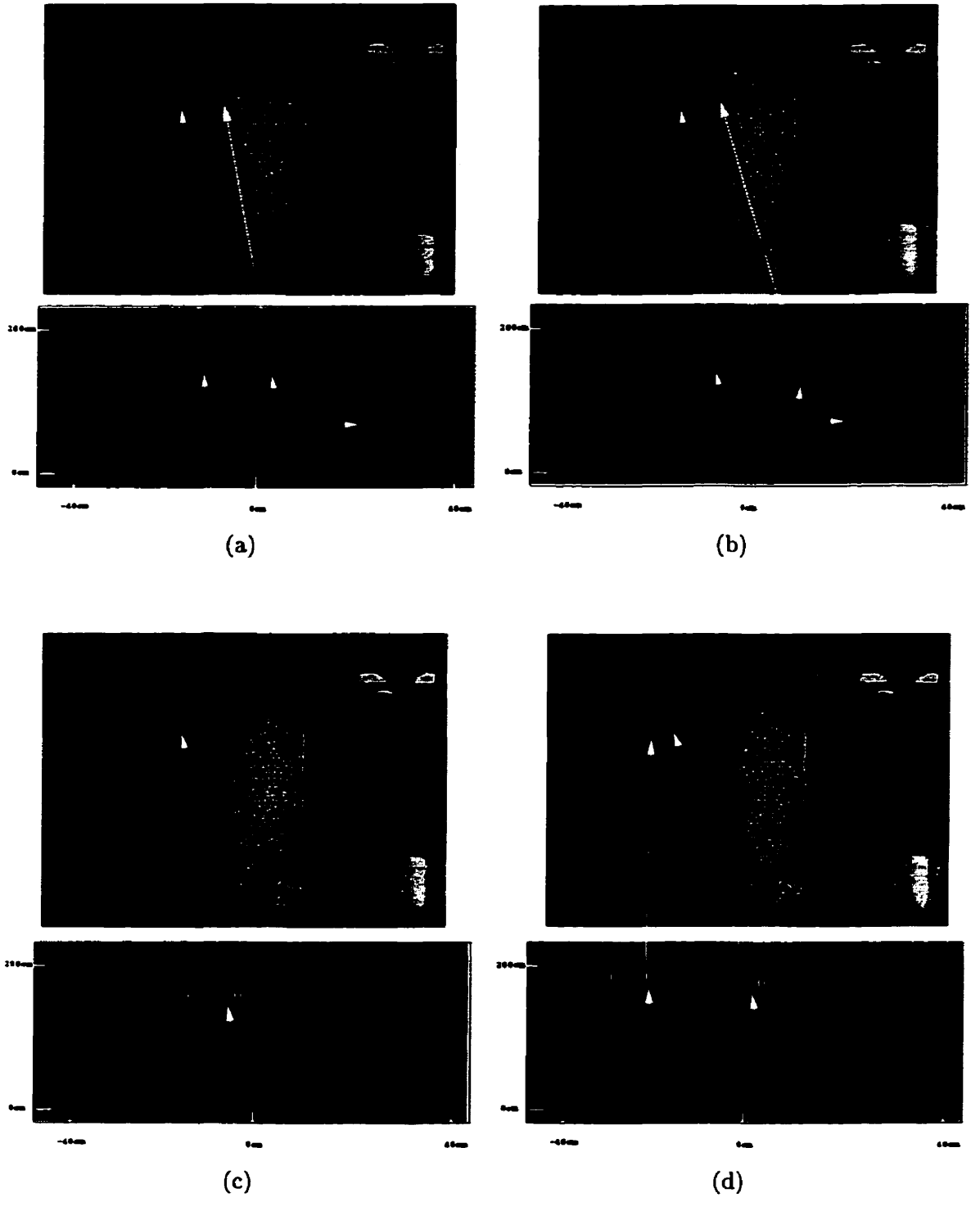


FIGURE 6.3. **A Closed Door Scanning Sequence.** (a) ORB detects the doorway opening by the step increase in range data beyond the wall. ORB notes the position of the beginning of the frame as being the last point on the wall before this step increase. (b)&(c) ORB pans the BIRIS sensor across the opening at 5° increments searching for the other end of the doorway frame. ORB also uses these scans to categorize the state of the doorway. In this case, the constant 10 cm step increase from the wall indicates that the door is closed. (d) The other boundary of the doorway frame is detected. The width of the opening is computed and compared to the range of widths for doorways on the CIM floor. Since the opening is found to be 78 cm wide, ORB concludes that the opening is indeed a doorway, and that the door is *closed*.

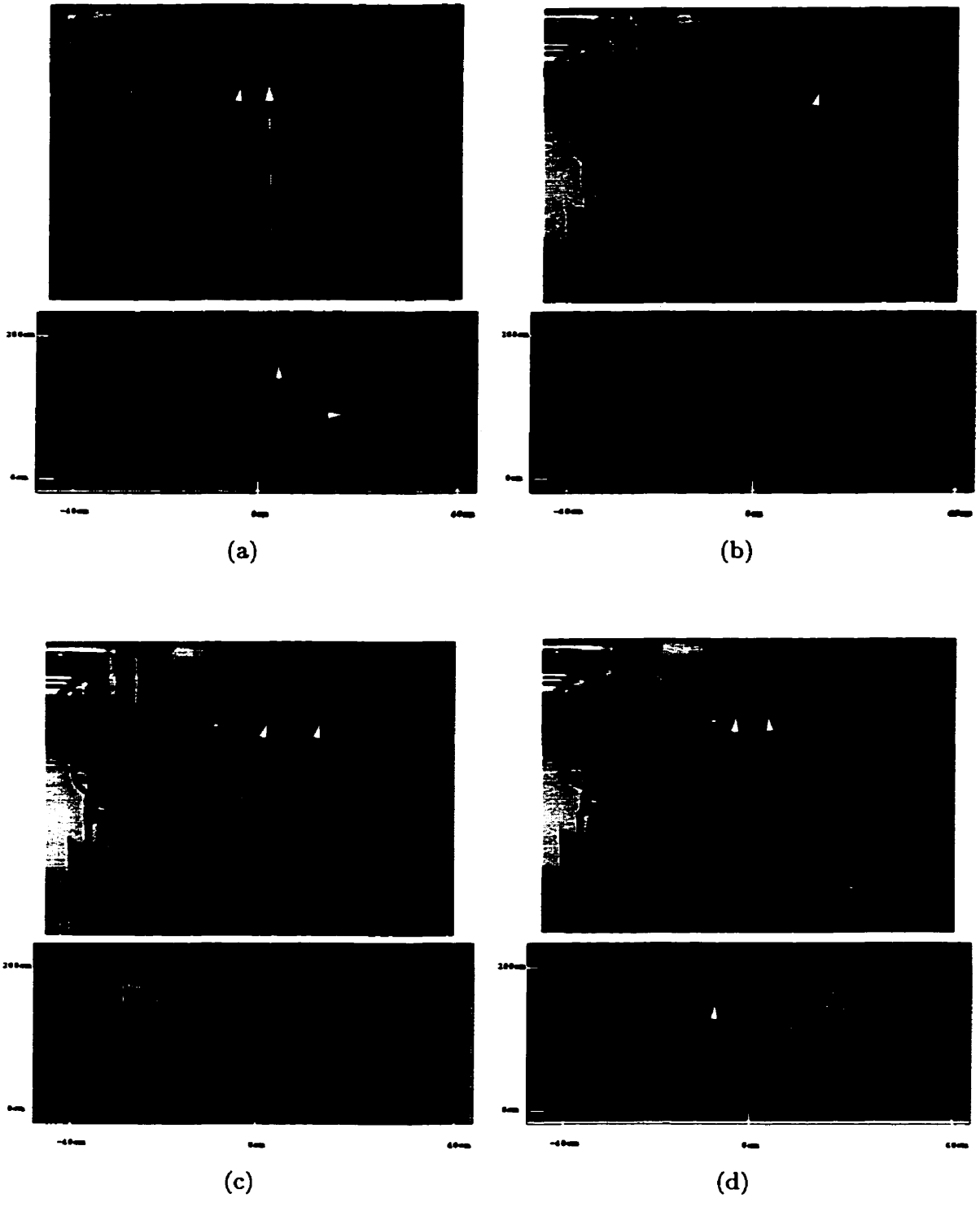


FIGURE 6.4. **A Partially Open Door Scanning Sequence.** (a) ORB detects the doorway opening by an increase in range data well beyond that of the wall. (b) ORB scans across the opening in a search for the other boundary while also compiling data to determine the state of the door. In this case, the range scans are greater than 10 cm, but the door is still in view of the rangefinder, indicating that the door is *partially open*. (c) The doorway frame comes into view as ORB detects a decrease in range data back towards the position of the surrounding wall. (d) The other boundary of the doorway frame is detected as the wall on the other side comes into view. The width of this doorway, 76 cm, conforms to the standards set for doors on the CIM floor. Therefore, ORB concludes that this is a doorway that is *partially open*.

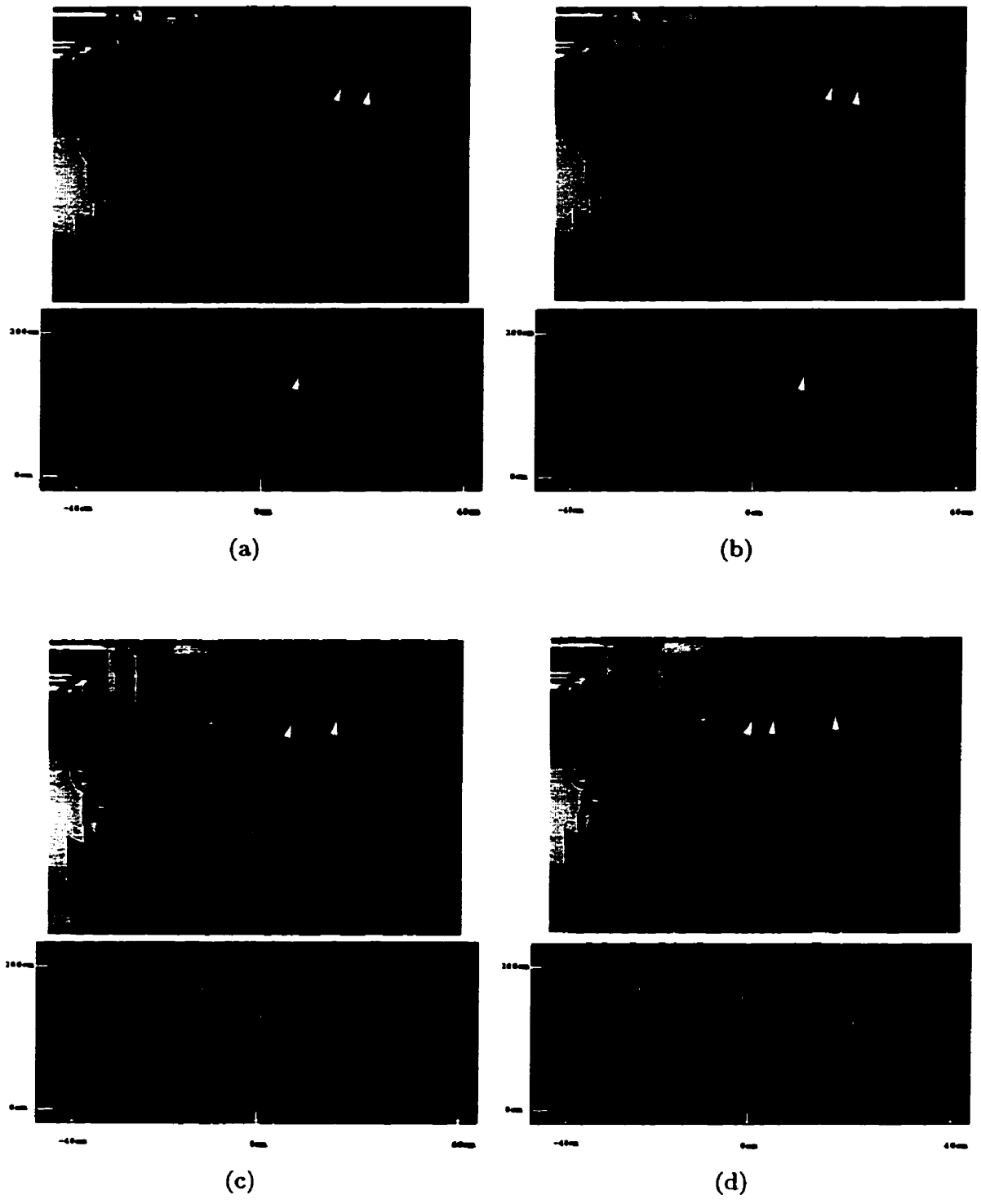


FIGURE 6.5. **An Open Door Scanning Sequence.** (a) ORB detects the doorway opening by a sharp increase in range data, indicating that the wall has come to an end. Either a doorway or a hallway will cause this to happen. The start of the doorway frame is defined as the point where the wall ends and the opening begins. (b) As ORB scans across the opening, no data is found indicating that opening is unobstructed. If it is a doorway, the door is *open*. (c) ORB scans further across the open doorway and the other end of the door frame comes into view. (d) The other boundary of the doorway frame is detected as the juncture between the doorway frame and the wall. The width of this door is found to be 77 cm, which is within the range defined for doorways on the CIM floor.

2.3. Mapping

ORB uses the QUADRIS platform to provide sensor updates for SPOTT. SPOTT relies on sensor readings (i.e., sonar, QUADRIS, infrared proximity and bumper system) to navigate through an office environment. ORB operates in two modes: *Scan Hallway* mode and *Scan Room* mode. SPOTT keeps track of the robot's position and sends a signal to ORB indicating which of the two modes ORB should operate in. *Scan Hallway* mapping results were previously shown to present the initialize and updating routines. Scanning a room is different from scanning a hallway only in that ORB's focus of attention varies slightly for each situation. ORB will concentrate on walls to either side of the robot in a hallway, while in a room, ORB will focus equally to the front as well as to the sides of the robot. A room is usually more spacious than a hallway, so the resulting data will be at a greater range and will hence be noisier. A sample map of a room as produced by ORB is shown in Figure 6.6.

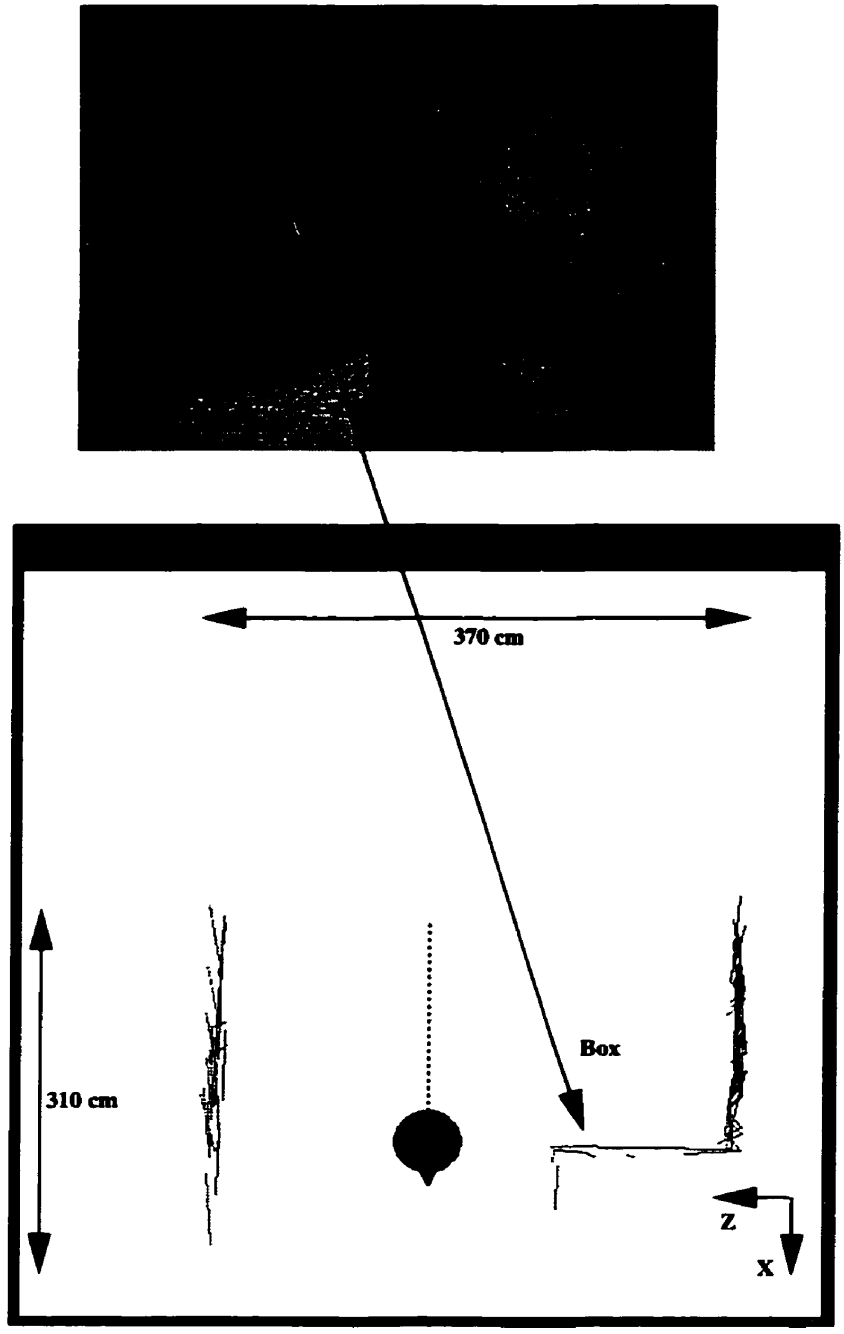
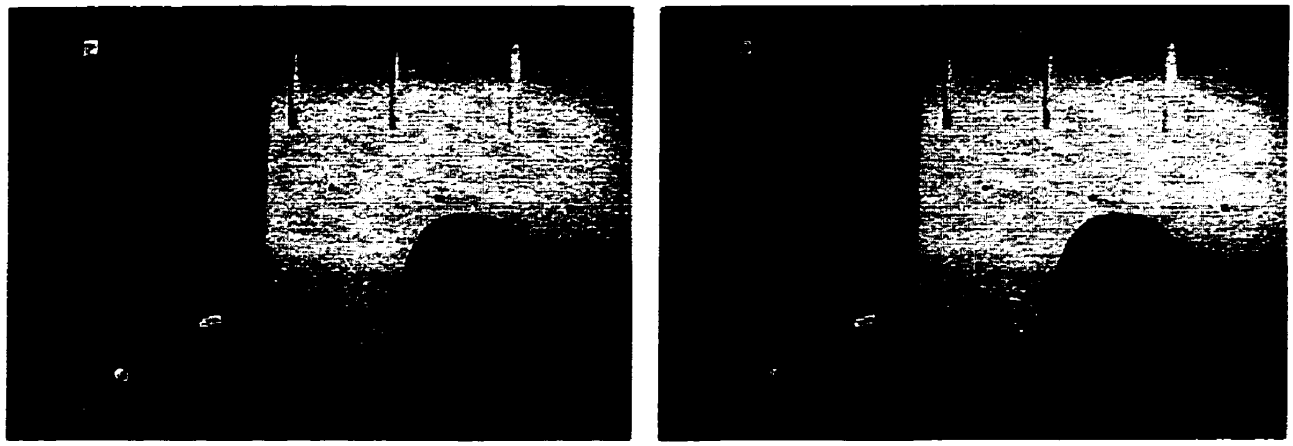


FIGURE 6.6. **ORB's Partial Map of a Room.** This is a sample map of a room as created by ORB. The box in the corner of the room is marked as a visual reference relating the picture to the map. The range readings of the room's walls are slightly noisier than that obtained when scanning a hallway because their distances are closer to the limits of the BIRIS sensor's optimal working range (2 metres). For closer objects like the box, the range readings are much cleaner.



(a) (b)

FIGURE 6.7. **Scanning the Chair Backrest and Seat.** (a) ORB in the process of scanning a chair backrest.(b) ORB scanning the seat.

3. Movable Objects

The movable objects are distinguished by their planar features, and are built at dimensions specified by Architectural Standards[1]. The results of recognizing the movable objects (chairs, tables and desks) are now presented. The range scans of each object are displayed using the software package *rim*[5].

3.1. Chairs

Chairs are identified by the planar features of the seat and backrest. These surfaces are located using knowledge of the chair's dimensions, and are detected by sweeping BIRIS range scans over them. This scanning process is depicted in Figure 6.7. The chair is assumed to have been localized before ORB performs its recognition procedure. The mobile robot is positioned in front of the chair by SPOTT as part of a "Find Chair" task, or manually by the user when testing ORB's functionality. The ideal view is a frontal one with both the backrest and seat visible. A chair can also be recognized from the side, as long as the seat is still in view.

ORB's chair model is shown in Figure 6.8, in a comparison with the set of range data obtained in the experiments shown above in Figure 6.7. ORB searches for a sparse set [6] of BIRIS scan lines to represent the seat and the backrest. The scan lines of the backrest are

[5] *Rim* (Range Image Monitor) was developed at CIM by the Artificial Perception Laboratory.

[6] The number of scan lines required to represent a movable object's planar surface is usually six or seven scans, which was the amount used for these experiments. This number can be selected by the user.

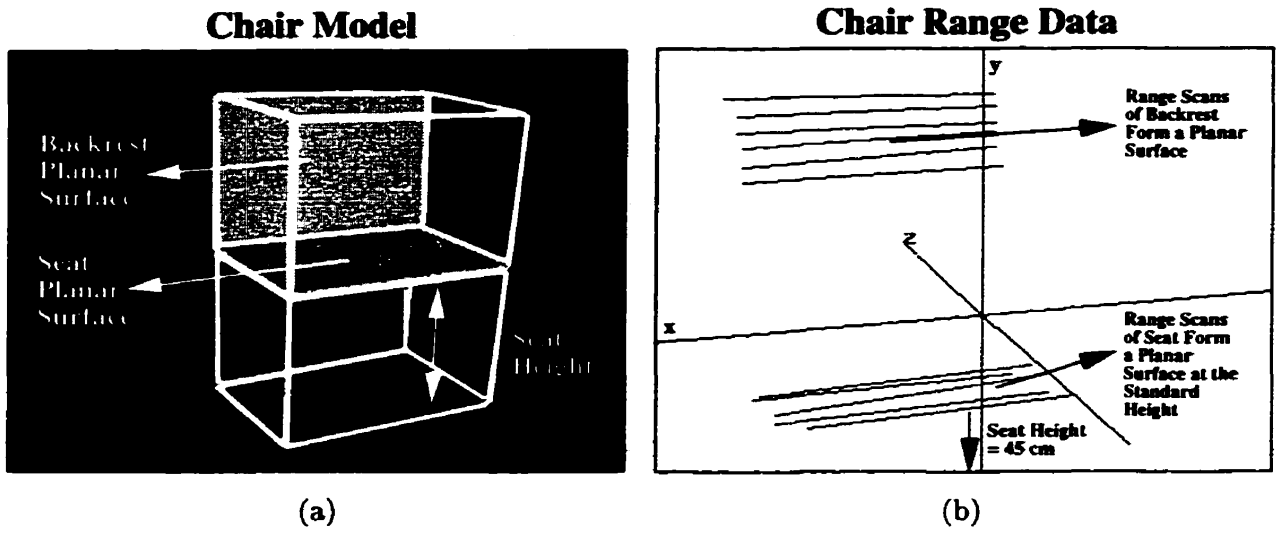


FIGURE 6.8. **The Chair Model and Chair Range Scans.** (a) ORB's chair model consists of planar surfaces representing the seat and backrest. The presence of the seat plane is a necessary condition for a positive identification of a chair. The seat must also be located at the standard height for chairs found in the CIM office space[1]. (b) This is the range data collected from the experiments shown previously in Figure 6.7. Both the backrest and seat surfaces were scanned over a length of 16 cm and a width of 37 cm. The seat is found at a height of 45 cm, which is within the acceptable limits (43 to 48 cm) for chairs in the CIM environment.

found at equal distances to the rangefinder, forming a vertical plane. This plane is within a 7 cm threshold to account for uncertainty in the range readings caused by the material construction of the backrest[7]. Scan lines of the seat are found at the same height from the floor, forming a horizontal plane within a 7 cm threshold. The height of this plane should fall within the range of seat heights expected to be found in the experimental office area. For chairs found in the CIM environment, the range of seat heights vary from 43 cm to 48 cm. This chair seat was found at 45 cm.

For a different view of the chair model and the experimental range data, a frontal view of both is found in Figure 6.9. From this view, the range scans of the seat are seen to fit a plane within the 7 cm threshold. Also, the surface area of the backrest covered by BIRIS scans can be seen. A view from above the chair in Figure 6.10 shows how the backrest scans fit to a plane, within the 7 cm threshold, and what surface area of the seat was covered by ORB.

[7]The chair is made of a darkish material which absorbs some of the laser light, affecting the BIRIS rangefinder's accuracy. Therefore, the threshold allows for some uncertainty in the range readings.

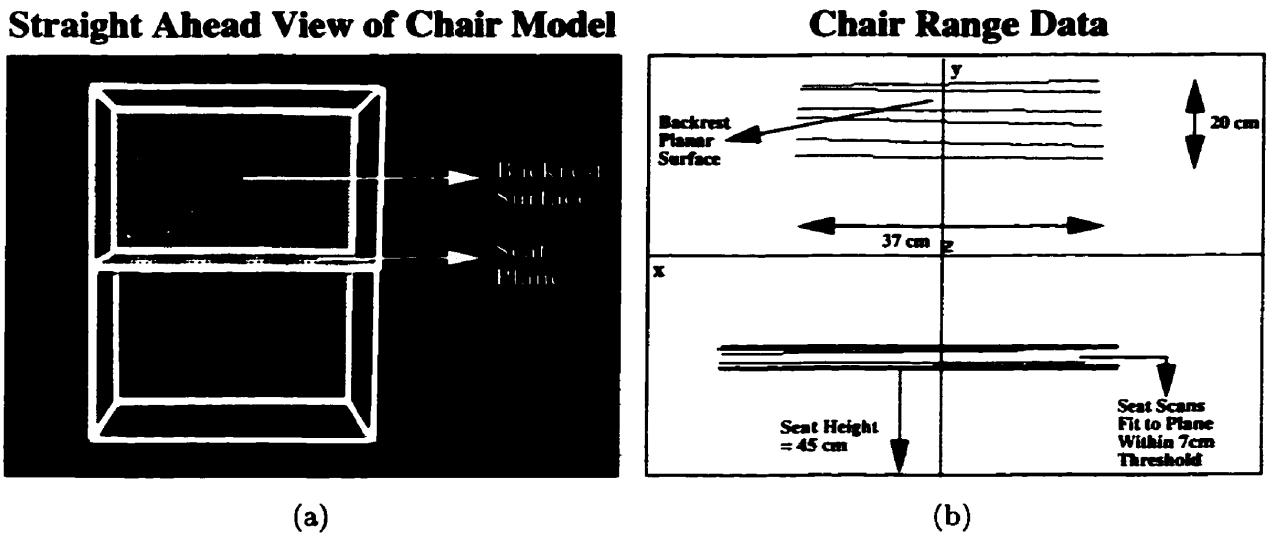


FIGURE 6.9. **Straight Ahead View of Chair.** (a) The chair model as seen from a "straight ahead" point of view (i.e., looking towards the front of the chair). (b) ORB's range data of a chair, as seen from a frontal view, shows the backrest surface represented by six BIRIS laser line scans. These scans form a plane that spans 20 cm in height and 37 cm in width. The seat scans are shown to form a plane within a threshold of 7 cm. The threshold for this experiment was chosen to account for uncertainty in the range data due to the material construction of the chair.

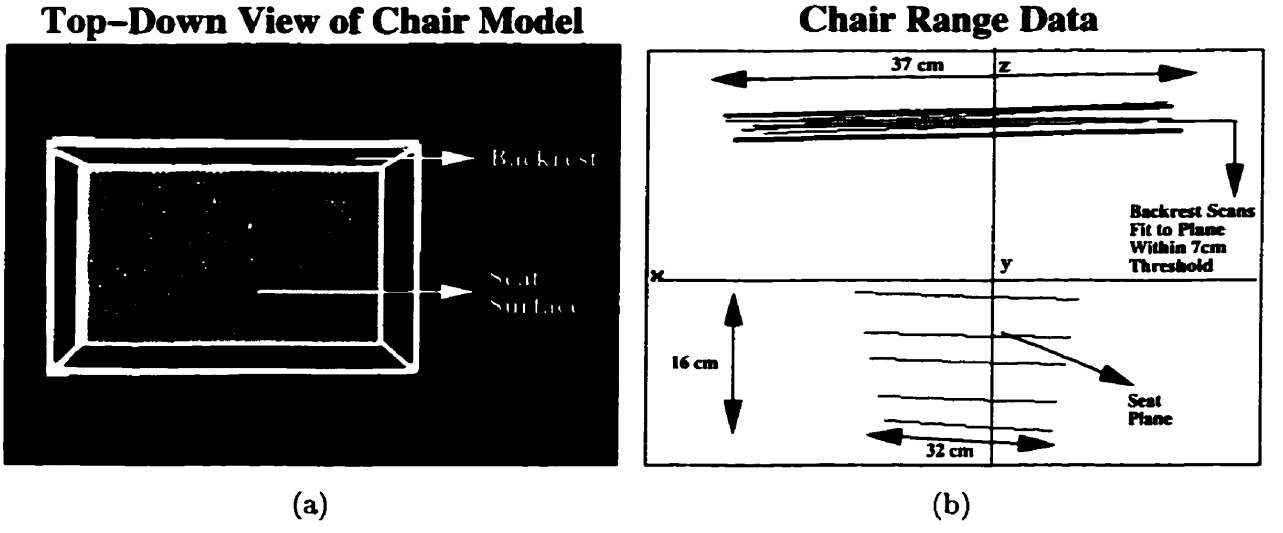


FIGURE 6.10. **Top-Down View of Chair.** (a) The chair model as seen from above looking down (i.e., a view from the ceiling). (b) ORB's chair range data as seen from this angle shows the seat surface represented by five BIRIS scan lines, covering an area of 16 cm in height by 32 cm in width. The backrest range scans fit a plane within the 7 cm threshold. The number of scans chosen to model the seat is arbitrary, and five scans were deemed sufficient for this series of experiments.

FIGURE 6.11. **Scanning a Desktop.** ORB in the process of scanning a desktop.

3.2. Desks

Tables and desks are similar in that they are both comprised of one main component, the tabletop/desktop. This horizontal planar surface is the main feature used in recognizing a desk, and is found at a standard height as defined in [1]. Desks are differentiated from tables by their planar vertical supports. Tables and desks are indistinguishable when viewed *frontally*, since the desks' support surfaces are hidden from view. When a desk is approached from the side however, ORB is able to scan both the desktop as well as a side support, as seen in Figure 6.11.

The desk model is shown in Figure 6.12 and consists of the desktop planar surface, found at a standard height (65 cm to 72 cm[1] in the CIM office space), along with three vertical planar supports. ORB is designed to recognize *movable objects* from a stationary position, since a complete representation of the object is not required for recognition. The set of range data acquired from a position beside a desk, as shown above in Figure 6.11, will only represent the desktop and one side support. This provides sufficient information for ORB to determine that the object is a desk, as can be seen from the results shown in Figure 6.12. The desktop surface was found within the acceptable range of desktop heights (70 cm), and a vertical planar surface was found in the supporting position.

A view from above the desk model and the corresponding range data is shown in Figure 6.13. From this angle, the desktop surface area scanned by ORB can be seen. A set of seven scans is used to represent the desktop surface. The side support forms a vertical

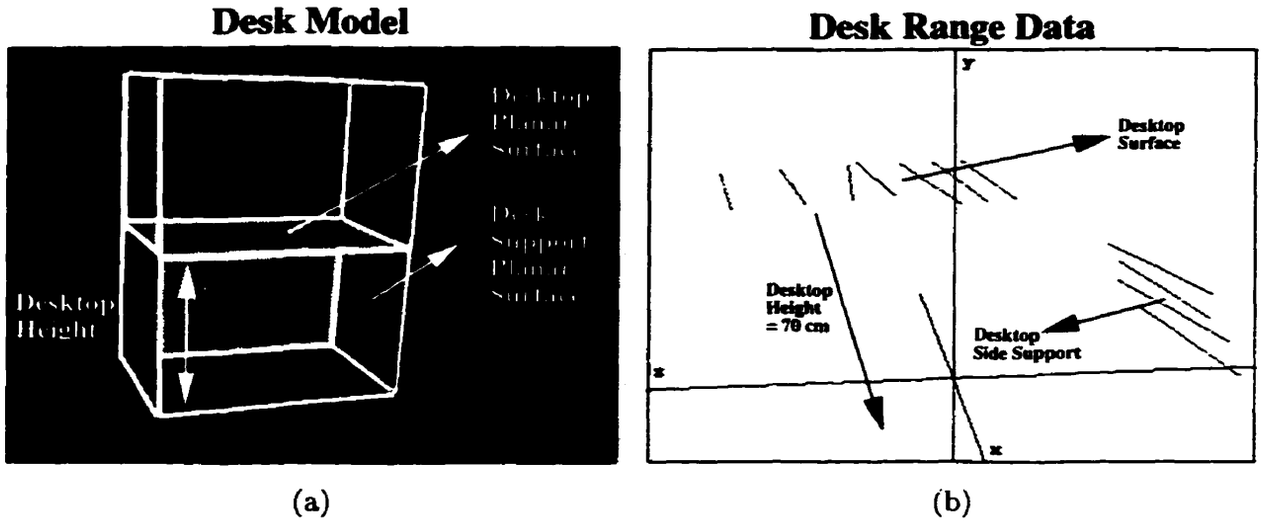


FIGURE 6.12. **The Desk Model and Desk Range Data.** (a) ORB's desk model consists of planar surfaces for the desktop and its three vertical planar supports. The desktop is found at a standard height for desks found in the CIM office area. (b) This is ORB's range data of a desk as collected in the experiment depicted in Figure 6.11. The desktop surface was found at a height of 70 cm, which is within the range (65 cm to 72 cm[1]) defined for desks in the CIM environment. Only one side support can be seen when scanning from this stationary position. The support on the opposite side as well as the third support are occluded from view. A more complete model may be produced if the robot were to move around the desk to collect extra readings. But for the sake of speed and simplicity, ORB is designed to recognize movable objects from a single stationary position.

plane that fits within a 5 cm threshold. The planar surfaces of a desk are smooth and flat, so the threshold chosen to fit a desk's planar surfaces will be more strict than that needed for a chair. ORB assumes that a section of the desktop surface will be clear for scanning. A frontal view of the model and the range data is shown in Figure 6.14. This angle shows how scans of the two planar surfaces, the desktop and side support, fit to planes within a threshold of 5 cm.

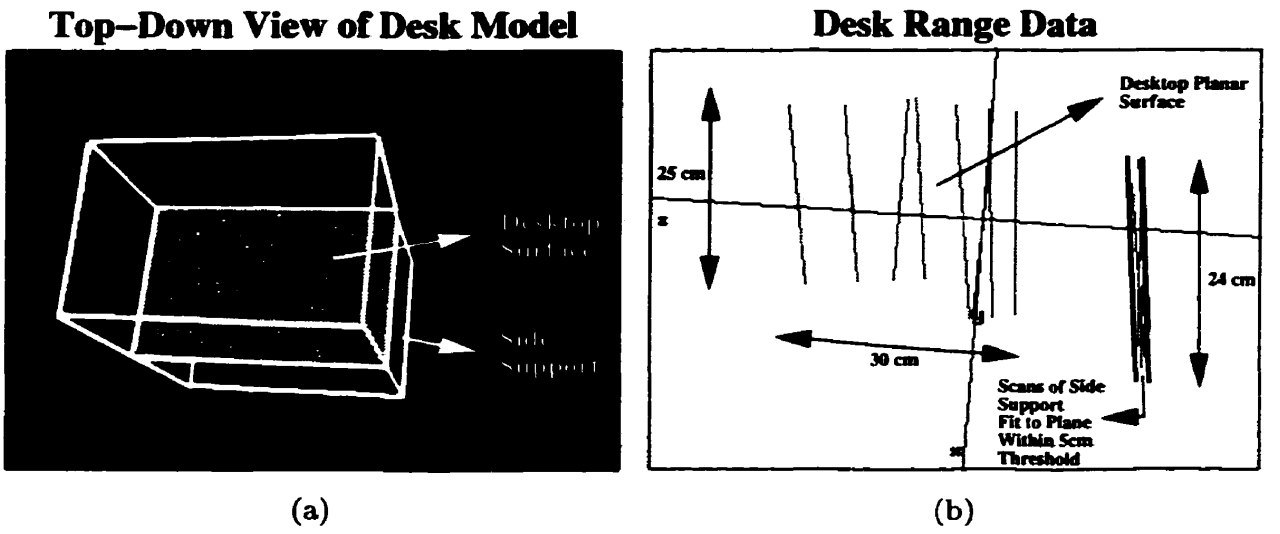


FIGURE 6.13. **Top-Down View of the Desk Model and Desk Range Data.** (a) This is a view of the desk model as seen from above. (b) ORB's range data of a desk from this view shows the desktop surface sweeped by seven scans covering a width of 30 cm and a height of 25 cm. Scans of the side support are shown to fit to a plane within a 5 cm threshold.

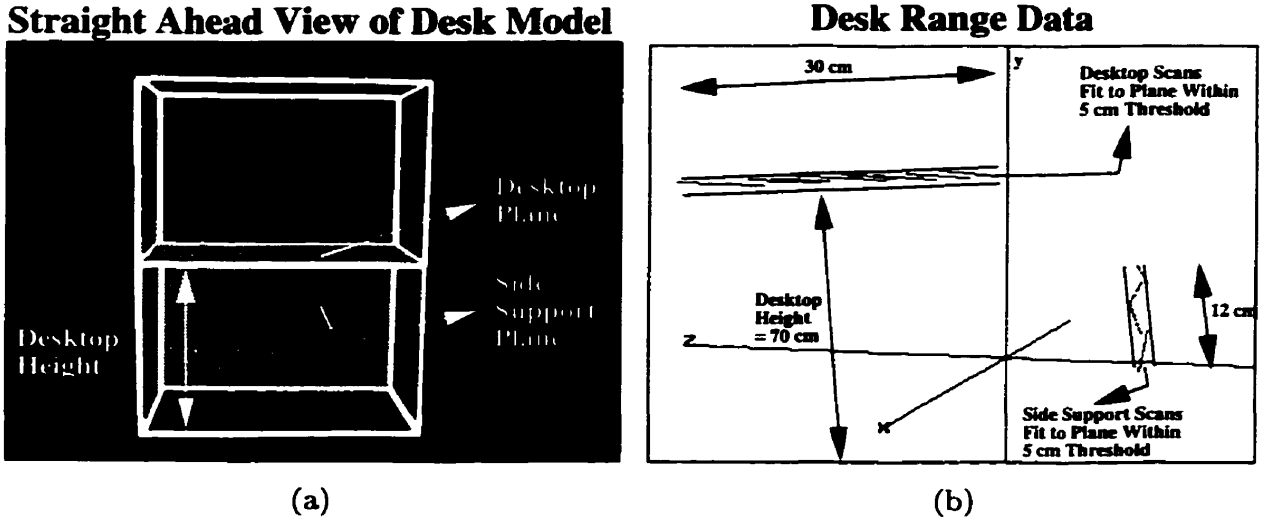


FIGURE 6.14. **Straight Ahead View of the Desk Model and Desk Range Data.** (a) This is the desk model as seen from behind, from the viewpoint of a person seated at the desk. (b) The desk range data from this angle shows how range scans of the two planar surfaces, the desktop and side support, both fit to planes within a 5 cm threshold. The desktop is at a height of 70 cm, which is within the acceptable range for the types of desks found in the CIM environment.

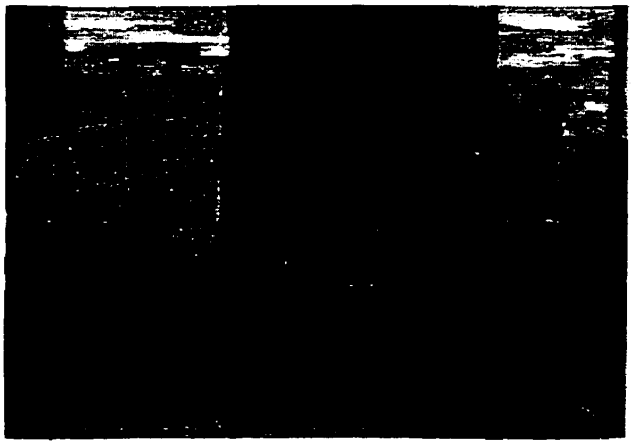


FIGURE 6.15. **Scanning a Tabletop.** A scene of ORB scanning a tabletop in the Mobile Robotics Lab (MRL) at CIM.

3.3. Tables

The main feature of a table is the tabletop, which is a horizontal planar surface found at a standard height (e.g., 65 cm to 72 cm in the CIM environment[1]). A table is also marked by the *absence* of any vertical planar supports underneath its tabletop. A scene of ORB scanning a table is shown in Figure 6.15.

The process of recognizing a table is very similar to that of scanning a desk. ORB searches for a horizontal planar surface, followed by a determination of whether there are vertical planar supports underneath. Both the table model and the range data acquired in the above experiment are shown in Figure 6.16.

The experimental data is presented the same way the desk results were shown, since the data is similar. The top-down view shown in Figure 6.17 provides a look at the area of the tabletop surface scanned by ORB. The frontal view presented in Figure 6.18 demonstrates how the tabletop range scans fit to a plane within a 5 cm threshold. The tabletop plane is found at 72 cm in height from the ground, which is within the range of standard heights defined for tables in the CIM office space. The absence of a planar support indicates that the object is a table rather than a desk.

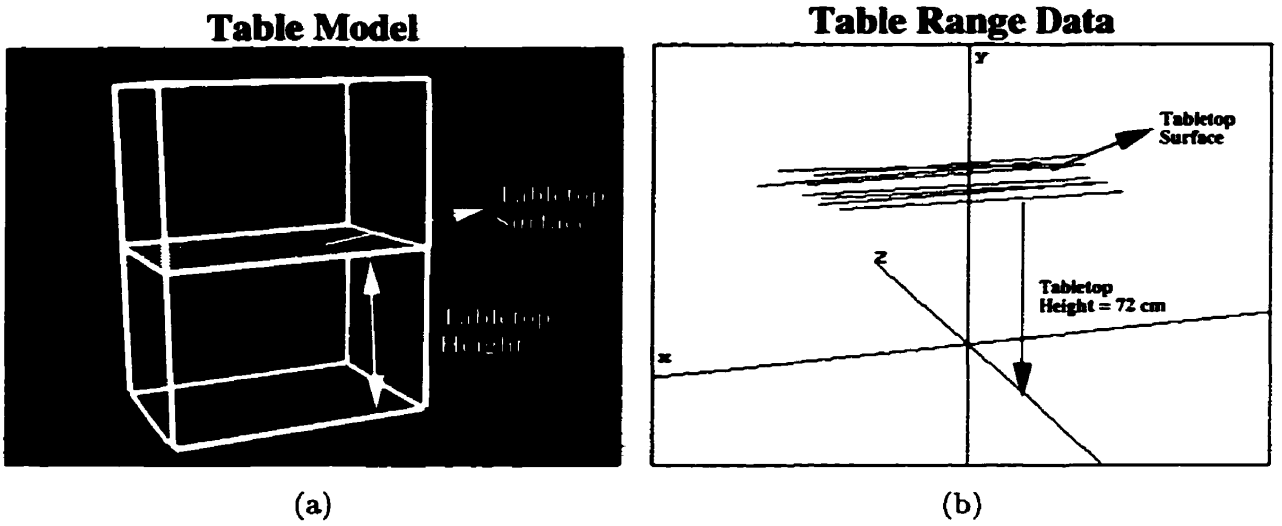


FIGURE 6.16. **The Table Model and Table Range Scans.** (a) ORB's table model consists of a single planar surface representing the tabletop. ORB's movable object models consist solely of planar features, so the legs of a table are not considered. (b) This is a set of range data collected by ORB in the experiment shown in Figure 6.15. The tabletop height was found to be 72 cm, which is within the standard range (65 cm to 72 cm[1]) for tables found at CIM.

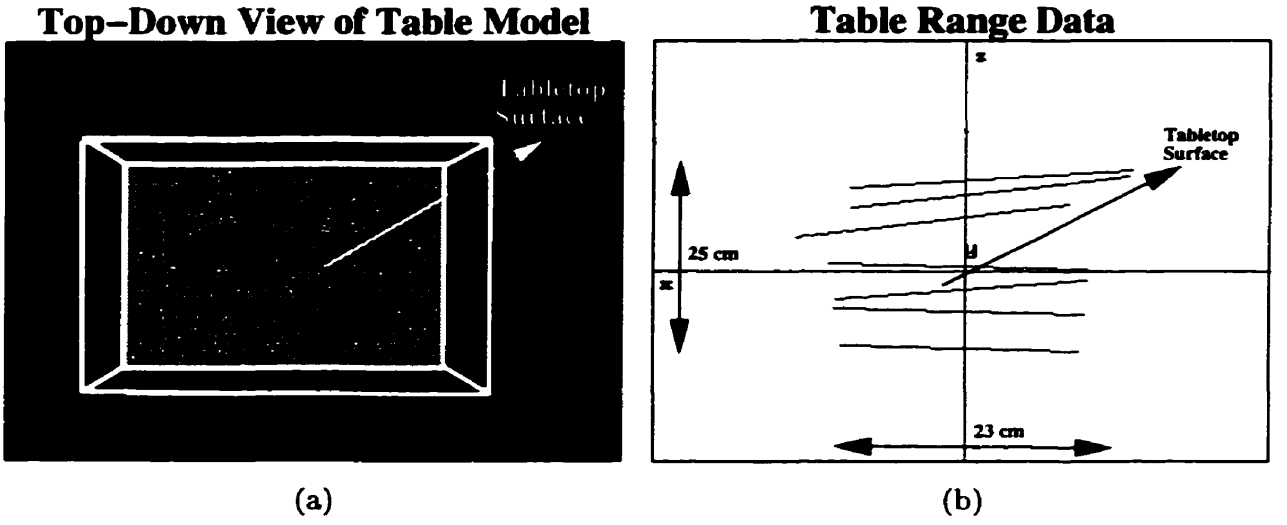


FIGURE 6.17. **Top-Down View of the Table Model and Table Range Data.** (a) This is a view of the table model as seen from above. (b) From this view, the set of range data of the table shows how the tabletop surface is represented by seven BIRIS scans, spanning a length of 25 cm and a width of 23 cm.

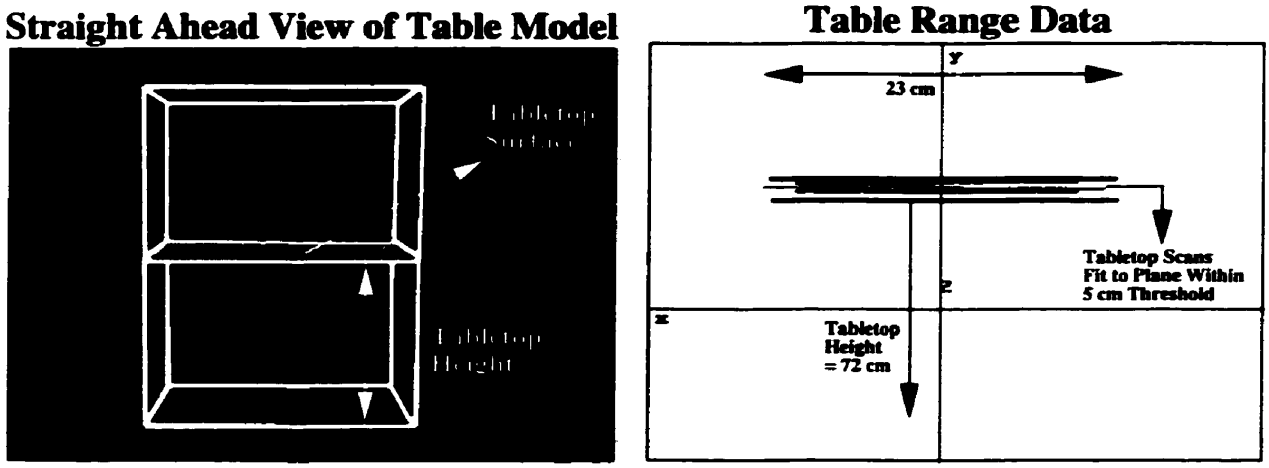


FIGURE 6.18. **Straight Ahead View of the Table Model and Table Range Data.** (a) This is a frontal view of the table model. (b) The table range data as seen from this angle demonstrates how the range scans of the tabletop are fit to a plane within a 5 cm threshold.

4. Discussion

These experiments have demonstrated that ORB is able to accomplish the following:

(i) Provide SPOTT with reliable laser range data, in the form of line segments, to aid in mapping an environment for navigational purposes.

(ii) Recognize and label the *structural* objects in an office area while sensing the surroundings for SPOTT.

(iii) Recognize *movable* objects as part of the "*Find Object*" series of task commands.

These experiments have also unearthed several issues that need to be addressed for future upgrades of ORB. They are outlined as follows:

- The uncertainty of QUADRIS range data in particular situations, as shown in Chapter 5, will cause some problems for ORB. ORB's perception of the environment is only as accurate as the data it receives. QUADRIS is susceptible to producing erroneous results when faced with extreme lighting conditions (e.g., bright overhead indoor lights or direct sunlight) or when scanning dark coloured or unreflective objects. Future work with QUADRIS should concentrate on its robustness in different environmental conditions.
- Some engineering problems that were encountered include the wireless communication problems addressed in Chapter 5. Communications with the mobile robot degraded when there was no line of sight between the transmitter on-board the robot and the receiver. The same problem was found with QUADRIS' video transmitter. The BIRIS video image will occasionally degenerate when line of sight is lost between the transmitter and receiver. ORB has provisions for these instances, but if the problem persists for an extended period of time, ORB is basically "blind" until QUADRIS functions properly again. This problem was investigated for some time and various solutions were proposed. But since it was found that the problems only occurred at the extremities of the CIM floor, it was decided that the experimental setup would function sufficiently well with the receiver placed at a central location on the floor. Further research into this problem is needed if the system is to function in a more elaborate setting. Another practical implementation problem is the mobile robot battery life span, which is typically 5 hours or so. Towards the end of this cycle, the QUADRIS hardware behaves erratically and the range data returned are not as reliable.

- The rate at which ORB returns QUADRIS range data and their corresponding labels (at approximately 1 frame/sec) is adequate for SPOTT's navigational tasks, considering the target speed for the mobile robot is 1 metre/sec. ORB's processing is highly dependent on its communications with QUADRIS hardware components as well as with SPOTT. Inherent delays with the QUADRIS components (e.g., camera selection) are unavoidable and cannot be improved. Communications with SPOTT (e.g., obtaining current robot position) is dependent on the network load. Inexplicable problems in allocating sockets for connections with the daemon processes *cportd* and *ptud*[8] surface occasionally. The only recourse currently in use is to terminate both daemon processes and start anew.

The results presented in this chapter represent typical output to be expected of ORB if the experimental conditions are normal. This means that the QUADRIS platform operates optimally, without returning any erroneous data, and that ORB has no trouble communicating with the QUADRIS hardware or with SPOTT. However, the issues mentioned above illustrate the types of problems that may be encountered during one of ORB's recognition routines. ORB's performance is not guaranteed under unforeseen circumstances such as extremely bright lighting that may corrupt the QUADRIS range data.

While scanning a hallway, the problems observed were connected to environmental conditions that, for the most part, cannot be avoided. The lighting problem previously documented has appeared on occasion, though very rarely. This problem presents itself in two scenarios. The first one occurs when the lighting in the hallway is excessively bright and leaves reflections on the walls and doorways. QUADRIS might produce data that is the result of these lighting effects rather than actual objects. ORB's filtering of these corrupted images, as described in Chapter 5, effectively reduces the occurrence of this phenomenon. This has not been an issue in the CIM environment, but it did occur during an off site experiment at the 1996 IRIS-PRECARN conference, held at the Queen Elizabeth Hotel in Montreal, Quebec. The other scenario takes place when scanning through an *open door*. In small offices, ORB might scan right through the room towards an open window. The sunlight could produce artifacts in the QUADRIS data, as exhibited in chapter 5. These artifacts could mislead ORB into believing that the door is closed or that an "unknown object" is blocking the doorway.

[8]Cportd is the interface to the camera selection hardware and ptud is responsible for controlling the PTUs (Pan-Tilt Units).

Movable objects present another set of problems on top of the hardware and environmental issues described above. QUADRIS data is less reliable when the object being scanned has a dark colour or is made of unreflective material. Tables and desks are usually brown or black, which could create some uncertainty in QUADRIS' range readings. Chairs are often composed of unreflective material for the seat and backrest which could cause some problems for ORB if the QUADRIS range scans are inaccurate. One solution to this dilemma is to use a higher intensity laser to obtain a more powerful signal from these problem surfaces. The laser used for these experiments provided satisfactory results, although some problems were encountered for objects with very dark material (i.e., dark brown desks).

Tables and desks are made to hold paper, pens, and books among other things. Therefore, it is expected that the tops of these objects will be filled with clutter. ORB assumes that a tabletop/desktop surface is clear when scanned, although this is not always possible. ORB relies on SPOTT, or the user, to place the robot in a position whereby ORB has a view of a clear section of the tabletop/desktop. When scanning a chair, ideally both the seat and backrest should be visible. Again, ORB depends on SPOTT or the user to properly position the mobile robot in front of the chair. ORB assumes that the "attention problem" has been solved in each case and that the target object has been localized in the scene. This version of ORB relies on external entities (i.e., SPOTT or the user) to provide this functionality. The next phase in ORB's development would have to give ORB a more active role in localizing its target objects.

CHAPTER 7

Conclusions

This thesis has presented an object recognition system called ORB, for use on a mobile robot. ORB offers specialized recognition capabilities in addition to its primary responsibility of supplying the navigation system (i.e., SPOTT) with laser range data of the environment. ORB was shown to produce range scan updates at a fast enough rate to work with SPOTT, a real-time[1] mobile robot control architecture (see Chapter 6). ORB utilizes the QUADRIS platform to perform its sensing activities. QUADRIS range data is accurate enough to reliably map a hallway and provide reasonable results when scanning a room (see Chapter 6)[2].

ORB operates in two modes. The first concerns the recognition of the *structural objects*, which is done while scanning a hallway. As SPOTT navigates the mobile robot through an office space, ORB updates the map database with line segments of QUADRIS range data[3]. During this mapping procedure, ORB also identifies where the walls and doorways are located (see Chapter 3). ORB recognizes these landmarks using simple 2D models that require only a minimal number of horizontal laser line scans. Walls are located during an initialization stage, in which ORB obtains its knowledge for the "wall model". ORB continually monitors the walls to update the map database. The line segments of range data which match the "wall model" are sent to SPOTT for inclusion in its map, along with a "wall" label. When a range scan is found not to match the "wall model", two possibilities exist. (i) If the range scan indicates that something has been found *in* the hallway (i.e., closer to the robot than the walls are), then this object is labelled as "unknown". It is also referred to as an *obstacle*. (ii) The other scenario is that ORB discovers an opening in the wall. This is detected by a range reading that lies beyond the scanned wall, by a

[1]This means that SPOTT reacts faster then the times taken by changes in the environment.

[2]QUADRIS range data is accurate to +/- 2cm up to a distance of 2 meters. Its accuracy degrades gradually thereafter.

[3]The raw QUADRIS range data, which is an array of 640 range points, is segmented using Ramer's[53] method.

threshold determined a priori (7 cm was used in these experiments). Two types of openings are found along a wall: a doorway or a hallway. They are distinguishable by their widths, since hallways are usually wider than doorways. The doorway model contains the range of doorway widths found on the office floor. ORB scans across the opening to determine if its width falls within the range specified in the doorway model. If so, ORB then determines what state the doorway is in (i.e., open, partially open, or closed) from data obtained from the scans across the breadth of the opening (see Chapter 3 and Chapter 6).

ORB's second recognition mode pertains to the *movable objects* (see Chapter 4). These are furniture items found in all office spaces, namely chairs, tables and desks. These objects are represented by idealized models in which the object's surfaces are modelled by planes. The physical dimensions of these models are based on standards defined in *Architectural Standards*[1]. These models are used to ease the recognition process, since the planar surfaces are easily reconstructed by a minimal set of laser line scans (unusually numbering 5 or 6). The *movable objects* are assumed to have been localized in the scene before ORB performs its recognition routine. ORB scans a *movable object* by centering it in view before searching for its planar surfaces. ORB's focus of attention (i.e., where to position the rangefinder) is determined by the *movable object's* model. Each *movable object* requires a specific scanning routine, as they have unique structures. Recognition is complete once ORB has successfully detected the planar features of the object, as described by its model. ORB's models for each of the *movable objects* are presented in Chapter 4. The experimental results are discussed in Chapter 6.

1. Contributions

The contributions of this thesis relate to the recognition of certain landmarks found in office environments, as performed by a sensing system on-board a mobile robot. These landmarks are categorized as either *structural* or *movable*. The *structural objects* are identified as the mobile robot travels through the hallways of an office space. The recognition of the *movable objects* is part of a set of task commands available for the mobile robot that are issued by SPOTT. An example of one such task is: "Find a Chair". In this case, once an object that is suspected of being a chair has been located[4], ORB is used to scan and determine if it is indeed a chair. The contributions of this work are outlined as follows:

[4]The attention problem, or process of locating the object in the scene, is not addressed in this work. It is assumed the object has been localized before the recognition process has started.

- The use of simple 2D models to represent the *structural objects* is proposed (see Chapter 3). ORB identifies the *structural objects* in an office space by exploiting a priori knowledge of the environment's structural dimensions. ORB introduces a novel approach to recognizing these landmarks in a fast and efficient manner, requiring only a minimal number of horizontal range scan lines.
- A unique method of modelling and identifying the *movable objects* is presented (see Chapter 4). ORB presents an idealized model for the *movable objects* whereby they are represented by a series of horizontal and vertical planes. ORB uses planar surfaces to model these objects in order to simplify the recognition process. ORB only requires a minimal number of horizontal laser range scans (e.g., 5 or 6) to reconstruct each of the object's surfaces.

2. Discussion

The experimentation done for this thesis has shown that ORB is capable of offering the following functionality:

(i) ORB provides an effective means of sensing and mapping an office environment for navigational purposes. Sample maps of a room and hallway as produced by ORB were presented in Chapter 6.

(ii) ORB identifies and labels the *structural objects* in an office area while sensing its surroundings. The results of ORB's scanning of walls in a hallway are shown in Chapter 6. The same chapter also presents ORB's recognition of doorways and its determination of the state of each doorway (i.e., whether its open, closed, or partially open).

(iii) ORB recognizes *movable objects* as part of the "*Find Object*" series of task commands. ORB's range scans of chairs, tables and desks found in the CIM environment are presented in Chapter 6.

The main challenges encountered in implementing the ORB system was in integrating it with SPOTT, as well as in dealing with environmental factors that caused problems for QUADRIS, as described in Chapter 5. Implementation of the ORB system required the incorporation of many software modules (SPOTT, robodaemon, cport, and ptud) and hardware devices (QUADRIS platform, Nomadics mobile robot). The complexity of the system gave rise to some practical implementation challenges. For instance, there were

engineering issues that needed to be addressed, such as the wireless communication problems described in Chapter 5. The QUADRIS video signal and the radio transmissions to the mobile robot would be corrupted occasionally when line of sight was lost between the transmitter on-board the robot and the receiver at the workstation.

ORB's optimal performance occurs when QUADRIS is functioning as expected, without returning any erroneous data, and when communications between the hardware and software modules in the system are properly completed. However, ORB's results are not guaranteed under unforeseen circumstances that corrupt the QUADRIS range data or interfere with the wireless video signals. For example, extremely bright lighting will hamper the QUADRIS data, causing ORB to misinterpret its surroundings. Environmental conditions that have created problems for QUADRIS are documented in Chapters 5 and Chapter 6. *Movable objects* with unreflective surfaces also cause uncertainty in QUADRIS range readings, affecting ORB's ability to properly identify the object. For example, tables and desks of dark brown or black colour attenuate the laser signal, making readings more difficult. Also, chairs whose seat and backrest are made of cushion may also produce less than accurate results.

At this time, ORB assumes that the target *movable object* has been localized in the scene before performing its recognition routine. However, this could lead to problems with occlusion, another factor that ORB must deal with when scanning *movable objects*. When scanning a chair, the seat and backrest must be visible. However, it was shown in Chapter 4 how these surfaces may be occluded from view depending on the pose of the chair with respect to the camera. Tabletop and desktop surfaces are also prone to being covered by clutter like books and coffee mugs. ORB needs a clear view of the surface to complete its task.

The issues outlined above are addressed in the next section concerning potential future work for ORB.

3. Future Work

The following research issues could be investigated to further improve ORB's performance:

(i) For this thesis the attention problem, or act of locating the target *movable object* in the scene, is assumed to have been handled by another process (i.e., SPOTT or the user). The next stage in ORB's development should allow ORB to become more actively involved in locating these objects. One idea would be to add a video

camera to the sensing platform to locate objects using video data. A 3D scene interpreter that was proposed by Sandakly and Giraudon[**56**] is one approach that could be considered, since it has actually been tested on scenes with chairs and tables. Another method that can be investigated, and has been employed to search for geons[**47**], exploits the colour characteristics of the target object. A colour camera is used to track and locate the geon based on its colour. Once localized, a laser rangefinder is used to scan the object. A similar approach can be used to search for *movable objects*, although the colour tracking might be more difficult since it will not be possible to manipulate (e.g., paint) the colours on a *movable object*. A search pattern that can be implemented is based on the assumption that desks are usually positioned up against a wall, and that chairs are placed in front of them. Once the mobile robot has entered a room, the search for desks and chairs should follow along each of the walls.

(ii) Some work could be done on the sensing system, QUADRIS, to improve its robustness in different environmental conditions. QUADRIS works optimally in controlled situations such as that found in the CIM Mobile Robotics Laboratory, but it has been found to produce unexpected results in foreign environments (see Chapter 5)[5] More experimentation needs to be done in order better establish QUADRIS's limitations. Experimentation with different filters and more powerful lasers would be one avenue to pursue. ORB will work with any system that supplies an array of range data, so the introduction of a new or improved sensor platform would benefit as well.

(iii) The wireless communication problems described in Chapter 5 and 6, should be addressed if ORB is to function properly in an elaborate setting. These problems arise when line of sight is lost between the transmitters on-board the robot and their receivers at the workstation. This occasionally affects the QUADRIS video transmissions and renders ORB "blind" until the video feed returns to normal. One approach to this problem would be to install a network of antennas throughout the floor connected to the receiver. This allows the receiver to be in contact with the transmitter no matter where the robot is on the floor. However, this would mean altering the environment a priori, which is not always feasible. The experiments for this thesis were done with the receiver located at a central location on the office floor.

[5]Some of these problems were the result of external lighting conditions or the scanning of unreflective (dark coloured) objects.

The video transmissions were satisfactory with this set-up, except at the extremities of the floor where the signal became somewhat degraded.

(iv) Currently, the bottleneck in terms of ORB's processing times is in the communication between ORB and the QUADRIS hardware (discussed in Chapter 5 and 6). The rate at which ORB returns QUADRIS range data, at approximately 1 frame/sec, is adequate for SPOTT's navigational tasks considering the target speed for the mobile robot is 1 metre/sec. But if the hardware system could be improved to speed up processing, this should be pursued.

(v) Communications with SPOTT is critical as information is constantly being passed between ORB and SPOTT. Currently, these communications are dependent on the message passing implementation called PVM (see Chapter 5). Alternative means of communication between processes, such as shared memory or threads, could be investigated to improve on PVM's performance [**69**].

(vi) Further additions to the *movable object* model database can be made to make it more complete. Objects that meet *movable object* criteria include cabinets and bookcases. These objects are representable by a set of planes in the general object model (see Chapter 4).

(vii) As described in Chapter 6, clutter is inevitably found on tables and desks in the form of books, papers and magazines. At present time, ORB needs a clear view of the tabletop/desktop surface in order to scan and recognize these objects. To give ORB more flexibility when working in a given office environment, it should be able to take this clutter into account. This can be achieved by allowing for an extra threshold that accounts for any objects that may be found on the tabletop/desktop. As long as the main tabletop/desktop surface is found at the standard height, a given uncertainty for books and other clutter would be allowed.

4. Summary

This work introduces an object recognition system for use on a mobile robot, called ORB. ORB functions mainly as a sensing system that utilizes the QUADRIS platform to scan the surroundings and provide laser range data for the navigation system called SPOTT. ORB performs additional sensory and perceptual tasks that aid SPOTT in completing its navigational responsibilities. These include the recognition of *structural* and *movable* objects.

REFERENCES

[1] C. Ramsey, H. Sleeper, R. Packard, Editor. *Architectural Standards*, Seventh Edition. John Wiley & Sons, Inc.

[2] F. Arman and J.K. Aggarwal. Model-Based Object Recognition in Dense-Range Images – A Review. In *ACM Computing Surveys*, Vol. 25, No. 1, March 1993. pp. 5-43.

[3] R. Bajcsy, L. Lieberman. Texture Gradient as a Depth Cue. In *Computer Graphics and Image Processing*, Vol. 5, pp. 52-67, 1976.

[4] P.J. Besl. Active Optical Range Imaging Sensors Springer-Verlag, 1989

[5] I. Biederman. Human Image Understanding: Recent Research and a Theory. In *Computer Vision, Graphics, and Image Processing*, 32:29-73, 1985.

[6] J. Borenstein, H.R. Everett, and L. Feng. Navigating Mobile Robots, Systems and Techniques. A.K. Peters, Ltd., 1995

[7] H. Blaasvaer, P. Pirjanian, H. Christensen. AMOR - An Automous Mobile Robot Navigation System. In *1994 IEEE International Conference on Systems, Man, and Cybernetics*, San Antonio, Texas, October 2-5, 1994. pp. 2266-2771.

[8] F. Blais, M. Rioux, and J. Domey. Optical Range Image Acquisition for the Navigation of a Mobile Robot. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, California, April 1991. pp. 2574-2580.

[9] F. Blais, M. Rioux. Sensing and Environment Perception for a Mobile Vehicle. In *SPIE Vol. 1480, Sensors and Sensor Integration*, April 4, 1991, Orlando, Florida. pp. 94-101.

[10] F. Blais, M. Lecavalier, J. Domey, P. Boulanger, and J. Courteau. Application of the BIRIS Range Sensor for Wood Volume Measurement. In *National Research Council Canada Technical Report*, October 1992.

[11] J.P. Brady, N. Nandhakumar and J. Aggarwal. Recent Progress in the Recognition of Objects from Range Data. In *9th International Conference on Pattern recognition*, Rome, Italy, November 14-17,1988.

[12] R.A. Brooks. Visual Map Making for a Mobile Robot. In *Proceedings of the 1995 IEEE International Conference on Robotics and Automation*, St. Louis, Missouri, March 25-28, 1985. pp. 824-829.

[13] R.A. Brooks. A Robust Layered Control System For A Mobile Robot In *IEEE Journal of Robotics and Automation.*, Vol.RA-2, No.1, March 1986.

[14] J. Budenske, M. Gini. Why Is It So Difficult For a Robot To Pass Through a Doorway Using Ultrasonic Sensors. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, San Diego, CA., May 8-14, 1994. pp. 3124-3129.

[15] R. Chatila, J.P. Laumond. Position Referencing and Consistent World Modeling for Mobile Robots In *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, St. Louis, Missouri, March 25-28, 1985. pp. 138-145.

[16] K. Cheng, T.S. Collett, A. Pickhard, R. Wehner. The Use of Visual Landmarks by Honeybees: Bees Weight Landmarks According to their Distance from the Goal. In *Journal of Comparative Physiology A*, Vol. 161, pp. 469-475.

[17] C.H. Chien, J.K. Aggarwal. Model Construction and Shape Recognition from Occluding Contours. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11, No. 4, April 1989. pp. 372-389.

[18] R. Chin, C. Dyer. Model-Based Recognition in Robot Vision. In *Computing Surveys*, Vol.18, No.1, March 1986. pp. 67-108.

[19] D. Coombs, K. Roberts. Centering Behavior Using Peripheral Vision In *Proceedings of 1993 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, New York City, New York, June 15-18, 1993. pp. 440-445.

[20] I.J. Cox and G.T. Wilfong. *Autonomous Robot Vehicles.* Springer-Verlag, 1990

[21] J. Crowley. Dynamic World Modeling for an Intelligent Mobile Robot Using a Rotating Ultra-Sonic Ranging Device. In *Proceedings of the 1995 IEEE International Conference on Robotics and Automation*, St. Louis, Missouri, March 25-28, 1985. pp. 128-135.

[22] G. Dudek, M. Jenkin, E. Milios, D. Wilkes. Reflections on Modelling a Sonar Range Sensor. *Technical Report CIM-2-9. Centre for Intelligent Machines*, McGill University, Montreal, Quebec, Canda.

[23] G. Dudek and M. Jenkin. A Multi-Layered Distributed Development Environment for Mobile Robotics. In *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS-3).* February 1993. pp. 542-550.

[24] M. El-Gamal. *Long Range, Accurate and Practical Algorithms for the BIRIS Range Finder.* Unpublished technical report, Centre for Intelligent Machines, McGill University, 1996.

[25] S.P. Engelson, D. McDermott. Error Correction in Mobile Robot Map Learning In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation,* Nice, France, May 1992. pp. 2555-2560.

[26] H.R. Everett. Survey of Collision Avoidance and Ranging Sensors for Mobile Robots. In *Robotics and Autonomous Systems*, Vol.5, No.1, May 1989. pp. 5-67.

[27] H.R. Everett. *Sensors for Mobile Robots: Theory and Application.* A.K. Peters, 1995.

[28] M. Ezzati. *Fast Image Segmentation Using Stereo Vision* Masters thesis, McGill University, Dept. of Electrical Engineering, September 1995.

[29] A. Geist, A. Beguelin, J. Dongarra, W. Jian, R. Manchek, V. Sunderam. *PVM: Parallel Virtual Machine - A Users' Guide and Tutorial for Networked Parallel Computing.* MIT Press, 1994.

[30] J. Gonzalez, A. Ollero, A. Reina. Map-Building for a Mobile Robot equipped with a 2D Laser Rangefinder. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, San Diego, CA., May 8-14, 1994. Vol.3, pp. 1904-1909.

[31] S. Gordon, W. Seering. Locating Polyhedral Features From Sparse Light-Stripe Data. In *Proceedings of the 1987 IEEE International Conference on Robotics and Automation,* pp. 801-806.

[32] P. Grandjean, M. Ghallab, E. Dekneuvel. Multisensory Scene Interpretation: Model-based Object Recognition In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, CA., April 9-11, 1991.

[33] B.K.P. Horn. *Robot Vision.* MIT Press. Cambridge, Massachusettes. 1986.

[34] M. Idesawa, T. Yatagai, T. Soma. A Method for Automatic Measurement of Three-Dimensional Shape by New Type of Moire-Fringe Topography. In *Proceedings of the Third International Joint Conference on Artificial Intelligence.*, Coronada, CA. Nov. 8-11, 1976. pp.708-712.

[35] K. Ikeuchi and T. Kanade. Modeling Sensors: Toward Automatic Generation of Object Recognition Program. In *Computer Vision, Graphics, and Image Processing*, 48:50-79, 1989.

[36] R.A. Jarvis. A Perspective on Range Finding Techniques for Computer Vision. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-5, No. 2, March 1983. pp. 122-139

[37] J.J. Koenderik, A.J. Van Doorn The Internal Representation of Solid Shape with Respect to Vision. In *Biological Cybernetics* Vol. 32, pp. 211-216.

[38] E. Krotkov. *Active Computet Vision by Cooperative Focus and Stereo.* Springer-Verlag. 1989.

[39] B. Kuipers, R. Froom, W. Lee, and D. Pierce. The Semantic Hierarchy in Robot Learning In *Robot Learning.* Kluwer Academic Publishers. 1993, pp. 141-170.

[40] S. Lang, L. Korba, F. Blais, and M. Lecavalier. Characterization and Testing of the BIRIS Range Sensor. In *IEEE Instrumentation and Measurement Technology Conference*, Orange County, California, May 18-20, 1993. pp. 459-464.

[41] J. Leonard, H.F. Durrant-Whyte. Dynamic Map Building for an Autonomous Mobile Robot. In *The International Journal of Robotics Research*, Vol. 11, number 4, August 1992. pp. 286-298.

[42] M.D. Levine. *Vision in Man and Machine.* McGraw-Hill, 1985.

[43] P. Mackenzie, G. Dudek. Precise Positioning Using Model-Based Maps. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, San Diego, CA., May 8-13, 1994, Vol.2, pp. 1615-1621.

[44] P. Mackenzie. An Overview of the QUADRIS System Unpublished technical report, Centre for Intelligent Machines, McGill University, 1996.

[45] D. Marr, T. Poggio. Cooperative Computation of Stereo Disparity. In *Science*, 194:pp.283-287.

[46] D. Marr, T. Poggio. A Computational Theory of Human Stereo Vision In *Proceedings of the Royal Society of London*, Vol. B204, pp. 301-328, 1979.

[47] R. Ng.. *Thesis in Preparation* M. Eng. thesis, McGill University, Dept. of Electrical Engineering, July 1997.

[48] *Nomadic Host Software Development Environment 1993.* Nomadic Technologies Inc., 2133 Leghorn Street, Mountain View, CA.

[49] A.P. Pentland. Recognition by Parts. In *Proceedings of the First International Conference in Computer Vision.*, London, England. 1987. pp.612-620.

[50] K. Prazdny. Motion and Structure from Optical Flow. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence.*, Tokyo, Japan, 1979. pp. 702-704.

[51] A. Geist, A. Beguelin, J. Dongarra, W. Jian, R. Manchek and V. Sunderam. *PVM: Parallel Virtual Machine - A Users' Guide and Tutorial for Networked Parallel Computing* MIT Press, 1994

[52] S. Qiuang, R. Mohr, K. Tombre. Recognizing and Locating Polyhedral Objects From Sparse Range Data. In *9th International Conference on Pattern Recognition*, Rome, italy, November 14-17, 1988. pp. 104-106.

[53] U. Ramer. An Iterative Procedure for the Polygonal Approximation of Plane Curves. In *Computer Graphics and Image Processing*, Vol. 1, 1972. pp. 244-256.

[54] M. Rioux, F. Blais, J. A. Beraldin, and P. Boulanger. Range Imaging Sensors Development at NRC Laboratories. In *Proceedings of the Workshop on Interpretation of 3D Scenes*, Austin, Texas, November 27-29, 1989. pp. 154-160.

[55] D. Rosenberg, M.D. Levine, S.W. Zucker. Computing Relative Depth Relationships from Occlusiopn Cues. In *Proceedings of the Fourth International Joint Conference on Artificial Intelligence.* Kyoto, Japan. Nov. 7-10, 1978. pp. 765-769.

[56] F. Sandakly, G. Giraudon. Reasoning Strategies for 3D Object Detection. In *Proceedings of International Symposium on Computer Vision*, Coral Gables, Florida, November 21-23, 1995. pp. 557-562.

[57] J. Santos-Victor, G. Sandini, F. Curotto, S. Garibaldi. Divergent Stereo for Robot Navigation: Learning from Bees. In *Proceedings of 1993 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, New York City, New York, June 15-18, 1993. pp. 434-439.

[58] K. Sarachik. Visual Navigation: Constructing and Utilizing Simple Maps of an Indoor Environment. Technical Report 1113. MIT Artificial Intelligence Laboratory, Massachusetts Institute of Technology. March 1989.

[59] M. Srinivasan, M. Lehrer, S.W. Zhang, G.A. Horridge. How Honeybees Measure their Distance from Objects of Unknown Size. In *Journal of Comparative Physiology A*, Vol. 165, 1989, pp. 605-613.

[60] M. Srinivasan, M. Lehrer, W.H. Kirchner, S.W. Zhang. Range Perception Through Apparent Image Speed in Free Flying Honeybees. In *Visual Neuroscience*, Vol.6, May 1991, pp. 519-535.

[61] M. Srinivasan. How Bees Exploit Optic Flow: Behavioural Experiments and Neural Models. In *Philosophical Transactions of the Royal Society of London*, Vol. 337, September 1992, pp. 253-259

[62] L. Stark, K. Bowyer. Achieving Generalized Object Recognition through Reasoning about Association of Function to Structure. In *IEEE Tranasactions on Pattern Analysis and Machine Intelligence*, Vol.13, No.10, October 1991. pp. 1097-1104.

[63] L. Stark and K. Bowyer. Function-Based Generic Recognition for Multiple Object Categories. In *CVGIP: Image Understanding*, Vol.59, No.1, January 1994, pp. 1-21.

[64] J. Stewman and K. Bowyer. A "Viewing Sphere" Approach to Computer Vision. In *IEEE Southeastcon '87*, Tampa, Florida, April 5-8, 1987. pp. 25-29.

[65] G. Taubin, R.M. Bolle, D.B. Cooper. Representing and Comparing Shapes Using Shape Polynomials. In *Proceedings of Computer Vision and Pattern Recognition.*, San Diego, CA, June 4-8, 1989. pp. 510-516.

[66] K. Wall and P. Danielsson A Fast Sequential Method for Polygonal Approximation of Digitized Curves. In *Computer Vision and Image Processing*, Vol.28, No.2, November 1984, pp. 220-227.

[67] K. Wu. *Computing Parametric Geon Descriptions of 3D Multi-Part Objects* Ph.D. thesis, McGill University, Dept. of Electrical Engineering, March 1996.

[68] J.S.Zelek and M.D. Levine *SPOTT: A Mobile Robot Control Architecture for Unknown or Partially Known Environments* In *1996 AAAI Spring Symposium on Planning with Incomplete Information for Robot Problems*, March 25-27, 1996 at Stanford University.

[69] J. Zelek. *A Real-Time, Distributed and Scalable Architecture for Autonomous Mobile Robot Control* Ph.D. thesis, McGill University, Dept. of Electrical Engineering, July 1996.