SENSOR PLANNING FOR OBJECT SEARCH

by

Yiming Ye

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Abstract

SENSOR PLANNING FOR OBJECT SEARCH

Yiming Ye
Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto
1997

This thesis studies the almost-unexplored field of sensor planning for object search. Object search is the task of efficiently searching for a given 3D object in a given 3D environment by an agent equipped with a camera for target detection and, if the environment configuration is not known, a method of calculating depth, like stereo or laser range finder. Sensor planning for object search refers to the task of selecting the sensing parameters so as to bring the target into the field of view of the camera and to make the image of the target easily detected by the available recognition algorithms. In this thesis, the task of sensor planning for object search is formulated as an optimization problem. This problem is proved to be NP-Complete, thus an approximate solution employing a one step look-ahead strategy is proposed. This approximation is equivalent to the optimal solution under certain conditions. The search region is characterized by the probability distribution of the presence of the target. The goal is to find the desired object reliably with minimum effort. The control of the sensing parameters depends on the current state of the search region and the detecting ability of the recognition algorithm. The huge space of possible sensing actions is decomposed into a finite set of actions that must be considered. In order to represent the surrounding environment of the camera and to determine efficiently the sensing parameters over time, a concept called the sensed sphere is proposed, and its construction, using a laser range finder, is derived.

Acknowledgements

First of all, I would like to thank my supervisor, Prof. John K. Tsotsos for his constant support and encouragement throughout this thesis. His insight and experience about computer vision research, his guidance, his patience, and his unflagging optimism make him one of the best advisors a Ph.D student could hope for.

I am also grateful to my committee members, Chris Brown, Eugene Fiume, Andrew Goldenberg, Allan D. Jepson, Hector Levesque, and Demetri Terzopoulos for spending their valuable time to critique my work and guide me through the process. In particular I would like to thank Chris Brown, my external examiner, for his detailed comments and suggestions concerning the manuscript.

There were many people who provided the technical and emotional support I needed to continue my journey. Dave Wilkes has given me many encouragement and help during various stages of my study. Hui Li and Victor Lee have been my great friends and have offered many help. Richard Mann, Chris Williams, and Sean Culhane helped me with my Qualifying Examination, especially Chris, who read my document in detail, and Richard, who helped me with my slides and presentation. Piotr Jasiobedzki has read my technical report on object search and given many valuable comments. Eric R. Harley has read my thesis in great detail and offered many valuable comments that greatly improved the presentation. Chakra Chennubhotla, Lawrence Chung, Sven Dickinson, Paul Kearney, D H Kulkarni, Feng Lu, James MacLean, Yasmina Miler, Brian Nixon, Fernando NuFlo, Andrew Prior, Arnold Rosenbloom, Kevin Schlueter, Gilbert Verghese, Eric Yu and many others have offered help in various aspects.

I would like to thank the Graduate School of University of Toronto for supporting me as a Open Fellowship student (1991-1993) and the Department of Computer Science of University of Toronto for supporting me as a Research Assistant student (1993-1995). I would like to thank IBM Centre for Advanced Studies for supporting me as a Research Fellow Student (1995-1996) and a Post Doctoral Research Fellow (1996-1997), and to the people at CAS, particularly Karen Bennet, Jacob Slonim, and Kelly Lyons.

I am indebted to my parents, Prof. Jizhong Ye and Prof. Chuying Wang, and my brothers Prof. Yongjian Ye and Mr. Wen Ye for their love and support. They have provided me with a warm family. They always try their best to help me whenever I needed and no matter how much effort they have to pay. I am glad that my little accomplishment has brought them great joy and happiness. I miss them, especially at this moment.

Words along cannot express my gratitude to my wife, Haiyan Wang, who always believed in me more than I did and always offered help in every possible way. She endured all my doubts, frustration, and impatience before my research took shape and shared all my excitements whenever I made a little progress ... sending me to the lab in the morning and greeting me back home late at night ... it is not easy. Her loving presence has made everyday of my graduate study an enjoyable experience and a sweet memory. Her love and effort will be treasured.

For my parents and my wife

Contents

Chapter 1

Introduction

1.1 Problem Statement

Object search is the task of finding a given three-dimensional object in a given three-dimensional environment by using a camera or a robot equipped with a camera. It is clear that exhaustive, brute-force search may suffice for its solution; however, our goal is the design of efficient strategies for search because exhaustive search will be computationally and mechanically prohibitive for non-trivial situations. Generally speaking, this task contains three parts. The first is selection of sensing parameters so as to bring the target into the field of view of the sensor with sufficient image quality that it can be detected by the recognition algorithm. This is the sensor planning problem for object search. The second is the manipulation of the hardware so that the sensing operators can reach the state specified by the planner. This is possible because of the recent availability of sophisticated hardware for sensor/platform control and real-time image processing ([77], [6], [87]). The third is search for the target within the image. This is the object recognition and localization problem, which attracts a lot of attention within the computer vision community (see ([19]) for a review). In this thesis, we study in detail the first of these subdivisions of the object search task, i.e., the problem of sensor planning.

1.2 Background

Vision is our most elaborate sense, and as such has challenged philosophers over many centuries to explain the mystery of its mechanism. As early as the 19th century, biologist Hermann von Helmholtz (1821-1894) began the study of this problem. Psychologists have also studied this problem from the behavioral point of view and the results are reported in a series of books by Gibson [30][31]. Computer scientists began to examine this problem because of the advent of the digital computer. The discipline of computer vision is still young.

Most vision research until 1975 was of an "engineering" nature — trying to construct entire vision systems, i.e., systems that exhibit some integration and use knowledge at all levels, including domain specific information. Such research produced algorithms for boundary detection using a variety of edge detection techniques or region growing techniques. These techniques were used to segment 2-D images based on gray value intensities. In general, engineering approaches to vision require ideal input data or specific domain dependent assumptions. These approaches were rejected by David Marr [50], because they were often *ad hoc*, with no sound theoretical basis relating the type of processing to the actual visual content. As a result, these approaches could not be generalized to arbitrary scene interpretation. Marr and his colleagues saw the need to return to fundamental principles of vision, and therefore concentrated on the nature of the computational problem common to both biological and machine systems. In this way the foundations of modern computer vision were established, and attention shifted from restrictions on the domain of application of a vision system to restrictions on visual abilities.

The focus of modern research is thus defined more in terms of topics that correspond to identifiable modules in the human visual system. It is hoped that producing such modules will result in a computer vision system having most, if not all, of the capabilities of the human visual system. Marr's effort in articulating a comprehensive vision theory in computational terms was enormously valuable in defining the principle issues. Vision is slowly becoming a well defined science. Now, greater significance is attached to demonstrating explicit links between image understanding, psychology, psychophysics and neurophysiology.

Despite the contributions of Marr's theory, it leaves some problems unsolved. In Marr's paradigm, the task of vision is construed as the construction of a detailed representation of the physical world. This construction is regarded as independent of larger tasks. Most work has only involved extensive static analysis of passively sampled data. But almost every basic problem in passive machine perception is very difficult. The first difficulty has to do with employing the right assumptions. Some of the algorithms are based on assumptions which despite generality are frequently not valid in the real world, and so these algorithms fail when applied to a variety of natural images. The second difficulty has to do with uniqueness properties of the resulting algorithms. Some of the problems cannot have a unique solution. In order to reduce the space of all solutions to a unique point, assumptions are made about the world which usually are not general enough and the algorithms may fail when applied to some real images. The third difficulty is the possible lack of robustness or stability of the resulting algorithms. Even if theoretical analysis shows that given the constraints at hand a particular problem has a unique solution, in practice it turns out that for some problems the solution might be unstable. In other words, a very small error in the input might result in a catastrophic error in the output.

Bajcsy [6] noticed that there are some major issues that are not addressed in Marr's paradigm, one of them being the effect of the perceiver's behavior. She proposed that a passive sensor be used in an active fashion, purposefully changing the sensor's state parameters according to sensing strategies. The problem becomes one of intelligent control applied to the data-acquisition process that depends on the current state of data interpretation including recognition. This new approach is called "active perception" or "active vision". It has led a growing number of researchers in the computer vision community to realize that the ability to actively control the parameters of the sensory apparatus, and more generally, the ability to combine vision with behavior is vital to achieve robust, real-time perception for a robot interacting with a complex world.

Since 1985, a significant amount of work in computer vision has focused on the problems of systems that act and respond to their environment. Recent research in active vision has shown the advantages of this strategy. Active vision algorithms can be more robust than static algorithms, and are often computationally efficient since irrelevant information is ignored. Furthermore, there are some tasks (e.g., model based object recognition) which are naturally suited to active vision and which conventional vision systems find very difficult to perform. It has been shown [4] that an active observer can solve basic vision problems (e.g., shape from shading, shape from contour, shape from texture and structure from motion) in a much more efficient way than a passive observer. Some problems that are ill-posed, nonlinear or unstable for a passive observer can be converted into well-posed, linear or stable problems by an active observer. This improvement is due to the availability of extra constraints from the additional images obtained in the active vision process. As summarized by Tsotsos [78], active vision can provide dynamic changes in the image acquisition process and is useful in at least the following ways:

- to see a portion of the visual field otherwise hidden;
- to compensate for spatial non-uniformity of a processing mechanism;
- to increase spatial resolution;
- to disambiguate aspects of the visual world (through induced motion, or lighting change, etc.);
- to enhance the efficiency of processing by restricting the search space;
- to segment areas of interest in the image precategorically by using gaze control;
- to provide for a better mathematical problem formulation and reduce computational complexity.

Tsotsos [78] pointed out that in order to obtain the above advantages, there are significant hurdles to overcome, one of them being efficiency. It can be shown [79] that if a problem can be solved by both active and passive methods, then the active approach will be more efficient only under certain constraints. In fact, we must consider the cost imposed on a perceptual system if it must actively manage the sensor. As listed in [78], these costs include:

- deciding that a change of visual field is needed and why (specifying why helps determine which change is best);
- deciding on which change is best (priority sequence: eye movement, head movements, body movement);
- executing change;
- adapting the system to the new viewpoint;
- corresponding objects and events between old and new viewpoints.

As a typical active vision task, an object search system must take all of the above factors into consideration. The work reported in this thesis can be viewed as a logical extension and elaboration of Tsotsos' work [79] concerning the efficiency of active and passive visual search.

1.3 Motivation

Object search is a task that is particularly well suited to the active vision approach. Thus, the study of object search may produce results that will lead to an understanding of various aspects of an active vision system. Furthermore, the sensor planning task for object search is very important from practical considerations. For example, a robot in its workspace often needs to use sensory information to determine the position of the objects in order to manipulate them. Typically, it is impractical to determine the location before the object is needed. Due to the camera's limited field of view, occlusion, the limited depth of field of lenses, the lighting conditions and other factors, it is often necessary to obtain many images using many camera configurations and viewpoints (see [86] for further discussion). Also, a robot often needs to search for known landmarks in order to estimate its own position and navigate in an industrial environment. For the above two examples, the solution to the problem of where to efficiently direct the sensor is critical.

Although sensor planning for object search is very important in practice, it is interesting to note that research on this topic within the computer vision community is limited ([77], [89], [29], [54], [63]). Connell et al. [20] have constructed a robot that roams an area searching for and collecting soda cans. The planning is very simple since the robot just follows the walls of the room and the sensor only searches the area immediately in front of the robot. This may not be very efficient since the likely presence of the target is not considered when the robot is roaming. Rimey and Brown [63] used composite Bayes net and utility decision rules to plan the sensor in their task-oriented system, TEA. The sensor is directed to the center of mass of the expected area for a certain object based on the belief value of the net. Probability of presence is used in their system, but the purpose of the sensor planning is mainly answering average queries that can be posed on a Bayes Net rather than searching.

The indirect search mechanism proposed by Garvey [29] first directs the sensor to search for an "intermediate" object that commonly participates in a spatial relationship with the target, and then directs the sensor to examine the restricted region specified by this relationship. Wixson and Ballard [91] presented a mathematical model of search efficiency and predicted that indirect search can improve efficiency in many situations. The problem with indirect search is that the spatial relationship between the target and the "intermediate" object may not always exist and detection of the "intermediate" object may not be always easier than detection of the target. Furthermore, there is no sensor planning involved in searching the environment for the "intermediate" object or in searching the restricted region for the target. It is interesting to note that the operational research community has done a lot of research on optimal search [42]. Their purpose is to determine how to allocate effort to search for a target, such as a lost submarine in the ocean or an oil field within a certain region. Although the results are elegant in a mathematical sense, they cannot be directly applied here because the searcher model is too abstract and general, and there is no sensor planning involved in their approach.

1.4 Thesis Statement and Overview

As an elaboration of Tsotsos' work on $3D$ object search [77] and work concerning the efficiency of active and passive visual search [78], this thesis provides the first formalization of the sensor planning task for object search and a practical sensor planning strategy. Instead of examining different aspects of object search individually and in some degree of isolation, the formulation emphasizes more the relationship between action and image processing and integrates the two into a complete system. As motivated by Tsotsos' discussion of incremental active perception [78], our formulation combines the influence of the search agent's initial knowledge and the influence of the performance of the available recognition algorithms together and formalizes the task as an optimization problem, thus transforms Tsotsos' general notion about the cost and efficiency of a general active vision system into the corresponding specific notion with respect to a specific active vision system: the object search system. In order to reveal basic insights into the structure of object search task, we analyze the task theoretically based on the formulation and prove that the sensor planning task is NP-Complete. Instead of looking for an optimal algorithm for sensor planning task, we propose to use greedy strategy and prove that the greedy strategy can generate optimal answers in certain situations. For practical search strategy, the search region is characterized by the probability distribution of the presence of the target. The control of the sensing parameters depends on the current state of the search region and the detection abilities of the recognition algorithm. In order to efficiently determine the sensing actions over time, the huge space of possible actions is reduced to a finite set of actions that must be considered. The result of each sensing operation is used to update the status of the search space. Both simulation experiments and real experiments are presented in support of our theory.

As a conclusion, this thesis studies the almost-unexplored field of object search. It first provides a formalization of the problem of how to search for an object in a 3D world and show that no optimal solution is likely in general. It then points to a sufficient method for achieving an approximate solution and tests the method with simulation and real experiments.

1.5 Thesis Outline

Part one formulates the sensor planning task for object search, pinpoints many important properties of this task and proves that this task is NP-Complete. Part two discusses in detail the detection function for a recognition algorithm. Part three proposes a strategy for the sensor planning task. Part four presents the experimental results. Both simulation tests and real experiments are presented.

Part I

Problem Formulation

Chapter 2

Problem Formulation

2.1 General Discussion

In its most general form, the state of an object search system at time τ can be denoted as $V(\tau) = (E_{conf}(\tau), S(\tau), K_{env}(\tau), T_{pos}(\tau), A)$, where the system parameters are defined as follows. $E_{conf}(\tau)$ is the configuration of the environment which may be dynamic or static. When it is dynamic, the environment configuration will change as time changes. $S(\tau) = (s_1(\tau), \ldots, s_m(\tau))$ is the vector of state parameters of the sensor. It is a function of time because the sensor parameters change during the search process. $K_{env}(\tau)$ is the system's knowledge about the environment. This knowledge is updated after each search action because the agent acts to acquire knowledge of the location of the target. $T_{pos}(\tau)$ is the position of the target. When the target is allowed to move during the search process, $T_{pos}(\tau)$ is not a constant. $A = (a_1, \ldots, a_{n_a})$ is the set of available recognition algorithms used to detect the target in the image.

At a given time τ, with respect to the current environmental configuration $E_{conf}(\tau)$ and the target position $T_{pos}(\tau)$, there exists a group of desired possible sensor parameters $S_{desired}(\tau) = \{S_1^*(\tau), \ldots, S_r^*(\tau)\}$, such that if we take an image with the sensor state parameters set to $S_i^*(\tau)$ (i can be $1, \ldots, r$) at time τ and analyze the image using the existing recognition algorithms, the target can be detected. The purpose of $3D$ object search is to achieve efficiently one of the desired sensor states and take an image according to this state.

In this thesis, we consider a simpler problem — we assume that the environment is static and that the target is not allowed to move. Under this assumption, the state of the object search system becomes $V(\tau) = (E_{conf}, S(\tau), K_{env}(\tau), T_{pos}, A)$. Only the sensor state parameters $S(\tau)$ and the agent's knowledge about the world is allowed to change. For the given E_{conf} and T_{pos}, there exists a group of desired sensor states $S_{desire} = \{S_1^*, \ldots, S_r^*\}$. These desired sensor states do not change as time changes. The goal of object search becomes to adjust the sensor state to one of the desired states, take an image under this state and analyze the image using the given recognition algorithms. The purpose of sensor planning is to try to realize the goal efficiently.

Suppose that all the possible states for the sensor are $S_1, \ldots, S_N$, then the sensor planning task becomes: select a state S from $S_1, \ldots, S_N$, such that $S \in S_{desired}$. This task is very difficult because normally the number N of possible states is huge and the number r of desired sensor states is relatively small. It is almost impossible to use brute force strategy (to try each possible state one by one) to solve this problem. In order to search this huge space for a desired sensor state, the agent must use the available knowledge to guide the search process and it must incorporate new knowledge from the results of the sensing operations into $K_{env}(\tau)$.

The task of sensor planing for object search is actually a task to focus the agent's limited resources to examine a certain $3D$ region of the space. The region is determined by the camera's position, the viewing axis and the visual angle size, etc. This task is similar to the visual attention problem where limited resources are focused on a part of an image (2D). Thus, the object search problem is also an attention problem [78].

2.2 The Search Agent

The search agent has two components. The first is a sensor, which is used to detect the target and to sense the environment. Here, the sensor is a camera with zoom capabilities. When the environment configuration is not known, we assume that the sensor also contains a laser range finder or that the sensor is composed of two cameras that can be used for stereo computation. The second is some means of controlling the state configurations of the sensor, for example, a mobile robot or a robotic arm.

2.2.1 The Sensor Model and its State Parameters

In this section, we define the sensor model and its state parameters.

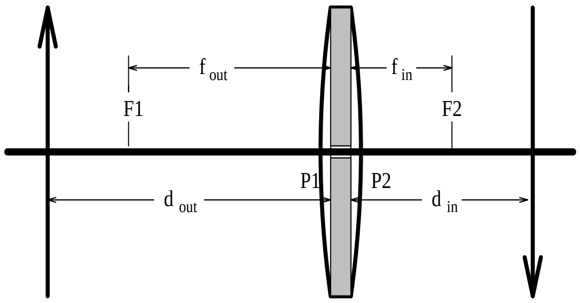


Figure 2.1: Thick-lens geometry. $P1$ and $P2$ are the primary and secondary principal planes respectively, and they intersect the optic axis at the principal points. The front focal plane $F1$ is a plane normal to the optic axis, situated at distance f_{out} in front of the lens. The back focal plane $F2$ is a plane normal to the optic axis, situated at distance f_{in} in back of the lens. d_{out} is the distance between the object and the primary principal point. d_{in} is the distance between the image plane and the secondary principal point.

The camera, together with a digitizer, reports the average light intensity for parts of the scene projecting onto each pixel in the image plane through a thick lens. Figure 2.1 illustrates the thick-lens geometry. Here we assume the camera has the same medium (air) between the lens and the sensor as between the lens and the scene, so $f_{out} = f_{in} = f$. We also assume that the front and back nodal points of the lens coincide with the entrance and exit pupils of the camera, respectively. The thick lens gathers all of the light radiating from an object point toward the lens' finite aperture, and brings this light into focus at a single distant image point. This process obeys the Gaussian lens law:

$$\frac{1}{d_{out}} + \frac{1}{d_{in}} = \frac{1}{f}$$

The **focal length** f is the distance from the secondary principal planes at which any object point that is located at infinity is brought into focus. The **focal plane** is the plane perpendicular to the optical axis at a distance f from the secondary principal planes. The focal length of a thick lens can be adjusted by motors on the zoom lenses. The distance from the lens to the detector plane can also be adjusted by another motor. Here we assume that the detection plane is always coincident with the focal plane. So the distance from the secondary principal plane to the detector plane is always adjusted to be f. Figure 2.2(a) illustrates the thick-lens image formation process. Since the distance between P_1 and P_2 has no influence on the position of the object point on the image, we can assume that it is 0 and thus get the equivalent image formation process as shown in Figure 2.2(b). This is actually a thin lens camera model. From now on, we assume the camera model is the same as the thin lens model with the property that the focal length can be adjusted and that the image plane is coincident with the focal plane.

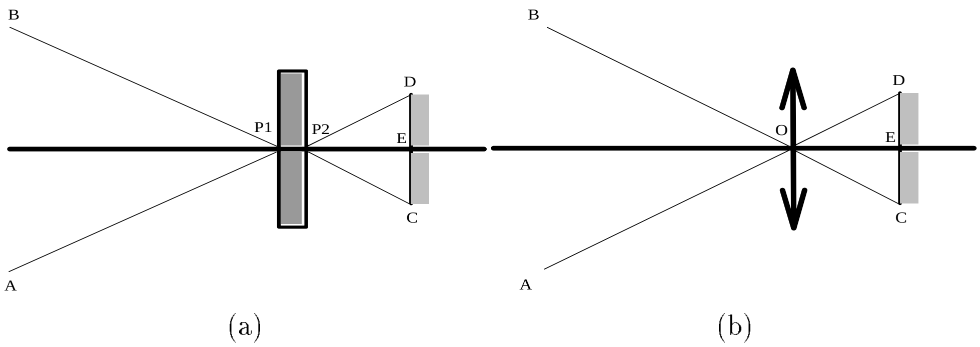


Figure 2.2: (a) $2D$ illustration of the thick lens image formation process. C is the image of B; D is the image of A. $\mid P_2E \mid= f$. BP_1 is parallel to CP_2, AP_1 is parallel to DP_2. (b) The equivalent image formation geometry, actually a thin lens geometry.

The **field of view** describes the volume which comprises all the directions from which rays of light strike the **image plane** after passing through the effective center of projection of the lens. Here, we assume that the image plane is a rectangle with size $W \times H$. Thus the field of view is in fact a rectangular pyramid with size (w, h), where w (width) is the angle size in the horizontal direction of the viewing volume and h (height) is the angle size in the vertical direction of the viewing volume. From Figure 2.2(b), it is easy to see that only those points that are within angle $\angle AOB$ can be projected onto the detection plane DC. So the angle size h equals angle $\angle DOC$, and it can be calculated by:

$$h = 2arctan(\frac{\mid DE \mid}{\mid OE \mid}) = 2arctan(\frac{\frac{H}{2}}{f})$$

Similarly,

$$w = 2arctan(\frac{\frac{W}{2}}{f})$$

Since W and H are fixed, the field of view is uniquely determined by the focal length f.

In summary, the parameters of the sensor that can be controlled during the search process are: the position of the **center of the camera** (P_1 in Figure 2.2(a) or O in Figure 2.2(b)); the direction of the **viewing axis** of the camera; and the size of the **viewing volume** $< w, h >$ [1], where w and h must satisfy the relation $\frac{tan(\frac{h}{2})}{tan(\frac{w}{2})} = \frac{H}{W}$. These parameters should be purposefully controlled so as to search efficiently for the target.

2.2.2 The Search Agent Used in This Thesis

Throughout this thesis, we assume that the sensor is a zoom camera equipped with a laser range finder, and that it is situated on a mobile platform. The camera is used to detect the target, and the laser range finder is used to construct the surrounding environment when the configuration of the environment is not known before the search process.

This searcher model is based on the ARK [2] robot (Figure 2.3), which is a mobile platform equipped with a special sensor: the Laser Eye (Figure 2.3). The mobile platform is a 3 wheeled Cybermotion K2A platform equipped with a small number of sonar transducers. The Laser Eye is mounted on a robotic head with pan and tilt capabilities. It consists of a camera with controlled focal length (zoom), a laser range-finder and two mirrors. The mirrors are used to ensure collinearity of effective optical axes of the camera lens and the range finder. The robot can move freely in an office-like environment or a known industrial environment. The laser can be "emitted" from the center of the camera (see [57] for detail) in any direction so as to measure the distance of objects in the environment to the center of the camera along this direction. The camera is mounted on the robot at a fixed height, so the position of the camera is totally determined by the position of the robot. The direction of the viewing axis and the size of the view angle can be adjusted according to the current visual task.

[1] The reason that we use $< w, h >$ instead of f is to make the discussions in the following chapters more intuitive

[2] The ARK (Autonomous Robot for a Known environment) project is undertaken as a collaboration between Ontario Hydro, Atomic Energy Canada Limited, The National Research Council of Canada, York University and the University of Toronto. The objective of the project is to build a mobile robot capable of navigating in partially known industrial environment, using a variety of sensors, including color video cameras, laser range finders, sonar and infrared sensors.

(a) (b)

Figure 2.3: (a) The Laser Eye. At the top is the Optech laser range finder; at the bottom is the zoom and focus controlled lens. The two mirrors are used to ensure collinearity of effective optical axes of the camera lens and the range finder. The pan-tilt unit is operated by two DC servocontrolled motors from Micromo Electronics, equipped with gearboxes and optical encoders. (b) The ARK mobile platform.

2.3 The Coordinate Systems

The coordinate systems used in this thesis are the World Coordinate System (WCS), the Head Coordinate System (HCS) and the Camera Coordinate System (CCS).

The World Coordinate System is fixed, and corresponds to the search region Ω. The Head Coordinate System corresponds to the robot head. The origin of the Head Coordinate System is the same as the head center. The direction of the Head Coordinate System is the same as that of the World Coordinate System (Figure 2.4(a)). The head center can be moved by moving the robot platform. We assume the center of the head is the same as the camera center (front nodal point of the camera, e.g. $P1$ in Figure 2.2(a)). The viewing axis of the camera (Figure 2.4(b)) is a half line that originates from the origin of the Head Coordinate System. The direction of the viewing axis is specified by pan and tilt (p, t) (Figure 2.4(c)), where p is the angle needed to rotate the camera viewing axis around the Z axis of the Head Coordinate System counter-clockwise until it falls onto the first quarter of the XZ plane of the Head Coordinate System, and t is the size of the angle between the camera viewing axis and the Z axis of the Head Coordinate System. The camera is not allowed to rotate around its viewing axis. The Camera Coordinate System is used to specify the relative position of the search region point with respect to the camera center. The origin of the Camera Coordinate System is the same as the origin of the Head Coordinate System. The direction of the Z axis of the Camera Coordinate System is the same as the direction of the view axis of the camera (Figure 2.4(d)). The direction of the X axis of the Camera Coordinate System is parallel to the horizontal direction of the image plane (AB in figure (Figure 2.4(d))). The direction of the Y axis of the Camera Coordinate System is parallel to the vertical direction of the image plane (DC in figure (Figure 2.4(d))). Let (x, y, z) be the coordinate of a point in the Camera Coordinate System. We will use (θ, δ, l) to represent the relative position of (x, y, z) to the camera, where θ, δ are the horizontal and vertical direction offsets to the viewing axis of the camera, and l is the length of the projection of the line segment, which connects the camera center and the target center, on the viewing axis. Since the camera viewing volume is symmetric about its viewing axis, we have:

$$\theta = \mid arctan(\frac{x}{z}) \mid \tag{2.1}$$

$$\delta = \mid arctan(\frac{y}{z}) \mid \tag{2.2}$$

$$l = \mid z \mid \tag{2.3}$$

Suppose the width and height of the viewing volume are (w, h). Then the condition that position (x, y, z) is within the viewing volume is

$$\theta \leq \frac{w}{2}$$
$$\delta \leq \frac{h}{2}$$
$$z > 0$$

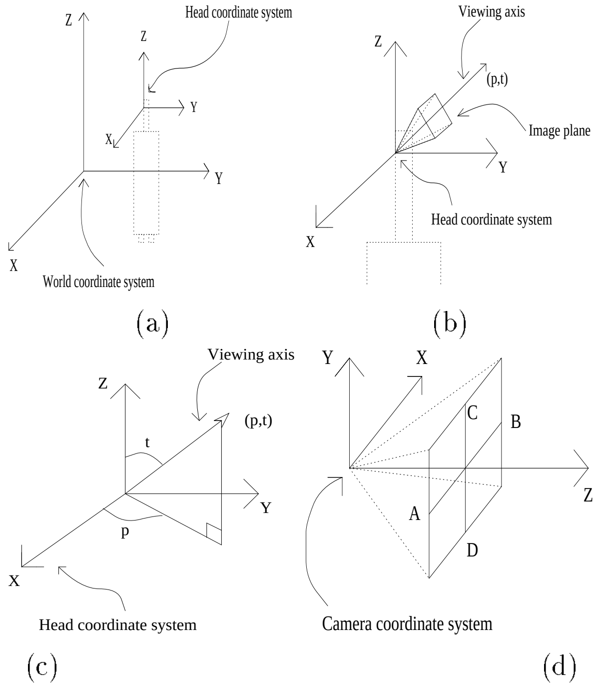


Figure 2.4: (a) The world coordinate system and the head coordinate system; (b) The head coordinate system and the camera viewing volume; (c) The head coordinate system and the camera viewing axis direction (p, t). (d) The camera coordinate system and the image plane.

2.4 The Formulation

Although it is important to examine different aspects of object search individually and in some degree of isolation, it is even more important to study their relationship and how to integrate them into a whole search system.

A natural question to be asked is: what is object search? One might think that the answer is straight forward. But when we analyze the problem more deeply, it is not easy at all. First, before the search process, the agent has varying degrees of knowledge regarding the location of the target. Any knowledge should be used to guide the search process. Second, there is no perfect recognizer. Even if the target is within the field of view of the camera, the recognition algorithm might fail to detect the target. Third, occlusion might occur during the search process. Because of these factors, we need to formulate the object search process by taking these factors into its consideration.

2.4.1 Some Concepts

In order to formulate the problem, we first define some concepts.

The search region Ω is the $3D$ space to be searched. This region can be in any form and it is assumed that we know the boundary of Ω exactly, but that we do not know its internal configuration. The world coordinate system corresponds to this region. Our task is to specify the position (x, y, z) of the center of the target if it is in this region, or stop the search process if it is not in this region. In order to make our approach practical, we will tessellate the region Ω into a series of elements c_i. Thus, we have: $\Omega = \bigcup_{i=1}^{n} c_i$ and $c_i \bigcap c_j = 0$. Our task is to find out which element contains the center of the target object. In the rest of the paper, we assume the search region is an office-like environment, and we tessellate the space into little cubes of the same size.

An operation $\mathbf{f}$ on Ω is an action of the searcher within the region Ω. The action means to take a **perspective** projection picture according to the current camera configuration $S(\tau)$ and to analyze the image to find out whether the target is present in the image. The camera configuration $S(\tau)$ is determined by the camera position (x_c, y_c, z_c) (i.e. the coordinates of the camera center with respect to the world coordinate system), the viewing axis direction (p, t), and the width and height of the solid viewing angle (w, h) (Note: the units of w and h are radians). For convenience, here we use (w, h) instead of the focal length of the camera as the parameter of the operation, although (w, h) is actually uniquely determined by focal length. An operation $\mathbf{f}$ on Ω can be represented as $\mathbf{f} = \mathbf{f}(S(\tau), a)$, where a is the recognition algorithm used to search the image. For convenience, we represent it as $\mathbf{f} = \mathbf{f}(x_c, y_c, z_c, p, t, w, h, a)$. This operation returns *true* if the target is found and *false* if the target is not found.

The cost function $\mathbf{t}(\mathbf{f})$ for operation $\mathbf{f}$ gives the time required to finish this operation. It is the sum of (1) the time to manipulate the hardware to the status specified by $\mathbf{f}$; (2) the time to take a picture; and (3) the time to run the recognition algorithm and update the environment. All times are in seconds.

The target distribution can be specified by a probability density function $\mathbf{p}'$. Since this distribution will be updated after each operation, $\mathbf{p}'$ can be considered as a function of both position and time, i.e. for $(x, y, z) \in \Omega$ ((x, y, z) is in world coordinate system) and time τ, $\mathbf{p}'((x, y, z), \tau)$ gives the probability that the center of the target is at (x, y, z) when the time is τ. Usually this probability distribution is assumed to be known at the beginning of the search and it is determined by our knowledge of the world. If we know nothing about the distribution, then we can assume a uniform distribution at the beginning. After discretization of the search space, the probability that the center of the target falls into cube c_i is: $\mathbf{p}(c_i, \tau) = \int_{c_i} \mathbf{p}'((x, y, z), \tau) dxdydz$.

The detection function on Ω is a function $\mathbf{b}'$, such that $\mathbf{b}'((x, y, z), \mathbf{f})$ gives the conditional probability of detecting the target, given that the center of the target is located at (x, y, z) and that the operation is $\mathbf{f}$. For the region after the discretization, we define

$$\mathbf{b}(c_i, \mathbf{f}) = \frac{\int_{c_i} \mathbf{b}^{'}((x, y, z), \mathbf{f}) dxdydz}{vol(c_i)}$$

as the conditional probability of detecting the target given that the center of the target is located within c_i and the operation is $\mathbf{f}$, where $vol(c_i) = \int_{c_i} dxdydz$. In practice, we use $\mathbf{b}^{'}((x_{i_c}, y_{i_c}, z_{i_c}), \mathbf{f})$ to approximate $\mathbf{b}(c_i, \mathbf{f})$, where $(x_{i_c}, y_{i_c}, z_{i_c})$ is the center of c_i. For any operation, if the projection of the center of the cube c_i is outside the image, we have $\mathbf{b}(c_i, \mathbf{f}) = 0$; if it is inside the image, the value of $\mathbf{b}(c_i, \mathbf{f})$ will be determined by various factors. It is obvious that the probability of detecting the target by applying action $\mathbf{f}$ is given by

$$\int_{\Omega} \mathbf{p}^{'}((x, y, z), \tau_{\mathbf{f}}) \mathbf{b}^{'}((x, y, z), \mathbf{f}) dxdydz \tag{2.4}$$

where $\tau_{\mathbf{f}}$ is the time just before $\mathbf{f}$ is applied. After discretization, the probability of detecting the target by action $\mathbf{f}$ is given by

$$P(\mathbf{f}) = \sum_{i=1}^{n} \mathbf{p}(c_i, \tau_{\mathbf{f}}) \mathbf{b}(c_i, \mathbf{f}) \tag{2.5}$$

Let $\mathbf{O}_{\mathbf{\Omega}}$ be the set of all possible operations that can be applied on region Ω. Then the effort allocation $\mathbf{F}$ is the ordered set of operations applied in the search, $\mathbf{F} = \{\mathbf{f}_1, \ldots, \mathbf{f}_k\}$, where $\mathbf{f}_i \in \mathbf{O}_{\mathbf{\Omega}}$.

2.4.2 The Formulation of the Problem

The problem of object search can thus be defined as follows. Let $\mathbf{K}$ be the total time available for search. Then for any effort allocation $\mathbf{F} = \{\mathbf{f}_1, \ldots, \mathbf{f}_k\}$, the probability of detecting the target by this allocation is:

$$\begin{aligned} P[\mathbf{F}] = & \sum_{i=1}^{n} \mathbf{p}(c_i, \tau_{\mathbf{f_1}}) \mathbf{b}(c_i, \mathbf{f_1}) \\ & + [1 - \sum_{i=1}^{n} \mathbf{p}(c_i, \tau_{\mathbf{f_1}}) \mathbf{b}(c_i, \mathbf{f_1})] \times [\sum_{i=1}^{n} \mathbf{p}(c_i, \tau_{\mathbf{f_2}}) \mathbf{b}(c_i, \mathbf{f_2})] \\ & + \ldots \\ & + \{ \prod_{j=1}^{k-1} [1 - \sum_{i=1}^{n} \mathbf{p}(c_i, \tau_{\mathbf{f_j}}) \mathbf{b}(c_i, \mathbf{f_j})] \} \times [\sum_{i=1}^{n} \mathbf{p}(c_i, \tau_{\mathbf{f_k}}) \mathbf{b}(c_i, \mathbf{f_k})] \end{aligned} \tag{2.6}$$

and the total time for applying this allocation is (following [78]):

$$T[\mathbf{F}] = \sum_{\mathbf{f} \in \mathbf{F}} \mathbf{t}(\mathbf{f}) \tag{2.7}$$

The **task of object search** is to find an allocation $\mathbf{F} \subset \mathbf{O}_{\mathbf{\Omega}}$ which satisfies $T(\mathbf{F}) \leq \mathbf{K}$ and maximizes $\mathbf{P[F]}$.

2.4.3 The Environment

The environment is assumed to be an office-like space or an industrial workspace. The robot knows the boundary of the space but it has no idea about the objects present in this space. The space is divided into little cubes and the environment is represented as a union of these cubes. This is actually a discrete approximation of the search space by regular $3D$ grid. Each cube is characterized by the following properties: the **probability** and **solidity**. The **probability** will take a value between 0 and 1. The **solidity** can be either *solid* or *unknown*. It is *solid* when the cube is occupied by solid objects. It is *unknown* when we are not sure whether the cube is occupied by solid objects (it might be empty or it might be occupied by solid object). The solidity properties are updated whenever the robot gets to a new position. The probabilities are updated whenever a visual operation is conducted. This is a process of environmental mapping.

The idea to tessellate the space into grids and assign each grid with property values is not new. Elfes [26] has studied the problem of acquiring and handling information about the existence and localization of solid objects and empty spaces by composing information coming from multiple sonar readings, and has built a coherent world-model that reflects the information acquired. This world model is expected to serve as a basis for essential operations such as path planning, obstacle avoidance, etc. To do this, he proposes the idea of occupancy maps, which are two dimensional arrays of cells corresponding to a horizontal grid imposed on the area to be mapped. Each cell in the final occupancy map contains its occupancy status (unknown, empty, or occupied) with an associated certainty factor, using the following convention: unknown 0; empty $[-1, 0)$; occupied $(0, 1]$. A cell is considered unknown if no information concerning it is available. Cells can be empty with a certainty factor ranging from 0 to -1 and occupied with a certainty factor ranging from 0 to 1. The final map is computed from two separate arrays derived from the empty and occupied certainty distribution introduced above. One important step of Elfes [26] approach is the combination of the constraints from individual sonar readings with the sonar map. A sonar range reading is interpreted as making an assertion about two volumes in $3D$ space: one that is probably empty f_E and one that is somewhere occupied f_O. f_E measures the confidence that a point inside the cone of the beam is empty, and f_O measures the uncertainty about the location of the surface patch that caused the echo, somewhere on the range surface of the cone. These functions are defined based on the geometry of the beam and the spatial sensitivity pattern of the sonar sensor. Formally, the process of combining evidence proceeds along the following steps.

1. Initialization: The sonar map is set to unknown.

2. Superposition of empty areas: The probabilities corresponding to empty areas are combined using the probability addition formula $p_E(cell) \leftarrow p_E(cell) + p_E(reading) - p_E(cell) \times p_E(reading)$

3. Superposition of occupied areas: The probabilities corresponding to occupied areas are ini-

tially weakened by the evidence of the empty certainty factors, using $p_O \leftarrow p_O(reading) \times (1 - p_E(cell))$. Elfes then normalized the occupied probabilities over the beam front. Finally, the occupied probabilities are combined using $p_O(cell) \leftarrow p_O(cell) + p_O(reading) - p_O(cell) \times p_O(reading)$

4. Thresholding: The final occupation value attributed to a cell in the sonar map is given by comparing the relative strengths of the empty and occupied values

In our approach, although each $3D$ grid is associated with the solidity property, we do not use such a complex procedure as above to infer this property. The reason is that our task is to search for a target rather than to reconstruct the environment. As a result, the solidity property of a grid is obtained only as a by-product of the search process (this will be addressed later). However, the process to handle the knowledge regarding the whereabouts of the target is not so simple. This knowledge is explicitly encoded in our representation for each grid element c_i as a probability distribution $\mathbf{p}(c_i, \tau_\mathbf{f})$ and is inferred using a complex procedure. In previous work on sensor planning for object search, this knowledge is also used to varying extents. In the indirect search approach ([29], [89]), although this knowledge is not explicitly represented, it is implicitly used, because the target has a high probability of presence within the region specified by the geometric relation between the intermediate object and the target. Thus, when an intermediate object is detected, the region to be examined is greatly constrained. In Connell's approach ([20]), this knowledge is also implicitly used, as the search region is constrained to the table top where the soda can often appears. In the system TEA ([63]), this knowledge is explicitly encoded into the Bayes Net.

The knowledge about the position of the target should be updated whenever an action falls. We cannot use Elfes's strategy described above to update the probability knowledge. In his approach, the solidity property of one grid element has no relation with the solidity property of another. In our situation, the probability associated with any grid element is closely related with the probabilities of the others. So, we need a probability updating rule that can take the correlation of the probabilities of all the cubes into consideration. Bayes' formula is used here to incorporate the new recognition results into the old distribution. The search region can be approximated as: $\Omega = \bigcup_{i=1}^{n} c_i$. Let α_i be the event that the center of the target is in cube c_i, α_{out} be the event that the center of the target is outside the search region Ω, and let β be the event that after applying a recognition action, the recognizer successfully detects the target. Since the above events $\alpha_1, \ldots, \alpha_n, \alpha_{out}$ are mutually complementary and exclusive, we get the following formula (for $i = 1, \ldots, n, out$):

$$P(\alpha_i \mid \neg\beta) = \frac{P(\alpha_i)P(\neg\beta \mid \alpha_i)}{P(\alpha_{out})P(\neg\beta \mid \alpha_{out}) + \sum_{j=1}^{n} P(\alpha_j)P(\neg\beta \mid \alpha_j)} \tag{2.8}$$

where $P(\alpha_i \mid \neg\beta)$ is the probability that the center of the target is within cube c_i, under the condition that the current recognition action failed to detect the target. $P(\neg\beta \mid \alpha_i)$ is the probability that the recognizer failed to detect the target given that the target is at c_i. $P(\neg\beta \mid \alpha_{out})$ is the probability that the recognizer failed to detect the target given that the target is outside the search region. We have $P(\neg\beta \mid \alpha_i) = 1 - P(\beta \mid \alpha_i) = 1 - \mathbf{b}(c_i, \mathbf{f})$ and $P(\neg\beta \mid \alpha_{out}) = 1$.

Therefore, after each operation, the previous probability distribution $\mathbf{p}(c_i, \tau_{\mathbf{f}})$ and $\mathbf{p}(c_{out}, \tau_{\mathbf{f}})$ will be replaced by $P(\alpha_i \mid \neg\beta)$ and $P(\alpha_{out} \mid \neg\beta)$, where $\tau_{\mathbf{f}}$ is the time just before $\mathbf{f}$ is applied. Thus, we have the following updating rule (Note: $\tau_{\mathbf{f}+}$ refers to the time just after the action $\mathbf{f}$ is applied):

$$\mathbf{p}(c_i, \tau_{\mathbf{f}+}) \leftarrow \frac{\mathbf{p}(c_i, \tau_{\mathbf{f}})(1 - \mathbf{b}(c_i, \mathbf{f}))}{\mathbf{p}(c_o, \tau_{\mathbf{f}}) + \sum_{j=1}^{n} \mathbf{p}(c_j, \tau_{\mathbf{f}})(1 - \mathbf{b}(c_j, \mathbf{f}))}, \quad i = 1, \ldots, n, o \tag{2.9}$$

Figure 2.5 shows a $1D$ example of the environmental updating rule. The environment contains 9 cells. The sensor is able to check 3 consecutive cells with detection ability of 0.9, 0.95, 0.9, respectively. It is assumed that before the search operations the probability that the target is inside each cell is 0.1, and the probability that the target is not in the search region is 0.1. After the fourth operation, the probability that the target is outside the search region is pretty high which suggests that the target may not be present in the region.

It is easy to see from the above example that after each failed detection operation, the probability for cubes outside the search region will increase and the total sum of the probabilities of the cubes inside the search region will decrease. This dynamic updating process is very important for selecting the next action.

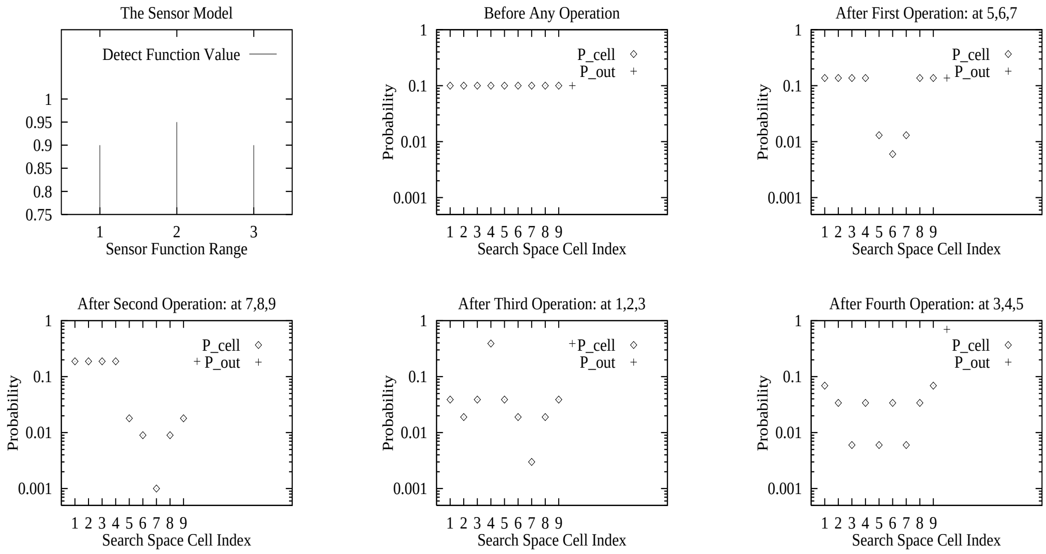


Figure 2.5: The sensor model and the probability update after four operations at: 5,6,7; 7,8,9; 1,2,3; and 3,4,5 respectively. Here we assume independence between actions

In our approach, the environment is tessellated into identical unit cubes. There are many other approaches to model a three-dimensional environment or a three-dimensional object.

Among them, octree representation [35] [59] is an economical approaches in terms of memory space. It provides a representation method that falls into the "recursive" category. The environment represented by an oct-tree is assumed to be $2^u \times 2^u \times 2^u$ arrays of unit cubes. Each unit cube has associated with it a label, either 1 or 0 respectively. The diameter of a $2^u \times 2^u \times 2^u$ environment is 2^u. An environment array of diameter 2^u may be divided into eight cube-shaped arrays of diameter 2^{u-1}. This process may be continued, where at each step an array of diameter 2^i is divided into eight cube-shaped arrays of diameter 2^{i-1}. Any cubic region obtained in this manner is referred to as an octant. Using oct-trees, an object array is divided into octants, each octant is in turn divided into octants, etc., until all the unit cubes within an octant have the same label. Octree representation can save memory because for any octant, if all its sub-octants are of the same label, then these sub-octants are not necessary to be represented.

For object search task, octree representation is not very attractive. Octree representation only wins when we have an exact method to measure the property of the smallest octant (thus assigning a value of either 0 or 1), and in most situations the octants with the same property should be able to be grouped together to form an upper level octants. These are not the situations for the target distribution representation in object search, because the probability value of each small cube is a number between 0 and 1, but not a binary value (either 0 or 1). Furthermore, the probability value of each cube is changing during the search process. Octree representation is also not very attractive in representing the solidity property of the environment for object search. Because the property of solidity is not measured exactly, and the number of solid cubes only increasing as the search continues and this number is usually not very big. Thus, we do not need a lot of memory to store the solidity properties of the cubes, because we only need to store the solid cubes accumulated during the search process.

2.5 Some Properties of the Object Search Process

It is interesting to explore the properties of the object search process defined above. Our purpose is to find regularities of this process so as to reveal some basic insights into the structure of the problem.

Suppose $\Omega = \bigcup_{i=1}^{n} c_i$, $\mathbf{O_\Omega} = \{\mathbf{f_1^*}, \mathbf{f_2^*}, \ldots, \mathbf{f_m^*}\}$. For any operation $\mathbf{f} \in \mathbf{O_\Omega}$, we define its influence range as $\Omega(\mathbf{f}) = \{c_i \mid \mathbf{b}(c_i, \mathbf{f}) \neq 0 \wedge c_i \in \Omega\}$. Its complement form is: $\overline{\Omega(\mathbf{f})} = \Omega - \{c_i \mid \mathbf{b}(c_i, \mathbf{f}) \neq 0 \wedge c_i \in \Omega\}$.

For any effort allocation $\mathbf{F} = \{\mathbf{f}_1, \ldots, \mathbf{f}_q \mid \mathbf{f}_i \in \mathbf{O_\Omega}\}$, the initial probability distribution is denoted as $\mathbf{p}^{[0]}(c_1), \mathbf{p}^{[0]}(c_2), \ldots, \mathbf{p}^{[0]}(c_n), \mathbf{p}^{[0]}(c_o)$. After the application of the operation $\mathbf{f}_1$, the distribution is denoted by $\mathbf{p}^{[1]}(c_1), \mathbf{p}^{[1]}(c_2), \ldots, \mathbf{p}^{[1]}(c_n), \mathbf{p}^{[1]}(c_o)$. Generally, after the application of the operation $\mathbf{f}_i$, the distribution is denoted by $\mathbf{p}^{[i]}(c_1), \mathbf{p}^{[i]}(c_2), \ldots, \mathbf{p}^{[i]}(c_n), \mathbf{p}^{[i]}(c_o)$, where $1 \leq i \leq q$. Let $P(\mathbf{f}_i)$ represent the probability of detecting the target by applying the action $\mathbf{f}_i$ with respect to the allocation $\mathbf{F}$. Then of course we have $P(\mathbf{f}_i) = \sum_{j=1}^{n} \mathbf{p}^{[i-1]}(c_j)\mathbf{b}(c_j, \mathbf{f}_i)$.

Let $P^{[0]}(\mathbf{f}_i)$ represent the probability of detecting the target by applying the action $\mathbf{f}_i$ when no action has been applied before. Then we have $P^{[0]}(\mathbf{f}_i) = \sum_{j=1}^{n} \mathbf{p}^{[0]}(c_j)\mathbf{b}(c_j, \mathbf{f}_i)$.

To get some insight into the object search process, we are most interested in the following regularities. First, we want to know the general expression for the probability distributions after we have executed an action sequence. Second, we want to know the general expression for the detection probability $P(\mathbf{f}_i)$ by applying an action $\mathbf{f}_i$ with respect to $\mathbf{F}$. Third, we want to know the general expression for the expected probability of detecting the target by a given effort allocation F. We hope to express the above three quantities by using those quantities that are already known (the initial probability distribution and the detection probabilities of each action when no action has been applied before) as much as possible, and avoiding using the intermediate probability distributions. Although the updating rule looks complex, we can actually obtain these general expressions as stated below in Properties 7, 8, and 9. We are also interested in the regularities of the probability changes during the search process. The following lists the related properties.

Assumption 1: *In the following discussions, we assume that the situation that the calculated value of $P(\mathbf{f})$ is 1 but the target is not detected by $\mathbf{f}$ will not happen during the search process. In other words, if the target is not detected by action $\mathbf{f}$, than $P(\mathbf{f}) < 1$.*

Property 1 *The total sum of the probabilities of all the cubes (including the outside probability) after each action is 1.*

$$\sum_{i=1}^{n,o} \mathbf{p}^{[k]}(c) = 1 \tag{2.10}$$

Note: see Appendix A for the proof.

Property 2 *For any given action $\mathbf{f}$, we have*

Part A

$$0 \leq P(\mathbf{f}) \leq 1 \tag{2.11}$$

Part B

$P(\mathbf{f}) = 0$ *if and only if* $\forall c$, $\mathbf{p}(c, \tau_\mathbf{f}) = 0$ *or* $\mathbf{b}(c, \mathbf{f}) = 0$.

Part C

$P(\mathbf{f}) = 1$ *if and only if* $\mathbf{p}(c_o, \tau_\mathbf{f}) = 0$ *and* $\forall c, \mathbf{p}(c, \tau_\mathbf{f}) \neq 0$, $\mathbf{b}(c, \mathbf{f}) = 1$.

Note: see Appendix A for the proof.

Property 3 *After a failed action* $\mathbf{f}$*, the probability for any cube that is outside the influence range of* $\mathbf{f}$ *is increasing, that is:*

$$\mathbf{p}(c, \tau_{\mathbf{f}+}) \geq \mathbf{p}(c, \tau_{\mathbf{f}}) \tag{2.12}$$

for $c \notin \Omega(\mathbf{f})$

Note: see Appendix A for the proof.

Property 4 *After a failed action* $\mathbf{f}$*, the sum of the probabilities of the set of all cubes that is outside the influence range of* $\mathbf{f}$ *is increasing, that is:*

$$\sum_{c \notin \Omega(\mathbf{f})} \mathbf{p}(c, \tau_{\mathbf{f}+}) \geq \sum_{c \notin \Omega(\mathbf{f})} \mathbf{p}(c, \tau_{\mathbf{f}}) \tag{2.13}$$

Note: see Appendix A for the proof.

Property 5 *After a failed action* $\mathbf{f}$*, the sum of the probabilities of any cubes that are inside the influence range of* $\mathbf{f}$ *is decreasing, that is:*

$$\sum_{c \in \Omega(\mathbf{f})} \mathbf{p}(c, \tau_{\mathbf{f}+}) \leq \sum_{c \in \Omega(\mathbf{f})} \mathbf{p}(c, \tau_{\mathbf{f}}) \tag{2.14}$$

Note: see Appendix A for the proof.

Property 6 *Suppose two cubes* c *and* c^* *have the same probability value before an action* $\mathbf{f}$ *is applied,* $\mathbf{p}(c, \tau_{\mathbf{f}}) = \mathbf{p}(c^*, \tau_{\mathbf{f}})$*. Cube* c *belongs to the influence range of* $\mathbf{f}$*,* $c \in \Omega(\mathbf{f})$*. Cube* c^* *does not belong to the influence range of* $\mathbf{f}$*,* $c^* \notin \Omega(\mathbf{f})$*. Then after* $\mathbf{f}$ *is applied and fails, we have*

$$\mathbf{p}(c, \tau_{\mathbf{f}+}) < \mathbf{p}(c^*, \tau_{\mathbf{f}+})$$

Note: see Appendix A for the proof.

Property 7 *For allocation* $\mathbf{F} = \{\mathbf{f}_1, \ldots, \mathbf{f}_q \mid \mathbf{f}_i \in \mathbf{O}_{\mathbf{\Omega}}\}$*, we have*

$$\mathbf{p}^{[k]}(c) = \frac{\mathbf{p}^{[0]}(c) g(c, \mathbf{f}_1) g(c, \mathbf{f}_2) \ldots g(c, \mathbf{f}_k)}{(1 - P(\mathbf{f}_1))(1 - P(\mathbf{f}_2)) \ldots, (1 - P(\mathbf{f}_k))}, \qquad k = 1, \ldots, q \tag{2.15}$$

where

$$g(c, \mathbf{f}) = \begin{cases} 1 - \mathbf{b}(c, \mathbf{f}) & \text{if } c \in \Omega(\mathbf{f}) \\ 1 & \text{otherwise} \end{cases}$$

Note: see Appendix A for the proof.

Property 8 *For allocation* $\mathbf{F} = \{\mathbf{f}_1 \dots \mathbf{f}_q\}$, *we have*

$$\begin{aligned} P(\mathbf{f}_k) &= \frac{1}{(1-P(\mathbf{f}_1))\dots(1-P(\mathbf{f}_{k-1}))}\Bigg\{P^{[0]}(\mathbf{f}_k) \\ &\quad +(-1)^1\sum_{i_1=1}^{k-1}\Big(\sum_{c\in\Omega(\mathbf{f}_{i_1})\bigcap\Omega(\mathbf{f}_k)}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_{i_1})\mathbf{b}(c,\mathbf{f}_k)\Big) \\ &\quad +(-1)^2\sum_{1\le i_1<i_2\le k-1}\Big(\sum_{c\in\Omega(\mathbf{f}_{i_1})\bigcap\Omega(\mathbf{f}_{i_2})\bigcap\Omega(\mathbf{f}_k)}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_{i_1})\mathbf{b}(c,\mathbf{f}_{i_2})\mathbf{b}(c,\mathbf{f}_k)\Big) \\ &\quad +(-1)^3\sum_{1\le i_1<i_2<i_3\le k-1}\Big(\sum_{c\in\Omega(\mathbf{f}_{i_1})\bigcap\Omega(\mathbf{f}_{i_2})\bigcap\Omega(\mathbf{f}_{i_3})\bigcap\Omega(\mathbf{f}_k)}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_{i_1})\mathbf{b}(c,\mathbf{f}_{i_2})\mathbf{b}(c,\mathbf{f}_{i_3})\mathbf{b}(c,\mathbf{f}_k)\Big) \\ &\quad +\dots \\ &\quad +(-1)^{k-1}\sum_{c\in\Omega(\mathbf{f}_1)\bigcap\dots\bigcap\Omega(\mathbf{f}_{k-1})\bigcap\Omega(\mathbf{f}_k)}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_1)\mathbf{b}(c,\mathbf{f}_2)\dots\mathbf{b}(c,\mathbf{f}_{k-1})\mathbf{b}(c,\mathbf{f}_k)\Bigg\} \end{aligned} \tag{2.16}$$

for $k = 2, \dots, q$.

Note: see Appendix A for the proof.

Property 9 *For the given allocation* $\mathbf{F} = \{\mathbf{f}_1, \dots, \mathbf{f}_q\}$, *we have*

$$\begin{aligned} P(\mathbf{F}) &= \sum_{i_1=1}^{q} P^{[0]}(\mathbf{f}_i) \\ &\quad +(-1)^{2+1}\sum_{1\le i_1<i_2\le q}\Big(\sum_{c\in\Omega(\mathbf{f}_{i_1})\bigcap\Omega(\mathbf{f}_{i_2})}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_{i_1})\mathbf{b}(c,\mathbf{f}_{i_2})\Big) \\ &\quad \dots \\ &\quad +(-1)^{r+1}\sum_{1\le i_1<i_2<\dots<i_r\le q}\Big(\sum_{c\in\Omega(\mathbf{f}_{i_1})\bigcap\Omega(\mathbf{f}_{i_2})\bigcap\dots\bigcap\Omega(\mathbf{f}_{i_r})}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_{i_1})\mathbf{b}(c,\mathbf{f}_{i_2})\dots\mathbf{b}(c,\mathbf{f}_{i_r})\Big) \\ &\quad \dots \\ &\quad +(-1)^{q+1}\Big(\sum_{c\in\Omega(\mathbf{f}_1)\bigcap\Omega(\mathbf{f}_2)\bigcap\dots\bigcap\Omega(\mathbf{f}_q)}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_1)\dots\mathbf{b}(c,\mathbf{f}_q)\Big) \end{aligned} \tag{2.17}$$

Note: see Appendix A for the proof.

We give an intuitive explanation of Property 9. For a set of actions $\mathbf{f}_{i_1}$, $\mathbf{f}_{i_2}$, ..., and $\mathbf{f}_{i_k}$, if we define a virtual intersect action

$$\mathbf{f} = \mathbf{f}_{i_1} \cap \dots \cap \mathbf{f}_{i_k} \tag{2.18}$$

with property

$$\mathbf{b}(c, \mathbf{f}) = \prod_{j=1}^{k} \mathbf{b}(c, \mathbf{f}_{i_j}). \tag{2.19}$$

Then of course

$$\Omega(\mathbf{f}) = \Omega(\mathbf{f}_{i_1}) \cap \ldots \cap \Omega(\mathbf{f}_{i_k}) \tag{2.20}$$

Thus, Property 9 can be rewritten as:

For the given allocation $\mathbf{F} = \{\mathbf{f}_1, \ldots, \mathbf{f}_q\}$*, we have*

$$P(\mathbf{F}) = \sum_i P^{[0]}(\mathbf{f}_i) + \ldots + (-1)^{r+1} \sum_{i_1,\ldots,i_r} P^{[0]}(\mathbf{f}_{i_1} \cap \ldots \cap \mathbf{f}_{i_r}) + \ldots + (-1)^{q+1} P^{[0]}(\mathbf{f}_{i_1} \cap \ldots \cap \mathbf{f}_{i_q}). \tag{2.21}$$

The above calculation looks quite similar to the calculation of the total number of elements of the union of q sets A_1, ..., A_q:

$$\mid A_1 \cup \ldots \cup A_q \mid = \sum_i \mid A_i \mid + \ldots + (-1)^{r+1} \sum_{i_1,\ldots,i_r} \mid A_{i_1} \cap \ldots \cap A_{i_r} \mid + \ldots (-1)^{q+1} \mid A_1 \cap \ldots \cap A_q \mid \tag{2.22}$$

Property 10 *For a given allocation* $F = \{\mathbf{f}_1, \ldots, \mathbf{f}_q\}$*,* $P[\mathbf{F}]$ *is not influenced by the order of the applied actions. That is to say:* $P[\mathbf{F}]$ *is symmetric about actions* $\mathbf{f}_1, \ldots, \mathbf{f}_q$*.*

Note: this is a direct result from Property 9 and it agrees with common-sense expectation.

Property 11 *For an allocation* $\mathbf{F} = \{\mathbf{f_1}, \ldots, \mathbf{f_q}\}$*, if we restrict the available actions such that there is no cube belonging to any two action ranges:* $\Omega(\mathbf{f}_j) \bigcap \Omega(\mathbf{f}_i) = \phi$*, if* $i \neq j$*, then* $P[\mathbf{F}]$ *can be calculated by the following:*

$$P(\mathbf{F}) = \sum_{i=1}^{q} P^{[0]}(\mathbf{f}_i) \tag{2.23}$$

Note: this is also a direct result from Property 9.

2.6 The Complexity of the Object Search Task

In this section, we present the complexity level analysis of the sensor planning task for object search. Complexity considerations are commonplace in the biological and computational vision literature. For example, Tsotsos [75] shows that the general problem of visual search (search for a target within an image) is computationally intractable in a formal, complexity-theoretic sense.

Tsotsos [78] also ties the concept of active perception to attentive processing in general, and to his complexity level analysis of visual search, and proves that active unbounded visual search is NP-Complete. Kirousis and Papadimitriou [40] show that the problem of polyhedral scene labeling is inherently NP-Complete. Many other vision researchers ([32],[58], etc.) routinely provide an analysis of the complexity of their proposed algorithms. Complexity level analysis of robotics and vision problems is important because it can reveal basic insights into the structure of the problem and delimit the space of permissible solutions in a formal and theoretical fashion.

We prove in the following that the object search task is NP-complete. During the proof we assume that the detection function value $\mathbf{b}(c_i, \mathbf{f})$ is only influenced by the operation $\mathbf{f}$ and cube c_i. It is not influenced by the previous applied actions. We also assume that the cost of manipulating the hardware is the same for any action. Thus, the cost of an action $\mathbf{f}$ is influenced only by the cost of performing the recognition algorithm. It is not influenced by the geometric difference of the imaging parameters of the current action and the action just before the current action [3].

2.6.1 The Problem

First, the maximization problem is changed into the equivalent decision problem.

WHAT WE HAVE: A search region $\Omega = \bigcup_{i=1}^{n,o} c_i$ and the corresponding target distribution; a finite set $\mathbf{O_\Omega} = \{\mathbf{f}_1^*, \mathbf{f}_2^*, \ldots, \mathbf{f}_m^*\}$ (the set of all possible operations that can be applied); an expected probability function $P[\mathbf{F}']$ as described in Formula (2.6); a cost function $T[\mathbf{F}']$ as described in Formula (2.7) for an effort allocation $\mathbf{F}'$, where $\mathbf{F}' = \{\mathbf{f}_1, \ldots, \mathbf{f}_k \mid \mathbf{f}_i \in \mathbf{O_\Omega}\}$ is the ordered set of operations applied in the search; an environmental updating rule as described in Formula (2.9); a cost constraint K; and an expected probability goal M.

WHAT WE WANT TO KNOW: Is there an effort allocation $\mathbf{F}$ such that $T[\mathbf{F}] \leq K$ and $P[\mathbf{F}] \geq M$?

Now, we describe the problem in a more theoretical fashion.

The search region is divided into n little cubes, $c_1, \ldots, c_n$. We use c_o to represent the outside region. Each of c_i ($1 \leq i \leq n$, or $i = o$) is assigned a probability value $\mathbf{p}(c_i, \tau_0)$ before the search process. We use an array $A[n]$ to represent these probability distributions.

$$\begin{aligned} A[0] &= \mathbf{p}(c_o, \tau_0) \\ A[1] &= \mathbf{p}(c_1, \tau_0) \\ &\vdots \\ A[n] &= \mathbf{p}(c_n, \tau_0) \end{aligned}$$

[3] In some situations, this assumption is not true. For example, in the search model discussed in this thesis, if the robot position of the previous action is different from the current action, then the cost is much bigger then the cost when the position of the previous action is the same as the current position. But there also exists situations such that this assumption is true. For example, when the searcher is only a pan, tile, and zoom camera and the camera's position is not able to change. To prove a NP problem is a NP-complete problem, we just need to prove a special instance of the NP problem is NP hard. So, this assumption does not influence the NP-complete property of the task

The values of $A[i]$ $(1 \leq i \leq n$, or $i = o)$ satisfy

$$A[0] + A[1] + \cdots + A[n] = 1$$

Each $A[i] \geq 0$ is assumed to be a rational number

$$A[i] = \frac{p_i}{q_i} > 0$$

where $p_i \in Z^+$ and $q_i \in Z^+$.

Before the search process, we are given a set of actions, $\mathbf{O_\Omega} = \{\mathbf{f}_1^*, \mathbf{f}_2^*, \ldots, \mathbf{f}_m^*\}$, that can be selected from. For any action $\mathbf{f}_j^*$ and any cube c_i, there is a detection function $\mathbf{b}(c_i, \mathbf{f}_j^*)$ associated with them. We use a $2D$ array $B[i][j]$ $(0 \leq i \leq n$ and $0 \leq j \leq m)$ to represent it,

$$B[i][j] = \mathbf{b}(c_i, \mathbf{f}_j^*)$$

Each $B[i][j]$ $(0 \leq i \leq n$ and $0 \leq j \leq m)$ is assumed to be a rational number between 0 and 1, $B[i][j] = \frac{p_{ij}}{q_{ij}}$, where $p_{ij} \in Z^+$, $q_{ij} \in Z^+$.

After an action $\mathbf{f}_j^*$ is applied, the value of $\mathbf{p}(c_i, \tau)$ is updated according to Formula (2.9). In our new representation, this updating rule can be represented as

$$A[i] \leftarrow \frac{A[i](1 - B[i,j])}{A[0] + \sum_{k=1}^{n} A[k](1 - B[k,j])}, \quad i = 0, 1, \ldots, n \tag{2.24}$$

So, the value of $A[i]$ will be updated according to Formula (2.24) each time when an action is applied.

The probability of detecting the target by a given action $\mathbf{f}_j^*$ is given by

$$P(\mathbf{f}_j^*) = \sum_{i=1}^{n} \mathbf{p}(c_i, \tau_{\mathbf{f}_j^*})\mathbf{b}(c_i, \mathbf{f}_j^*)$$

In the new representation, $P(\mathbf{f}_j^*)$ is represented as $P[\langle j \rangle]$, thus

$$P[\langle j \rangle] = \sum_{i=1}^{n} B[i,j]A[i] \tag{2.25}$$

Note, the value of $A[i]$ used in Formula (2.25) is the most recent updated value of $A[i]$.

For effort allocation $\mathbf{F} = \{\mathbf{f}_{j_1}^*, \ldots, \mathbf{f}_{j_k}^*\}$, the probability of detecting the target is given by

$$\begin{aligned} P[\mathbf{F}] \quad &= \quad P(\mathbf{f}_{j_1}^*) + [1 - P(\mathbf{f}_{j_1}^*)]P(\mathbf{f}_{j_2}^*) \\ &\quad\quad + \ldots \\ &\quad\quad + \{\prod_{i=1}^{k-1}[1 - P(\mathbf{f}_{j_i}^*)]\}P(\mathbf{f}_{j_k}^*) \end{aligned} \tag{2.26}$$

In the new representation, this effort allocation is represented as $\mathbf{F} = \langle j_1, \ldots, j_k \rangle$, and the probability of detecting the target is calculated by the following formula:

$$
\begin{aligned}
P[\langle j_1, \ldots, j_k \rangle] \quad &= \quad P[\langle j_1 \rangle] + [1 - P[\langle j_1 \rangle]] P[\langle j_2 \rangle] \\
&\quad + \ldots \\
&\quad + \{ \prod_{i=1}^{k-1} [1 - P[\langle j_i \rangle]] \} P[\langle j_k \rangle]
\end{aligned} \tag{2.27}
$$

Note, in the above calculation, the value of $A[l]$ $(1 \leq l \leq n)$ used in calculating $P[\langle j_i \rangle]$ is the value of $A[l]$ just before the action $\langle j_i \rangle$ is applied.

In the new representation, the cost by $\mathbf{f}_i^*$ is represented as $C[i]$ $(C[i] = \mathbf{t}(\mathbf{f}_i^*))$. So, the cost for $\langle j_1, \ldots, j_k \rangle$ is represented by

$$T[\langle j_1, \ldots, j_k \rangle] = \sum_{i=1}^{k} C[j_i] \tag{2.28}$$

With the above description, the task of sensor planning for object search can be described as

INSTANCE

- an index set $\Big\{1, \ldots, m\Big\}$;
- a probability set $\Big\{A[i] \mid 0 \leq i \leq n\Big\}$, $A[i]$ $(0 \leq i \leq n)$ are a rational numbers, $A[i] = \frac{p_i}{q_i}, p_i \in Z^+, q_i \in Z^+, 0 \leq A[i] \leq 1$, $A[i]$ satisfy $\sum_{i=0}^{n} A[i] = 1$;
- a detection function set $\Big\{B[i][j] \mid 0 \leq i \leq n, 0 \leq j \leq m\Big\}$, $B[i][j]$ $(0 \leq i \leq n, 0 \leq j \leq m)$ are rational numbers, $B[i][j] = \frac{p_{ij}}{q_{ij}}, p_{ij} \in Z^+, q_{ij} \in Z^+, 0 \leq B[i][j] \leq 1$;
- a cost set $\Big\{C[i] \mid 1 \leq i \leq m\Big\}$, $C[i]$ ($1 \leq i \leq m$) are rational numbers, $C[i] = \frac{C_{pi}}{C_{qi}}$, $C_{pi} \in Z^+, C_{qi} \in Z^+$;
- a cost limit constant K (K is a rational number, $K = \frac{K_p}{K_q}$);
- a goal M (M is a rational number, $M = \frac{M_p}{M_q}$).

QUESTION

Is there an ordered set $\langle j_1, \ldots, j_k \rangle$ ($j_i \in \{1, \ldots, m\}$ for $1 \leq i \leq k$), such that $T[\langle j_1, \ldots, j_k \rangle] \leq K$ and $P[\langle j_1, \ldots, j_k \rangle] \geq M$?

2.6.2 Object Search Task is an NP Problem

The following nondeterministic algorithm solves this problem:

```
procedure NondeterministicObjectSearch(A,B,C,K, M)
    rational A[0:n], B[0:n][1:m], C[1:m], K,M
    integer E[0:m], F[0:m], len
    for i ← 1 to m do
        E[i] ← choice(0, 1, 2, ..., m)
    end for
    for i ← 1 to m do
        F[E[i]] = i
    end for
    len = 0
    i = 1
    while F[i] ≠ 0
        len ← len + 1
        i++
    end while
    for i = 1, i ≤ len, i++
        P[i] = 0
        for j = 1, j ≤ n, j++
            P[i] = P[i] + B[j, i] * A[j]
        end for
        for j = 1, j ≤ n, j++
            Temp_a = A[j] * (1 − B[j, i])
            Temp_b = 0
            for k = 0, k ≤ n, k++
                Temp_b = Temp_b + A[k](1 − B[k, i])
            end for
            A[j] = Temp_a / Temp_b
        end for
    end for
    P_F = 0
    Term_a = 1; Term_b = 0
    F[0] = 0; P[0] = 0
    for i = 1, i ≤ len, i++
        Term_a = Term_a * (1 − P[F[i − 1]])
        Term_b = P[F[i]]
        P_F = P_F + Term_a * Term_b
    end for
    T_F = 0
```

```
(38)            for i = 1, i ≤ len, i++
(39)                T_F = T_F + C[F[i]]
(40)            end for
(41)            if T_F > K or P_F < M then
(42)                failure
(43)            else
(44)                success
(45)            endif
(46)    end NondeterministicObjectSearch
```

In the following, we analyze the above nondeterministic object search algorithm.

- (4) —– (6)
  - **Explain** - Array $E[0:m]$ is defined as follows: if $E[i] = k^{'}$ $(1 \leq k^{'} \leq m)$ then $\mathbf{f}_i^*$ will be the $k^{'}$th action in the effort allocation; if $E[i] = 0$, then $\mathbf{f}_i^*$ does not belong to the effort allocation.
  - **Steps**: m
- (7) —– (14)
  - **Explain** - Array $F[0:m]$ gives the ordered set of actions selected as the effort allocation. len gives the number of actions in the effort allocation. The ordered actions for the effort allocation are: $\langle F[1], \ldots, F[len] \rangle$.
  - **Steps**: $m + 2 + 2m$.
- (15) —– (28)
  - (16) —– (19)
    - **Explain** - $P[i]$ gives the probability of detecting the target for the ith action in the effort allocation (Formula (2.25)).
    - **Steps**: $n + 1$.
  - (20) —– (27)
    - **Explain** - $A[0:n]$ gives the target distribution. This set of commends update the distribution after the ith action is applied (Formula (2.24)).
    - **Steps**: $n(n + 3)$
  - **Steps** for ((15) —– (28)): m(n+1 + n(n+3))
- (29) —– (36)

  - **Explain** - P_F gives the probability of detecting the target by the selected effort allocation (Formula 2.27).
  - **Steps** - $3 + m$

- (37) —– (40)
  - **Explain** - T_F gives the cost by the selected effort allocation.
  - **Steps** - $m + 1$.

From the above discussion, it is obvious that the computation steps in the nondeterministic algorithm *NondeterministicObjectSearch* is of the order of $O\Big(m + (m+2+2m) + m(n+1+n(n+3)) + (3+m) + (m+1)\Big)$, that is $O\Big(mn^2\Big)$.

The input length to the algorithm is of the order of $\Big(m+n+max\{\lceil log_2 p_i\rceil\} + max\{\lceil log_2 q_i\rceil\} + max\{\lceil log_2 p_{ij}\rceil\} + max\{\lceil log_2 q_{ij}\rceil\} + max\{\lceil log_2 C_{pi}\rceil\} + max\{\lceil log_2 C_{qi}\rceil\} + max\{\lceil log_2 K_p\rceil\} + max\{\lceil log_2 K_q\rceil\} + max\{\lceil log_2 M_p\rceil\} + max\{\lceil log_2 M_q\rceil\}\Big)$

Thus, it is obvious that sensor planning task belongs to NP.

It is interesting to point out that if we calculate P_F using other methods, the complexity can be exponential. For example, if we use Property 9 to calculate, the complexity is exponential. If we first transform P_F like following, we also get an exponential complexity:

$$\begin{aligned} P[\mathbf{F}] &= P(\mathbf{f}^*_{j_1}) + [1 - P(\mathbf{f}^*_{j_1})]P(\mathbf{f}^*_{j_2}) + \cdots + \{\prod_{i=1}^{k-1}[1 - P(\mathbf{f}^*_{j_i})]\}P(\mathbf{f}^*_{j_k}) \\ &= [\sum_{i=1}^{k} P(\mathbf{f}^*_{j_i})] - [\sum_{1\le i<j\le k} P(\mathbf{f}^*_{j_i})P(\mathbf{f}^*_{j_j})] + \cdots + [(-1)^{k+1}P(\mathbf{f}^*_{j_1})\cdots P(\mathbf{f}^*_{j_k})] \end{aligned} \tag{2.29}$$

2.6.3 The Sensor Planning for Object Search Task is NP-Complete

In this section, we use proof by restriction to prove that the object search task belongs to the NP-complete class. An NP-completeness proof by restriction for a given problem $\Pi \in NP$ consists simply of showing that Π contains a known NP-complete problem Π' as a special case. The heart of such a proof lies in the specification of the additional restrictions to be placed on the instances of Π so that the resulting restricted problem will be identical to Π'.

Theorem 1 The sensor planning for object search task described in Section 2.6.2 is NP-Complete.

Proof:

We prove the theorem by restriction.

The $KNAPSACK$ problem is a well known NP-Complete problem [27]. We will restrict the object search task to $KNAPSACK$ by allowing instances in which any two operations in $\mathbf{O_\Omega} = \{\mathbf{f_1}^*, \mathbf{f_2}^*, \ldots, \mathbf{f_m}^*\}$ do not have a common influence region. This is to say that $\Omega(\mathbf{f_i}) \bigcap \Omega(\mathbf{f_j}) = \phi$ for any $i \neq j$.

After this restriction, by combining Property 10 and Property 11, the task becomes

INSTANCE

- A finite action set $U = \Big\{1, \ldots, m\Big\}$
- An initial detection probability $P^{[0]}[u]$ for each $u \in U$
- A cost $C[u]$ for each $u \in U$
- a cost limit constant K (K is a rational number);
- a goal M (M is a rational number).

QUESTION: Is there a subset $\mathbf{F} \subseteq U$ such that

$$\sum_{u \in \mathbf{F}} P^{[0]}[u] \leq K$$

and

$$\sum_{u \in \mathbf{F}} C[u] \geq M$$

Obviously the above problem is equivalent to the $KNAPSACK$ problem. Thus, the task of sensor planning for object search is NP-Complete.

$\Box$.[4]

Note, the task of sensor planning for object search is not equivalent to the $KNAPSACK$ problem, only a special case of the sensor planning task is equivalent to the $KNAPSACK$ problem.

The conclusion that the task of sensor planning for object search is NP-Complete means that no provably optimal algorithm in polynomial time exists, and thus a good approximation algorithm is probably all one can get.

[4]To get a more understanding of the above proofs, please refer to Appendix B

2.7 The Simplified Version: a One Step Look Ahead Problem

As noted in Section 2.4, the task of object search is actually a task of finding the "best" sequence of actions $\mathbf{F} = \{\mathbf{f}_1, \ldots, \mathbf{f}_k\}$ that will maximize the probability of detecting the target $P[\mathbf{F}]$ (see Formula (2.6)) given the condition that the total cost $T[\mathbf{F}]$ (see Formula (2.7)) is less than a constant. Since this task is NP-complete, and usually the number of the available candidate actions is big, it is necessary to relax the original problem. Instead of looking for an algorithm that always generates an optimal solution, we will simply use heuristics that will generate a feasible solution for the original problem. Since the calculation of the detection probability for a given operation is time consuming, it is not practical to use complex heuristics strategies. Here we use the greedy strategy. The greedy method suggests that one can devise an algorithm which works in stages, considering one input at a time. At each stage, based on some optimization measure, the next candidate is selected and is included into the partial solution developed so far. We make the object search task simpler by changing it to a "one step look ahead problem": decide only which is the very next action to execute, considering the effect and cost of that action.

2.7.1 The Objective Function

Suppose we have already executed q ($q \geq 1$) actions $\mathbf{F}_q = \{\mathbf{f}_1, \ldots, \mathbf{f}_q\}$. We now want to find the next action $\mathbf{f}_{q+1}$ to execute, with the hope that our strategy of finding the next action may lead to an **approximate** solution of the object search task. Suppose $P[\mathbf{F}_q]$ is the expected probability of detecting the target by allocation $\mathbf{F}_q$. $T[\mathbf{F}_q]$ is the cost of allocation $\mathbf{F}_q$. When we execute a next action $\mathbf{f}$, the effort allocation becomes $\mathbf{F}_{q+1} = \{\mathbf{f}_1, \ldots, \mathbf{f}_q, \mathbf{f}\}$. The probability of detecting the target becomes

$$P[\mathbf{F}_{q+1}] = P[\mathbf{F}_q] + \{ \prod_{j=1}^{q} [1 - \sum_{i=1}^{n} \mathbf{p}(c_i, \tau_{\mathbf{f}_j}) \mathbf{b}(c_i, \mathbf{f}_j)] \} \times [\sum_{i=1}^{n} \mathbf{p}(c_i, \tau_{\mathbf{f}}) \mathbf{b}(c_i, \mathbf{f})]$$

The total cost becomes

$$T[\mathbf{F}_{q+1}] = T[\mathbf{F}_q] + \mathbf{t}(\mathbf{f})$$

where $\mathbf{t}(\mathbf{f})$ is the time needed to select and execute action $\mathbf{f}$.

So, the contribution of $\mathbf{f}$ to the probability of detecting the target is

$$\Delta_P(\mathbf{f}) = \{ \prod_{j=1}^{q} [1 - \sum_{i=1}^{n} \mathbf{p}(c_i, \tau_{\mathbf{f}_j}) \mathbf{b}(c_i, \mathbf{f}_j)] \} \times [\sum_{i=1}^{n} \mathbf{p}(c_i, \tau_{\mathbf{f}}) \mathbf{b}(c_i, \mathbf{f})]$$

The additional cost needed in order to get this probability contribution is $\Delta_T(\mathbf{f}) = \mathbf{t}(\mathbf{f})$.

Our strategy for selecting the next action is: the next action $\mathbf{f_{q+1}}$ should be selected that maximizes the term

$$\frac{\Delta_P(\mathbf{f})}{\Delta_T(\mathbf{f})}$$

Since the term

$$\{ \prod_{j=1}^{q}[1 - \sum_{i=1}^{n} \mathbf{p}(c_i, \tau_{\mathbf{f}_j})\mathbf{b}(c_i, \mathbf{f}_j)] \}$$

is fixed no matter what next action $\mathbf{f}$ is selected, our strategy becomes (Note: this is also the strategy when $q = 0$): the next action $\mathbf{f_{q+1}}$ should be selected that maximizes the term

$$\mathbf{E}(\mathbf{f}) = \frac{\sum_{i=1}^{n} \mathbf{p}(c_i, \tau_{\mathbf{f}})\mathbf{b}(c_i, \mathbf{f})}{\mathbf{t}(\mathbf{f})}$$

That is, we should select the action that has the maximum utility value $\mathbf{E}(\mathbf{f})$.

2.7.2 Minimize the Expected Time of Detecting the Target

In our approach, the task of sensor planning for object search is formulated as a task to maximize the probability of detecting the target with a given cost. We have also proved in Section 2.5 that for a given effort allocation, the probability of detecting the target is not influenced by the order of the applied actions. But this result does not suggest that we can apply a given set of actions in any random order. Although the criterion of detection probability is very important, it is not the only criteria that a search agent should take into the consideration. Another very important criterion is the time used to detect the target. It is obvious that the search agent wants to find the target as early as possible. For this reason, we introduce the concept of "expected time to detect the target" for a given effort allocation $\mathbf{F} = \{\mathbf{f}_1, \ldots, \mathbf{f}_k\}$.

Definition 1 *For a given effort allocation* $\mathbf{F} = \{\mathbf{f}_1, \ldots, \mathbf{f}_k\}$*, the expected time to detect the target by applying* $\mathbf{F}$ *is defined as*

$$\begin{aligned} \Theta[\mathbf{F}] &= \mathbf{t}(\mathbf{f}_1) \times P(\mathbf{f}_1) \\ &\quad + \Big(\mathbf{t}(\mathbf{f}_1) + \mathbf{t}(\mathbf{f}_2)\Big) \times [1 - P(\mathbf{f}_1)]P(\mathbf{f}_2) \\ &\quad + \ldots \\ &\quad + \Big(\mathbf{t}(\mathbf{f}_1) + \ldots + \mathbf{t}(\mathbf{f}_k)\Big) \times \{\prod_{i=1}^{k-1}[1 - P(\mathbf{f}_i)]\}P(\mathbf{f}_k) \end{aligned} \tag{2.30}$$

By using Property 8, it is easy to get the following property.

Property 12 *For an allocation* $\mathbf{F} = \{\mathbf{f_1}, \ldots, \mathbf{f_q}\}$*, if we restrict the available actions such that there is no cube belonging to any two influence ranges:* $\Omega(\mathbf{f}_j) \bigcap \Omega(\mathbf{f}_i) = \phi$*, if* $i \neq j$*, then* $\Theta[\mathbf{F}]$ *can be calculated by the following:*

$$\begin{aligned} \Theta[\mathbf{F}] &= \mathbf{t}(\mathbf{f}_1) \times P^{[0]}(\mathbf{f}_1) \\ &\quad + \Big(\mathbf{t}(\mathbf{f}_1) + \mathbf{t}(\mathbf{f}_2)\Big) \times P^{[0]}(\mathbf{f}_2) \\ &\quad + \ldots \\ &\quad + \Big(\mathbf{t}(\mathbf{f}_1) + \ldots + \mathbf{t}(\mathbf{f}_k)\Big) \times P^{[0]}(\mathbf{f}_k) \end{aligned} \tag{2.31}$$

Before pointing out some advantages of our approach, we first give the following property of our strategy.

Property 13 *For a set of actions* $\{\mathbf{f}_1, \ldots, \mathbf{f}_q\}$*, suppose that*
(1) There is no cube belonging to any two influence ranges: $\Omega(\mathbf{f}_j) \bigcap \Omega(\mathbf{f}_i) = \phi$*, if* $i \neq j$*;*
(2) $\frac{P^{[0]}(\mathbf{f}_1)}{\mathbf{t}(\mathbf{f}_1)} > \frac{P^{[0]}(\mathbf{f}_2)}{\mathbf{t}(\mathbf{f}_2)} > \ldots > \frac{P^{[0]}(\mathbf{f}_q)}{\mathbf{t}(\mathbf{f}_q)}$.
then the order of applying these actions by our one step look ahead strategy must be: $\mathbf{f}_1 \rightarrow \mathbf{f}_2 \rightarrow \ldots \rightarrow \mathbf{f}_q$.

Note: see Appendix A for the proof.

One important advantage of our one step look ahead strategy is stated in Property 14.

Property 14 *For a group of actions* $\{\mathbf{f_1}, \ldots, \mathbf{f_q}\}$*, if there is no cube belonging to any two influence ranges:* $\Omega(\mathbf{f}_j) \bigcap \Omega(\mathbf{f}_i) = \phi$*, if* $i \neq j$*, then the application order of the actions selected by our strategy minimizes the expected time of detecting the target with respect to this group.*

Note: see Appendix A for the proof.

2.7.3 How Good is Our Algorithm

The greedy strategy cannot always generate the best sequence. The following is a simple example of situation where it fails. Suppose the search region $\Omega = \bigcup_{i=1}^{5} c_i$. The available actions are: $\mathbf{O_\Omega} = \{\mathbf{f}_1, \ldots, \mathbf{f}_5\}$. The cost for these actions are: $\mathbf{t}(\mathbf{f}_1) = \mathbf{t}(\mathbf{f}_2) = \mathbf{t}(\mathbf{f}_3) = 5$, $\mathbf{t}(\mathbf{f}_4) = \mathbf{t}(\mathbf{f}_5) = 4.5$. The cost constraint is $K = 19$. The probability distribution and the detection functions are listed in the following:

c_i	$\mathbf{p}(c_i)$	$\mathbf{b}(c_i, \mathbf{f}_1)$	$\mathbf{b}(c_i, \mathbf{f}_2)$	$\mathbf{b}(c_i, \mathbf{f}_3)$	$\mathbf{b}(c_i, \mathbf{f}_4)$	$\mathbf{b}(c_i, \mathbf{f}_5)$
1	0.19	0.7	0	0	0	0
2	0.19	0	0.7	0	0	0
3	0.19	0	0	0.7	0	0
4	0.19	0	0	0	0.7	0
5	0.19	0	0	0	0	0.7
c_o	0.05	0	0	0	0	0

The effort allocation generated by the greedy strategy are $\mathbf{F}_{greedy} = \{\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3\}$. The probability of detecting the target is: $P[\mathbf{F}] = 0.399$.

The optimal effort allocation are $\mathbf{F}_{optimal} = \{\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_5, \mathbf{f}_4\}$. The probability of detecting the target is: $P[\mathbf{F}] = 0.437$.

Although the greedy strategy does not always generate the best sequence, it can generate satisfactory answers in general. Usually, when we search for an object, we want to maximize the probability of detecting the target at any time during the search process, so that we can find the target as early as possible. This justifies our "one step look ahead" strategy: always apply the most promising operations.

In some situations, our strategy may even generate the optimal answers as stated in Property 15.

Property 15 *Suppose the available operations* $\mathbf{O}_{\mathbf{\Omega}} = \{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_m\}$ *satisfy:*
(1) $\mathbf{t}(\mathbf{f}_i) = \mathbf{t}(\mathbf{f}_j)$, $1 \leq i, j \leq m$.
(2) $\Omega(\mathbf{f}_j) \bigcap \Omega(\mathbf{f}_i) = \phi$, $1 \leq i, j \leq m, i \neq j$.
Then, our simple strategy will generate the optimal answer.

Note: see Appendix A for the proof.

Since the evaluation of $E(\mathbf{f})$ is very difficult for actions whose positions are different from current position, we further simplify the above problem by dividing it into two subproblems. The first is "where to look next" problem, which answers the question how to select the next action so as to maximize $E(\mathbf{f})$ for a fixed camera center position. The second is the "where to move next" problem, which answers the question how to select the next robot position when the benefit with respect to the current position is too low. Although this further simplification may not always provide an optimal answer for the above simplified problem when the operation set is $\mathbf{O}_{\mathbf{\Omega}}$, it does provide a suboptimal answer when the operation set is restricted to actions of current position.

Part II

The Detection Function

Chapter 3

The Detection Function

In Chapter 1, we identified three subtasks for an object search system, one of them being the task of how to determine whether or not the target is within the image taken by the camera. This is the object recognition problem. In Chapter 2, we argued that a sensor planning strategy should take the ability of the object recognition algorithm into consideration. In Section 2.4, we propose to use the concept of detection function and the cost for a given action to incorporate the ability of the recognition algorithm into the sensor planning strategy. It is important to stress that the detection function as used here is not intended to capture all aspects of object recognition. It is needed in order to avoid incurring the expense of executing object recognition algorithms where they would be fruitless. Although, in principle, the problem we address can indeed be solved by brute-force search for object in all possible images, we wish to devise a much more efficient method by using the detection function to select those imaging geometries which have the best potential for yielding useful results from object recognition algorithms.

In the following sections, we discuss in detail the concept of detection function. We first give a brief overview of research in object recognition in order to summarize the important issues which must be considered for the detection function. We then introduce the concept of a standard detection function. Finally we discuss the concept of detection function in the general situation.

3.1 Object Recognition

Object recognition is the task of finding and labeling parts of a two-dimensional image of a scene that correspond to an object in the scene [14] [68] [5] [86]. To perform the object recognition task, we need to establish models, or general descriptions of each object to be recognized. A model may include shape, texture, and context knowledge about the occurrence of such objects in a scene. A model label can be thought of as a tag pinned to an area in the image that we believe shows an instance of the corresponding object model. We may think of object recognition as the process of taking image data as input and emitting one or more labels for subsets of image data, such that each subset is a representation of an object and the corresponding label is the object identity. There are several desirable properties that a recognizer should have [86]. The first is that when an object in the system's known universe is present, the system should emit a label for the object; when the object is not present, its label should not be emitted. The second is that the recognizer should exhibit competence in a wide variety of circumstances. The third is that a recognizer should exhibit a pleasing tradeoff, with competence and robustness on the one hand, and computational resources used on the other [86].

Object recognition is difficult. One of the difficulties comes from the process of identifying objects in the image. Because a combination of factors must be used in the identification process. These factors may include restrictions on allowable shapes, the semantics of the scene context, and the information present in the image itself. The other difficulty comes from the image formation process. Because the state parameters of the camera and the condition of the surrounding environment have a great influence on the quality of the resulted image. From the feature detection aspect, if the features used by the recognizer are not detectable in the image, the recognizer may produce a negative response even when the desired object is in the field of view. Feature detection can be hindered by three kinds of imaging phenomena: low contrast, poor focus, or insufficient resolution. Low contrast between scene parts is due to unfavorable interaction between the existing illumination and surface reflectance and geometry. Insufficient resolution is caused by the unfavorable relation among the camera's viewing angle size, the scene feature size and the distance of the feature to the camera center. With respect to image projection, there are problems caused by occlusion, view degeneracy and non-inverting projection property of the image formation process. Features may be occluded by other objects between the camera and the target object, or occluded by the target object itself. Features may also be missing because they are out of the field of view of the camera. Due to the accidental alignment of spatially distinct scene parts from the camera viewpoint, features may be difficult to identify. The aligned parts may be part of the same object or different objects. This superposition of features caused by view degeneracy can lead to erroneous part counts or bad parameterizations of a model. From the image formation geometry, the effects of projection of a three-dimensional world onto two dimensions are irreversible. The three-dimensional structure of the world cannot be recovered from a single two-dimensional image without making very restrictive assumptions about the world being imaged. This adds difficulty in the inference process of object recognition, as the system need explicitly account for the effects of projection by considering any possibility of variation in camera viewpoint.

Although object recognition is very difficult, a lot of work has been done in this area. One of the reasons being that if the object recognition problem could be solved successfully by a vision system, that system would be extremely useful in a wide variety of applications, including automatic inspection and assembly, and autonomous vehicle navigation, etc. The computational strategies used by previous work in object recognition may be roughly classified into four classes according to two main characteristics: their suitability for complex image data and their suitability for complex models [68]. The first class is the "Feature Vector Classification" method. Feature vector methods rely on a trivial model of an object's image characteristics and are typically applied only to simple data. The second class is the "Fitting Models to Photometry" method. Such methods search for features with predetermined global shapes and photometric properties. These methods are particularly suitable when simple models are sufficient but the photometric data are noisy and ambiguous. The third class is the "Fitting Models to Symbolic Structures" method. Such approaches typically look for instances of objects by matching data structures that represent relations among object parts. These procedures are appropriate when complex models are required but reliable symbolic structures can be accurately inferred from simple data. The fourth class is the "Combined Strategies" method. This approach is required when both the data and the desired model instances are complex.

Because there is no perfect object recognizer, it is very important to evaluate the performance of different object recognition algorithms. Since different researchers in object recognition do not evaluate their systems in any consistent manner, the information in the literature makes it practically impossible to compare existing object-recognition systems quantitatively. Thus, there is very little work on quantitative performance evaluation of different recognition algorithms. Researchers normally only make subjective comparisons of different techniques.

But for the object search task, in order to incorporate the ability of the recognition algorithm into the planning strategy, we need to construct a performance evaluation matrix such that the recognition ability of the algorithm can be quantitatively represented. One way to compare different recognition algorithms might be to run the recognition algorithms on standardized sets of images and to compare the number of successful recognition cases for each algorithm. This approach however is not appropriate, because it is very difficult to select the standard test images and the resulting test images might be subjective. Furthermore, this approach cannot be incorporated into the sensor planning process because the camera's state parameters are not considered. What we need is an evaluation matrix that can take into consideration the influences of the camera's state parameters, the relative configuration between the camera and the target object, and the success rate of the recognition algorithm. In order to do this, we propose the concept of "detection function" for a recognition algorithm, and we argue that the "detection function" representation can reasonably represent the ability of the recognition algorithms and can incorporate that ability into the sensor planning process. A detection function as described here is novel, our purpose is not to do research in object recognition, but rather to derive a measure useful for object search representing quantitative performance with varying system geometries.

From a certain point of view, the task of object recognition is different from the task of detection. Object recognition refers to the task of finding and labeling parts of a two-dimensional image of a scene that correspond to an object in the scene. Detection refers to the task of identify an object with the given camera's parameter settings and the given recognition algorithm. Thus, there are more factors considered in detection. The concept of detection function is a function used to describe the statistical detection result with a given set of camera parameters and a given recognition when the target orientation and the background situation is allowed to change.

It is interesting to note that Schiele and Crowley [66] recently propose a statistical approach to answer the question "where to look next". Their approach is based on the use of multidimensional histograms of local neighborhood operators and a network of salient points. The multidimensional receptive field histograms serve as basis to generate a hypothesis for the presence of an object. The network of salient points enables the technique to determine the next region of interest to drive fixation and to verify the generated hypothesis. The hypothesis generation is mainly based on a probability recognition algorithm which can calculate probabilities for each object at an arbitrary image location. By choosing the most probable objects one can localize the actual region of interest in the image in order to find a salient point of one of the objects in the object database. Once a salient point of one of the most probable object is found, they *hypothesize* this object. Using the network of salient points of the hypothesized object one can choose the most significant (or the next most significant) salient point in the network to determine the next region of interest (*where to look next*). The verification steps is continued until the evidence of the object is sufficient for confirmation of the hypothesis.

3.2 The Concept

The "detection function" representation can specify the detection ability of the recognition algorithm by incorporating various factors that influence the performance of the recognition algorithm. The detection function $\mathbf{b}^*(<\theta, \delta, l>, <w, h>)$ gives the probability of detecting the target when the relative position of the center of the target to the camera (see Section 2.3) is (θ, δ, l) and the camera's view angle size is $<w, h>$. Angles θ and δ are the horizontal and vertical direction offset to the viewing axis of the camera, and l is the length of the projection of the line segment, which connects the camera center and the target center, on the viewing axis (see Section 2.3 for details). Among the parameters θ, δ, l, and $<w, h>$, the influences of $<w, h>$ and l on the detection function value are big, because they determine the resolution and scale of the target image; the influences of θ and δ on the detection function value are small, because object recognition algorithms are usually pretty shift-invariant and thus the perspective distortion does not influence much. The center of the target refers to the geometric center of the target. Its coordinate (x_c, y_c, z_c) in the Camera Coordinate System can be calculated as

$$x_c = \frac{\int\int\int_{\Omega_{Target}} x dx dy dz}{\int\int\int_{\Omega_{Target}} dx dy dz} \tag{3.1}$$

$$y_c = \frac{\int\int\int_{\Omega_{Target}} y dx dy dz}{\int\int\int_{\Omega_{Target}} dx dy dz} \tag{3.2}$$

$$z_c = \frac{\int\int\int_{\Omega_{Target}} z dx dy dz}{\int\int\int_{\Omega_{Target}} dx dy dz} \tag{3.3}$$

where Ω_{Target} is the $3D$ region occupied by the target object. The above calculations are performed with respect to the Camera Coordinate System.

When (x_c, y_c, z_c) is available, (θ, δ, l) can then be calculated according to equation (2.1), (2.2), and (2.3).

The detection function representation $\mathbf{b}^*(<\theta, \delta, l>, <w, h>)$ has fixed the factors of the camera's viewing angle size and the relative position of the target center. The influences of other factors such as the relative orientation of the target with respect to the camera, the illumination condition, and the background situations are incorporated as statistical values in the function $\mathbf{b}^*(<\theta, \delta, l>, <w, h>)$. Theoretically, we can put the target at $<\theta, \delta, l>$ and perform experiments under all the possible situations. The final value of $\mathbf{b}^*(<\theta, \delta, l>, <w, h>)$ equals to the total number of successful recognitions divided by the total number of experiments.

Suppose (1)the available relative orientations of the target with respect to the camera configuration are $\mathbf{O} = \{O_1, \ldots, O_{k_o}\}$ (Note, it is obvious that there is an infinite number of such orientations). We assume here that this infinite set of orientations is quantized uniformally into a set of finite cases); (2)the different illumination situations are $\mathbf{I} = \{I_1, \ldots, I_{k_i}\}$ (We assume the different illumination situations can be divided reasonably into a finite set); (3) the background and partial occlusion situations are $\mathbf{C} = \{C_1, \ldots, C_{k_c}\}$ (We assume different background and partial occlusion situations can be reasonably divided into a finite set). Then the set of all possible situations is $\Gamma_0 = \{<I_i, O_j, C_k> \mid I_i \in \mathbf{I}, O_j \in \mathbf{O}, C_k \in \mathbf{C}\}$. For each triple $<I_i, O_j, C_k> \in \Gamma_0$, we can perform an experiment and run the recognition algorithm. It is obvious that the total number of experiments is $N_{total} = \mid \mathbf{I} \mid \times \mid \mathbf{O} \mid \times \mid \mathbf{C} \mid$. Let $N_{success}$ be the total number of successful recognitions, we have

$$\mathbf{b}^*(<\theta, \delta, l>, <w, h>) = \frac{N_{success}}{N_{total}} = \frac{N_{success}}{\mid \mathbf{I} \mid \times \mid \mathbf{O} \mid \times \mid \mathbf{C} \mid} = \frac{N_{success}}{k_o k_i k_c} \tag{3.4}$$

In practice, we may not able to perform these experiments because of the huge space of possibilities. Therefore, we can just assign a value to $\mathbf{b}^*(<\theta, \delta, l>, <w, h>)$ according to our experience, or we can just perform a limited number of experiments and use the results as the estimated value of the detection function. We may even give an analytical expression of $\mathbf{b}^*(<\theta, \delta, l>, <w, h>)$ according to our experience.

For a given camera viewing angle size, we need to know the detection function value of all points inside the camera's viewing volume. If we have an analytical expression for the detection function, we can use the expression to obtain the value. If we do not have the corresponding analytical expression, we need to sub-divide the infinite space formed by

$$\Theta \times \Delta \times L = \{<\theta, \delta, l> \mid \theta \in \Theta, \delta \in \Delta, l \in L\}$$

into finite elements so as to record the average detection function value with respect to each element in a tractable manner. Where Θ is the set of all the possible values of the horizontal offset of points to the viewing axis of the camera, Δ is the set of all the possible values of the vertical offset of points to the viewing axis of the camera, and L is the set of all the possible length of the projection of the line segment, which connects the camera center and the target center, on the viewing axis.

We first tessellate L. For the fixed camera angle size and the given recognition algorithm, there exists an upper bound l_{nl} such that no features of the target can be detected because of the low resolution of the feature image (Figure 3.1(a)). There also exists a lower bound l_0, such that either each feature image is too big to be detected or the target center can not be at that distance because the position is occupied by the camera's lens (Figure 3.1(a)). So, the detection function value will be nonzero only when $l \in [l_0, l_{n_l}]$. Thus $L = [l_0, l_{n_l}]$. We uniformly divide L into n_l segments $[l_0, l_1]$, $[l_1, l_2]$, ..., $[l_{n_l-1}, l_{n_l}]$ (Figure 3.1(b)). Then, for each given $[l_i, l_{i+1}]$, we tessellate the value of Θ and Δ.

We consider the value of Θ for slice $[l_i, l_{i+1}]$ (Figure 3.1(c)). It is obvious that if no features of the target are brought into the camera's field of view, the detection function value will be 0. Suppose the diameter of the target is D_{Target} (the longest distance between any two points of the target). In order to guarantee that no part of the target is brought into the field of view of the camera when the target center is within slice $[l_i, l_{i+1}]$, the distance of point E to OA must be bigger than or equal to D_{Target}. Suppose $\angle AOD = \alpha_i$, we have:

$$\frac{l_i sin\alpha_i}{cos(\alpha_i + \frac{w}{2})} \leq D_{Target}.$$

Thus

$$\alpha_i \geq arctan\{\frac{cos(\frac{w}{2})}{\frac{l_i}{D_{Target}} + sin(\frac{w}{2})}\}.$$

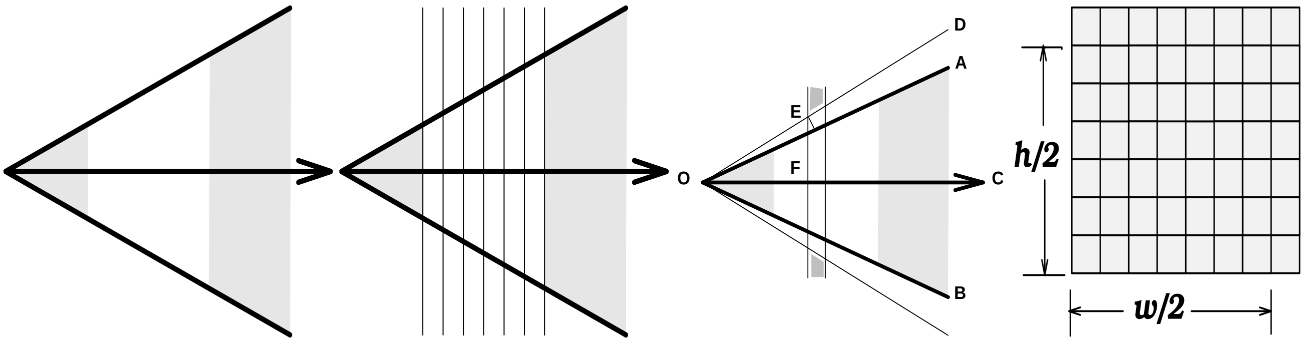
Let

$$\alpha_i = arctan\{\frac{cos(\frac{w}{2})}{\frac{l_i}{D_{Target}} + sin(\frac{w}{2})}\}.$$

Then the range for Θ will be $[0, \frac{w}{2} + \alpha_i]$. We uniformally divide $[0, \frac{w}{2}]$ into $n_w - 1$ parts. Finally we get the tessellation $[\theta_0, \theta_1]$, $[\theta_{n_w-2}, \theta_{n_w-1}]$, $[\theta_{n_w-1}, \frac{w}{2} + \alpha_i]$.

Similarly we can tessellate Δ. For a given distance slice $[l_i, l_{i+1}]$, the final tessellated $\Theta \times \Delta$ is shown in Figure 3.1(d).

In this way, we tessellate $\Theta \times \Delta \times L$ into a series of elements. Each element e_{ijk} is in the form of $l \in [l_i, l_{i+1}]$, $\theta \in [\theta_j, \theta_{j+1}]$, $\delta \in [\delta_k, \delta_{k+1}]$. Then, the detection function value of any point inside e_{ijk} can be simply approximated by the average value of the detection function of all the points within e_{ijk}. Thus, we are able to represent the detection function values of a given camera viewing angle size. Theoretically, the average detection function value inside e_{ijk} can be calculated by



(a) (b) (c) (d)

Figure 3.1: The tessellation of $\Theta \times \Delta \times L$. (a) The detection function value is 0 in the shaded area. (b) Uniformally divide L. (c) Tessellate Θ. Angle AOB is the viewing volume. Angle $AOC = \frac{w}{2}$, where w is the width of the viewing angle. No part of the target can be brought into the field of view of the camera when the center of the target is outside angle DOC (for a target which is above OC). (d) Tessellated $\Theta \times \Delta$.

$$\mathbf{b}^*(e_{ijk}, < w, h >) = \frac{\int\int\int_{e_{ijk}} \mathbf{b}^*(< \theta, \delta, l >, < w, h >) d\theta d\delta dl}{\int\int\int_{e_{ijk}} d\theta d\delta dl}$$

Practically, we can use the detection function value of the point at the center of e_{ijk} to approximate $\mathbf{b}^*(e_{ijk}, < w, h >)$. These detection function values can be stored in a look up table and retrieved when necessary.

3.3 Approximating Detection Function Values for Different Angle Sizes

It is not necessary to record the detection function values of all the different sizes of the solid viewing angles. We only need the detection values of one camera angle size (the *reference angle*), and those of the other camera angle sizes can be approximately transformed to those of the known camera angle size.

Suppose we know the detection function values for viewing angle size $< w_0, h_0 >$. We want to find the detection function values for viewing angle size $< w, h >$.

To get the value of $\mathbf{b}^*(< \theta, \delta, l >, < w, h >)$, for a given $< \theta, \delta, l >$, we need to find the values of $< \theta_0,\ \delta_0,\ l_0) >$ for angle size $< w_0, h_0 >$ that make the following approximation true:

$$\mathbf{b}^*(< \theta, \delta, l >, < w, h >) \approx \mathbf{b}^*(< \theta_0, \delta_0, l_0 >, < w_0, h_0 >) \tag{3.5}$$

The approximation relation (Formula (3.5)) means that when we use the recognition algorithm to analyze the image taken with parameters $(< \theta, \delta, l >, < w, h >)$ and the image taken with parameters $(< \theta_0, \delta_0, l_0 >, < w_0, h_0 >)$, we should get the same results for the two sets of parameters in almost all the situations. To guarantee this, the images of the target object should be almost the same for both sets of imaging parameters. This means that the two pictures must satisfy at least two geometric factors, namely a scale factor and a position factor. The scale factor means that the size of the projection of the target object on the image plane should be almost the same for both $\mathbf{f}$ and $\mathbf{f}_0$. The position factor means that the projection of the center of the target object should be almost at the same position on the image plane for both $\mathbf{f}$ and $\mathbf{f}_0$.

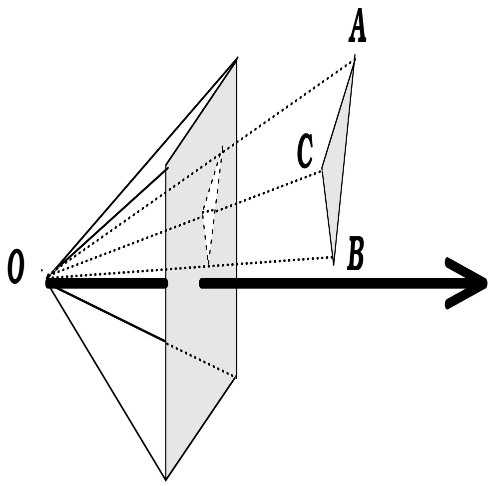


Figure 3.2: The projection of triangle ABC onto the image plane.

First we use the scale factor to find the value of l_0 when l is given. The sizes of the projection of the target object on the image planes for ($<\theta,\delta,l>,<w,h>$) and ($<\theta_0,\delta_0,l_0>,<w_0,h_0>$) are approximately determined by l and l_0 respectively. The scale factor tells us that for a target patch that is parallel to the image plane, the area of its projection on the image plane for ($<\theta,\delta,l>,<w,h>$) should be same as the area of its projection on the image plane for ($<\theta_0,\delta_0,l_0>,<w_0,h_0>$). Suppose that the width of the image plane is W, and the height of the image plane is H. Since the size of the image plane remains constant for different focal lengths, W and H will be same for any focal length.

Suppose the target patch is a triangle $\triangle ABC$ (Figure 3.2) with area S_{ABC} (Any complex convex polygon can be divided into a series of triangles). Suppose the coordinates of $\triangle ABC$ in the Camera Coordinate System for parameters ($<\theta,\delta,l>,<w,h>$) are $A(x_1,y_1,l)$, $B(x_2,y_2,l)$, and $C(x_3,y_3,l)$. Denote the image of $\triangle ABC$ in the image plane as $\triangle A'B'C'$. Let the area of $\triangle A'B'C'$ be $S_{A'B'C'}$, and the coordinates of A', B', and C' be $A'(x_1',y_1',f)$, $B'(x_2',y_2',f)$, $C'(x_3',y_3',f)$. Suppose for parameter ($<\theta_0,\delta_0,l_0>,<w_0,h_0>$), the image of $\triangle ABC$ in the image plane is $\triangle A_0'B_0'C_0'$, and the area of $\triangle A_0'B_0'C_0'$ is $S_{A_0'B_0'C_0'}$.

Then,

$$x_1' = \frac{f}{l}x_1, y_1' = \frac{f}{l}y_1, x_2' = \frac{f}{l}x_2, y_2' = \frac{f}{l}y_2, x_3' = \frac{f}{l}x_3, \text{ and } y_3' = \frac{f}{l}y_3$$

$$S_{A'B'C'} = \frac{1}{2}\begin{vmatrix} x_1' & y_1' & l \\ x_2' & y_2' & l \\ x_3' & y_3' & l \end{vmatrix} = \frac{1}{2}\begin{vmatrix} \frac{f}{l}x_1 & \frac{f}{l}y_1 & l \\ \frac{f}{l}x_2 & \frac{f}{l}y_2 & l \\ \frac{f}{l}x_3 & \frac{f}{l}y_3 & l \end{vmatrix} = \frac{1}{2}\frac{f^2}{l^2}\begin{vmatrix} x_1' & y_1' & l \\ x_2' & y_2' & l \\ x_3' & y_3' & l \end{vmatrix} = \frac{f^2}{l^2}S_{ABC} \tag{3.6}$$

Since

$$tan(\frac{w}{2}) = \frac{\frac{W}{2}}{f}$$

and

$$tan(\frac{h}{2}) = \frac{\frac{H}{2}}{f}$$

we have

$$S_{A'B'C'} = \frac{f^2}{l^2}S_{ABC} = \frac{WH}{4l^2tan(\frac{w}{2})tan(\frac{h}{2})}S_{ABC} \tag{3.7}$$

Similarly,

$$S_{A_0'B_0'C_0'} = \frac{WH}{4l_0^2tan(\frac{w_0}{2})tan(\frac{h_0}{2})}S_{ABC} \tag{3.8}$$

To guarantee $S_{A'B'C'} = S_{A_0'B_0'C_0'}$, we get:

$$l_0 = l\sqrt{\frac{tan(\frac{w}{2})tan(\frac{h}{2})}{tan(\frac{w_0}{2})tan(\frac{h_0}{2})}} \tag{3.9}$$

Second, we use the position factor to find the values of θ_0, δ_0, when θ and δ are given. Suppose D is the center of $\triangle ABC$, and $D'(x', y', z')$ is the image of D on the image plane with respect to $(<\theta, \delta, l>, <w, h>)$, and $D_0'(x_0', y_0', z_0')$ is the image of D on the image plane with respect $(<\theta_0, \delta_0, l_0>, <w_0, h_0>)$. Then we have

$$x' = ftan(\theta) = \frac{\frac{W}{2}}{tan(\frac{w}{2})}tan(\theta) = \frac{W}{2}\frac{tan(\theta)}{tan(\frac{w}{2})} \tag{3.10}$$

Similarly,

$$y' = \frac{H}{2}\frac{tan(\delta)}{tan(\frac{h}{2})} \tag{3.11}$$

$$x_0' = \frac{W}{2}\frac{tan(\theta_0)}{tan(\frac{w_0}{2})} \tag{3.12}$$

$$y_0' = \frac{H}{2}\frac{tan(\delta_0)}{tan(\frac{h_0}{2})} \tag{3.13}$$

To guarantee $x' = x_0'$ and $y' = y_0'$, we get

$$\theta_0 = arctan[tan(\theta)\frac{tan(\frac{w_0}{2})}{tan(\frac{w}{2})}] \tag{3.14}$$

and

$$\delta_0 = arctan[tan(\delta)\frac{tan(\frac{h_0}{2})}{tan(\frac{h}{2})}] \tag{3.15}$$

So, when we want to find the detection function value for parameters $< \theta, \delta, l >$ with respect to the camera angle size $< w, h >$, we can first find the corresponding $< \theta_0, \delta_0, l_0 >$, then retrieve the detection function value for $\mathbf{b}^*(< \theta_0, \delta_0, l_0 >, < w_0, h_0 >)$ from the look up table or get the value from the analytical formula.

3.4 Consistency with the Misfocus Property

It is interesting to note that the calculations (3.9), (3.14) and (3.15) derived in Section 3.3 are consistent with respect to the misfocus property of the two visual angle sizes: the area of the blur circle for $(< \theta, \delta, l >, < w, h >)$ is the same as the area of the blur circle for $(<\theta_0, \delta_0, l_0>, <w_0, h_0>)$. It is clear from the *Gaussian lens law*, that only points within one particular object plane are brought into focus on the image plane by an ideal lens. This distance is called the *focusing distance*. A point, which is nearer than the focusing distance, is brought into focus at a place behind the image plane. A point, which is farther than the focusing distance, is brought into focus to a place in front of the image plane. Usually we always place the image plane at the same position as the *focal plane*. So only points that are at infinity are brought into focus. Those that are nearer than infinity are imaged as blur circles, each blur circle is formed by the intersection of the corresponding cone of light rays with the image plane. Suppose the diameter of the blur circle produced by $(< \theta, \delta, l >, < w, h >)$ is c, the focus distance for l is v, and the diameter of aperture is a. We have

$$\frac{c}{a} = \frac{v-f}{v} \tag{3.16}$$

$$\frac{1}{l} + \frac{1}{v} = \frac{1}{f} \tag{3.17}$$

Thus, $c = \frac{af}{l}$. Similarly, we have $c_0 = \frac{af_0}{l_0}$, where c_0 is the diameter of the blur circle produced by $(< \theta_0, \delta_0, l_0 >, < w_0, h_0 >)$.

Since

$$l_0 = l\sqrt{\frac{tan(\frac{w}{2})tan(\frac{h}{2})}{tan(\frac{w_0}{2})tan(\frac{h_0}{2})}} = l\frac{f_0}{f} \tag{3.18}$$

then

$$c_0 = \frac{af_0}{l_0} = \frac{af_0}{l\frac{f_0}{f}} = \frac{af}{l} = c \tag{3.19}$$

Thus, calculations (3.9), (3.14) and (3.15) derived in section 3.3 are consistent with the misfocus property.

Usually an object point is considered to be well focused if the diameter of the blur circle is less than the pixel size on the image plane. The depth of field refers to the distance from which an object point is well focused. When the **image plane** is different from the **focal plane**, the depth of field is a range of depth around the **focusing distance**. We assume in this document that the image plane is coincident with the focal plane, thus the focusing distance will be infinity. Therefore, in our case, the depth of field will extend to infinity starting at a distance where the size of the blur circle is the same as the size of the sensor pixel.

3.5 The Detection Function During the Search Process

In this section, we discuss the correlation properties of the detection function value with respect to a sequence of actions. Suppose we need to get the value of $\mathbf{b}(c, \mathbf{f})$ during the search process, where c is a cube in the environment, and $\mathbf{f} = \mathbf{f}(x_c, y_c, z_c, p, t, w, h, a)$ is the action to be evaluated. We can use the value of $\mathbf{b}'((x, y, z), \mathbf{f})$ to approximate $\mathbf{b}(c, \mathbf{f})$, where (x, y, z) is the coordinate of the center of cube c in the World Coordinate System. To obtain $\mathbf{b}'((x, y, z), \mathbf{f})$, we can first transform (x, y, z) into the Camera Coordinate System and calculate the corresponding $< \theta, \delta, l >$, and then retrieve the value from the lookup table or get the value from the analytical formula.

There is an interesting phenomenon that we have not addressed yet. It appears that if we apply the same action $\mathbf{f}$, which fails, again and again, the probability distributions within the field of view of the camera will be reduced each time $\mathbf{f}$ is applied and finally become 0. This is wrong. The reason is that the correlation among different actions has not yet been considered. As a matter of fact, after the first time $\mathbf{f}$ is applied, the detection function values $\mathbf{b}'((x, y, z), \mathbf{f})$ should be 0 instead of those obtained from $\mathbf{b}^*(< \theta, \delta, l >, < w, h >)$. In the following, we will analyze this phenomenon in detail and propose a strategy that can reduce the inaccuracy caused by ignoring correlation among actions.

3.5.1 Correlation Between Different Actions

Suppose $\mathbf{b}'(\phi, (x, y, z), \mathbf{f})$ is the detection function when no actions have been previously applied. Then the value of $\mathbf{b}'(\phi, (x, y, z), \mathbf{f})$ can be obtained as described above. Suppose $\mathbf{b}'(\langle \mathbf{f}_1 \rangle, (x, y, z), \mathbf{f})$ is the detection function for $\mathbf{f}$ when an action $\mathbf{f}_1$ has been applied before and failed to detect the target, where $\mathbf{f}_1 = \mathbf{f}(x_1, y_1, z_1, p_1, t_1, w_1, h_1, a)$. Suppose $\beta_{\mathbf{f}_1}$ is the event that $\mathbf{f}_1$ successfully detects the target, and $\beta_{\mathbf{f}}$ is the event that $\mathbf{f}$ successfully detects the target. Let α_ϕ be the event in which no actions have been applied before. Then we have

$$\mathbf{b}'(\phi, (x, y, z), \mathbf{f}) \quad = \quad p(\beta_{\mathbf{f}} \mid \alpha_\phi)$$

$$\mathbf{b}'(<\mathbf{f}_1>, (x,y,z), \mathbf{f}) \quad = \quad p(\beta_{\mathbf{f}} \mid \neg\beta_{\mathbf{f}_1})$$

It is obvious that the correlation between $\mathbf{f}$ and $\mathbf{f}_1$ can be neglected when the target does not appear in both of their images. In this case, we assume $\mathbf{b}(\phi, (x,y,z), \mathbf{f}) = \mathbf{b}(<\mathbf{f}_1>, (x,y,z), \mathbf{f})$.

When the target appears in both images, the two actions correlate with each other. Suppose

- (θ, δ, l): The position of the target with respect to action $\mathbf{f}$.
- $(\theta_1, \delta_1, l_1)$: The position of the target with respect to action $\mathbf{f_1}$.
- $\Gamma_{0-\mathbf{f_1}-fail}$: The set of situations that make $\mathbf{f_1}$ fail.
- $\Gamma_{\neg\mathbf{f_1}-\mathbf{f}}$: The set of possible situations to be considered by $\mathbf{f}$ when the first action $\mathbf{f_1}$ failed.
- $< I_{\neg\mathbf{f_1}}, O_{\neg\mathbf{f_1}}, C_{\neg\mathbf{f_1}} >$: The real situation encountered when $\mathbf{f_1}$ is applied ($\mathbf{f_1}$ failed to detect the target). Of course, we have $< I_{\neg\mathbf{f_1}}, O_{\neg\mathbf{f_1}}, C_{\neg\mathbf{f_1}} > \in \Gamma_{0-\mathbf{f_1}-fail}$.
- $< I_{\neg\mathbf{f_1}-\mathbf{f}}, O_{\neg\mathbf{f_1}-\mathbf{f}}, C_{\neg\mathbf{f_1}-\mathbf{f}} >$: The situation encountered by $\mathbf{f}$ after $\mathbf{f_1}$ is applied. Of course, we have $< I_{\neg\mathbf{f_1}-\mathbf{f}}, O_{\neg\mathbf{f_1}-\mathbf{f}}, C_{\neg\mathbf{f_1}-\mathbf{f}} > \in \Gamma_{\neg\mathbf{f_1}-\mathbf{f}}$.

When the parameters of the camera change from action $\mathbf{f_1}$ to action $\mathbf{f}$, the extrinsic parameters change from $(x_{c1}, y_{c1}, z_{c1}, p_1, t_1)$ to (x_c, y_c, z_c, p, t), and the intrinsic parameters change from $< w_1, h_1 >$ to $< w, h >$. The transformation T_r^* that changes the extrinsic parameters of the camera consists of a translation that changes (x_{c1}, y_{c1}, z_{c1}) to (x_c, y_c, z_c) and a rotation that changes (p_1, t_1) to (p, t). If we take the camera coordinate system as our reference, then there is a transformation T_r corresponding to T_r^*, which changes the target position from $(\theta_1, \delta_1, l_1)$ for $\mathbf{f}_1$ to (θ, δ, l) for $\mathbf{f}$, and also changes the situation from $< I_{\neg\mathbf{f_1}}, O_{\neg\mathbf{f_1}}, C_{\neg\mathbf{f_1}} >$ to $< I_{\neg\mathbf{f_1}-\mathbf{f}}, O_{\neg\mathbf{f_1}-\mathbf{f}}, C_{\neg\mathbf{f_1}-\mathbf{f}} >$. Actually, we have the following relations:

$$T_r((\theta_1, \delta_1, l_1)) = (\theta, \delta, l)$$

and

$$\begin{aligned}
T_r(\Gamma_{0-\mathbf{f_1}-fail}) &= T_r(\{< I_{\neg\mathbf{f_1}}, O_{\neg\mathbf{f_1}}, C_{\neg\mathbf{f_1}} > \mid < I_{\neg\mathbf{f_1}}, O_{\neg\mathbf{f_1}}, C_{\neg\mathbf{f_1}} > \in \Gamma_{0-\mathbf{f_1}-fail}\}) \\
&= \{T_r(< I_{\neg\mathbf{f_1}}, O_{\neg\mathbf{f_1}}, C_{\neg\mathbf{f_1}} >) \mid < I_{\neg\mathbf{f_1}}, O_{\neg\mathbf{f_1}}, C_{\neg\mathbf{f_1}} > \in \Gamma_{0-\mathbf{f_1}-fail}\} \\
&= \{< I_{\neg\mathbf{f_1}-\mathbf{f}}, O_{\neg\mathbf{f_1}-\mathbf{f}}, C_{\neg\mathbf{f_1}-\mathbf{f}} > \mid < I_{\neg\mathbf{f_1}-\mathbf{f}}, O_{\neg\mathbf{f_1}-\mathbf{f}}, C_{\neg\mathbf{f_1}-\mathbf{f}} > \in \Gamma_{\neg\mathbf{f_1}-\mathbf{f}}\}
\end{aligned}$$

The following table lists the factors that influence the values of detection functions $\mathbf{b}(<\mathbf{f}_1>, (x,y,z), \mathbf{f})$ and $\mathbf{b}(<\phi>, (x,y,z), \mathbf{f})$.

	$\mathbf{b}'(<\mathbf{f}_1>, (x,y,z), \mathbf{f})$	$\mathbf{b}'(<\phi>, (x,y,z), \mathbf{f})$
Angle Size	$< w, h >$	$< w, h >$
Relative Position	$< \theta, \delta, l >$	$< \theta, \delta, l >$
Situations	$\Gamma_{\neg \mathbf{f_1} - \mathbf{f}} = T_r(\Gamma_{0-\mathbf{f_1}-fail})$	Γ_0

From the above table, we see that when we calculate $\mathbf{b}'(<\mathbf{f}_1>, (x,y,z), \mathbf{f})$ and $\mathbf{b}'(<\phi>, (x,y,z), \mathbf{f})$, the visual angle sizes and the relative positions of the target with respect to the camera are same. But the situations $\Gamma_{\neg \mathbf{f_1} - \mathbf{f}}$ considered by $\mathbf{b}'(<\mathbf{f}_1>, (x,y,z), \mathbf{f})$ are more constrained than the situations Γ_0 considered by $\mathbf{b}(<\phi>, (x,y,z), \mathbf{f})$. This is because $\Gamma_{\neg \mathbf{f_1} - \mathbf{f}} \subseteq \Gamma_0$. This constrained situation is dependent on the previous action $\mathbf{f_1}$, and it is actually a transformation of $\Gamma_{0-\mathbf{f_1}-fail}$. So, in general situations $\mathbf{b}'(<\mathbf{f}_1>, (x,y,z), \mathbf{f})$ and $\mathbf{b}'(<\phi>, (x,y,z), \mathbf{f})$ may not always be equal. When the difference between the configurations of $\mathbf{f_1}$ and $\mathbf{f}$ are big, we will assume independence between $\mathbf{f_1}$ and $\mathbf{f}$. When the difference between the configurations of $\mathbf{f_1}$ and $\mathbf{f}$ are very small, we cannot assume independence between actions. Figure 3.3 gives an explanation of the above analysis.

Suppose $\mathbf{b}'(\mathbf{H}, (x,y,z), \mathbf{f})$ represents the detection function value for action $\mathbf{f}$ when the previous history is $\mathbf{H}$, i.e., $\mathbf{H}$ is the sequence of actions applied before $\mathbf{f}$. Then the value of $\mathbf{b}'(\mathbf{H}, (x,y,z), \mathbf{f})$ is influenced by $\mathbf{H}$. This influence should be considered in real applications. Section 3.5.2 gives the summary of our strategy.

3.5.2 Real Applications

In real application, we can only get $\mathbf{b}(\phi, (x,y,z), \mathbf{f})$ by retrieving from the database according to the relative position of the target with respect to the camera center. For $\mathbf{b}(\mathbf{H}, (x,y,z), \mathbf{f})$, we either use approximate values based on $\mathbf{b}(\phi, (x,y,z), \mathbf{f})$ or consider the correlation of current action with the previous actions in $\mathbf{H}$. In the following, we summarize the strategies in estimating $\mathbf{b}(\mathbf{H}, (x,y,z), \mathbf{f})$.

Case 1: The Previous Action is Exactly the Same as the Current Action

Since the first action $\mathbf{f_1}$ and the current action $\mathbf{f}$ are the same, we have $(x_{c1}, y_{c1}, z_{c1}) = (x_c, y_c, z_c)$, $(p_1, t_1) = (p, t)$, and $(w_1, h_1) = (w, h)$. Thus, the transformation T_r described above is the identity. So $(\theta_1, \delta_1, l_1) = (\theta, \delta, l)$. The situations are also same: $\langle I_{\neg \mathbf{f_1}}, O_{\neg \mathbf{f_1}}, C_{\neg \mathbf{f_1}} \rangle = \langle I_{\neg \mathbf{f_1} - \mathbf{f}}, O_{\neg \mathbf{f_1} - \mathbf{f}}, C_{\neg \mathbf{f_1} - \mathbf{f}} \rangle$. Since $\langle \theta_1, \delta_1, l_1, I_{\neg \mathbf{f_1}}, O_{\neg \mathbf{f_1}}, C_{\neg \mathbf{f_1}} \rangle$ makes $\mathbf{f_1}$ fail and $\mathbf{f_1} = \mathbf{f}$, then, $\langle \theta_1, \delta_1, l_1, I_{\neg \mathbf{f_1}}, O_{\neg \mathbf{f_1}}, C_{\neg \mathbf{f_1}} \rangle$ also makes $\mathbf{f}$ fail. Thus, $\langle \theta, \delta, l, I_{\neg \mathbf{f_1} - \mathbf{f}}, O_{\neg \mathbf{f_1} - \mathbf{f}}, C_{\neg \mathbf{f_1} - \mathbf{f}} \rangle$ makes $\mathbf{f}$ fail. So, in this situation, $\mathbf{b}_1'((x,y,z), \mathbf{f}) = 0$. In a real application, we should avoid repeated operations.

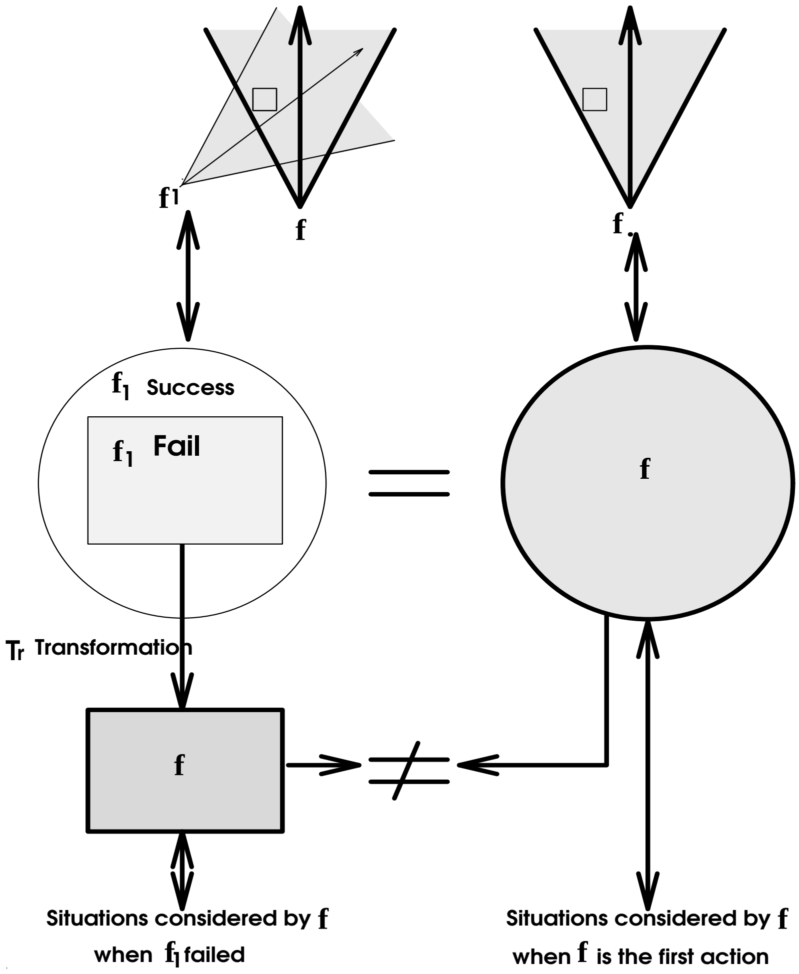


Figure 3.3: The correlation between actions

Case 2: The Intrinsic Parameters and the Position of the Two Actions are the Same, but the Direction of the Two Actions are Different

We have $(x_{c1}, y_{c1}, z_{c1}) = (x_c, y_c, z_c)$, and $(w_1, h_1) = (w, h)$, but $(p_1, t_1) \neq (p, t)$. In this case, the position factors are different but the situation factors are the same. That is, $(\theta_1, \delta_1, l_1) \neq (\theta, \delta, l)$ and $< I_{\neg\mathbf{f_1}}, O_{\neg\mathbf{f_1}}, C_{\neg\mathbf{f_1}} >=< I_{\neg\mathbf{f_1}-\mathbf{f}}, O_{\neg\mathbf{f_1}-\mathbf{f}}, C_{\neg\mathbf{f_1}-\mathbf{f}} >$. Since the camera angle is usually not very big, to keep the target within the image, the change in position of the target with respect to the camera is also not very big. Since the intensity, orientation and occlusion factors are same, the patterns of the target in the image should not change very much by just changing the viewing direction of the camera. Since the previous image makes the recognition algorithm fail, the current image will probably also make the recognition algorithm fail. Thus, in this case, we approximate the detection function value as 0.

In a real application, when a cube appears on the images of two actions with the same intrinsic parameters and camera position, the detection function value of the later action with respect to this cube is zero.

Case 3: The Positions of the Two Actions are Same, but the Intrinsic Parameters are Different.

If the difference in direction and intrinsic parameters are small, we approximate the detection function value of the later action as $\mathbf{b_1'}((x, y, z), \mathbf{f}) = 0$. Otherwise, we assume independence between the actions.

Case 4: The Position of the Two Actions are Different.

We assume independence between the actions.

Case 5: The Layered Sensed Sphere and the Approximation of $\mathbf{b}(\mathbf{H}, (x, y, z), \mathbf{f})$.

One benefit of using the layered sensed sphere (to be addressed later) to select actions is to reduce the error caused by the dependence relation between actions. Each layer in the layered sensed sphere corresponds to a given set of intrinsic parameters of action. The difference between sets of intrinsic parameters for different layers is large. We only select actions whose parameters appear in the layered sensed sphere. So in the search process, if a cube has been examined more than once by actions belonging to same layer, then the detection function value will be 0 for the later actions. Otherwise, if it is the first time a cube is examined by an action belonging to a layer, then the detection function value will be retrieved according to the database.

3.6 Detection Function in A General Sense

In the previous section, we have discussed the detection function $\mathbf{b}'((x,y,z),\mathbf{f})$ which gives the probability of detecting the target given that the operation is $\mathbf{f}$ and that the target center position is at (x,y,z). This detection function can be used to calculate the probability of detecting the target by a given action $\mathbf{f}$, thus it can be used to select actions in the sensor planning process.

In this section, we discuss detection function more generally. First we identify the factors that influence the detection results of a given action.

- Sensor's state parameters: Camera's position (x_c, y_c, z_c), viewing direction (p,t), and viewing angle size (w,h).
- Available recognition algorithms $a_1, \ldots, a_{n_r}$.
- Target T_{tar}.
- The relative viewing direction between the camera and the target d_c.
- The position of the target (x,y,z).
- The background situation B.
- The lighting conditions I.

We represent the most general detection function as $D(x,y,z,x_c,y_c,z_c,p,t,w,h,a,T_{tar},d_c,B,I)$ (Note: a is one of $a_1, \ldots, a_{n_r}$). The most general detection function gives the detection situation of the action $\mathbf{f}$. When the target is detected, it get value 1, when the target is not detected, it get value 0. For $D(x,y,z,x_c,y_c,z_c,p,t,w,h,a,T_{tar},d_c,B,I)$, every factor that can influence the detection result is fixed. Thus, the detection function value is also fixed, be either 1 or 0. When the target is detected by $\mathbf{f}$, the value of $D(x,y,z,x_c,y_c,z_c,p,t,w,h,a,T_{tar},d_c,B,I)$ is 1; when the target is not detected, the value of $D(x,y,z,x_c,y_c,z_c,p,t,w,h,a,T_{tar},d_c,B,I)$ is 0.

Since the target is given at the beginning of the search process, so, the parameter T_{tar} in $D(x,y,z,x_c,y_c,z_c,p,t,w,h,a,T_{tar},d_c,B,I)$ is fixed. Thus, T_{tar} is a constant in function D, instead of a variable parameter. So, we omit T_{tar} in the representation of function D, with an underline assumption that the target is T_{tar}. We now represent D as

$$D(x,y,z,x_c,y_c,z_c,p,t,w,h,a,d_c,B,I).$$

Since $x_c, y_c, z_c, p, t, w, h, a$ gives the parameters that specifies the action $\mathbf{f}$, thus, we use $\mathbf{f}$ to replace $x_c, y_c, z_c, p, t, w, h, a$ in the representation of function D. Thus, the representation of function D becomes $D(x,y,z,\mathbf{f},d_c,B,I)$.

Now, we consider the lighting situation I and the background situation B. Since in real action selection process, we cannot predict the lighting situations I and the background situations B at a given position (x, y, z), what we know is only the probability of the presence of the target at (x, y, z). So we have to relax the situation I and the situation B when considering the function D and we have to include the possible variations of $< B, I >$ into the value of D in a probabilistic fashion. Thus, the value of D will be a probabilistic value and the variable B and I will be taken out from the parameter list in the representation of function D. $D(x, y, z, \mathbf{f}, d_c)$ refers to the probability of detecting the target when the target center position is (x, y, z), the action is $\mathbf{f}$, and the relative viewing direction of the camera with respect to the target is d_c. Suppose there are N_{IB} total situations of $< B, I >$, they are $< B, I >_1, \ldots, < B, I >_{N_{IB}}$. If when action $\mathbf{f}$ is applied, the situation is $< B, I >_i$, then because everything concerned with D is fixed, the function of D can be either 0 or 1. If the target is detected, the value is 1, otherwise, it is 0. Suppose among $< B, I >_1, \ldots, < B, I >_{N_{IB}}$, there are $N_{IB}^{success}$ situation of $< B, I >$ that make the action $\mathbf{f}$ successfully detect the target, then the value of D is $\frac{N_{IB}^{success}}{N_{IB}}$. In real applications, if the background situation can be constrained before the value of D is used, then we might be able to get a more accurate value of D. For example, when we are looking for a white object in the environment, if the white target is not detected, then the value of D for those cubes whose center's projection on the image are within white regions should have a relative lower value than we thought before the image is taken.

Now, let's consider the influence of the relative direction of the viewer with respect to the target. Of course, different viewing direction will bring different aspect of the target to the camera. Thus, will bring different results for the action. So, the influence of different viewing direction relative to the target is very important. The most general detection function should include the relative direction d_c as a factor. Since the relative direction corresponds to the relative target orientation (e.g. which aspect is facing the viewer) with respect to the viewer, we should take the target's self-rotation (the aspect facing the viewer) d_c as a factor.

For d_c, we have two choices here. One is to include it as a parameter of the detection function D; the other is to include the variation of d_c as a probabilistic value of the function D, thus delete d_c from the parameter list of D.

If we include d_c as a parameter of the detection function D, the representation of D is $D(x, y, z, \mathbf{f}, d_c)$. Since the purpose of introducing the concept of detection function is to calculate the probability of detecting the target by a given action $\mathbf{f}$, we **have to** provide a probability distribution function $\mathbf{p}^{d_c}(x, y, z, d_c, \tau)$ in order to calculate the effectiveness of a selected action. $\mathbf{p}^{d_c}(x, y, z, d_c, \tau)$ gives the search agent's belief at time τ about the probability that the target is at position (x, y, z) and that the target's orientation relative to the camera is d_c. The probability of detecting the target can thus be calculated as:

$$\int_{\Omega} \{ \int_{vs} [\mathbf{p}^{d_c}(x, y, z, d_c, \tau) D((x, y, z), \mathbf{f}, d_c) dd_c] \} dx dy dz \qquad (3.20)$$

Where $\int_{vs}[\mathbf{p}^{d_c}(x,y,z,d_c,\tau)D((x,y,z),\mathbf{f},d_c)dd_c]$ means taking the integration of d_c over the whole **v**iewing **s**phere regarding the position (x,y,z). The **v**iewing **s**phere for a point (x,y,z) refers to a sphere with this point as the sphere's center. If the target is assumed at point (x,y,z) and the camera's center is assumed at point p_{vs} of the viewing sphere, then the *latitude* and the *altitude* of the point p_{vs} can be used to represent the relative viewing direction of the camera to the target.

But, the difficulty comes from the above formulation is that what the agent know at the beginning of the search is only the possible position of the target within Ω. It is unreasonable to assume that the agent also knows the probability of the possible orientation of the target at all points of the environment. Even if the target is able to know the probability of the target's possible orientation at all points, it is too time consuming to calculate Equation 3.20. Thus, it is **impractical** to take d_c as a parameter of function D. So we are **forced to** include the variation of the relative viewing direction of the camera to the target (or, to say it in other words, the relative orientation of the target with respect to the camera) as a probabilistic value of the detection function. Thus, the representation of the detection function D is $D((x,y,z),\mathbf{f})$. This is actually the detection function $\mathbf{b}'((x,y,z),\mathbf{f})$ used in formulating the sensor planning task in Section 2.4.1. When $\mathbf{b}'((x,y,z),\mathbf{f})$ is used, the agent need to know only the probability distribution of the target within the environment in order to calculate the probability of detecting the target by a given action.

So, the reason that we do not include the relative viewing direction of the camera to the target as a parameter in the detection function is **not** because the relative viewing direction does not influence the detection result (they **do** influence the detection result), but because it is impractical to include it as a parameter. The influence of the relative view direction of the camera to the target is incorporated as a probabilistic value in the detection function $\mathbf{b}'((x,y,z),\mathbf{f})$.

As a conclusion, we propose the concept of detection function to capture a statistical regularity linking object recognition algorithm, target object, and viewpoint parameters. If computing and checking the detection function is sufficiently low, then it has utility and can be used to predict the probability of detecting the target by a given action. If the cost of running a recognition algorithm is high, then the detection function has the role of reducing the number of times the recognition algorithm is invoked during the search process.

Part III

The Object Search Strategy

Chapter 4

Previous Work on Sensor Planning

In the relationship between viewer and viewed object, research in computer vision has generally placed an emphasis on the object to be observed. The problems of reconstructing the observed object, finding its shape and pose, have been studied extensively assuming that the viewpoint is given and that there is no control of its selection. On the other hand, the issues related to determining viewpoints that will be most suitable for the vision task at hand have received considerably less attention. This viewpoint selection problem, however, is very important since it is the viewpoint for the most part that determines the quality of the resulting image. Thus, an effective viewpoint may make a vision task feasible, as well as facilitate its execution. Furthermore, since image acquisition is usually much less costly than the subsequent image analysis, it seems advantageous to dedicate computational effort to determine appropriate values for the viewing parameters. With such control of the imaging process it may then be possible to perform a simpler analysis on the resulting image. It is for these reasons that active vision research has been of interest.

Previous work on sensor planning can be roughly divided into two categories according to whether or not the configurations of the camera's state parameters are specified. The works that do not explicitly specify the configurations of the camera's state parameters generally emphasize the benefits of adjusting the camera's state parameters. One benefit of purposefully moving cameras is to introduce more constraints so as to make some computations easier. For example, purposefully moving cameras may make the recovery of $3D$ shape from $2D$ contour and the recovery of 3D shape from texture easier ([4]). Using camera movement to constrain degrees of freedom through a fixation point can facilitate the computation of kinetic depth ([10]). When an observer is able to track an environmental point, the task of navigation and the task of depth computation will benefit ([13], [65]). Another benefit of purposefully moving cameras is to get a better view so as to make the task at hand easier. Wilkes and Tsotsos [87] propose the concept of *active object recognition*. Their idea is to divide the recognition task into two easier subtasks, one is to move the camera and primary light source to a position called the standard view of the unknown object, the other is to solve the $2D$ pattern recognition problem from the standard view. In their case, the the standard view is defined to be a position at which the lengths of two non-parallel object lines are maximized. They propose a set of behaviors that can move the camera to the standard view based on the low level image data. Kutulakos and Dyer [46] present a sensor planning approach in which the observer can purposefully change viewpoint in order to achieve a well-defined geometric relationship with respect to a $3D$ shape prior to its recognition. Maver and Bajcsy [53][54] develop a strategy to determine the sequence of different views in order to see a portion of the visual field otherwise hidden. Kim [39] has studied the problem of determining the camera viewpoints for successive views looking for distinguishing features of object. The distance of the camera to the object is determined by the size of the object and the size of the feature. Within this distance, the direction (the position of the camera on the sphere) is determined by the shape of the feature and the presence of occluding objects. An aspect graph with nodes being assigned values representing "goodness" of view is suggested to guide the motion of the camera on the sphere. Ikeuchi [34] precompiles a geometrical model into an interpretation tree which distinguishes among viewpoints based on the set of visible faces. This tree is then used to determine attitude by using the optimal features at each determining process. Grimson [82] proposes an optimal sensing strategy for disambiguating among alternative interpretations of scenes containing multiple polyhedral objects (multiple objects in known poses or a single object with several consistent poses). Whaite and Ferrie [84][85] propose a method that can use the spatial structure of uncertainty to guide the acquisition of additional data in a way that improves the knowledge of the scene. Wixson and Ballard [90] [89] study the benefits of combining a high resolution image (corresponding to wide camera angle size) with a low resolution image (corresponding to small camera angle size) in the task of object search. He shows that first using a large viewing angle size to locate a large intermediate object and then using small viewing angle size to examine around the intermediate object for the target can increase the search efficiency in many situations. Swain et al. [69][70] propose a strategy of using colour as a low resolution cue to guide the fovea (corresponding to the small viewing angle size) in order to find an object whose identity is known. In Rimey and Brown's task-oriented system TEA [63], the low resolution camera provides a peripheral image of the entire field of view, while the high resolution camera can be moved within the field of view to examine the object in detail and to do verification. Bolle et al. [15] examine the problem of how to combine information from the high resolution fovea-like window and information from the coarse resolution of the whole image so as to extract features that cannot be detected at either coarse or fine resolution alone. Their scheme has been used in an experiment to find a golf ball.

The other category is those works that explicitly specify the configurations of the camera's state parameters in order to perform certain task. Our work on sensor planning for object search belongs to this category. The basic task of this category is that in setting up an automated system for monitoring some process, the effectiveness of the system can largely be determined by the locations, types, and configurations of the sensor used. Some systems have been developed which, based on geometric models of an environment and models of the sensors, can generate sensor locations and settings which provide a robust view of specific features so that the features are detectable, recognizable, measurable, or meet some other task constraints. Krotkov[45][44] presents solutions to two problems arising in the context of automatically focusing a general-purpose servo-controlled video camera on manually selected targets. One is how to best determine the focus motor position providing the sharpest focus on an object point at an unknown distance. The other is how to compute the distance to a sharply focused object point. He uses a method based on maximizing the magnitude of the intensity gradient as the criterion function. The Fibonacci search technique is used to optimally locate the mode of the criterion function and then the thick-lens law is used to compute the distance. Cowan et al. [21] present an approach to automatically generating the possible camera locations for observing an object. Their approach, which uses models of the object and the camera, is based on meeting the requirements that the spatial resolution be above a minimum value, all surface points be in focus, all surfaces lie within the sensor field of view, and no surface points be occluded. Their approach converts each sensing requirement into a geometric constraint on the sensor location, from which the three-dimensional region of viewpoints that satisfy the constraint is computed. The intersection of these regions is the space where a sensor may be located. Tarabanis [72] also considers the above problems by analytically characterizing the domain of admissible camera locations, orientations, and optical settings. Their concept of a viewpoint is defined in a broader sense that includes not only viewer orientation and position, but also the optical settings associated with the viewpoint at hand. These settings are also observer attributes affecting the resulting image of the scene. Examples of such attributes include the focus and aperture settings of a regular lens as well as the focal length settings of a zoom lens. In their approach, the task constraints are characterized analytically. Using concepts from geometry and optics, each task requirement is modeled by an equivalent analytical relationship which in turn is satisfied in a domain of admissible values in the space of parameters to be planned. The admissible domain of the camera's sensing parameters is a region bounded by the hyper-surfaces that are characterized by these analytical relationships. These component admissible domains obtained for each task requirement separately are then combined in order to find parameter values that satisfy all constraints simultaneously.

One notable characteristic of most of the previous work on sensor planning is the use of already-acquired scene data to suggest new viewpoints. For the object search task, it is not suggested to acquire the scene data of the whole environment before the search process. Furthermore, the agent does not know the position of the target. So, the only possible way to efficiently perform the sensor planning task is to combine the agent's knowledge about the probability distribution of the target with properties of the recognition algorithm. In the following chapters of Part III, we propose a sensor planning strategy that makes use of the target distribution, the detection function for the available recognition algorithm, and the agent's crude knowledge of the surrounding environment of the camera. We first propose the concept of the "sensed sphere" and then present the planning algorithm.

Chapter 5

Sensed Sphere

To take the occlusion situations into consideration, a representation of the environment around the camera is needed. Any lines emitting from the center of the camera will hit a solid object in the environment. Each emitting line can be specified by its direction (pan, $tilt$) and length. The environment around the center of the camera can thus be represented by the union of all these emitting lines. This representation can be discretized by dividing the space around the camera with a series of solid angles. Each solid angle is associated with a radius which is the length of the emitting line in the direction of the central axis of the solid angle. The environment can thus be represented by the union of these solid angles. This representation is called the **sensed sphere** [77]. The sensed sphere can be constructed either by a laser or by stereo. To use stereo to construct the sensed sphere, we can first construct a sparse depth sampling of the environment by existing stereo algorithms [43][52][73][81] and then transform this into the representation of a sensed sphere. Here, we use a laser to construct the sensed sphere. In order to do this, we need to **tessellate** the surface of a unit sphere so as to get a series of uniform or near uniform patches which partition the surface. Then we can ping the laser at the center of each patch to construct the sensed sphere. There are many ways to tessellate the surface, one interesting way is to use geodesic domes. However, for our concern, this is not the best choice because the centers of those triangular facets are not regular, possibly requiring frequent changes of the pan and tilt together in order to ping the next center. This is not efficient when the pan and tilt operations cannot overlap in time, since the mechanical operations might waste time.

Figure 5.1(a) shows the head coordinate system and the corresponding unit sphere, both centered at the camera center O. The value of tilt of the emitting line from the center of the camera is the angle between this line and the positive direction of the Z axis. The value of pan of the emitting line is the amount of rotation needed to rotate the emitting line around the Z axis counter-clockwise so as to make the emitting line fall onto the XZ plane. pan and $tilt$ are in radians. The range of tilt is $0 \leq tilt \leq \pi$. The range of pan is $0 \leq pan \leq 2\pi$. So, the positive direction of the X axis corresponds to $tilt = \frac{\pi}{2}$ and $pan = 0$. The positive direction of the Y axis corresponds to $tilt = \frac{\pi}{2}$ and $pan = \frac{\pi}{2}$. The positive direction of the Z axis corresponds to $tilt = 0$. In order to make the tessellation as uniform as possible and to make the number of mechanical operations as small as possible, we use the following method to tessellate the surface of the unit sphere. First, we divide the horizontal big circle (corresponding to $tilt = \frac{\pi}{2}$) uniformly with angle α (i.e., for the horizontal big circle, the value of pan can be 0, α, 2α,... see Figure 5.1(b)). Then we divide the vertical big circle, which corresponds to $pan = 0$, with the amount of $tilt$ equal to α (i.e., for the vertical big circle, $tilt$ can be 0, α, 2α,..., see Figure 5.1(c)). The area of patches at the equator is approximately α^2. However, the area of each surface patch for other $tilt$s will differ with this tessellation scheme. In order to keep patch areas approximately equal, we adjust the amount of pan for each different $tilt$. In Figure 5.1(d) $\angle AOB = \alpha$, which corresponds to the value of pan when $tilt = \frac{\pi}{2}$; $\angle FOC = \eta$, which is the amount of tilt. We want to find the value of pan for $tilt = \eta$ such that the area of the surface patch corresponding to $tilt = \eta$ is approximately equal to the area of the surface patch corresponding to $tilt = \frac{\pi}{2}$. Thus, the big circle arc $\hat{AB}$ (which corresponds to $tilt = \frac{\pi}{2}$) should be same as the big circle arc $\hat{CD}$ (which corresponds to $tilt = \eta$) (Figure 5.1(d)). This follows $\mid AB \mid = \mid CD \mid$. After some calculations we obtain the amount of pan corresponding to $tilt = \eta$ is

$$2arcsin\{\frac{sin(\frac{\alpha}{2})}{sin\eta}\} \tag{5.1}$$

Figure 5.2 shows an example of a tessellated sphere.

Generally speaking, the value of α used in the tessellation is determined by the complexity of the environment and the pan and tilt resolution limits of the hardware. The more complex the environment geometry, the smaller the value of α should be in order to satisfactorily approximate the environment.

The sensed sphere constructed by the method above can be concisely represented as the union of all solid angles and the corresponding distances measured by laser:

$$SensedSphere = \bigcup_i SolidAngle_i(pan_{b_i}, pan_{e_i}; tilt_{b_i}, tilt_{e_i}; radius_i; prob_i)$$

where pan_{b_i}, pan_{e_i} are the beginning and ending of pan, and $tilt_{b_i}, tilt_{e_i}$ are the beginning and ending of $tilt$ for the solid angle $SolidAngle_i$. Parameter $radius_i$ is the "radius" of this angle, which is the distance measured using laser at $pan = \frac{pan_{b_i}+pan_{e_i}}{2}$ and $tilt = \frac{tilt_{b_i}+tilt_{e_i}}{2}$. Parameter $prob_i$ is the probability that the target is within the solid angle $SolidAngle_i$. Any point in the space whose $tilt, pan$ satisfy $tilt_{b_i} \leq tilt < tilt_{e_i}; pan_{b_i} \leq pan < pan_{e_i}$ belongs to $SolidAngle_i$. Each point in the space belongs to only one solid angle.

The sensed sphere can be used to register the solidity property of the search region — whether a cube is occupied by a solid object. Since the sensed sphere is an approximation of the real environment surrounding the camera center, we make the following assumption: for any point or volume within a given solid angle, if its distance to the camera center is almost the same as the radius of the corresponding solid angle, and its distance to the environmental point that is hit by the laser is within a certain range, then it is occupied by a solid object.

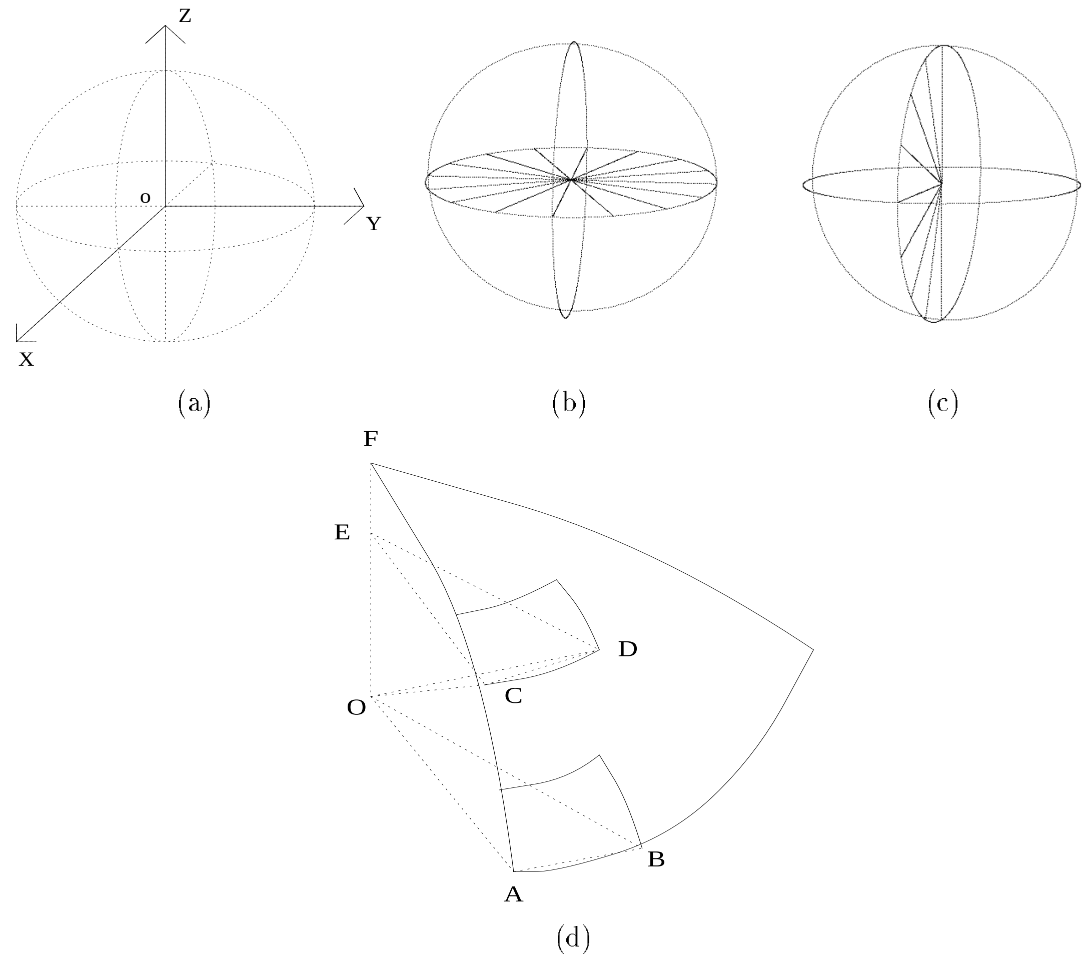
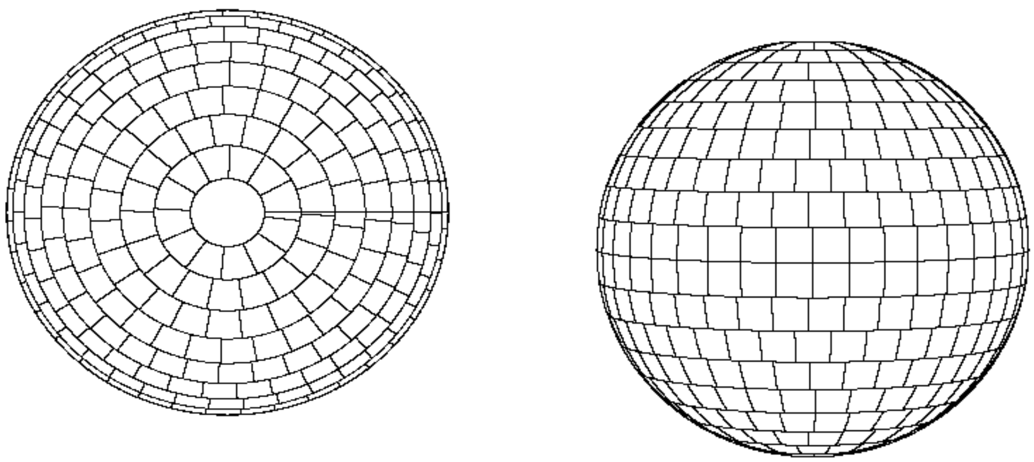


Figure 5.1: Tessellation of the unit sphere



(a) (b)

Figure 5.2: (a) The top view and (b) The side view of the tessellate sphere with $\alpha_0 = 0.157$ (9 degrees).

This is so because the laser, which emits from the center of the camera and along the central axis of the corresponding solid angle, hits a solid object with the distance equal to the radius of the solid angle during the construction of the sensed sphere.

For each cube c, its center belongs to only one solid angle A_c. Suppose the radius of A_c is r_{A_c} and the point hit by the laser beam along the central axis of A_c is P_{A_c}. Suppose the camera center is O. Let $\mid c - O \mid$ represent the distance between the center of c and the camera center, and $\mid c - P_{A_c} \mid$ represent the distance between c and P_{A_c}. Then for each cube c within the environment, we register its solidity property as follows. If $\mid c - O \mid \leq \delta_1$ and $\mid c - P_{A_c} \mid \leq \delta_2$, then the solidity of cube c is registered as *solid*, otherwise it is registered as *unknown*. Constants δ_1 and δ_2 are determined before the search process.

Constructing the full sensed sphere may be expensive if the mechanical operations takes time. An alternative way is to change the algorithm described above such that the sensed sphere is only computed in sectors, i.e., regions of the sphere where probabilities are high enough to warrant exploration. This can be a large improvement over the full sensed sphere when the cost of mechanical operations of manipulating the pan, tilt of the camera is high.

Chapter 6

Where to Look Next

6.1 The Objective Function

The sensed sphere will be used to select the field of view and the direction of the viewing axis of the camera. From Section 2.4, we know that an operation can be represented as $\mathbf{f} = \mathbf{f}(x_c, y_c, z_c, p, t, w, h, a)$. When (x_c, y_c, z_c) is fixed, we represent it as $\mathbf{f} = \mathbf{f}(p, t, w, h, a)$. Suppose the region of the viewing volume determined by p, t, w, h is V and the region confined by the current sensed sphere is C. Let $\Psi = V \bigcap C$. Then, of course, Ψ is those parts of Ω that can be "seen" by the current camera setting (see Figure 6.1 for illustration). The probability of detecting the target by operation $\mathbf{f} = \mathbf{f}(p, t, w, h, a)$ is:

$$\mathbf{P}_\Psi(\mathbf{f}) = \sum_{c_i \in \Psi} \mathbf{p}(c_i, \tau_\mathbf{f})\mathbf{b}(c_i, \mathbf{f}) \tag{6.1}$$

The utility function of operation $\mathbf{f}$ is

$$\mathbf{E}_\Psi(\mathbf{f}) = \frac{\sum_{c_i \in \Psi} \mathbf{p}(c_i, \tau_\mathbf{f})\mathbf{b}(c_i, \mathbf{f})}{\mathbf{t}(\mathbf{f})} \tag{6.2}$$

Our goal is to select an operation with the highest utility function value. We first solve a simpler problem: how to select the parameters p, t, w, h of the next operation when the recognition algorithm a is given.

The cost function $\mathbf{t}(\mathbf{f})$ is actually determined by the time used to perform state transition and the time used to take a picture and run the recognition algorithm. If we simplify the problem by assuming that the time used for state transition is the same for any operation, then $\mathbf{t}(\mathbf{f})$ is only influenced by the time used for recognition.

For a given algorithm a, the time needed to perform recognition is determined by the complexity of the image, which is again determined by the parameters p, t, w, h and the environment. Since the situation of the environment is hard to predict, we can test the recognition algorithm a under various background complexities and select the average value of time needed as the cost to run this algorithm. Thus for any operation $\mathbf{f_a}$ which takes a as its recognition algorithm, the cost function $\mathbf{t}(\mathbf{f_a})$ is the same for any p, t, w, h and background.

When recognition algorithm a is given, the problem of selecting an operation $\mathbf{f_a}$ with the highest utility function value is equivalent to the problem of selecting an operation $\mathbf{f_a}$ with the highest probability of detecting the target. Thus, we must maximize

$$\mathbf{P}_\Psi(\mathbf{f_a}) = \sum_{c_i \in \Psi} \mathbf{p}(c_i, \tau_{\mathbf{f_a}})\mathbf{b}(c_i, \mathbf{f_a}) \tag{6.3}$$

An enumeration approach would be to calculate the above equation for all the possible actions and select the one with maximum $\mathbf{P}_\Psi(\mathbf{f_a})$. This is time consuming because of the huge number of possible actions, as determined by the total number of camera configurations that can be achieved by the hardware. This number is the product of: (1) the number of different viewing angle sizes $N_{|<w,h>|}$ determined by the zoom lens of the camera; (2) the number of different pan angles $N_{|p|}$; and (3) the number of different tilt angles $N_{|t|}$. $N_{|<w,h>|}$, $N_{|p|}$, and $N_{|t|}$ are determined by the hardware of the camera and the robotic head. Now, let us analyze the time needed to find an optimum action among the available actions determined by the hardware. Suppose $N_{|<w,h>|} = 1500$, $N_{|p|} = 360$, and $N_{|t|} = 360$, then the number of possible actions is $1500 = 194400000$. Suppose the time needed to calculate $\mathbf{P}_\Psi(\mathbf{f_a})$ for a single action is one second (The calculation of $\mathbf{P}_\Psi(\mathbf{f_a})$ is time consuming in general). Then the system needs 54000 hours to find the optimum answer.

To make the problem tractable, we must decompose this huge space of possible sensing actions into a finite set of actions that must be considered and select the next action among the resulting limited set of actions. Note that the selected actions might not be the optimal action with respect to the original huge space of sensing actions, because the "real" optimal action might not belong to the limited finite set.

The following sections will illustrate how to decompose the huge space of possible actions into a limited set of actions that must be considered and how to select the next action among this limited set of actions. We assume that the **recognition algorithm** and the **target** are **given** in all the discussions.

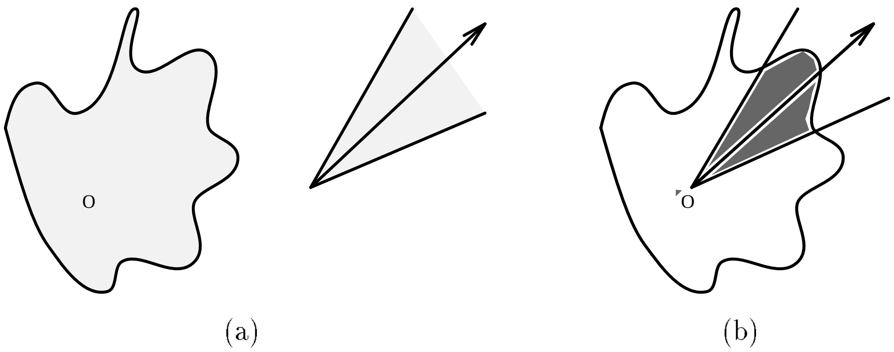


Figure 6.1: (a) The $2D$ illustration of the sensed sphere and the camera's parameter settings. O is the center of the camera. (b) The region which can be seen by the current camera settings.

6.2 Select Viewing Angle Sizes

In this section, we discuss the problem of selecting the candidate viewing angle sizes for a given recognition algorithm.

The size of the viewing angle has a close relationship to the expected distance and size of the target object. Of course it is not necessary to try each possible viewing angle size one by one; we try to select those sizes that are really necessary. The ability of the recognition algorithm and the value of the detection function is influenced by the image size of the target. Usually the recognizer can successfully recognize the target only when the image size of the target is within a certain range. If the image size of the target is too big, then some features of the target may be out of the view of the camera, thus the given recognizer may not be able to recognize the target. If the image size is too small then, because of the limited resolution of the image, some features of the target may not be correctly detected by the given recognition algorithm. When the target can be totally brought into the field of view of the camera and the features can be detected within certain precision, the recognition algorithm can be expected to function correctly. So, for an operation with a given recognition algorithm and a fixed viewing angle, the probability of successfully recognizing the target is high only when the target is within a certain distance range. Therefore, different sizes of the viewing angle will be associated with different distance ranges. We call this range the **effective range** of the corresponding viewing angle size (Figure 6.2(a)). Generally, the effective range can be judged from the detection function value for this viewing angle size. Our purpose here is to select those angles whose effective ranges will cover the whole distance of the depth D of the search region, and at the same time there will be no overlap of their effective ranges (Figure 6.2(b)). If the relationship between effective ranges and angles is known for a given recognition system, then, for a given search task, the selection of viewing angle sizes can proceed via table lookup indexed by the distance from the robot.

Suppose that the biggest viewing angle for the camera is $w_0 \times h_0$ and its effective range is $[N_0, F_0]$ (Figure 6.2(b)). This means that N_0 is the beginning of the effective range for this viewing angle, and F_0 is the end of the effective range for this viewing angle. When the distance of the target to the camera, along the direction of the viewing axis, is within the range of $[N_0, F_0]$, the value of the detection function is relatively high, and thus there is high probability that the recognition algorithm can detect the target. We will now use a geometric constraint to find other necessary viewing angles $w_1 \times h_1, \ldots, w_{n_0} \times h_{n_0}$ and their corresponding effective ranges $[N_1, F_1], \ldots, [N_{n_0}, F_{n_0}]$, such that $[N_0, F_0] \bigcup \ldots \bigcup [N_{n_0}, F_{n_0}] \supseteq [N_0, D]$ and $[N_i, F_i) \bigcap [N_j, F_j) = \emptyset$ if $i \neq j$ (Figure 6.2(b)). These $n_0 + 1$ angle sizes are enough to examine the whole depth of the search space with high probability.

Since the effective range $[N_1, F_1]$ of the next viewing angle $w_1 \times h_1$ should be just next to the effective range of the first viewing angle $w_0 \times h_0$, we have $N_1 = F_0$ (Figure 6.2(b)). To guarantee that the area of the image of the target patch of the first viewing angle $w_0 \times h_0$ at

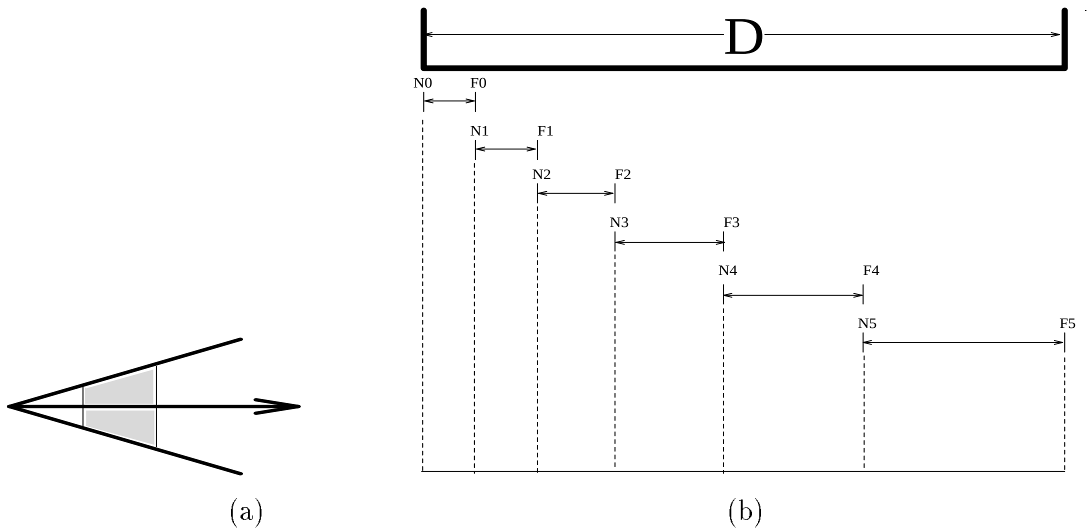


Figure 6.2: $2D$ illustration of the selection of the camera's angle size. (a) The effective range of a given angle size. (b) The viewing angle sizes should be selected such that their effective range can cover the whole distance of the depth D and at the same time there will be no overlap of their effective ranges.

N_0 and F_0 equal the area of the image of the target patch of the second viewing angle $w_1 \times h_1$ at N_1 and F_1 respectively, we get (using Equations (3.9))

$$w_1 = 2arctan(\frac{N_0}{N_1}tan(\frac{w_0}{2})); h_1 = 2arctan(\frac{N_0}{N_1}tan(\frac{h_0}{2})); F_1 = F_0(\frac{F_0}{N_0})$$

Similarly, we can get

$$w_i = 2arctan[(\frac{N_0}{F_0})^i tan(\frac{w_0}{2})] \tag{6.4}$$

$$h_i = 2arctan[(\frac{N_0}{F_0})^i tan(\frac{h_0}{2})] \tag{6.5}$$

$$N_i = F_0(\frac{F_0}{N_0})^{i-1}; F_i = F_0(\frac{F_0}{N_0})^i \tag{6.6}$$

where w_i, h_i is the ith angle size we select, and $[N_i, F_i]$ is the corresponding effective range. Since $N_i \leq D$, we obtain

$$i \leq \frac{ln(\frac{D}{F_0})}{ln(\frac{F_0}{N_0})} - 1 \tag{6.7}$$

Let

$$n_0 = \lfloor \frac{ln(\frac{D}{F_0})}{ln(\frac{F_0}{N_0})} - 1 \rfloor \tag{6.8}$$

then the angles to be considered by the sensing actions are $w_0 \times h_0, w_1 \times h_1, \ldots, w_{n_0} \times h_{n_0}$.

So, when the target position is between $[N_i, F_i]$, it can be detected with high probability by using angle size $w_i \times h_i$. When the target position is between $[N_{i+1}, F_{i+1}]$, it can be detected with high probability by using angle size $w_{i+1} \times h_{i+1}$. When the target position is exactly at the boundary that connect $[N_i, F_i]$ and $[N_{i+1}, F_{i+1}]$ (which means that the target position is at

F_i and N_{i+1}), then it can be detected with high probability by using both angle size $w_i \times h_i$ and angle size $w_{i+1} \times h_{i+1}$.

Figure 6.3 shows the influence of the characteristic ratio $\frac{F_0}{N_0}$ on the effective range, the size of the viewing angle, the effective viewing angle and the number of angles needed. Figure 6.3(a) shows that for big characteristic ratios, the beginning of the effective range will increase very quickly as the index of the effective camera angle increases. But Figure 6.3(b) tells us that we are not able to have many effective camera angles for big characteristic ratios, because the size of the angle will decrease so rapidly that the hardware may not be able to provide this size. Figure 6.3(c) shows that the total effective volume (the $3D$ space within the effective range that can be seen) will increase as the index of the effective camera angle increases. This suggests that it is better to remain in one place and search the space for as far as the camera can see, rather than drive the robot near the unsearched space. Figure 6.3(d) shows that for a relatively large characteristic ratio, only a few sizes of viewing angle are enough for an object search task.

6.3 Select Viewing Direction

6.3.1 Layered Sensed Sphere

For a given recognition algorithm and a given target, the sensed sphere can be further divided into several layers according to the viewing angle sizes $w_0 \times h_0, w_1 \times h_1, \ldots, w_{n_0} \times h_{n_0}$, where $w_i \times h_i$ are the effective angle sizes determined by the method of Section 6.2 (see Figure 6.5 for an illustration). Each layer corresponds to an effective viewing angle size. It can be represented as follows:

$$\bigcup_{<w,h>=<w_0,h_0>,\ldots,<w_{n_0},h_{n_0}>}$$

$$\bigcup_i LSolidAngle_i(pan_{b_i}, pan_{e_i}; tilt_{b_i}, tilt_{e_i}; radius_i; N_{<w,h>}, F_{<w,h>}; prob_i^{<w,h>})$$

where $radius_i$ is the radius of the corresponding $SolidAngle_i$; $N_{<w,h>}$, $F_{<w,h>}$ is the near side and far side of the effective range corresponding to angle size $w \times h$; $prob_i^{<w,h>}$ is the sum of the probability of all the cubes which satisfy the following (see Figure 6.4 for an illustration): (1) it is within $SolidAngle_i$, (2) the distance to the camera center is less than $radius_i$, and (3) the distance to the camera center is within the effective range of $< w, h >$, which means that the distance should be greater than $N_{<w,h>}$ and less than $F_{<w,h>}$.

6.3.2 Select Viewing Direction

In this section, we discuss the problem of selecting the viewing direction for a given recognition algorithm and a given effective angle size.

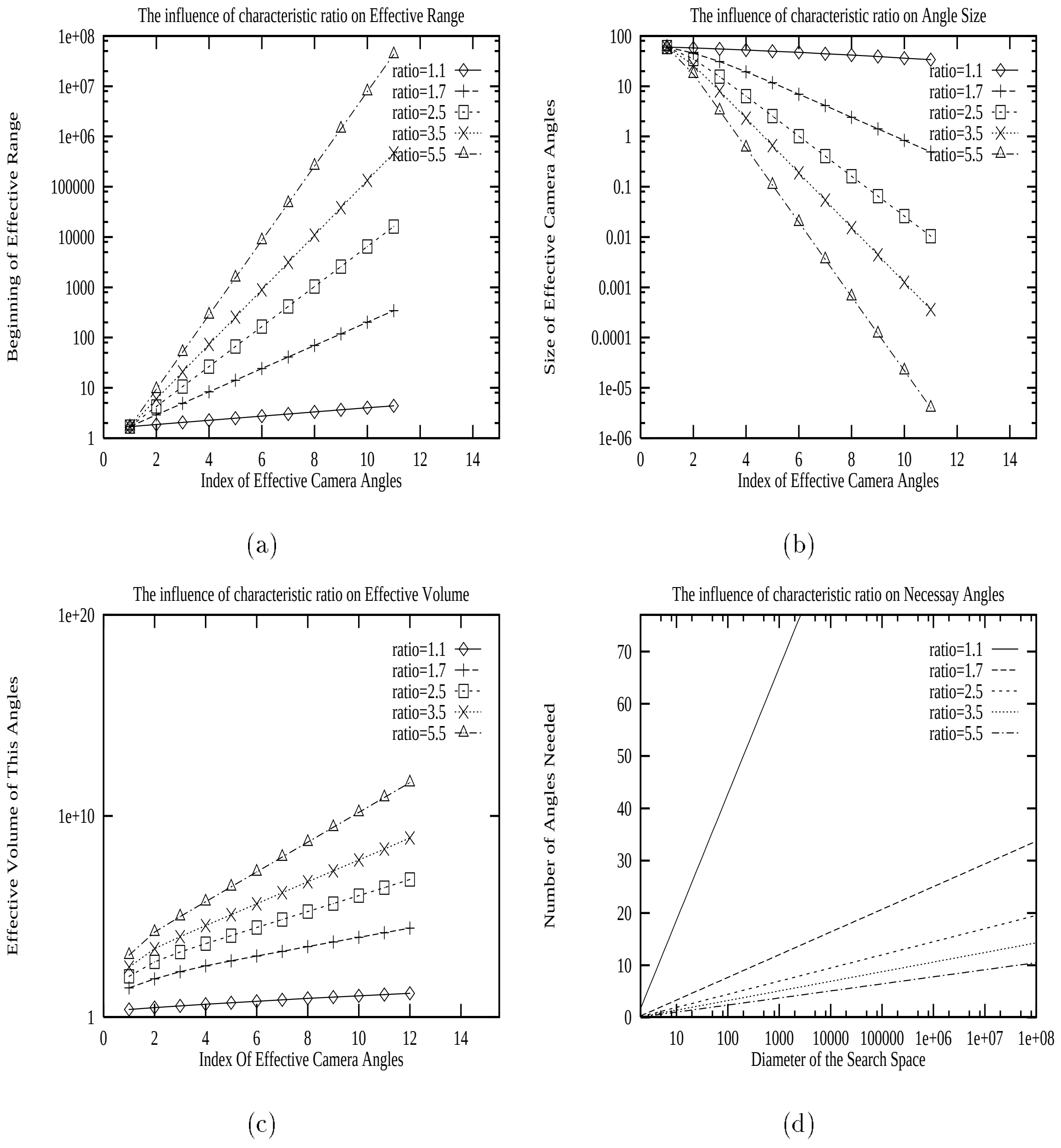


Figure 6.3: The Influence of Characteristic Ratio when $N_0 = 1.7$, $w_0 = h_0 = 1.046$ radian (60^o degrees)

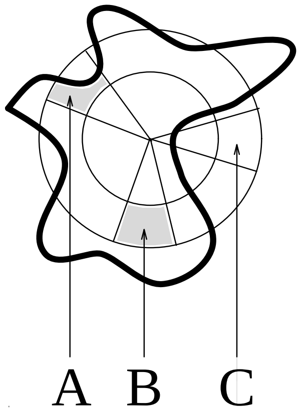


Figure 6.4: $2D$ illustration of the calculation of $prob_i^{<w,h>}$. For angle **A**, the probability is the sum of the shaded area, which is only part of the layered slice. For angle **B**, the probability is the sum of the shaded area, which is the whole layered slice. For angle **C**, the probability is 0.

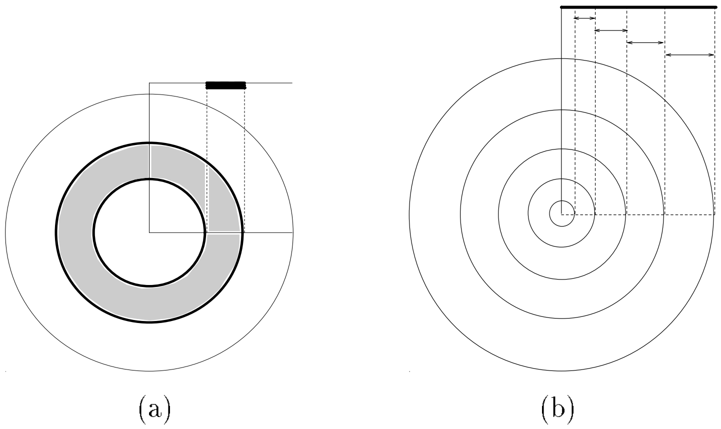


Figure 6.5: $2D$ illustration of the layered sensed sphere. (a) A single layer corresponding to a given effective range. (b) The layered sensed sphere.

The layered sensed sphere (LSS) for the given recognition algorithm and target created above can be used to select the viewing direction of the camera. For any given effective angle size $w_k \times h_k$, where $0 \leq k \leq n_0$, there is a layer in the layered sensed sphere that corresponds to it. We denote this layer as $LSS_{<w_k,h_k>}$. We now try to find a viewing direction $< p_k, t_k >$ that can "see" the region of $LSS_{<w_k,h_k>}$ which has the maximum probability of the presence of the target. The strategy is to first select a set of candidate directions, and then compare among them to select the best one.

The selection of the candidate directions is similar to the tessellation of the sensed sphere, but not the same (Please see Appendix C for details). We only consider those directions whose viewing volumes will have minimal overlap with each other and whose union of viewing volumes will cover the whole sphere (see Figure 6.6 for an illustration). Thus we can use the similar tessellation method as we did for the construction of the sensed sphere, with the tessellating angle size α equals to a value that is a slightly smaller than $min\{w_k, h_k\}$. The reason to make α slightly smaller than $min\{w_k, h_k\}$ is to avoid the boundary problem —— when the target is at the boundary of two neighboring viewing angle determined by the tessellation of using α, it can actually be totally bring into the field of view of the camera because the actual camera angle size is bigger than α. The central axis of each resulted solid angle will be taken as the candidate viewing direction. Please refer to Appendix C for details about the selection of α and the tessellation.

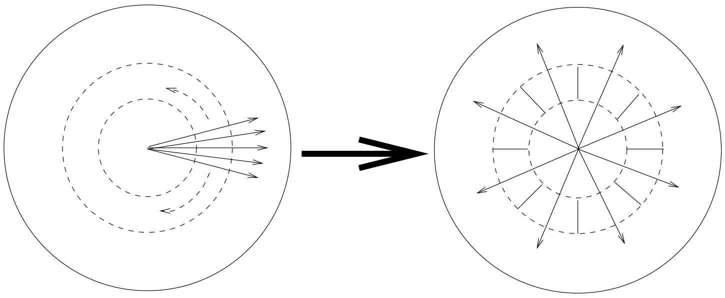


Figure 6.6: $2D$ illustration of the selection of viewing directions. For a given angle size, there are a huge number of candidate directions that can be selected from (left figure). We only consider those directions such that their effective volume will cover the whole layer, and there is very little overlap among these effective viewing volumes (right figure).

For any viewing candidate direction $< p, t >$, the presence of the target on $LSS_{<w_k,h_k>}$ is approximated by the sum of the probability on $LSS_{<w_k,h_k>}$ of those radii that fall within the viewing volume determined by $< p, t >$. Let V be the viewing volume determined by parameters t, p, w_k, h_k. In the following we will determine whether a radius of the sensed sphere, with direction $< t_i, p_i >$, will fall into the viewing volume V.

First, we change the head coordinate system so as to make the t, p become $(\frac{\pi}{2}, 0)$. This can be done by first rotating the sphere around Z axis the amount of p, and then rotating the sphere around Y axis the amount of $\frac{\pi}{2} - t$. In the new coordinate system, the region covered by this view volume can be represented by a pyramid $O-ABCD$ (Figure C.5(a)). The coordinates for points O, A, B, C, and D in the Head Coordinate System are:

$O(0,0,0)$,

$A(\frac{r}{\sqrt{1+2tan^2(\frac{\alpha}{2})}}, -\frac{rtan(\frac{\alpha}{2})}{\sqrt{1+2tan^2(\frac{\alpha}{2})}}, -\frac{rtan(\frac{\alpha}{2})}{\sqrt{1+2tan^2(\frac{\alpha}{2})}})$,

$B(\frac{r}{\sqrt{1+2tan^2(\frac{\alpha}{2})}}, \frac{rtan(\frac{\alpha}{2})}{\sqrt{1+2tan^2(\frac{\alpha}{2})}}, -\frac{rtan(\frac{\alpha}{2})}{\sqrt{1+2tan^2(\frac{\alpha}{2})}})$,

$C(\frac{r}{\sqrt{1+2tan^2(\frac{\alpha}{2})}}, \frac{rtan(\frac{\alpha}{2})}{\sqrt{1+2tan^2(\frac{\alpha}{2})}}, \frac{rtan(\frac{\alpha}{2})}{\sqrt{1+2tan^2(\frac{\alpha}{2})}})$,

$D(\frac{r}{\sqrt{1+2tan^2(\frac{\alpha}{2})}}, -\frac{rtan(\frac{\alpha}{2})}{\sqrt{1+2tan^2(\frac{\alpha}{2})}}, \frac{rtan(\frac{\alpha}{2})}{\sqrt{1+2tan^2(\frac{\alpha}{2})}})$.

Second, the direction of any radius $< t_i, p_i >$ in the new coordinate system can be calculated as follows: On the unit sphere, the point (x, y, z) that corresponds to this radius is:

$$\begin{aligned} x &= sin(t_i)cos(p_i) \\ y &= sin(t_i)sin(p_i) \\ z &= cos(t_i) \end{aligned}$$

After rotating around Z with the amount of p, the new coordinate is $(x^{'}, y^{'}, z^{'})$, where

$$\begin{aligned} x^{'} &= xcos(p) + ysin(p) \\ y^{'} &= -xsin(p) + ycos(p) \\ z^{'} &= z \end{aligned}$$

After rotating around Y with amount of $\frac{\pi}{2} - t$, the coordinates become

$$\begin{aligned} x^{''} &= x^{'}sin(\tfrac{\pi}{2} - t) - z^{'}cos(\tfrac{\pi}{2} - t) \\ y^{''} &= y^{'} \\ z^{''} &= x^{'}cos(\tfrac{\pi}{2} - t) + z^{'}sin(\tfrac{\pi}{2} - t) \end{aligned}$$

Then the new tilt will be

$$tilt_{new} = \begin{cases} arctan(\frac{\sqrt{(x^{''})^2+(y^{''})^2}}{z^{''}}) & \text{if } z^{''} > 0 \\ \pi + arctan(\frac{\sqrt{(x^{''})^2+(y^{''})^2}}{z^{''}}) & \text{if } z^{''} < 0 \end{cases}$$

The new pan will be

$$pan_{new} = \begin{cases} arctan(\frac{y^{''}}{x^{''}}) & \text{if } x^{''} > 0 \text{ and } y^{''} \geq 0 \\ \pi + arctan(\frac{y^{''}}{x^{''}}) & \text{if } x^{''} < 0 \\ 2\pi + arctan(\frac{y^{''}}{x^{''}}) & \text{if } x^{''} > 0 \text{ and } y^{''} < 0 \end{cases}$$

If $tilt_{new}$ and pan_{new} is within pyramid $O-ABCD$, then the radius with direction $< t_i, p_i >$ belongs to the viewing volume $\mathbf{V}$.

Since $prob_i^{<w_k,h_k>}$ gives the probability of the presence of the target object within both the solid angle $LSolidAngle_i$ of the sensed sphere and LSS_k of the layered sensed sphere, the total probability of presence within both V and LSS_k should be the sum of the corresponding probabilities within LSS_k of all those solid angles whose radii fall into the view volume V:

$$\sum_{radius_i \in \mathbf{V}} prob_i^{<w_k,h_k>} \tag{6.9}$$

Thus we can perform calculations (according to formula 6.9) for each candidate viewing direction t, p and select the direction with the maximum value as the best direction $< p_k, t_k >$ for the corresponding viewing angle size $< w_k, h_k >$.

6.4 Select the Best Action

For a given recognition algorithm a, we can find a list of candidate viewing angle sizes $w_0 \times h_0, w_1 \times h_1, \ldots, w_{n_0} \times h_{n_0}$ according to Section 6.2 (Note: the value of n_0 and $w_1 \times h_1, \ldots, w_{n_0} \times h_{n_0}$ is influenced by the recognition algorithm a). For each candidate angle size $< w_k, h_k >$ $(0 \leq k \leq n_0)$, we can find a best direction $< p_k, t_k >$ according to Section 6.3. So, we can get $n_0 + 1$ candidate actions $\mathbf{f}_0 = \mathbf{f}(p_0, t_0, w_0, h_0, a)$, $\mathbf{f}_1 = \mathbf{f}(p_1, t_1, w_1, h_1, a)$, ..., $\mathbf{f}_{n_0} = \mathbf{f}(p_{n_0}, t_{n_0}, w_{n_0}, h_{n_0}, a)$. Then we can calculate the probability of detection for each $\mathbf{f}_k, 0 \leq k \leq n_0$ according to Formula (6.1) and select the one with highest value as the best candidate action with respect to the recognition algorithm a.

For any recognition algorithm a, we can get a best candidate action $\mathbf{f_a}$ according to the above method. Suppose $a_1, a_2, \ldots, a_m$ are the recognition algorithms available for the search task. Then we can get m candidate actions $\mathbf{f_{a_1}}, \ldots, \mathbf{f_{a_m}}$. For each candidate action, we can calculate its utility function value according to Equation (6.2). Then the action with the highest utility value will be taken as the next action to be executed.

After applying each action, we will update the probability of any cubes within the search space. Since the updating process might take time, we can select several operations under current world state, and update the state after these operations are applied. The probability of presence within the current sensed sphere SS is $Prob_{SS} = \sum_{c_i \in SS} \mathbf{p}(c_i)$. This probability will be decreased each time an operation is applied. When the ratio of $Prob_{SS}$ and the probability of presence within the search space $\frac{Prob_{SS}}{1-\mathbf{p}(c_o)}$ is less than a certain threshold, or when the probability of detecting the target by the selected action is too small, we will want the robot to change position.

It is clear that our action selection process is guided by the target probability distribution. If there is no prior information, then it is guided by the size of the effective volume determined by the camera's state parameters. It is interesting to ask whether it is worth to use gradient descent method to find the maximum value of Equation (6.3). We believe that from the point of view of calculation cost, the gradient descent method may not be attractive. Because the target probability distribution is determined by the initial knowledge and the updated results of later actions, thus the data provided by $\mathbf{p}(c_i, \tau_{\mathbf{f_a}})$ has no regularity and cannot be represented as an analytical expression. Thus, the function $\mathbf{P}_{\Psi}(\mathbf{f_a})$ cannot in general represented as an analytical expression. So, the only way to calculate the gradient descent is to do it discretely. This will not save the computation cost as compared with our method.

Chapter 7

Where to Move Next

7.1 Overview of Robot Navigation Issue

When a change of position is needed, a robot performing a search task has two immediate problems that must be solved. The first is the localization problem, which gives the answer to the question "Where am I". The second is the guidance problem, which gives the answer of the question "What is the next destination and which path does the robot follow to arrive at the destination?".

Navigation is a fundamental requirement of autonomous mobile robots. It is defined as "the science of getting ships, aircraft, or spacecraft from place to place; esp: the method of determining position, course, and distance traveled" [55]. It is surprising to find that position estimation for mobile robots is still problematic, given the available knowledge and experience in the land, marine and aerospace communities. One reason might be a lack of knowledge regarding the navigation literature from other disciplines. But a more important reason is that autonomous vehicle research has, until quite recently, avoided any use of artificial beacons, that is, either active or passive beacons that are placed in the vehicle's environment for navigation purposes. This is because the primary interest in autonomous vehicles has been from research into artificial intelligence with the goal of developing "intelligent" mobile machines, the behavior of which closely resembles that of animals. As a consequence, vehicle navigation systems have had to identify naturally occurring landmarks in the environment, which is difficult due to noise and difficulties in interpreting the sensory information from computer vision or ultrasonic sensors.

Because the emphasize of this thesis is not robot localization but object search, we simplify the localization problem by assuming that beacons are available to assist. Placing easy to recognize beacons in the robot's workspace is an effective way of avoiding many of the problems associated with recognizing naturally occurring landmarks. Artificial beacons may be either active or passive. In the active case, the beacons may transmit periodic waveforms that are received by the (passive) receiver aboard the vehicle. Active beacons are a very common method of navigation for both aircraft and shipping, and have been used for mobile robot localization.

Durieu et al. [25] analyze the operation and performances of a navigation system for a mobile robot using ultrasonic/infra-red telemeter with active beacons, intended for a mobile inspection robot application. During the navigation process, the robot emits an ultrasonic pulse which is detected by the beacons with a delay proportional to the distance from the beacons (Figure 7.1(a)). Each beacon then returns a coded infra-red answer so that the robot has a measurement of its distance to the beacon. Based on the distance measurements to the available beacons, the robot position can be calculated accordingly. Passive beacons, for example, bar codes or retroflectors, are more commonly used for mobile vehicle applications, in which case the vehicle is usually equipped with an active transmitter such as a laser. Passive beacons have the advantages of robustness and reliability, because they require no power and are completely inert. They are also cheaper, since a single active transmitter on the vehicle can be served by many passive beacons. Robins [64] describes a automatic guided vehicles (AGV) developed by GEC Electrical Projects Ltd and been launched as a full product at the GEC Centenary Exhibition in 1986. There is a laser scanner on the vehicle, which is used to read passive bar-coded reflectors (or passive beacons) and to measure the angle of the reflectors relative to the axis of the movement of the AGV (Figure 7.1(b)). The reflectors are constructed from strips of retro-reflective and non-reflective material arranged to form a bar code. Retro-reflection is obtained when a ray of light enters the inside of a mirrored cube. The light is reflected around and emerges parallel with, but in the opposite direction to, the incident ray. The return beam to the scanner is converted by a photo-diode to an electrical signal which is a replica of the bar code.

From the above discussions we know that there exist ways to obtain the robot position within a reasonable precision range. We assume therefore that the error of the robot position estimation is within a certain small range. The following sections will address the problem of "where to move next".

7.2 The Selection of the Next Robot Position

In order to decide where to move next, first, we need to decide which points the robot **can reach**. Then we need to know which one is the **most beneficial** among these reachable points.

7.2.1 The Candidate Positions

In order to simplify the navigation problem, we limit the movement of the robot from current position to next position to a **straight line**.

As described in Section 2.4.3, the search space Ω is tessellated into a $3D$ grid. This tessellation also divides the horizontal plane of the search region into a $2D$ grid. To save computational time, we only consider the vertex (or the center) of these $2D$ grids as the possible position of the robot. Thus, there is only a limited number of next candidate positions to be considered.

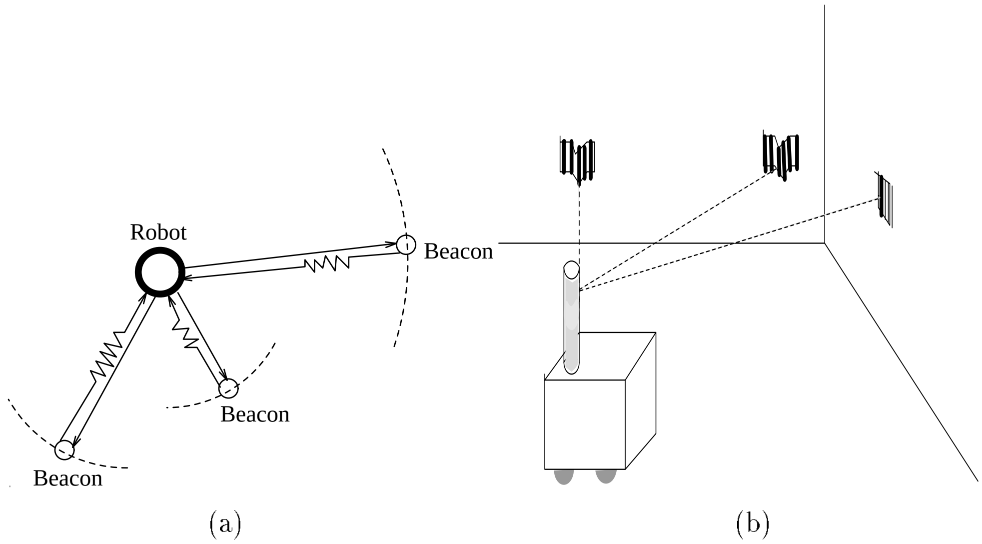


Figure 7.1: (a) The active beacon. (b) The passive beacon.

These next positions can be further reduced by considering whether the robot can reach these positions from its current position via a straight line. When the robot moves from its current position to a candidate position, there will be a volume of space that is traversed by the robot. According to the current registration of the space, if there are no solid cubes present within this volume, then we assume the robot can reach this candidate position, otherwise, we assume the robot cannot reach this candidate position. We only consider those candidate positions that the robot can reach by the above criteria.

7.2.2 Select the Next Robot Position

The Objective Function

For each candidate position, we give an objective criterion that estimates the benefit when the robot is moved to this position. In the following discussion, the target is assumed to be specific.

We first explain the concept of **cover range** of the camera and how to find the **near radius** and **far radius** of the cover range. The cover range means that when the target is within this range, we are able to adjust the camera's state parameters and select an appropriate recognition algorithms such that the target can be detected with high probability. The near radius of the cover range means the smallest distance of this range, the far radius of the cover range means the biggest distance of this range.

For a given camera, there are limits for the biggest angle size and the smallest angle size. This is determined by the mechanical structure of the zoom lens of the camera. For each recognition algorithm a_i, there exists a distance d_{a_i}, such that if the distance of the target to the camera is less than d_{a_i}, even if the camera angle size is the biggest, the probability of detecting the target is too small because the target is too near to the camera. We call this distance the **near distance** for recognition algorithm a_i with respect to the camera. This distance is the same as the near side $N_{0_{a_i}}$ of the first effective range $[N_{0_{a_i}}, F_{0_{a_i}}]$ of the recognition algorithm a_i. For every available recognition algorithm a_i, we can get a near distance. We call the smallest of these the **near radius** of the cover range for the camera.

Similarly, for each recognition algorithm a_i, there exists a distance D_{a_i}, such that if the distance of the target to the camera is larger than D_{a_i}, then even if the camera angle size is the smallest, the probability of detecting the target is too small because the target is too far from the camera. We call this distance the **far distance** for recognition algorithm a_i with respect to the camera. For every available recognition algorithm, we can get a far distance. We call the biggest of these the **far radius** of the cover range for the camera.

Any distance between the near radius and far radius of the camera has the property that if the target is at this distance, by adjusting the camera's viewing angle size and selecting an appropriate recognition algorithm, it can be detected with reasonable probability. Thus, the range between the near radius and far radius of the camera is called the **cover range** of the camera, which means that this range can be examined with high probability by the given camera and the available recognition algorithms.

Suppose that the robot is currently at position **B** of the environment and **A** is a candidate next position (Figure 7.2). We first assume that the robot knows the geometric configuration of the environment. Thus the sensed sphere from point **A** can be **calculated** (Figure 7.2 (c)(d)). We sum up the probabilities of all the cubes that belong to the sensed sphere and the cover range of the camera (Figure 7.2 (e)(f)). It is obvious that these cubes can be examined by the camera with satisfactory results. We call this the covering probability for candidate position **A**. This covering probability will be used as a criterion to select the best next position.

But generally, we may not know the exact geometric configuration of the environment. We can only make use of our knowledge about the geometric configuration obtained so far, and use this incomplete knowledge to guide our selection. From the sensed sphere constructed at the first position **A**, we can determine some solid cubes in the environment (Figure 7.2 (g)(h)). From these solid cubes and the boundary of the search region, we can obtain an expected sensed sphere (Figure 7.2 (i)(j)). The covering probability will then be calculated as the sum of the probabilities of all the cubes that belong to this expected sensed sphere and the covering range of the camera.

The Algorithm

Now we give the algorithm to select the next robot position among these reachable candidate positions. For each reachable candidate position of the robot, we can **calculate** its corresponding **expected sensed sphere** SS_e as follows. We calculate the radius for each solid angle of SS_e, assuming we pinged a laser in each direction from the candidate position. The boundary of the search region Ω (this is assumed to be known) is taken to be solid. We update these radii according to the current registration of the search space Ω. For each solid cube and its

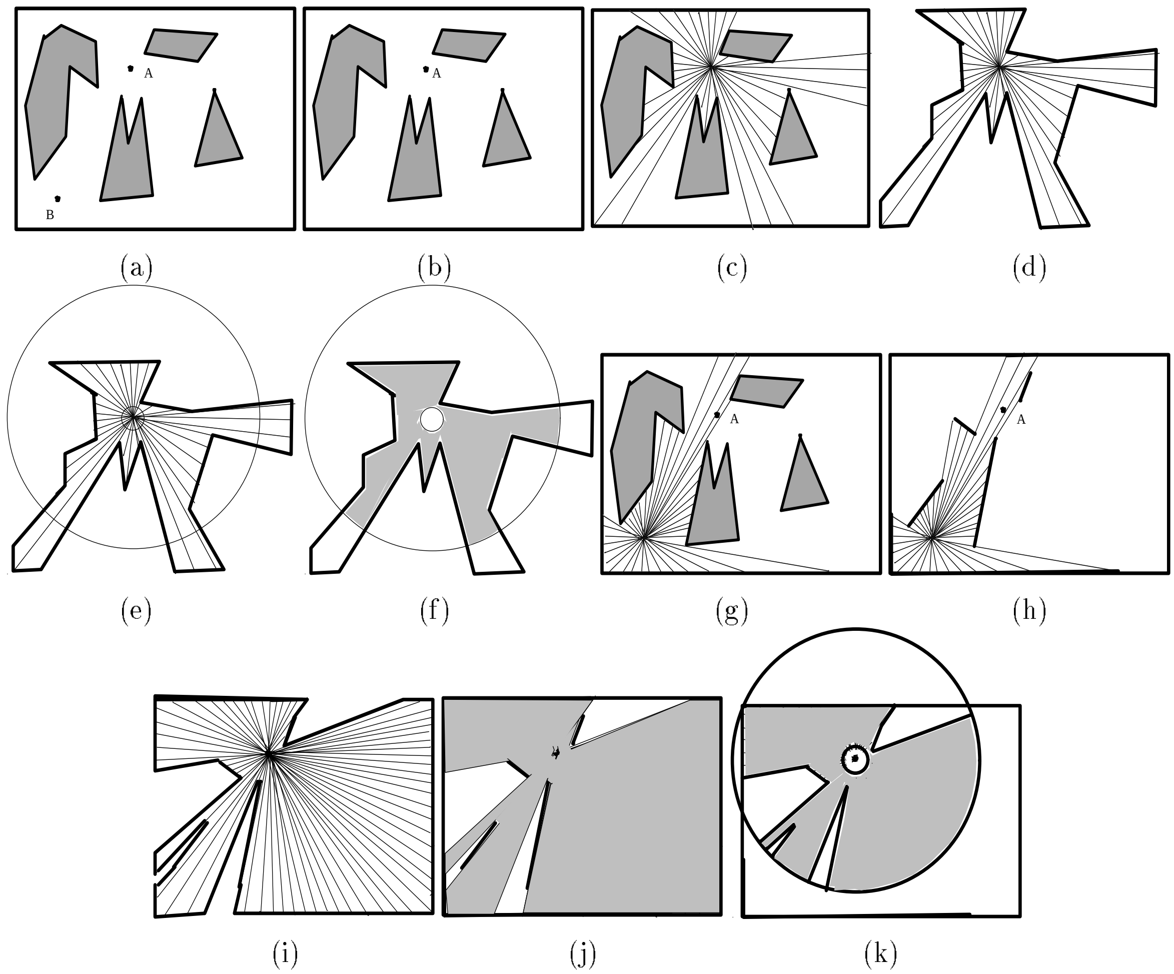


Figure 7.2: $2D$ illustration of the covering probability. (a) The environment. **B** is the camera's current position, **A** is a candidate position. The shaded areas are occupied by solid obstacles. (b),(c),(d) The candidate position and the constructed sensed sphere assuming the robot knows the geometric configuration of the environment. (e) The covering range of the camera. (f) The shaded area belongs to the covering range of the camera and the sensed sphere. The sum of the probabilities within this area is the covering probability of the candidate position **A**. (g),(h) The sensed sphere at position **B**. (i),(j) The expected sensed sphere at **A**. (k) The sum of the probabilities of the shaded area is the covering probability when the robot does not know the geometric configuration of the environment. Notice that the shaded areas in (f) and (k) are different. The correct situation is shown in (f), and (k) is the situation obtained by making use of the available knowledge accumulated so far.

associated solid angle with respect to the expected sensed sphere SS_e, if the distance of this cube to the center of the new camera position is less then the radius of the solid angle, we replace the radius of the solid angle by this distance. Otherwise, the radius of the solid angle is not updated. In this way, we construct the expected sensed sphere SS_e for the current candidate position. Each candidate position will be associated with an expected sensed sphere constructed by the above method. For each expected sensed sphere SS_e and the corresponding camera covering range C_{ov}, we can calculate the probability of presence of the target by summing up the probabilities of all the cubes that are within this expected sensed sphere and the camera covering range $\sum_{c_i \in SS_e \cap C_{ov}} \mathbf{p}(c_i)$. Thus, for each candidate position, we obtain its corresponding covering probability. We then select the position with the maximum covering probability as our next position.

The SS_e calculated by the above method is not a true representation of the environment. It is an approximation of the environment based on the knowledge the search agent accumulated so far. Thus, wrong choices, from an efficiency standpoint, are possible.

7.2.3 Drive the Robot to the New Position

After we choose the next position, we drive the robot to this position and begin our new search. Since the property of the cube and the sensed sphere may not correctly represent the real situation, the robot may stop in the middle of the journey. If it cannot reach the next position, then it will begin the "where to look next" computation at the place it has stopped.

7.3 The Combination of the Registration Space

We know from Chapter 5 that the sensed sphere can be used to register the search space (assigning the values of solidity to each cube). After we drive to a new position, we will use the laser to construct the sensed sphere again. After the construction, we can register the space according to this new sensed sphere. Then we will combine the new and old registration and update the registration file as follows:

Old	*New*	*Combined*
solid	solid	solid
solid	unknown	solid
unknown	solid	solid
unknown	unknown	unknown

Since we tessellate the search space based on Ω at the beginning of the search process, the cubes resulting from the tessellation and their orientation and position will not change with respect to the world coordinate system when the robot moves. Thus, there is no spatial registration problem for cubes. We only need to assign the solidity property to each registered cube when the robot moves from one position to another.

7.4 When to Stop Searching

The stopping criterion is determined by estimating the benefit that can be obtained if we continue searching. If the total probability that the target is inside the search region $(1 - p(c_o))$ is less then a certain threshold, or the covering probability of the next robot position (section 7.2) is less then a certain threshold, then we will stop the search process.

Chapter 8

Discussion

8.1 When the Robot Knows the Environment Configuration

We have presented our sensor planning strategy by assuming that we do not know the geometric configuration of the search environment. Thus we need to use a laser range finder to construct the sensed sphere and to infer the solidity property of the cubes which make up the environment. In some industry environments, the main obstacles that can cause occlusion are usually known and fixed. When we assume that we have perfect knowledge about the geometric structure of the environment, the sensed sphere can be constructed a prior instead of using a laser range finder. Suppose we represent all the occluding surfaces by a series of triangles. For a line emitting from the center of the camera along a given direction, we can calculate the intersection of this emitting line with each triangle. If the intersection point is within the given triangle, we then calculate the distance from the camera center to the intersection point. We call this distance the radius for this triangle. During the process of constructing the sensed sphere, the radius for a given direction (p, t) is the smallest radius among all the radii of the available triangles for the given direction. Thus, the sensed sphere can be calculated. By calculating rather then measuring the sensed sphere, we can save a lot of time during the search process. In warehouse and factory environments, the majority of the automatic guided vehicles (AGV) have been guided by clues associated with the floor, e.g. buried wires, white lines and metal tapes. Although differing in principle, all these methods provide the AGV with a line to follow, and the localization problem can be solved by recognizing special markers placed on or in the floor [64]. In this case, we can predefine a basic set of candidate positions such that the robot can view the whole search space without occlusion from these positions. Thus, the next position for the robot can be selected from this basic set of candidate positions.

8.2 Comparison with Wixson's Work

The existing works that are directly concerned with the task of searching for a $3D$ object within a $3D$ environment are: Connell et al.'s work [20] on searching for and collecting soda cans by an artificial creature, Garvey's proposal [29] for indirect search and Wixson et. al's work [88] on efficiency analysis of direct and indirect search and on viewpoint selection for object search.

Connell [20] describes a real, fully-functional mobile robot which operates in an unstructured environment to collect empty soda cans. The robot has infrared proximity sensors on the body for obstacle avoidance, a flux-gate compass for navigation and a high resolution laser range finder for locating and recognizing objects. The operation cycle is as follows. It starts by wandering around its environment. It locates soda cans with a laser light-striper, grabs them with a 2 degree of freedom arm, and brings them back using infrared proximity sensors and the compass. The cycle repeats as the robot ventures forth once again on its quest for cans. Although the task of the robot is to collect soda cans, the purpose of Connell's research is to test Brooks's subsumption architecture [60] [17] rather than to study object search. Connell's system is unusual in that it is controlled by a collection of independent behaviors rather than a centralized program, and because it uses a minimal amount of state. As a result, the robot's goal is not to maximize its efficiency in searching and collecting soda cans, as long as it can perform the task. It may face the possibility of entering infinite loops instead of achieving its true goal. The navigation is also simple, as the robot is constrained to only move along the walls, and it is not required that the robot be able to examine every part of the region.

Although the strategy of indirect search is proposed by Garvey [29], it is Wixson and Ballard who first study the theory of efficiency of direct and indirect search. From a certain point of view, Wixson is the first to seriously study the task of $3D$ object search and to propose the formula for search efficiency. Although our approach is quite different from Wixson's approach, many of the results are consistent. In this chapter, we compare Wixson's approach and our approach in detail. The comparison is divided into two parts. We first compare Wixson's work on efficiency concern of direct and indirect search with our "where to look next" strategy. We then compare Wixson's work on viewpoint selection with our work on "where to move next".

8.2.1 Efficiency of Object Search

Wixson's Approach

Wixson analyzes the conditions under which indirect search is advantageous over direct search. To do this, he determines the factors that affect the efficiency of searches conducted using these two approaches, and develops models of their effects. He defines the efficiency as $\frac{\gamma}{T}$, where T is the expected cost of the search, and γ is the probability that the search actually finds the desired object. He then defines a direct search of a spatial region r for an object of type j as a process of examining the region with an object recognizer Rec_j. Direct searches assume that the target may lie, with equal probability, at any location in r, and do not attempt to restrict the search only to certain subregions of r. Direct searches are simply concerned with the viewpoint selection problem of moving the camera to image previously-unviewed portions of r.

By lumping many of the factors that affect the search efficiency together into more observable parameters that reflect properties of the object recognizer and the search region as a whole, Wixson is able to formulate the search efficiency of object search. These parameters are assumed to be obtained from a "spanning" view sequence ν. The parameters obtained from this sequence are: the number of views V required to totally examine the search region (*i.e.*, the number of views in ν), the probability density function $P(R)$ of the number R of positive responses produced by Rec_j in the course of ν, the probability β that a given positive positive response is a false positive, and the cost c per view. Using these parameters, the success probability γ_{dir} of a direct search can be written as

$$\gamma_{dir}(P(R), \beta) = (1 - P(R = 0)) \times (1 - \beta) \tag{8.1}$$

The expected cost of a search is the expected effort expended before obtaining the first positive response.

The expected cost can thus be expressed as:

$$T_{dir}(P(R), V, c) = \left[P(R = 0)V + \sum_{r=1}^{\infty} P(R = r)\tau(1, r, V) \right] \times c \tag{8.2}$$

where $\tau(k, r, V)$ represents the expected number of images that must be examined before finding k positive responses, given that r positive responses occur in V images, and more than one positive response can occur per image.

$$\tau(k, r, V) = \sum_{v=1}^{V} v \times \sum_{a=0}^{k-1} \binom{r}{a}^{a} \left[\sum_{f=k-a}^{r-a} \binom{r-a}{f} \left(\frac{1}{V}\right)^{f} \left(\frac{V-v}{V}\right)^{(r-f-a)} \right]$$

Based on the above formulas, Wixson points out that in order to maximize the success probability and minimize the cost, one must (a) try to minimize β, (b) obtain a distribution $P(R)$ that maximizes the expected number $E(R)$ of responses that can be found in the search region, and (c) minimize the number of images V required to search the area. By assuming that one configuration of imaging parameters and internal thresholds has been selected for each recognizer, he concludes that variations in the properties describing a search $(\beta, P(R), V)$ can only be due to variations of properties of the region being searched, such as its size and the number of instances of the target that it contains. The central idea underlying indirect search is to use spatial relationships to select regions where these properties have especially favorable values, i.e., where the resulting β, $P(R)$, and V values improve search efficiency.

Basically, the indirect searches reduce the computationally expensive problem of searching for a target in a large volume at high resolution to a two-stage problem. In the first stage, one locates an intermediate object that typically participates in some spatial relationship with the target, and which can be found with lower resolution, i.e. with a wider field of view. In the second stage, the high-resolution search for the target is performed in the much smaller volume specified by the spatial relationship. Indirect search thus sacrifices the time required to find the intermediate object in order to reap the benefits of examining a smaller volume. As with the formulation of the direct search process, Wixson further formalizes the efficiency of indirect search. The efficiency model of direct and indirect search is then used to study and compare the efficiencies as various aspects of the search are systematically varied. The empirical experiments predict that, over typical values of these properties in indoor environments, indirect search is up to 8 times more efficient than direct search.

Wixson's work on efficiency concern is very important because his research reveals many important properties of the object search process.

Comparison

It is interesting to compare our work with Wixson's work. Although both efforts aim at increasing the search efficiency, the approaches take completely different paths.

- The main difference between our work and that of Wixson is that Wixson's formulation is used to compare the search efficiency of different strategies, while our formulation is used to perform the sensor planning task. In Wixson's formulation, the parameters considered are β, $P(R)$, and V. The state parameters of the camera are not explicitly represented. So, although his formulation is very effective in comparing different search strategies, it is not suitable for determining the camera's state parameters during the search process. In our formulation, the state parameters of the camera are explicitly represented. Because of this, these parameters can be determined during the search process. Thus, our formulation is useful in the sensor planning task.

- Wixson's formulation has a strong two dimensional flavor. He assumes that the camera's configuration (field of view) is fixed. Thus, the search region can be spanned by a view sequence. But for a strictly $3D$ situation, we may need many angle sizes such that their effective ranges can cover the whole distance of the search region. The resolution is also influenced by the target distance to the camera center, so that in $3D$ the resolution is also a relative factor. Our formulation of the problem is strictly $3D$.

- Our formulation can also explain why indirect search may increase search efficiency. In our sensor planning strategy, the sensor's state is guided by the probability distribution of the target. So, the initial distribution has a great influence on the performance of the sensor planning process. If a related intermediate object exists and the search of the intermediate object is not time consuming, then we can first direct the sensor to locate the intermediate object. After finding the intermediate object, we can update the probability distribution according to the spatial relationship of the intermediate object and the target. We then search for the target according to the new distribution. Thus, if we include the indirect aspects in the a priori probabilities (you are looking for a pencil, so the space immediately above a desk has a high probability for search), then indirect search fits in our strategy easily. It can be incorporated into our strategy as a two phase operation. The first phase is to determine the initial distribution by locating the intermediate object. The second phase is to search for the target.

- The indirect search strategy does not plan the sensor during the process of searching for the intermediate object and the process of searching for the target. Our approach plans the sensor state parameters during each phase.

8.2.2 View Point Selection

Wixson's Work

Searching for an object in a cluttered area is often complicated by the fact that the clutter hides portions of the area from view. Multiple viewpoints are often required before the object is brought into view. Wixson examines methods for automatically selecting these viewpoints in $2D$ simulations. The motivation of his research is to examine the necessity of map construction for selecting viewpoints in search tasks. He does so by comparing the performance of algorithms that rely on different representations of varying sophistication. The evaluation of a viewpoint selection strategy involves averaging the cost it incurs over a sample of possible geometric configurations of obstacles. To predict performance when the object is present, the cost incurred is plotted as a function of the fraction of the search region's free space that has been examined. To evaluate performance when the object is not present, the cost incurred is obtained by measuring the total cost incurred by each strategy before it gives up.

From the aspect of scene representation, there are two extremes. At the one extreme of the representation spectrum is the model-based methods. These kind of methods allow potential viewpoints to be evaluated on the basis of how much previous-unviewed area they will bring into view. At the opposite extreme are methods that make no attempt to select the next viewpoint based on an evaluation of potential viewpoints' utility. These methods either select viewpoints at random or follow a fixed sequence of pre-enumerated viewpoints, and construct no representation of what has been seen. It is obvious that there are many strategies that require representations that are between these two extremes.

Generally speaking, analytical evaluation of viewpoint-selection algorithms is difficult because their performance depends on the likely arrangements of obstacles in the search region. Such arrangements are difficult to model analytically. Because of this, Wixson evaluates various algorithms empirically in simulations using a large number of test cases. He concludes that the rate at which model-free strategies bring unseen area into view is very comparable to that of model-based strategies. Thus, model-free methods are very likely to be cheaper than the model based methods. He also concludes that when the desired object is not present, the lack of a good method of knowing when to give up in model-free strategies can cause them to incur significantly more cost than model based methods.

Comparison

In our sensor planning strategy, the next robot position is obtained by using the search region models obtained so far. We argue that it is important to plan the next robot position. There is a very important question to ask: does our argument contradict Wixson's empirical simulation results?

The answer is: no. Our argument consistent with his results, but in general we address different issues. The most important difference is that we consider the robot's knowledge about the target distribution a very important factor, and this knowledge is used to guide the next robot position. In Wixson's simulation experiments, the knowledge about the target distribution is not used. Thus, his goal in selecting the next viewing point is to view the unexamined region as much as possible, while our purpose is to get to a position such that the camera can examine the parts of the region with the maximum probability of presence of the target. The other difference is that Wixson's simulation is in $2D$ instead of $3D$. There may be situations in $3D$ where his results are not applicable.

Part IV

Testing the Model

Chapter 9

Simulations

To test the ideas presented in this document, we have done some simulation experiments. The searcher model is a mobile platform equipped with laser, camera and sonar as described in Section 2.2.2). The algorithm used in the simulation is as follows:

1. Determine the necessary angle sizes according to the diameter of the search region (see Section 6.2).

2. Generate the layered sensed sphere corresponding to those angle sizes obtained in (1) (see Section 6.3.1).

3. Find the best direction for each angle size (Section 6.3.2).

4. Use Formula (6.1) to compare the best directions of different angle sizes, and select several top actions among them.

5. Apply these selected top actions and update the environment (see Section 2.4.3).

6. Goto 2 when the probability within current sensed sphere is **not** small enough, otherwise, Goto 7 (see Section 6.4).

7. When the probability that the target is within the search region is small enough (see Section 7.4), stop search.

8. Determine the next candidate positions, and calculate the probability of those expected sensed spheres (see Section 7.2).

9. Select the best position (Section 7.2) and drive the robot to that position.

10. Combine the search region of the two positions (Section 7.3).

11. Go to 2 and begin the search at the new position.

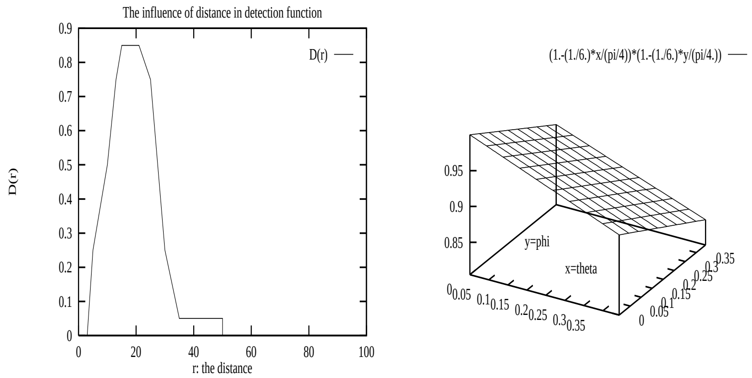
In our simulation experiments, the biggest angle for the sensor is $\pi/4$, and we assume only one recognition algorithm is available. The detection function for $\pi/4$ is:

$$b(c, f_{\pi/4}) = b(r, \varphi, \theta) = D(r)(1 - \frac{1}{6}\frac{\varphi}{\frac{\pi}{4}})(1 - \frac{1}{6}\frac{\theta}{\frac{\pi}{4}}) \tag{9.1}$$

where r is the distance of the cube to the camera center, φ is the offset of the cube with respect to the pan direction of the camera axis, and θ is the offset of the cube with respect to the tilt direction of the camera axis. The value of $D(r)$ is:

$$D(r) = \begin{cases} 0.5 + \frac{r-11}{11-5}(0.5 - 0.25) & \text{if } r \in (0, 2.5] \\ 0.25 + \frac{0.25}{1.5}(r - 5) & \text{if } r \in (2.5, 5] \\ 0.5 + \frac{r-11}{11-5}(0.5 - 0.25) & \text{if } r \in (5, 11] \\ 0.75 + \frac{r-13.5}{13.5-11}(0.75 - 0.5) & \text{if } r \in (11, 13.5] \\ 0.85 + \frac{r-15}{15-13.5}(0.85 - 0.75) & \text{if } r \in (13.5, 15] \\ 0.85 & \text{if } r \in (15, 21] \\ 0.75 + \frac{r-25}{25-21}(0.75 - 0.85) & \text{if } r \in (21, 25] \\ 0.25 + \frac{r-29}{29-25}(0.25 - 0.75) & \text{if } r \in (25, 29] \\ 0.05 + \frac{r-35}{35-29}(0.05 - 0.25) & \text{if } r \in (29, 35] \\ 0.05 & \text{if } r \in (35, 50] \\ 0 & \text{if } r \in (50, +\infty) \end{cases}$$

The influence of r, φ, θ to the detection function is illustrated in Figure 9.1. The effective region for $\pi/4$ is $[11, 27]$.



(a) (b)

Figure 9.1: (a) the influence of the distance r. (b) the influence of the angle φ, θ

In all the simulation experiments, the direction of the z axis corresponds to $tilt = 0$; the direction of x axis corresponds to $pan = 0$, $tilt = \frac{\pi}{2}$; and the direction of y axis corresponds to $tilt = \frac{\pi}{2}$, $pan = \frac{\pi}{2}$.

9.1 The General Case

In this simulation, the search region is a room of size $50 \times 50 \times 25$ units (Figure 9.2). The horizontal vertices of the room are $(0,0)$, $(50,0)$, $(50,50)$, $(0,50)$ respectively. There is one big obstacle, with size $5 \times 37 \times 19$, in the space that can cause occlusion. The horizontal vertices of the obstacle are $(36,10)$, $(41,10)$, $(41,47)$, $(36,47)$ respectively. The height of the robot is 15, and it is situated at $(45,10)$ before the search. See Figure 9.2 for an illustration. Since the diameter of the region is 75, only three angle sizes $0.785398(45^o)$,$0.386876(22.166346^0)$, $0.164481(9.424083^o)$ are needed in the search process. Their effective ranges are $[11,27]$, $[27,66]$, $[66,162]$, respectively. It is assumed that we do not know the probability distribution at the beginning, and we do not know whether the object is within the room or outside the room. We assume that the outside probability is 0.5, and the distribution within the room is uniform (the probability of each cell is 0.000008).

First, the robot constructs the sensed sphere (Figure 9.2).

Then it registers the space (Figure 9.3). There are 832 solid cubes.

For each angle size, the robot first selects 5 actions based on the layered sensed sphere. It then uses the detection function to compare these 15 actions and selects the top 5. Lastly, it executes those actions that belong to the top 5 that have not been executed before. After executing each action, the search region will be updated. This process is continued until the ratio of the probability of current sensed sphere and the probability of the search space is less then 0.16. The actions actually executed are shown in the following table and the top 7 actions are illustrated in Figure 9.4. It is clear that these actions are correctly selected.

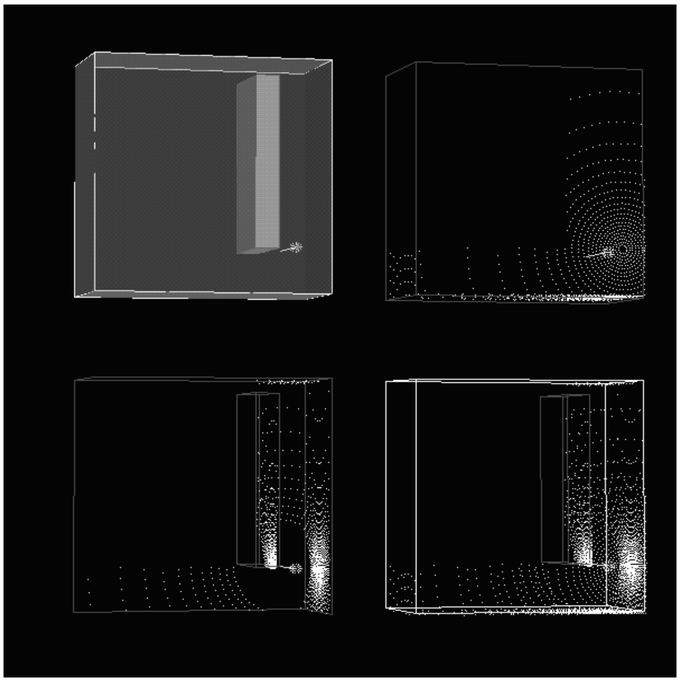
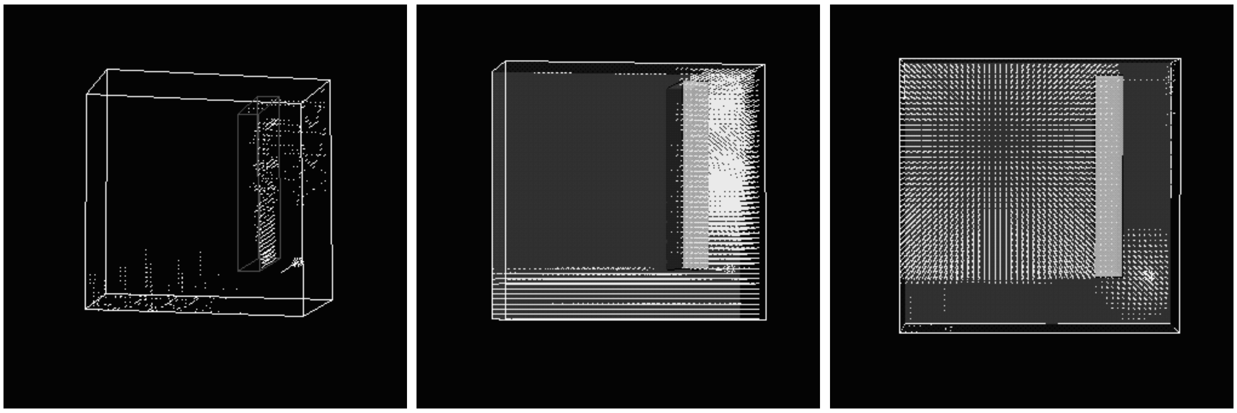


Figure 9.2: Top left: the environment for Simulation 1. There is a big obstacle within a room. The robot is at $(45, 10)$. Top right: a part of the sensed sphere constructed at $(45, 10)$. Bottom Left: another part of the sensed sphere constructed at $(45, 10)$. Bottom right: the complete sensed sphere constructed at $(45, 10)$. Among the figures above, only the bottom right shows the complete sensed sphere together with the environment. The upper right figure and the lower left figure each shows just a part of the sensed sphere. There is no overlap between them. When these two figures are combined, we get the lower right figure.



(a) (b) (c)

Figure 9.3: (a) The solid cube. (b) Cubes without occlusion with respect to camera position. (c) Cubes with occlusion with respect to camera position.

Angle Size	Pan	Tilt
0.785398	1.708314	1.963495
0.386876	3.140938	1.740941
0.785398	1.708314	1.178097
0.386876	3.170120	1.354065
0.785398	3.416629	1.963495
0.386876	1.585060	1.354065
0.386876	1.570469	1.740941
0.785398	3.416629	1.178097
0.386876	3.533556	1.740941
0.386876	3.299366	0.967189
0.785398	4.270786	1.963495

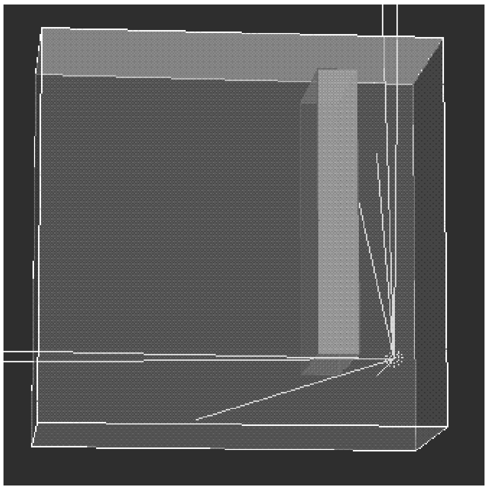


Figure 9.4: The top 7 actions selected. The long line corresponds to the viewing axis of angle size 22.166346^o. The short line corresponds to the viewing axis of angle size 45^o

Figure 9.5 shows the outside probability p_{out}, the probability within the current sensed sphere p_{SS}, and the fraction of the current sensed sphere probability with respect to the search region probability $\frac{p_{SS}}{1-p_{out}}$ after each selection and execution phase. After each phase of actions, we can see that the outside probability is increasing, and the current sensed sphere probability and its percentage within the search space is decreasing.

Figure 9.6 shows the probability from the layered sensed sphere and the real calculated probability of detection by formula $\sum_{c_i \in \Psi} \mathbf{p}(c_i, t_{\mathbf{f}})\mathbf{b}(c_i, \mathbf{f})$. From the figure we can see that the order of detecting probabilities calculated from the layered sensed sphere is quite similar to the order of probabilities calculated from the detection function. This demonstrates that the layered sensed sphere is very useful in selecting the candidate actions. We can also see that the

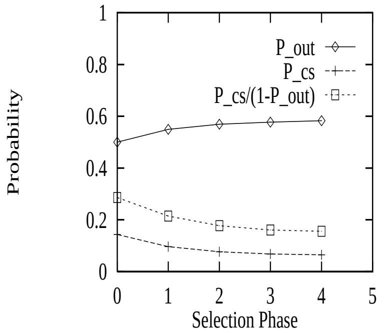
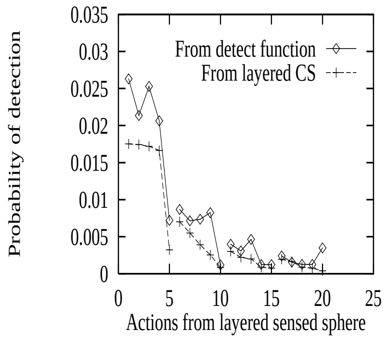
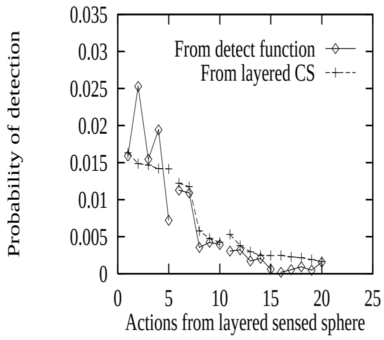


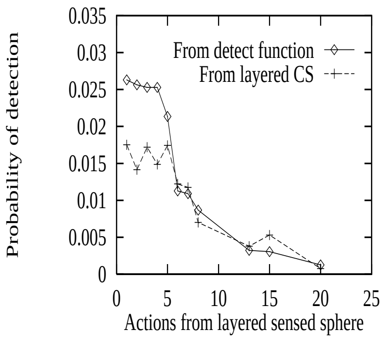
Figure 9.5: The probability change after each phase



(a)



(b)



(c)

Figure 9.6: Comparison of the probability from layered sensed sphere and those from detection function: (a) for angle 0.785398; (b) for angle 0.386876; and (c) for real executed actions.

more actions executed, the less benefit we obtain.

To select the next position, the robot will calculate the expected sensed sphere and its probability of presence for each possible position. To save time, the robot only considers the positions $x = 1, 6, 11, \ldots, 46$ while $y = 1, 2, \ldots, 50$. Figure 9.7 shows this probability for each candidate position. Among them, the position $(36, 1)$ has the maximum probability 0.371333. Therefore, the robot will move to $(36, 1)$ and begin the new search.

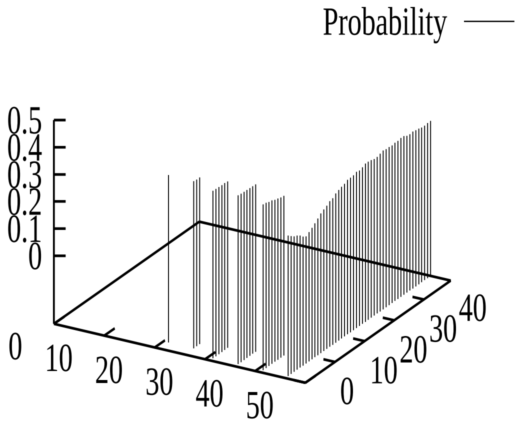


Figure 9.7: The probability of presence for the expected sensed sphere of each candidate position.

Figure 9.8 shows the expected sensed sphere before the constraint using the solid cube.

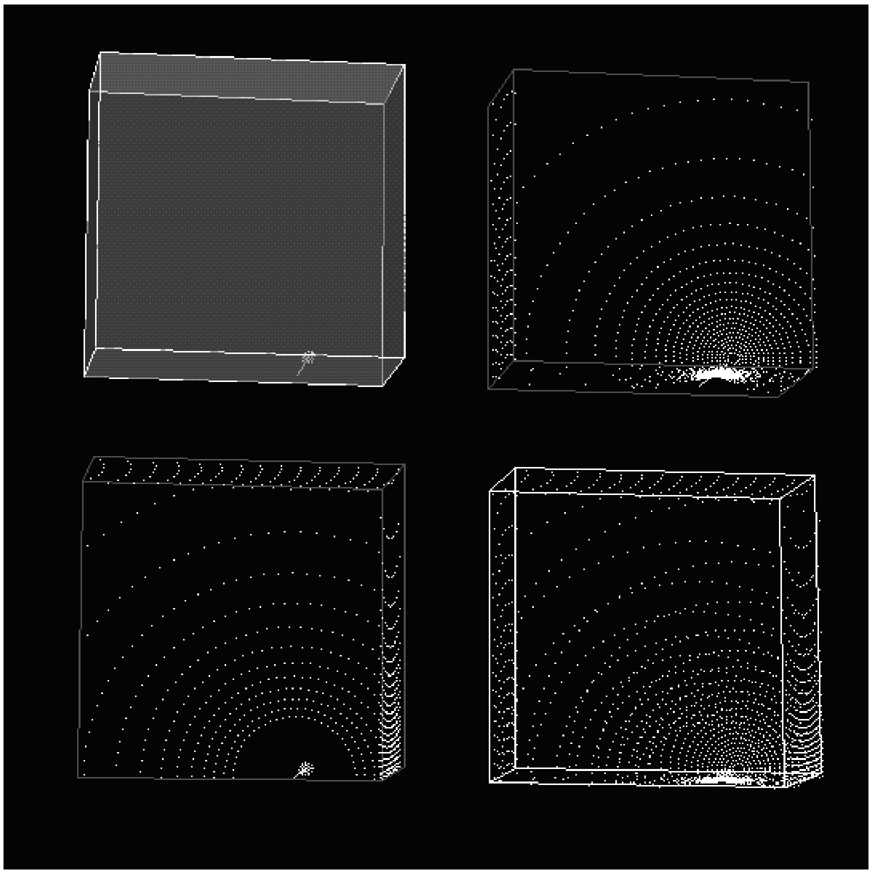


Figure 9.8: Top left: the expected environment for search space. Top right: a part of the expected sensed sphere at $(36, 1)$ before constraining using the solid cube. Bottom Left: another part of expected sensed sphere at $(36, 1)$ before constraining using the solid cube. Bottom right: the complete expected sensed sphere at $(36, 1)$ before constraining using the solid cube. Among the figures above, only the bottom right shows the complete sensed sphere together with the environment. The upper right figure and the lower left figure each shows just a part of the sensed sphere. There is no overlap between them. When these two figures are combined, we get the lower right figure.

Figure 9.9 shows the expected sensed sphere after the constraint using the solid cube.

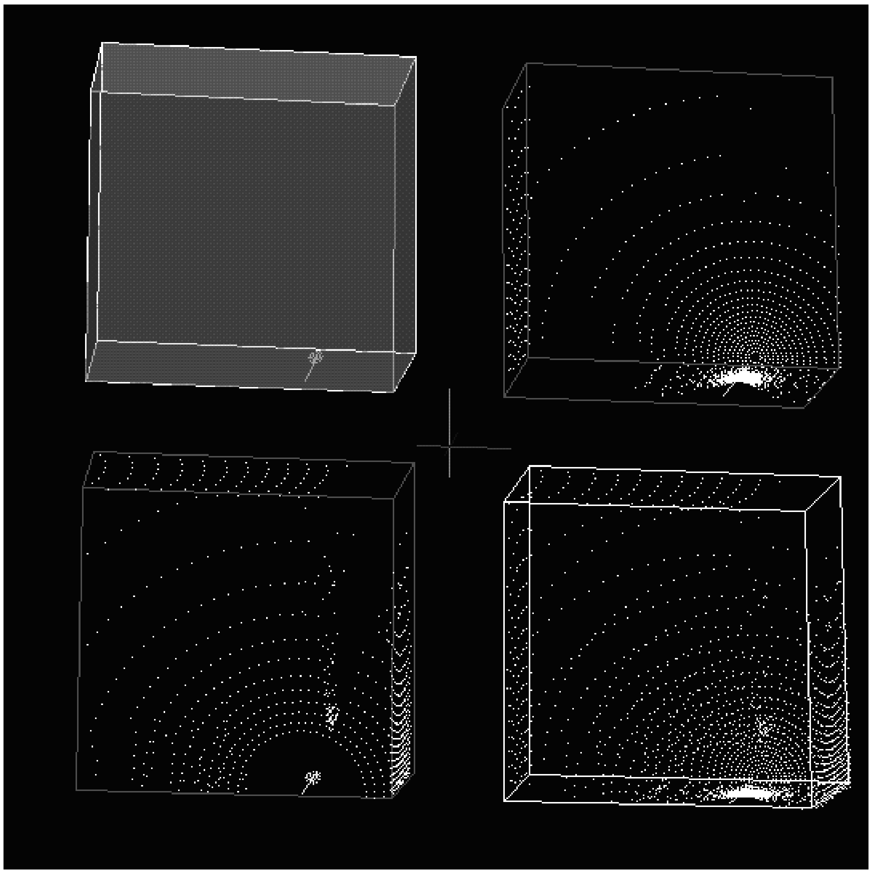


Figure 9.9: Top left: the expected environment for search space Top right: a part of expected sensed sphere at $(36, 1)$ after constraining using the solid cube. Bottom Left: another part of expected sensed sphere at $(36, 1)$ after constraining using the solid cube. Bottom right: the complete expected sensed sphere at $(36, 1)$ after constraining using the solid cube. Among the figures above, only bottom right shows the complete sensed sphere together with the environment. The upper right figure and the lower left figure each shows just a part of the sensed sphere. There is no overlap between them. When these two figures are combined, we get the lower right figure.

Figure 9.10 shows the environment and the sensed sphere when the robot drives to $(36, 1)$.

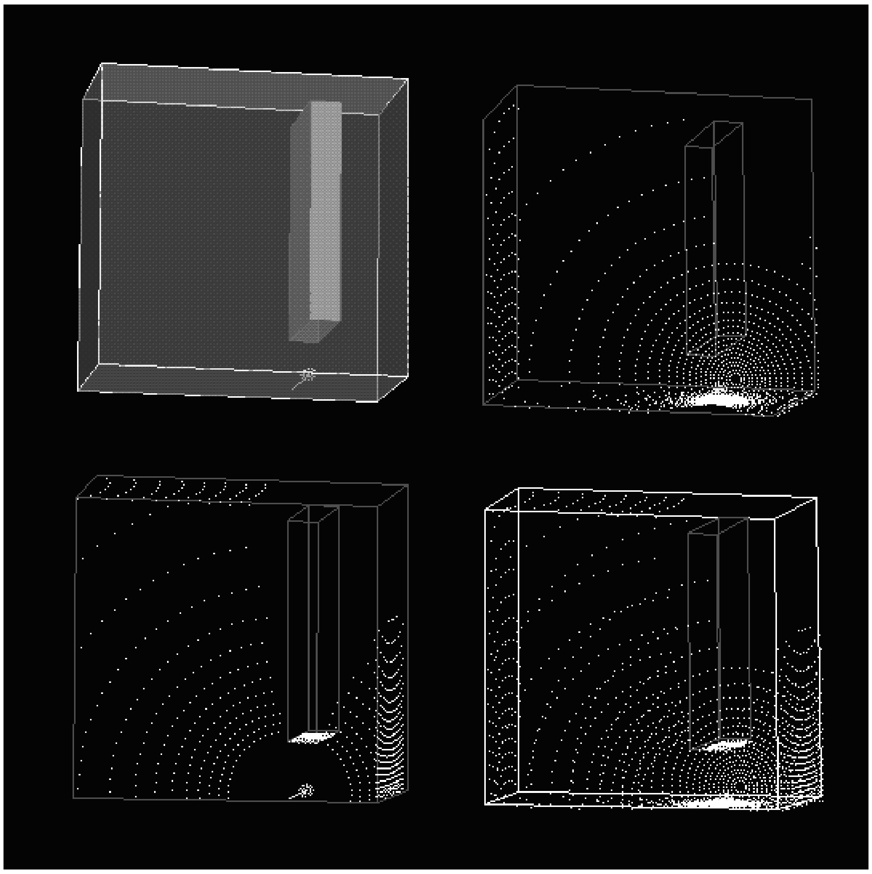
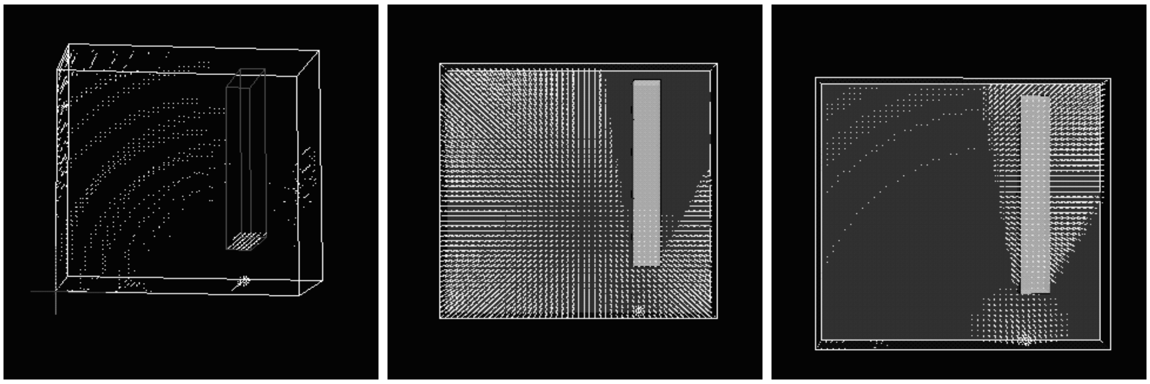


Figure 9.10: Top left: the environment. Top right: a part of sensed sphere at $(36, 1)$. Bottom Left: another part of sensed sphere at $(36, 1)$. Bottom right: the complete sensed sphere at $(36, 1)$. Among the figures above, only bottom right shows the complete sensed sphere together with the environment. The upper right figure and the lower left figure each just shows a part of the sensed sphere. There is no overlap between them. When these two figures are combined, we get the lower right figure.

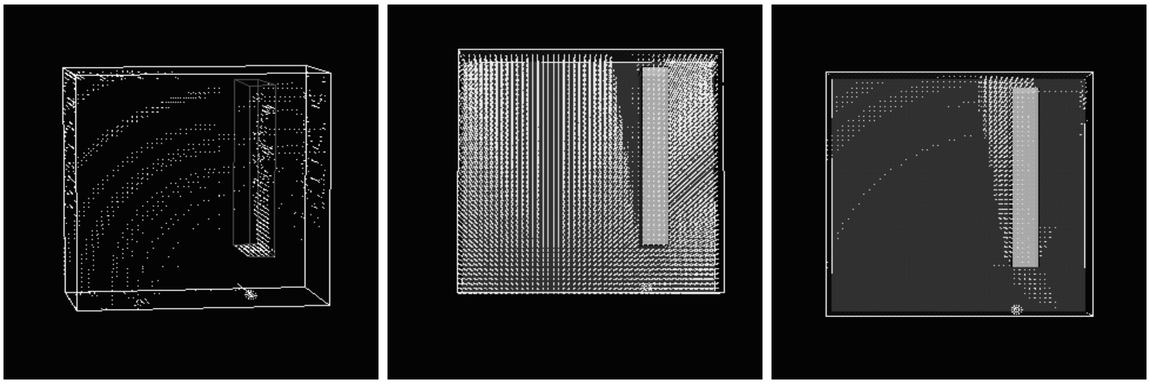
Figure 9.11 shows the solid cubes, the cubes without occlusion and cubes with occlusion when the robot drives to $(36, 1)$.



(a) (b) (c)

Figure 9.11: (a) The solid cube. (b) Cubes not occluded with respect to the camera position. (c) Cubes occluded with respect to the camera position;

Figure 9.12 shows the combined solid cubes and the combined occlusion situation of cubes.



(a) (b) (c)

Figure 9.12: (a) The combined solid cube. (b) Cubes without occlusion with respect to either the first camera position or the second camera position. (c) Cubes with occlusion with respect to both the first camera position and the second camera position.

This example shows that our scheme can effectively search the object when we have no knowledge about the distribution of the target. The sensed sphere representation is very useful in deciding where to look next and where to move next.

9.2 Given the Initial Distribution

This simulation is designed to show that our scheme can effectively incorporate previous knowledge into the search process. The search space is a room $119 \times 100 \times 41$. The horizontal vertices of the room are $(0, 0)$, $(119, 0)$, $(119, 100)$, and $(0, 100)$. There is a long table $100 \times 10 \times 11$

present in the room. The horizontal vertices of the table are $(11, 65)$, $(111, 65)$, $(111, 75)$, and $(11, 75)$. We know at the beginning that the target is probably present in the room and is most likely on the long table. The outside probability is 0.15. The probability that the target object is on the table is 0.79. The height of the robot is 25. The robot is at $(5, 55)$ before the search. Figure 9.13 shows the environment and the sensed sphere constructed at this position.

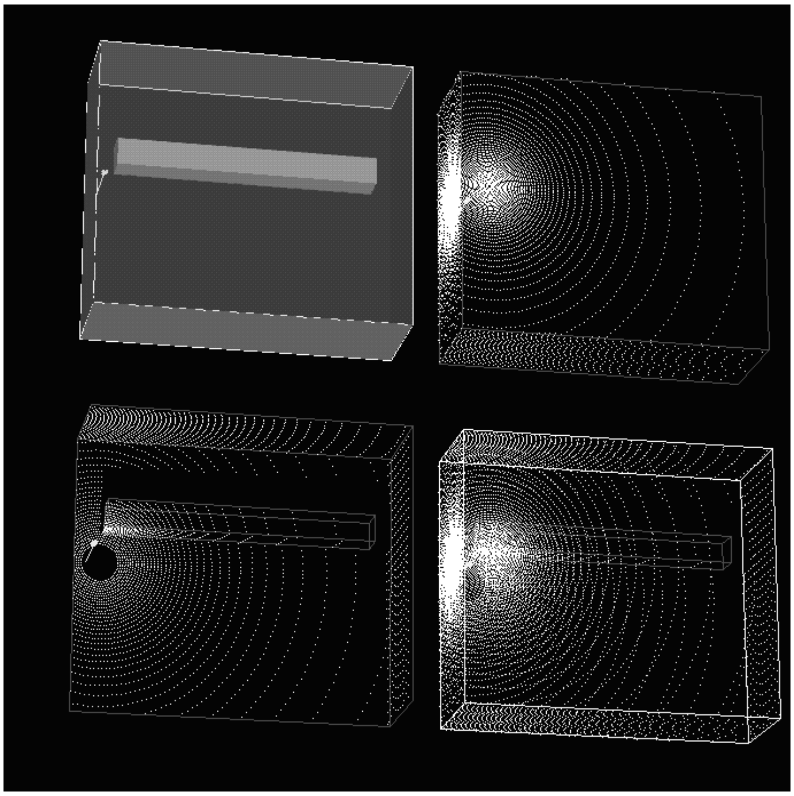
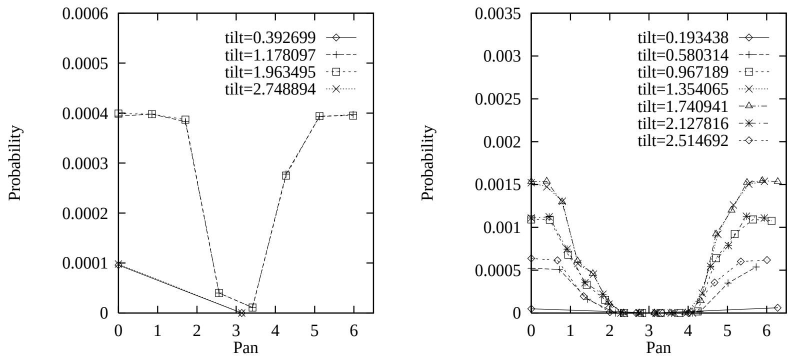


Figure 9.13: Top left: the environment for experiment 2. There is a big table in a room. The robot is at $(11, 55)$. Top right: a part of the sensed sphere constructed at $(11, 55)$. Bottom Left: another part of the sensed sphere constructed at $(11, 55)$. Bottom right: the complete sensed sphere constructed at $(11, 55)$.

Figure 9.14 and Figure 9.15 show the probability for the candidate action determined from each layer of the sensed sphere. From this we can see that the viewing direction is almost correctly directed to the table surface.

The top 7 actions for each angle from the layered sensed sphere are listed below:



(a) (b)

Figure 9.14: The probability calculated from the layered sensed sphere for various candidate $(tilt, pan)$ pairs. (a) when $angle = 0.785398$; (b) when $angle = 0.386876$.

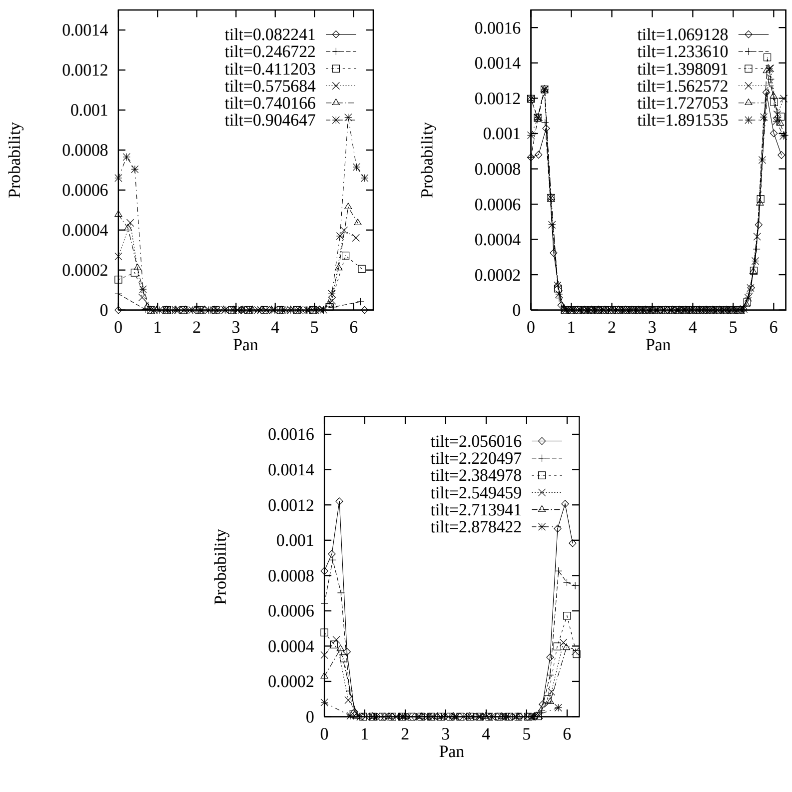


Figure 9.15: The probability calculated from the layered sensed sphere for various candidate $(tilt, pan)$ pairs when $angle = 0.164481$.

0.785398	**0.386876**	**0.164481**
(Tilt,Pan)	**(Tilt,Pan)**	**Tilt,Pan)**
(1.963495,0.000000)	(1.740941,5.889259)	(1.398091,5.843980)
(1.178097,0.854157)	(1.740941,0.392617)	(1.562572,5.921527)
(1.963495,0.854157)	(1.354065,5.943975)	(1.891535,5.893619)
(1.178097,5.979101)	(1.740941,0.000000)	(1.727053,5.828009)
(1.963495,5.979101)	(1.740941,6.281876)	(1.233610,5.926888)
(1.963495,5.124944)	(1.740941,5.496642)	(1.398091,0.333942)
(1.178097,0.000000)	(1.354065,0.000000)	(1.562572,0.328974)

The top seven actions selected by the detection function among them are listed in the following table and illustrated in Figure 9.16:

Angle	**Tilt**	**Pan**	**Probability**
0.386876	1.740941	0.392617	0.019095
0.386876	1.740941	0.000000	0.016860
0.386876	1.740941	6.281876	0.016706
0.386876	1.354065	0.000000	0.010288
0.785398	1.963495	0.854157	0.004473
0.785398	1.963495	0.000000	0.004243
0.386876	1.740941	5.889259	0.001841

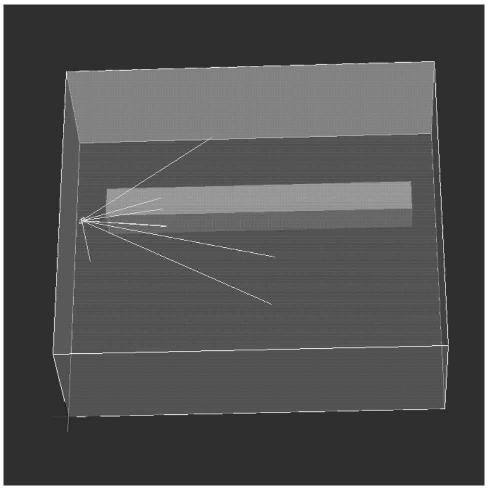


Figure 9.16: The top 7 actions selected for experiment 2. The long line corresponds to the viewing axis of angle size 22.166346^o. The short line corresponds to the viewing axis of angle size 45^o.

We can conclude that the layered sensed sphere and the detection function are very effective in selecting actions when we have knowledge of the probability distribution.

9.3 Comparison with the Brute Force Method

The main purpose of this simulation is to compare our strategy with the brute force strategy from the aspect of the expected probability of detecting the target.

The environment, the robot position and the sensor model for this simulation are the same as those shown in Figure 9.2, except that there is no obstacle in the room. The target distribution satisfies a 3-variate normal distribution $N(\mu, \Sigma)$, where the mean vector $\mu = (25, 25, 15)^T$, and the covariance matrix $\Sigma = diag(\sigma^2, \sigma^2, (\frac{\sigma}{2})^2)$. Thus

$$\mathbf{p}(x, y, z) = \frac{1}{(2\pi)^{\frac{3}{2}} \times 25 \times 25 \times 15} e^{-\frac{1}{2}[\frac{(x-25)^2}{\sigma^2} + \frac{(y-25)^2}{\sigma^2} + \frac{(z-15)^2}{(\frac{\sigma}{2})^2}]} \tag{9.2}$$

As stated in Section 2.4, the purpose of the sensor planning task is to maximize the probability of detecting the target for a given time constraint. For the environment and the sensor model used in this simulation, we need three camera angle sizes to cover the whole search space. Thus, we can get three layers from the layered sensed sphere. For each layer, we need a certain number of directions to cover it. Then for the three layers we can get a fixed number of actions that must be tried. For these fixed actions, from Property 10, the expected probability of detecting the target (Formula 2.6) is also fixed. We call this amount the limit probability of detecting the target for the available actions. In this context, the purpose of sensor planning strategy for object search can be taken as how to select the order of the action sequence such that the expected probability of detecting the target can reach the limit probability as early as possible. Here we compare the effects of our sensor planning strategy with the effects of the Non-Planning strategy. The Non-Planning strategy means that the action application sequence is not influenced by the agent's initial knowledge of the target probability distribution. For example, if $\{\mathbf{f}_1, \ldots, \mathbf{f}_j\}$ is the actions that can be selected. Then, for Planning strategy, the order of application of these actions is different for different initial knowledge. But for Non-Planning strategy, the order of application of these actions is fixed and is not influenced by the initial knowledge. Of course, there are different ways of determine the application order of Non-Planning strategy. Here, the Non-Planning strategy refers to the strategy of first selecting the actions that belong to the first layer, then selecting the actions that belong to the second layer, and last, selecting the actions that belong to the third layer.

Figure 9.17 compares the effects of the two strategies for different initial probability distributions. Notice that the number of actions needed to reach the detection limit for our strategy is much smaller than that for the Non-planning strategy. This illustrates that the planning strategy is more efficient in many situations in the sense of selecting the order of actions to be applied for a given action set.

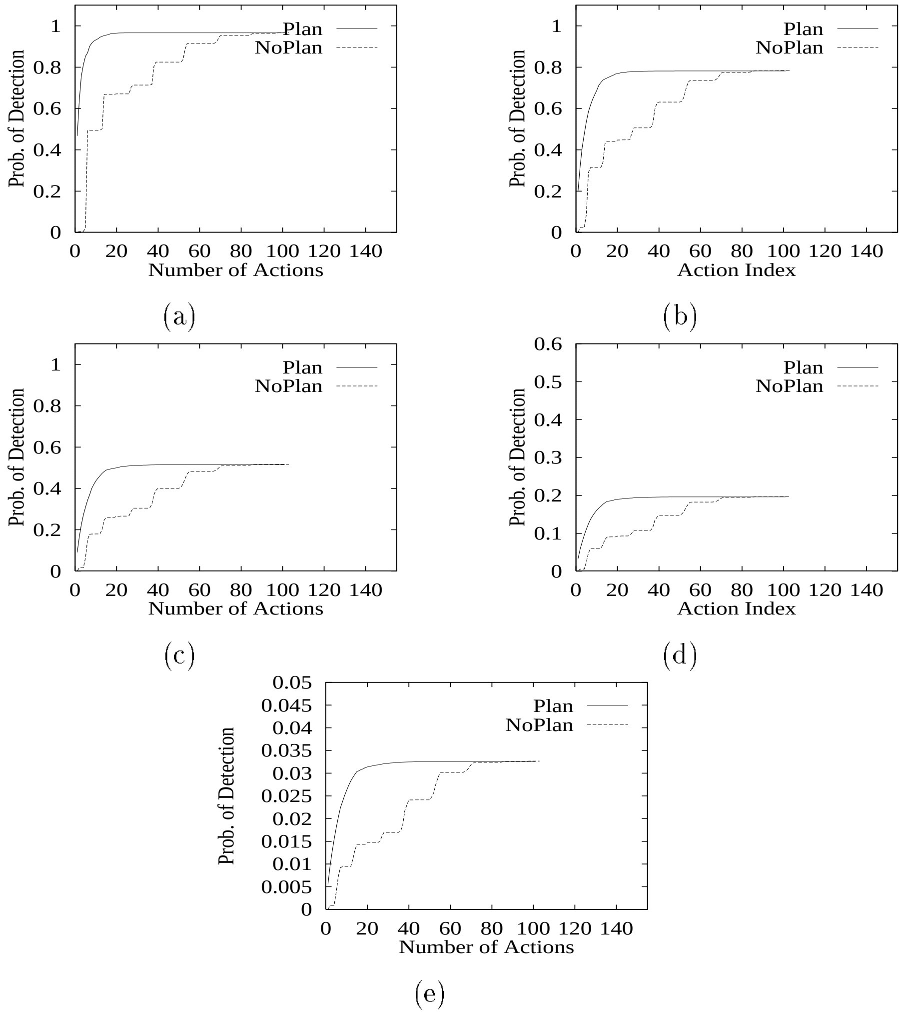


Figure 9.17: The detection probabilities $P[\mathbf{F}]$ of the planning strategy (Plan) and those of the Non-planning strategy (NoPlan). (a)$\sigma = 5$; (b)$\sigma = 10$; (c)$\sigma = 15$; (d)$\sigma = 25$; (e)$\sigma = 50$.

Chapter 10

Real Experiments

10.1 Where to Look Next: the Experiment with the Laser Eye

In addition to the simulation tests, we have also performed experiments using the real robot head described in Section 2.2.2. This chapter discusses results of some of these real experiments.

10.1.1 Search Region, Recognition Algorithm, and Sensed Sphere

The search region is a part of the Vision Lab of the Department of Computer Science at University of Toronto. The Laser Eye is situated within this region and the ARK robot can move within this region. Figure 10.1 shows the configuration of the search region. Figure 10.2 shows the images of the search region.

The world coordinate system is shown in Figure 10.1, where A is the origin. AX and AY are the x axis and y axis, respectively. The direction of Z axis is upward, and is not shown in Figure 10.1. The camera coordinate system is centered at W with axes parallel to the world coordinate system. The *pan* and *tilt* are measured with respect to the camera coordinate system. At present, the range of *pan* that the ARK head can reach is from 0 degrees to 90 degrees, and from 270 degrees to 360 degrees. The range of *tilt* that the ARK head can reach is from 49^o degrees to 105^o degrees. The biggest visual angle size for the head is $41^o \times 39^o$; the smallest visual angle size for the head is $7^o \times 6.7^o$. The image taken with the camera is a color image with size 480×502.

Our task is to search for a white baseball within this region. This task contains three subtasks: sensor planning, hardware manipulation and object recognition. The sensor planning component is influenced by the object recognition component —– different object recognition algorithms correspond to different sensor sequences. Since our intention here is to test the sensor planning strategy rather than to solve the object recognition problem, we use a simple recognition algorithm to locate the baseball within an image. Four threshold criteria are used in our algorithm. They are: intensity I_0, blob size B_{min}, B_{max}, and roundness percentage R_0. The following is the recognition algorithm.

1. Change the color image into black and white. The intensity range of the resulting image

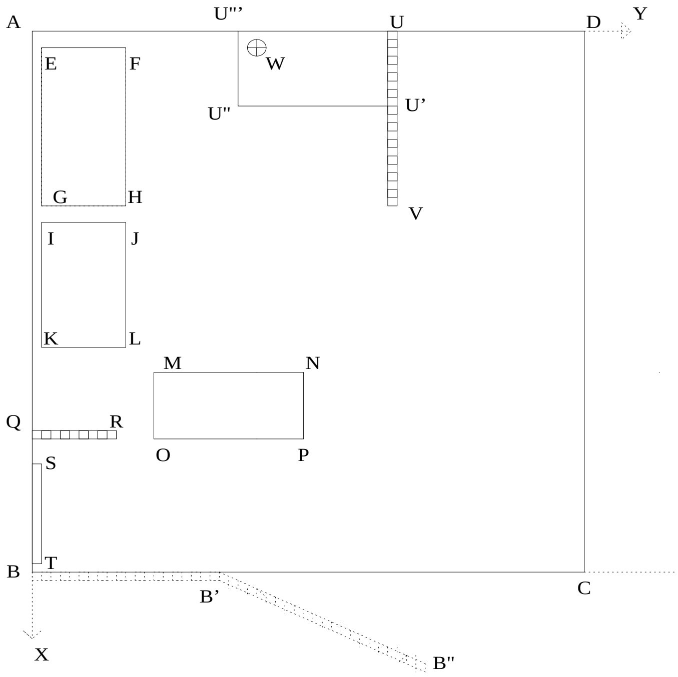


Figure 10.1: The top view of the search environment. A is the origin of the coordinate system. AX and AY are the x axis and y axis, respectively. ABCD is the horizontal boundary of the search region. W is the ARK head. EFGH, IJKL, MNOP, and $UU'U"U"'$ are four tables within this region. The head is on table $UU'U"U"'$. UV and QR are office walls present in the region. BB' and B'B" are office walls outside the search region. ST is a door into the lab. The height of the region is 2.54m. The height of the robot head relative to the table is 0.66m. The height of the table is 0.79m, thus the head is 1.45m above the floor. The horizontal coordinates are as follows (in meters): A(0,0), B(6.21,0), C(6.21,5.62), D(0,5.62), E(0.1,0.2), F(0.1,0.85), G(1.91,0.2), H(1.91,0.85), I(2.16,0.2), J(2.16,0.85), K(3.69,0.2), L(3.69,0.85), M(3.96,1.3), N(3.96,2.83), O(4.72,1.3), P(4.72,2.83), Q(4.72,0), R(4.72,0.91), S(4.91,0), T(5.9,0), U(0.1,3.73), V(1.81,3.73), W(0.1,2.37), B'(1.19,6.21), U'(0.75,3.73), U"(0.75,1.9), U"'(0.1,1.9). This search region is divided into little cubes with side length $0.1m$ when running the algorithm.

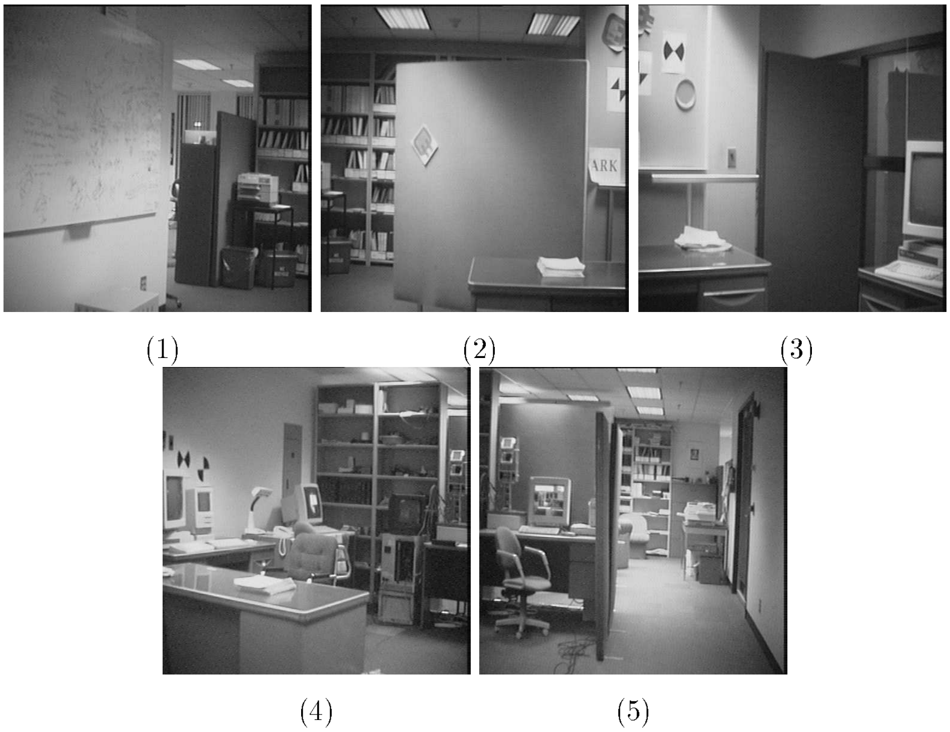


Figure 10.2: The search environment. (1) Part of boundary DC and the gap $B''C$. The image is taken from a place around W. The printer in the image is outside the search space. (2) Part of table $MNOP$; office wall $B'B''$; a little of office wall BB'. The image is taken from a place around W. The shelf in the image is outside the search region. (3) Office wall BB' and QR; a part of table $MNOP$ and table $IJKL$. The image is taken from a place around W. (4) Boundary AQ; a part of table $MNOP$; a part of table $IJKL$; table $EFGH$; a part of ARK head; a part of table $UU'U''U'''$; boundary AU''. The image is taken from a place around C. (5) Table $UU'U''U'''$; the ARK head; the region between office wall UV and boundary CD. The image is taken from a place around C. The shelf and printer in the image are outside the search boundary.

is from 0 to 255.

2. Generate a binary image by applying a threshold to the image from step (1). Pixels with intensity value bigger than I_0 will have the resulting intensity value 255. Pixels with intensity value less than or equal to I_0 will have the resulting intensity value 0.

3. Perform morphological operations on the image resulting from Step 2, first erosion then dilation.

4. Perform region growing on the image resulting from Step 3, and label white blobs.

5. Calculate the center and size (pixel number) of each blob.

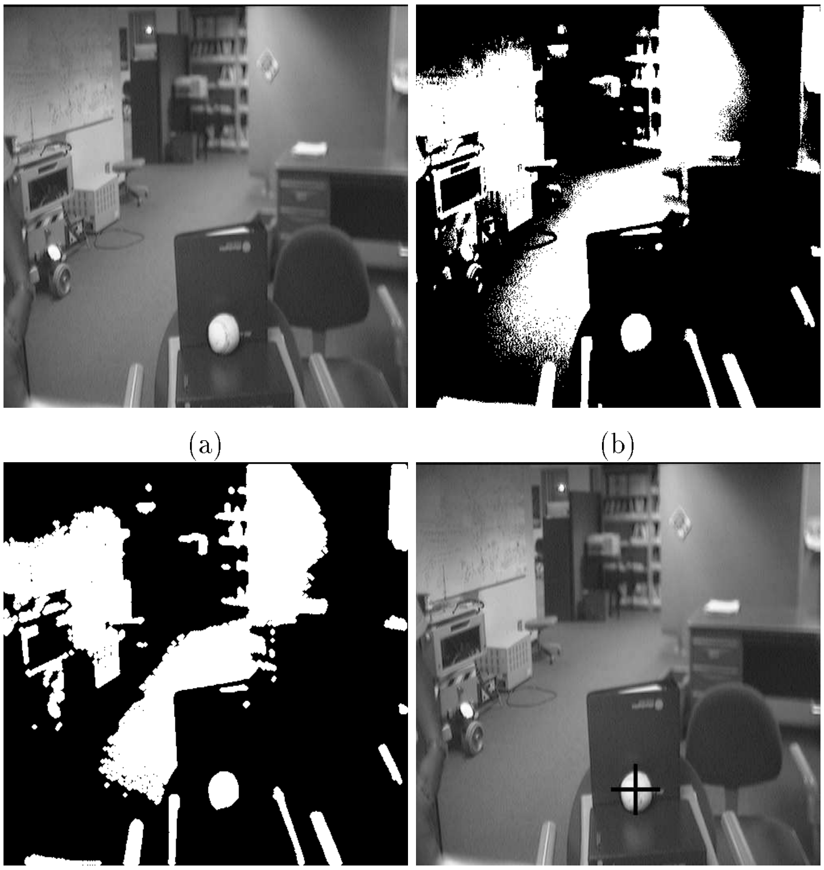
6. According to the center and size from Step 5, calculate the corresponding circle with respect to each blob, and the percentage of the blob that falls into the circle.

7. If a blob size is between B_{min} and B_{max}, and the percentage is bigger than R_0, then this blob is taken as an image of the ball.

In our experiment, the parameters for the threshold are: $I_0 = 119$, $B_{min} = 250$ pixels, $B_{max} = 1375$ pixels, and $R_0 = 0.91$ respectively. We take the biggest angle size as the first angle size, thus $w_0 = 41^o$, $h_0 = 39^o$. The effective ranges for this angle size are obtained by experiments. The nearest distance for which the blob size is smaller then B_{max} will be the value of N_0. The farthest distance for which the blob size is bigger than B_{min} will be taken as the value of F_0. Experimentally, we determine the effective range for $w_0 \times h_0$ to be $N_0 = 1.47m$, $F_0 = 3.01m$. The output of the recognition algorithm is listed below:

	Blob Size	Percentage
N_0	1366 pixels	0.9356
F_0	273 pixels	0.9158

The process of running the recognition algorithm for distance N_0 is shown in Figure 10.3



(a) (b) (c) (d)

Figure 10.3: (a) The image when angle size is $w_0 \times h_0$. The target is 1.47m from the camera. (b) The generated binary image. (c) The image after the morphological operation. (d) The final result: the target is detected.

The process of running the recognition algorithm for distance F_0 is shown in Figure 10.4

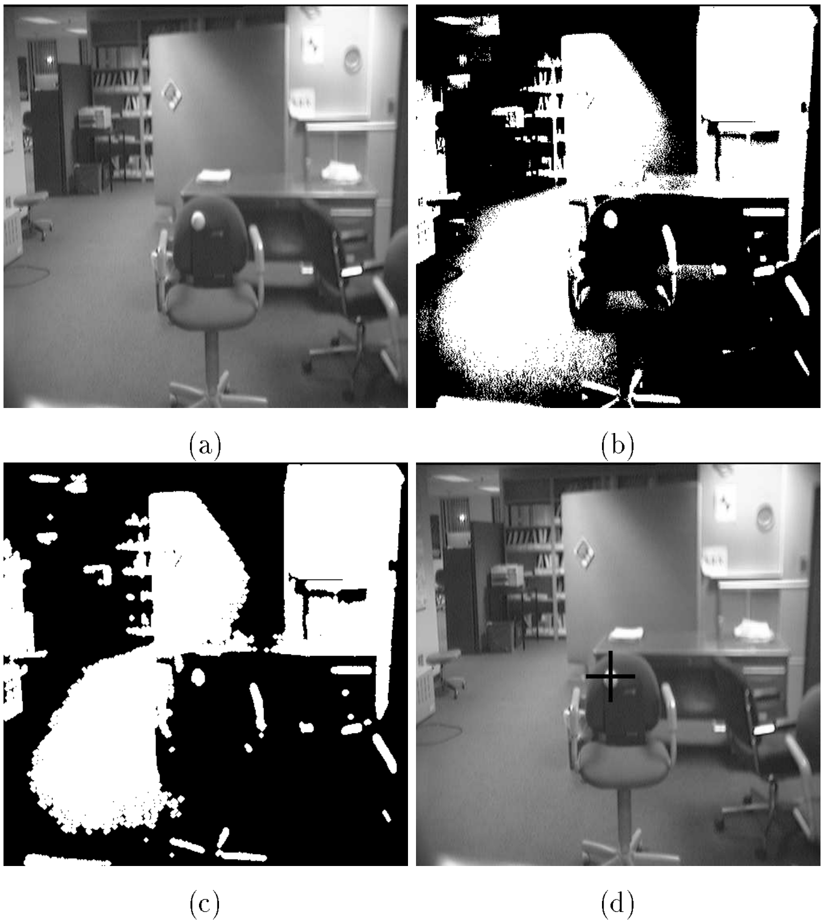


Figure 10.4: (a) The image when angle size is $w_0 \times h_0$. The target is 3.01m from the camera. (b) The generated binary image. (c) The image after the morphological operation. (d) The final result: the target is detected.

From equation (6.4), (6.5), and (6.6) of Section 6.2, we can get the next effective angle size as $20.7^o \times 19.7^o$, the next effective ranges are $N_1 = 3.01$, $F_1 = 6.16$. The outputs of the recognition algorithm when the angle size is $20.7^o \times 19.7^o$ and the target is put at N_1, F_1 are:

	Blob Size	Percentage
N_1	1332 pixels	0.9114
F_1	271 pixels	0.9077

The process of running the recognition algorithm for distance N_1, angle size $w_1 \times h_1$ are shown in Figure 10.5

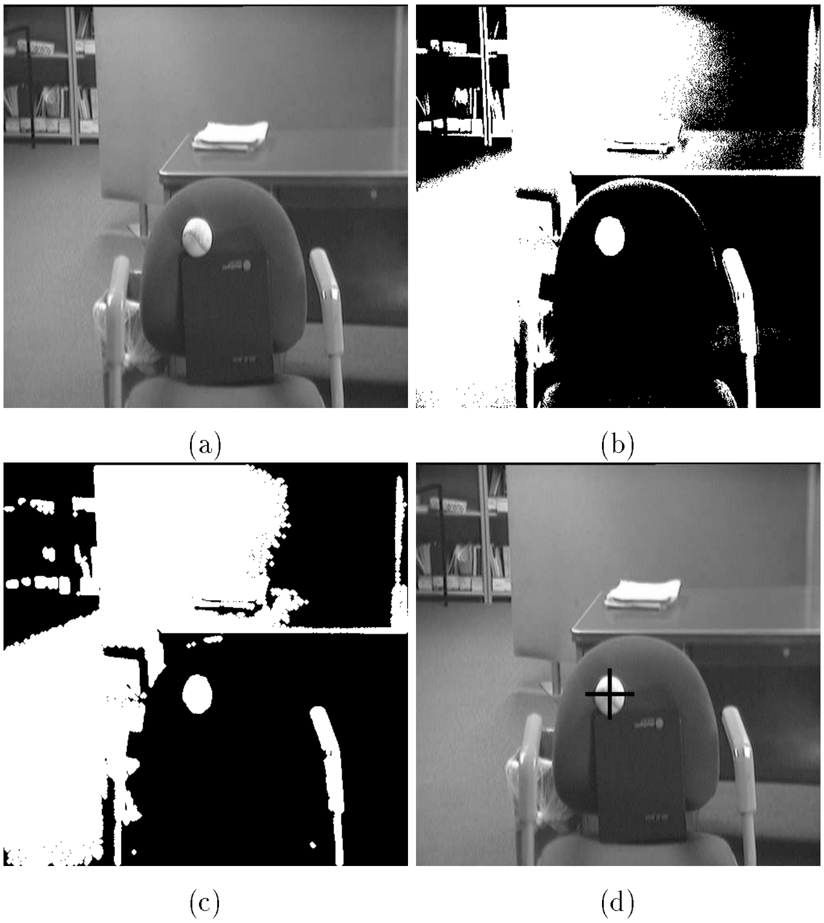


Figure 10.5: (a) The image when angle size is $w_1 \times h_1$. The target is 3.01m from the camera. (b) The generated binary image. (c) The image after the morphological operation. (d) The final result: target is detected

The process of running the recognition algorithm for distance F_1, angle size $w_1 \times h_1$ are shown in Figure 10.6

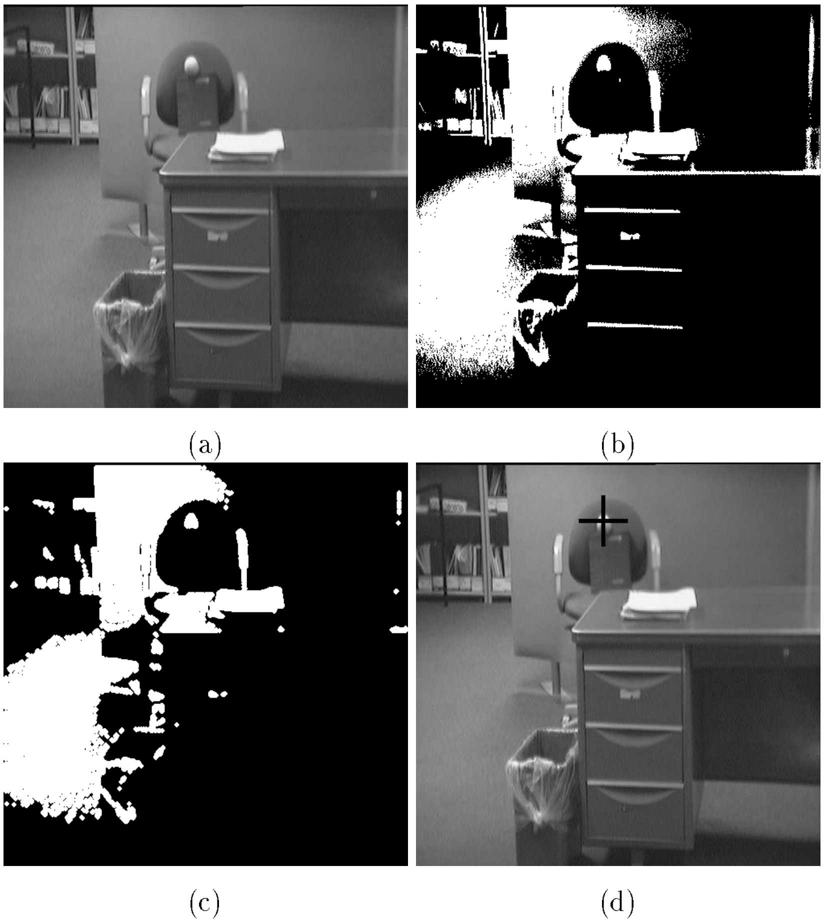


Figure 10.6: (a) The image when angle size is $w_1 \times h_1$. The target is 6.16m from the camera. (b) The generated binary image. (c) The image after the morphological operation. (d) The final result: the target is detected.

To make the problem simpler, we assume that if the target is within the effective range, it can be detected in most situations. Here we use the analytical expression for the detection function of the first effective angle:

$$b(c, f_{\pi/4}) = b(r, \varphi, \theta) = D(r)(1 - \frac{1}{16}\frac{\varphi}{\frac{41^o}{2}})(1 - \frac{1}{16}\frac{\theta}{\frac{39^o}{2}}) \tag{10.1}$$

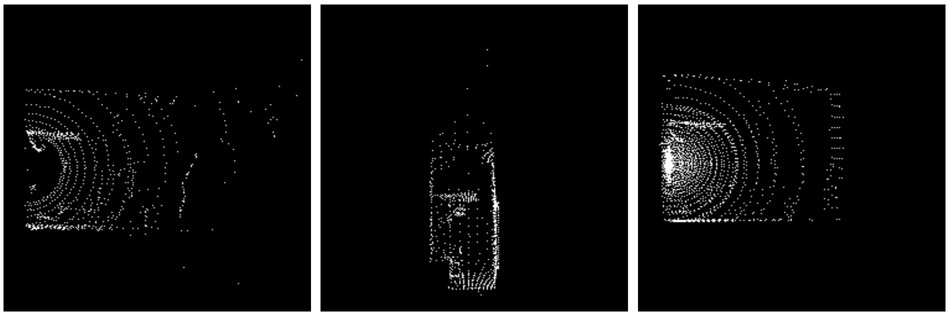
where r is the distance in meters of the cube c to the camera center, $D(r)$ is given by:

$$D(r) = \begin{cases} 0 & \text{if } r \leq 0.9 \\ 0.9 + (0.9 - 0)\frac{r-1.47}{1.47-0.9} & \text{if } 0.9 \leq r \leq 1.47 \\ 0.9 & \text{if } 1.41 \leq r \leq 3.01 \\ 0 - (0 - 0.9)\frac{r-3.5}{3.5-3.01} & \text{if } 3.01 \leq r \leq 3.5 \\ 0 & \text{if } r > 3.5 \end{cases}$$

φ (in degrees) is the offset of the cube with respect to the pan direction of the camera axis, and θ (in degrees) is the offset of the cube with respect to the tilt direction of the camera axis.

In constructing the sensed sphere, the number of divisions for tilt (which ranges from 0^o to 180^o) is 50. For the horizontal big circle, the number of divisions for *pan* is 100. When the head can reach the *pan* and *tilt* specified by the algorithm, then we adjust the head to this position and ping the laser. If the point hit by the laser is inside the search region, than this point is used to construct the sensed sphere. Otherwise, we calculate the distance by taking the search boundary as a solid plane and assume that we are pinging the laser at the corresponding *pan* and *tilt*. This calculated result will be used to construct the sensed sphere. When the head cannot reach the *pan* and *tilt* specified by the algorithm, we take the search boundary as the solid plane and calculate the distance as above. The total time used to construct this sensed sphere is approximately 45 minutes.

Figure 10.7 shows the sensed sphere constructed in our experiment.



(a) (b) (c)

Figure 10.7: (a),(b) Two views of the sensed sphere that is constructed by laser only (c) The sensed sphere taking into consideration the search region boundary. Laser points that are outside the search region are deleted.

10.1.2 When the Initial Distribution is Unknown

The first experiment assumes that we have no idea where the target is. The length of the action queue is 5. The algorithm first selects 5 actions for each angle size according to the layered sensed sphere (see Section 6.4). We can get a total of 10 actions as listed in the following table. We notice that the probability of presence for angle $20.7^o \times 19.7^o$ is generally higher than that of angle $41^o \times 39^o$. This is because the probability distribution is uniform initially, and the effective viewing volume of angle size $20.7^o \times 19.7^o$ is bigger than that of angle size $41^o \times 39^o$ (see Section 6.2 and Figure 6.3(c) for an illustration)

Angle	**Tilt**	**Pan**	**Probability of Presence**
$41^o \times 39^o$	1.788960	0.000000	0.020326
$41^o \times 39^o$	1.788960	5.870162	0.019852
$41^o \times 39^o$	1.073376	5.738306	0.014990
$41^o \times 39^o$	1.073376	0.000000	0.014859
$41^o \times 39^o$	1.788960	0.733770	0.014341
$20.7^o \times 19.7^o$	1.282817	0.000000	0.031588
$20.7^o \times 19.7^o$	1.649336	0.000000	0.031588
$20.7^o \times 19.7^o$	1.282817	0.382451	0.031103
$20.7^o \times 19.7^o$	1.649336	0.367665	0.031103
$20.7^o \times 19.7^o$	2.015855	0.000000	0.030841

The algorithm uses Equation (6.1) to compare these 10 actions and select the top 5. The top 5 actions and their detecting probabilities are listed below:

Angle	**Tilt**	**Pan**	**Probability of Detection**
$20.7^o \times 19.7^o$	1.649336	0.367665	0.055094
$20.7^o \times 19.7^o$	1.282817	0.382451	0.051286
$20.7^o \times 19.7^o$	1.649336	0.000000	0.048718
$20.7^o \times 19.7^o$	1.282817	0.000000	0.045231
$20.7^o \times 19.7^o$	2.015855	0.000000	0.040451

After executing these 5 actions, the algorithm updates the environment accordingly. It then constructs the layered sensed sphere according to the updated distribution. The next 5 actions, according to this new distribution, are listed below:

Angle	Tilt	Pan	Probability of Detection
$20.7^o \times 19.7^o$	1.649336	5.882644	0.040265
$41^o \times 39^o$	1.788960	5.870162	0.030158
$41^o \times 39^o$	1.788960	0.733770	0.029876
$20.7^o \times 19.7^o$	0.916298	6.026081	0.028314
$41^o \times 39^o$	1.073376	5.738306	0.026084

Figure 10.8 shows the top 10 images taken.

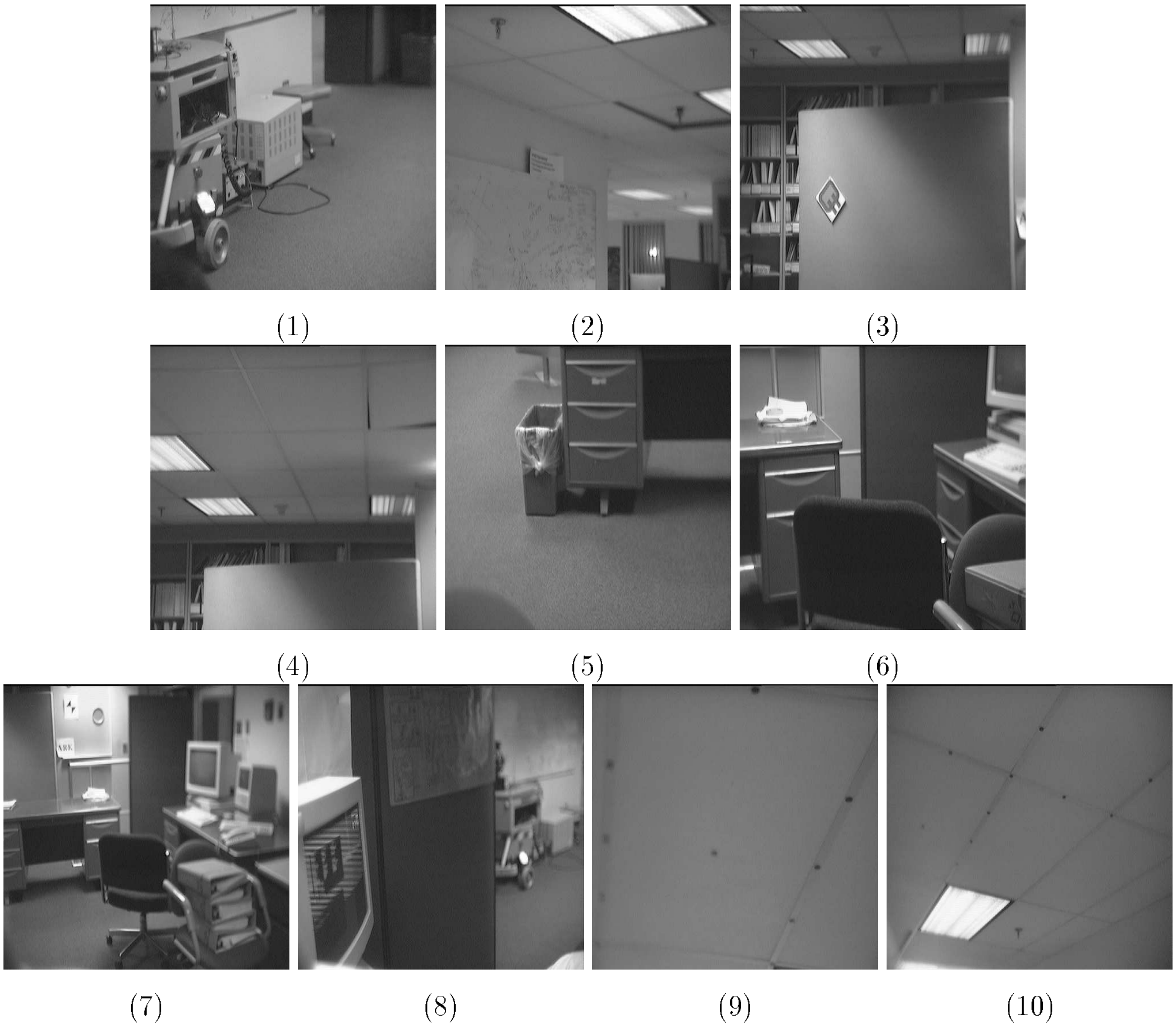


Figure 10.8: The top 10 images taken when the initial distribution is unknown.

We can see from Figure 10.8 that some of the actions are directed to the ceiling where the baseball can not possibly be. This is because we assume a uniform distribution at the beginning.

In a real situation, we may know something about the distribution of the target. From the viewpoint of efficiency, we should take anything we know about the probability distribution into consideration.

10.1.3 When the Initial Distribution is Known

This experiment tests our strategy when the distribution is known in advance. The sensor planning strategy and the recognition algorithm are exactly the same as the experiments described in Section 10.1.2 except that we do not initialize the distribution as uniform. We first initialize the probability of those cubes around table surfaces $EFGH$, $IJKL$ and $MNOP$ as 1 and the probability of other cubes within the search region as 0 (Figure 10.1). The three regions around the three table surfaces are as follows: for table $EFGH$ and $IJKL$: $0.1 \leq x \leq 3.6; 0.1 \leq y \leq 0.7; 0.73 \leq z \leq 0.9$ and for table $MNOP$: $4 \leq x \leq 4.7; 1.1 \leq y \leq 2.7; 0.73 \leq z \leq 0.9$.

We normalize the distribution by making the sum of the probability of cubes within the search region equal to 0.5. Thus the outside probability is 0.5. The top 5 actions selected according to this distribution and their corresponding detecting probabilities are listed below:

Angle	**Tilt**	**Pan**	**Probability of Detection**
$20.7^o \times 19.7^o$	1.649336	6.250309	0.136058
$41^o \times 39^o$	1.788960	5.136392	0.126364
$20.7^o \times 19.7^o$	1.649336	0.000000	0.123388
$20.7^o \times 19.7^o$	2.015855	0.000000	0.119216
$20.7^o \times 19.7^o$	1.282817	0.000000	0.117252

After executing these 5 actions and updating the environment, the next 5 actions according to the new distribution are as follows:

Angle	**Tilt**	**Pan**	**Probability of Detection**
$20.7^o \times 19.7^o$	2.015855	5.692438	0.116527
$20.7^o \times 19.7^o$	1.649336	5.882644	0.115142
$41^o \times 39^o$	1.073376	5.738306	0.105849
$41^o \times 39^o$	1.788960	5.870162	0.105441
$20.7^o \times 19.7^o$	1.282817	5.736768	0.097463

Figure 10.9 shows the top 10 images taken.

From the figure we see that the actions are as follows.

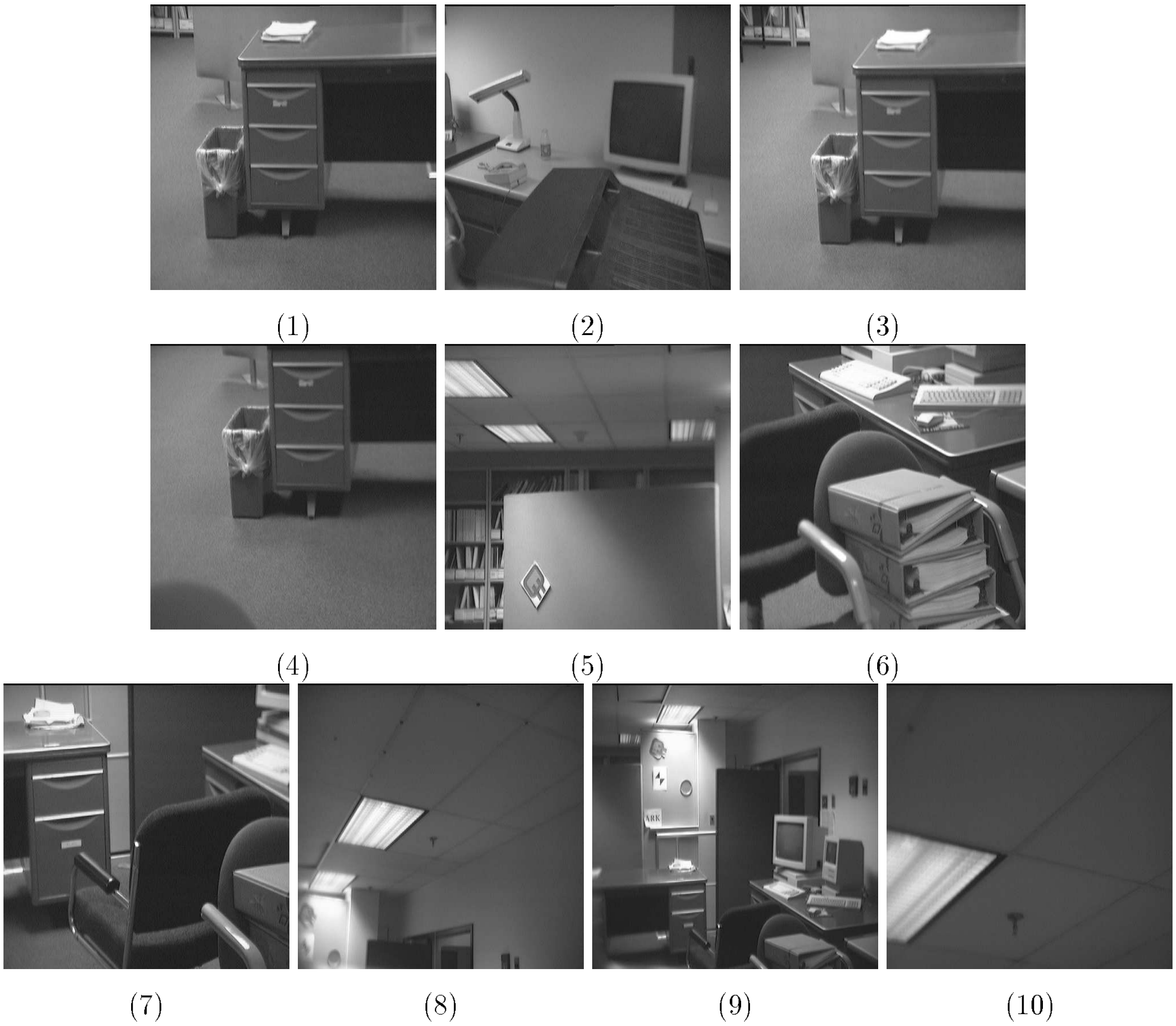


Figure 10.9: The 10 images taken when the initial distribution is known.

1. The first action examines a part of table $MNOP$.

2. The second action examines table $EFGH$.

3. The third action examines a part of table $MNOP$, which is quite similar to the first action. This is because the length of the action queue is 5. Thus, after the first action is applied, the distribution is not updated, and the probability of detecting the target for the third action is also high.

4. The fourth action also examines table $MNOP$. Since the region of probability we give is an approximation of the real situation, the Z range of the region is not exactly on the table top. This might cause this action to examine the region below the table.

5. The fifth action examines the region above the table $MNOP$.

6. The sixth action examines table $IJKL$ and a corner of $EFGH$.

7. The seventh action examines corners of both table $IJKL$ and table $MNOP$.

8. The eighth action examines region above table $IJKL$.

9. The ninth action examines table $IJKL$. From the detection function of the recognizer we know that table $MNOP$ is not examined, although it is within the field of view of the camera. This is because the distance of $MNOP$ to the camera is too far with respect to the angle size $41^o \times 39^o$.

10. The tenth action is not satisfactory. Two possible reasons are: 1. all the necessary actions are selected; 2. mechanical error of the device.

Generally speaking, the resulting image sequences are satisfactory, but note that there are four actions among the first 5 actions which examine table $MNOP$. This is because there are 5 actions on the action queue and we only update the distribution after these 5 actions are executed. This problem will be solved if the length of the action queue is 1. Thus, we update the distribution as soon as an action is executed. The benefit of making the length of the action queue bigger than 1 is that we might find the target during the first group of actions, thus we can save the time for distribution updating.

10.1.4 When the Action Queue Length is 1

Experiment 3 tests our algorithm when the length of the action queue is one. The search space and the probability distribution are exactly the same as experiment 2. Here we consider the first three actions executed.

The candidate actions selected based on the layered sensed sphere for each phase are listed below:

Phase	Angle	Tilt	Pan	Prob. of Presence
First	$41^o \times 39^o$	1.788960	5.870162	0.108861
First	$20.7^o \times 19.7^o$	1.649336	6.250309	0.041139
Second	$41^o \times 39^o$	1.788960	5.136392	0.125967
Second	$20.7^o \times 19.7^o$	1.649336	6.250309	0.013291
Third	$41^o \times 39^o$	1.788960	5.136392	0.067562
Third	$20.7^o \times 19.7^o$	1.649336	6.250309	0.015413

The real executed actions are listed below:

Phase	Angle	Tilt	Pan	Probability of Detection
First	$20.7^o \times 19.7^o$	1.788960	5.870162	0.135798
Second	$41^o \times 39^o$	1.788960	5.136392	0.143653
Third	$41^o \times 39^o$	1.649336	6.250309	0.102285

Figure 10.10 shows the three images taken.

(1) (2) (3)

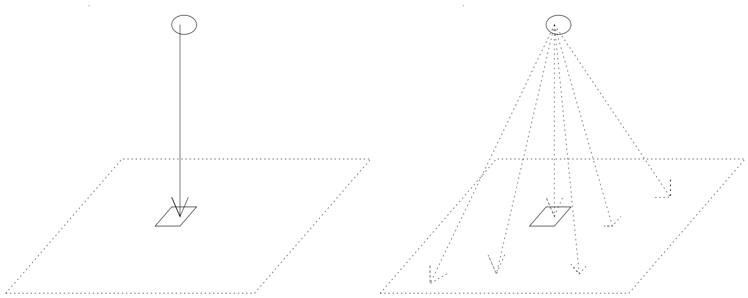
Figure 10.10: The 3 images taken for experiment 3.

We can see from the images that the first action examines table $IJKL$, the second action examines table $EFGH$, the third action examines table $MNOP$. This action sequence is satisfactory.

10.1.5 The Influence of Initial Knowledge on the Performance of the Sensor Planning System

Our formulation may be a little misleading regarding the concept of sensor planning for object search, because the real purpose for sensor planning is to bring the target into the field of view of the camera as early as possible, not to maximize the probability of detecting the target. The probability distribution is just an approximation of the real situation based on our knowledge. So, the best effort allocation refers to the best action sequence with respect to the initial distribution $\mathbf{p}$ from our knowledge, not the distribution $\mathbf{p}^*$ from the real world.

An intuitive explanation about the difference between $\mathbf{p}$ and $\mathbf{p}^*$ is shown in Figure 10.11. The correct probability distribution $\mathbf{p}^*$ devotes all the probability mass (totally 1) to the cube that contains the target center. The distributions obtained from our imperfect knowledge $\mathbf{p}$ has some of the probability masses scattered into other cubes. From a certain point of view, the purpose of object search is to bring those probability masses, that are mistakenly scattered into other cubes, back into the cube that contains the center of the target.



(a) (b)

Figure 10.11: (a) The probability distribution in the case of perfect knowledge $\mathbf{p}^*$. All the probability masses are devoted to the cube that contains the center of the target. (b) The probability distribution in the case of initial knowledge according to our perception of the world. Some probability masses are mistakenly scattered into other cubes.

In our formulation, the probability distribution $\mathbf{p}$ is used to guide the sensor planning process. Our knowledge about the initial distribution is very important in order to make our scheme work effectively. The more accurate our knowledge about the initial distribution, the more efficient our algorithm.

We have performed experiments to examine the influence of the initial knowledge on the effectiveness of our sensor planning strategy. Experimental results are listed in Table 1, which gives the average number of actions needed to bring the ball into the the effective viewing volume of the camera for planning and non-planning strategies. The meaning of the data from the table is as follows (Note: for convenience, we use A, B, and C to represent tables $EFGH$, $IJKL$, and $MNOP$ respectively). The first row of the second column means that the target is on table A. The second row of the second column means that the number of actions needed to bring the ball into the effective volume of the camera is 1 when we give the table surface A a high probability distribution. The third row of the second column means that the number of actions needed to detect the target by using the Non-planning strategy is 8. The fourth row of the second column means that the number of actions needed to bring the ball into the effective volume of the camera is 7 when we give the table surface C a high probability distribution (Note: the target is on A). The similar explanations can be applied to other columns of the table.

From the second and third row of Table 1, we can see that the planning strategy is much more efficient when our knowledge about the distribution is reasonable at the beginning. From the fourth row of Table 1 we can see that the performance of the planning strategy is not very satisfactory when we have misleading information at the beginning. One of the above test results is shown in Figure 10.12.

Target Pos.	A	B	C	A or B	A or C	A, B or C
Plan$\binom{correct}{knowledge}$	(A)1	(B)1	(C)1	(AB)1.5	(AC)1.5	(ABC)2.5
No Plan	8	9	10	8.5	9	9
Plan$\binom{incorrect}{knowledge}$	(C)7	(A)8.5	(B) 11.5	(C)6	(B)9	(C)4.3

Table 1

10.2 Where to Move Next: an Experiment with the ARK Platform

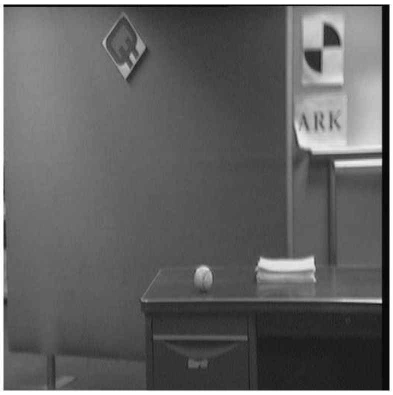
In this section we describe experiment in which the ARK platform is used to test the complete sensor planning strategy, especially the where to move next procedure.

10.2.1 Search Region and the Recognition Algorithm

The search region is the same as that in the Laser Eye experiment. The recognition algorithm is also the same as in the Laser Eye experiment, except that the threshold for the parameter values are different. Here we use $I_0 = 110$, $B_{min} = 170$ pixels, $B_{max} = 5500$ pixels, and $R_0 = 0.91$. The viewing angle size is fixed: $w_0 = h_0 = 30^o$. Thus we only have one effective range, $N_0 = 1m$, $F_0 = 5.5m$. The outputs of the recognition algorithm at N_0 and F_0 are listed below:

	Blob Size	Percentage
N_0	5478 pixels	0.9237
F_0	171 pixels	0.9474



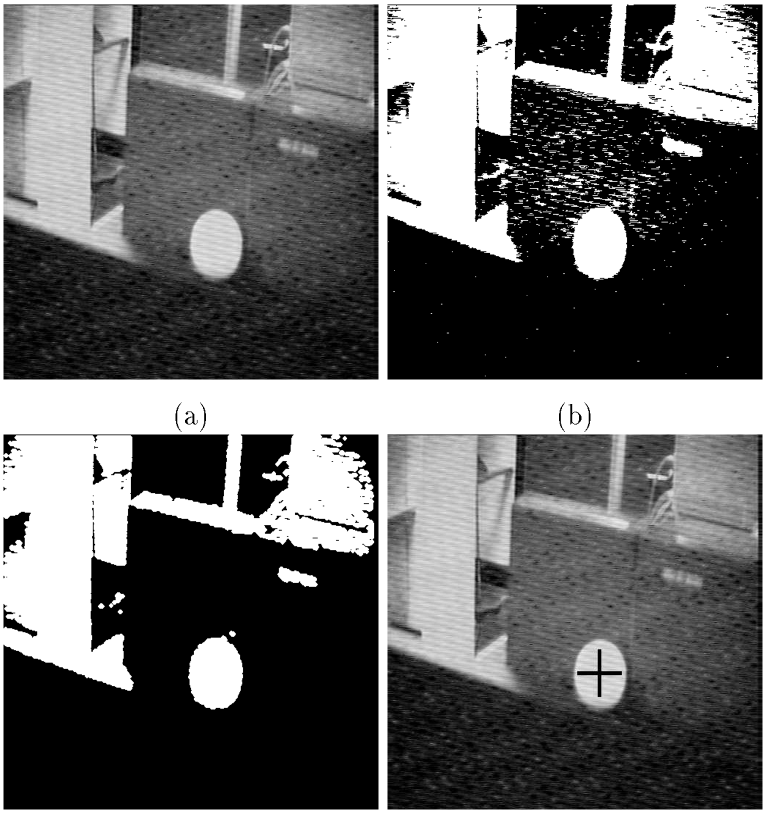


(a) (b) (c)


(d) (e)

Figure 10.12: One of the image sequences obtained using the planning strategy, where the target is assumed on A, B, or C and we give (ABC) high probability at the beginning. Although the ball appeared in the first image (a), the algorithm failed to detect it because it is outside the effective range of the action (size $41^o \times 39^o$). The third action ((c), size $20.7^o \times 19.7^o$) found the target. (d) is the image resulting from action (c) after region growing. (e) The final result of the analysis of image (c) by the given recognition algorithm; the target is detected.

The process of running the recognition algorithm when the target is at distance N_0 is shown in Figure 10.13



(a) (b)

(c) (d)

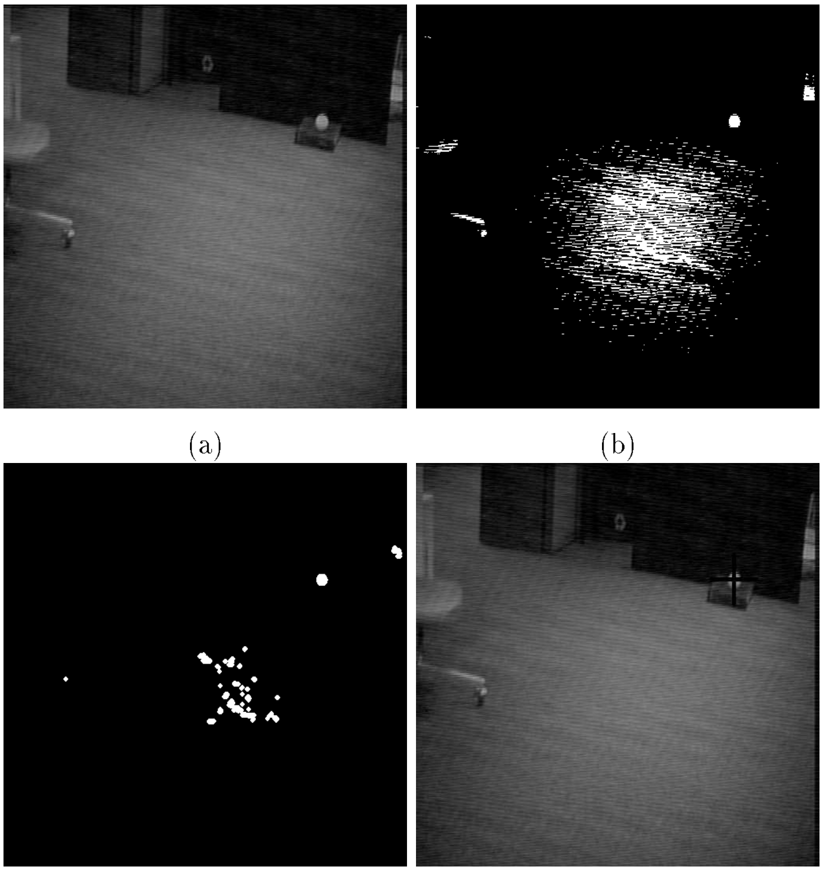
Figure 10.13: (a) The image when the target is at $1m$ from the camera. (b) The generated binary image. (c) The image after the morphological operation. (d) The final result: the target is detected.

The process of running the recognition algorithm when the target is at distance F_0 is shown in Figure 10.14

As in the experiments with the Laser Eye, we use the analytical expression for the detection function:

$$b(c, f_{\pi/6}) = b(r, \varphi, \theta) = D(r)(1 - \frac{1}{16}\frac{\varphi}{\frac{30^o}{2}})(1 - \frac{1}{16}\frac{\theta}{\frac{30^o}{2}}) \tag{10.2}$$

In the above equation, r (in meters) is the distance of the cube c to the camera center, φ (in degrees) is the offset of the cube with respect to the pan direction of the camera axis, θ (in degrees) is the offset of the cube with respect to the tilt direction of the camera axis, and $D(r)$ is given by



(a) (b)

(c) (d)

Figure 10.14: (a) The image when the target is at 5.5m from the camera. (b) The generated binary image. (c) The image after the morphological operation. (d) The final result: the target is detected

$$D(r) = \begin{cases} 0 & \text{if } r \leq 1 \\ 0.95 & \text{if } 1 \leq r \leq 5 \\ 0.95 + (0.95 - 0)\frac{r-5.5}{5.5-5} & \text{if } 5 \leq r \leq 5.5 \\ 0 & \text{if } r > 5.5 \end{cases}$$

In constructing the sensed sphere, the number of divisions for tilt (which ranges from 0^o to 180^o) is 50. For the horizontal big circle, the number of division for *pan* is 100. From our experience, the safest tilt range for the head is between 0^o and 115^o (Because of hardware limitations, the maximum tilt is 130^o. When the tilt approach this value, the camera may fall off from its home position). So, we only use the real laser when the tilt is between 0^o and 115^o. When the head can reach the *pan* and *tilt* specified by the algorithm, we adjust it to this position and ping the laser. If the point hit by the laser is inside the search region, than this point will be used to construct the sensed sphere. Otherwise, we will calculate the distance by taking the search boundary as a solid plane and assume that we are pinging the laser at the corresponding *pan* and *tilt*. This calculated result will be used to construct the sensed sphere. When the head cannot reach the *pan* and *tilt* specified by the algorithm, then we take the search boundary as the solid plane and calculate the distance as above. The total time used to construct this sensed sphere is approximately several hours (Note, here we use the slow speed of the robot).

At the beginning, we give the table top **B**, table top **C**, and the floor region **D** high probability of presence. The real position of the baseball is within region **D** (Figure 10.15). The length of the action queue is 1 during the selection phase. When the probability of detecting the target is less then 0.0065, the robot is required to change the position. The following subsections list the experimental results.

10.2.2 When the Robot is at Its First Position

In the beginning, the robot is at $(1.6, 2.9)$ (P_1 in Figure 10.15).

The sensed sphere constructed by the real laser at P_1 is shown in Figure 10.16 (a). We can see that some points are outside the search region. When the search region boundary is considered, a lot of outside points are deleted, and we get the real sensed sphere used to do the sensor planning task (Figure 10.16 (b)). Figure 10.16 (c) shows the solid cube generated according to the sensed sphere. Figure 10.16 (d) shows the cubes that are occluded by these solid cubes.

The following table lists the actions selected by the algorithm and their corresponding detection probability.

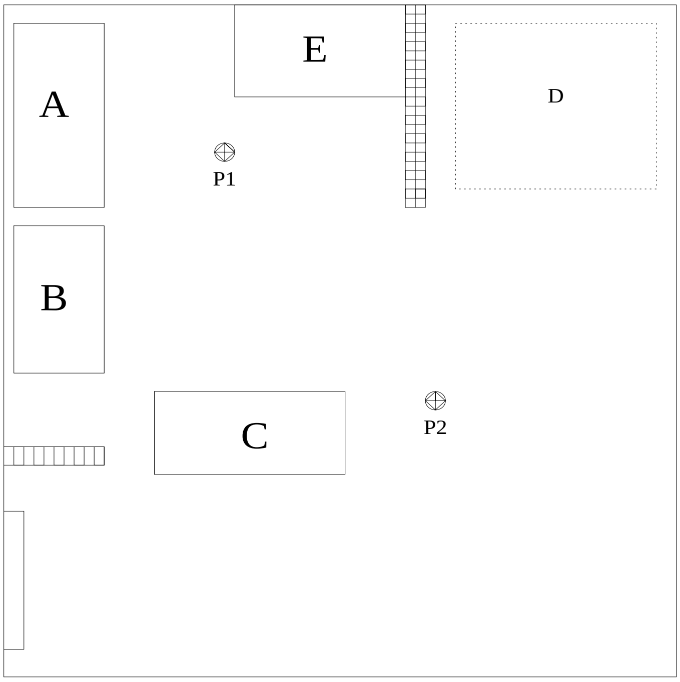


Figure 10.15: The environment for the search task using the ARK platform, where P_1 is the platform's initial position.

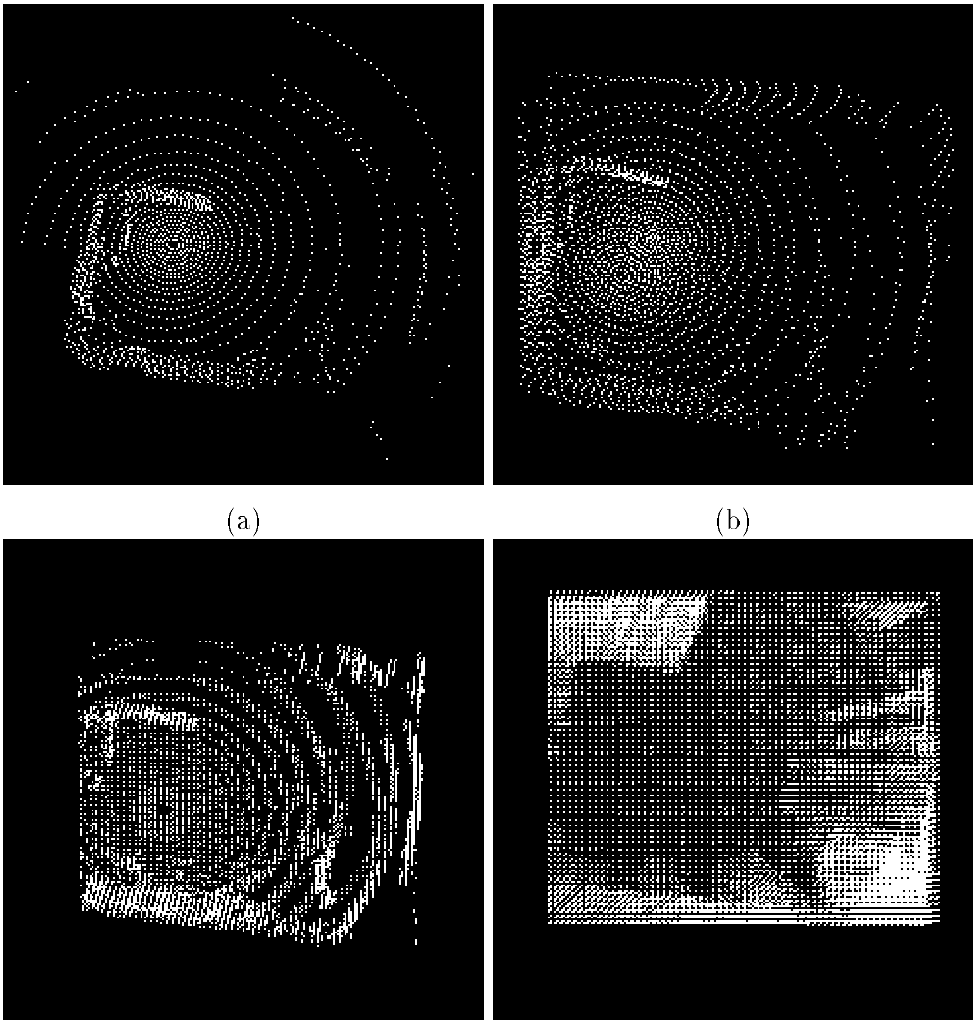
Tilt	Pan	Probability of Detection
1.801100	0.000000	0.037403
2.315700	5.655361	0.036140
2.315700	4.948441	0.036654
1.286500	5.366405	0.006105

We can see that the probability of detection of the last action is 0.006105 and that it is less than 0.0065. Thus the robot will quit the "where to look next" process and begin the "where to move next" process.

Figure 10.17 shows the four images taken by the head according to the four actions listed above. We can see that the first action examines table **C**, the second and third actions examine table **B**. The fourth action simply points upward, none of the specified region is examined. The probability of detecting the target by the fourth action is very low, because none of the high probability region is brought into the effective range of the camera. No action is selected to check region **D**, because **D** is occluded by the office wall.

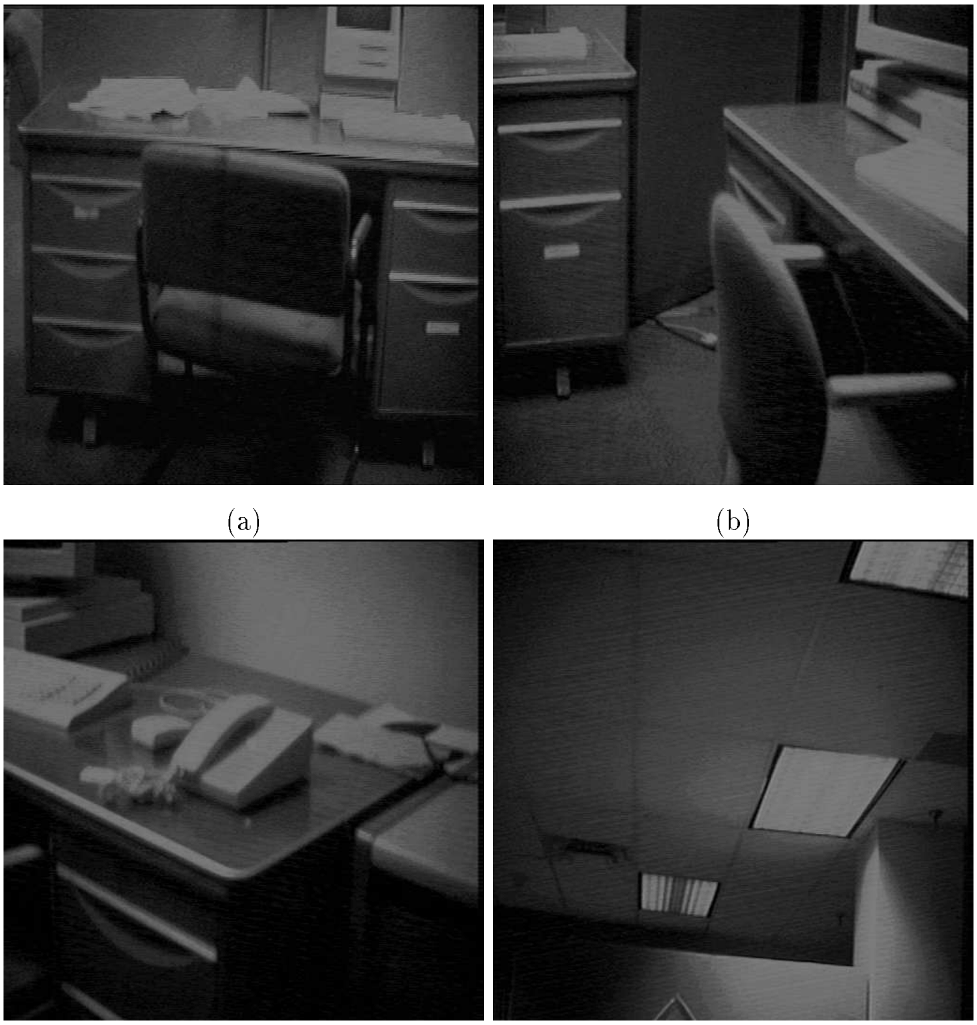
10.2.3 Where to Move Next

To select the next position, the robot first finds all the candidate positions that can be reached. Then, for each reachable candidate position, the robot calculates the expected sensed sphere



(a) (b) (c) (d)

Figure 10.16: (a) The sensed sphere constructed only by the real laser. (b) The sensed sphere constructed after consideration of the search region boundary. (c) The solid cube. (d) The cube that is occluded according to the sensed sphere.



(a) (b)

(c) (d)

Figure 10.17: The four actions generated by the sensor planning algorithm. (a) Action with $tilt$ = 1.801100 and pan = 0.000000. (b) Action with $tilt$ = 2.315700, pan = 5.655361. (c) Action with $tilt$ = 2.315700, pan = 4.948441. (d) Action with $tilt$ = 1.286500, pan = 5.366405.

and the probability of presence within this sphere. In our case, the robot only considers those positions for which $x = 0.1, 0.6, 1.1, \ldots, 6.1$ and $y = 0.1, 0.6, 5.6$. Figure 10.18 shows the probability for each candidate position. Among them, the position $(4.6, 3.6)$ has the maximum probability 0.238158. Figure 10.19 (a) shows the expected sensed sphere for $(4.6, 3.6)$ obtained by only considering the boundary of the search region. Figure 10.19 (b) shows the constrained sensed sphere obtained by considering the solid cubes at this position.

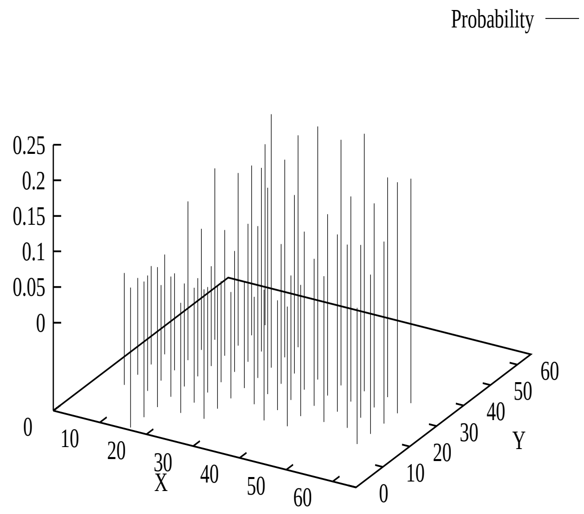
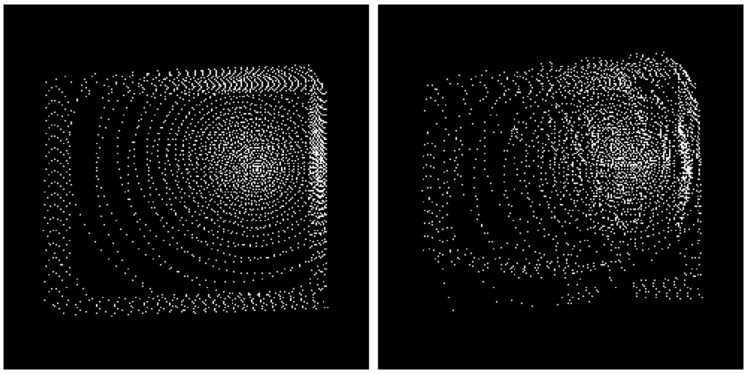


Figure 10.18: The probability of presence for each possible platform positions.



(a) (b)

Figure 10.19: The expected sensed sphere at $(4.6, 3.6)$ by only considering the boundary of the search region. The expected sensed sphere after the constraining using the solid cubes.

10.2.4 The New Position

When the robot drives to the next position $(4.6, 3.6)$, it constructs the sensed sphere and repeats the search process again. Figure 10.20(a) shows the sensed sphere constructed by the real laser at the new position. Figure 10.20(b) shows the sensed sphere obtained by considering the boundary of the search region. Figure 10.20(c) shows the solid cubes at the new position. Figure 10.20(d) shows the cubes that are occluded according to the new position. Figure 10.21(a) shows the combined solid cubes with respect to the previous and current position. Figure 10.21(b) shows the cubes that are occluded with respect to the previous and current positions.

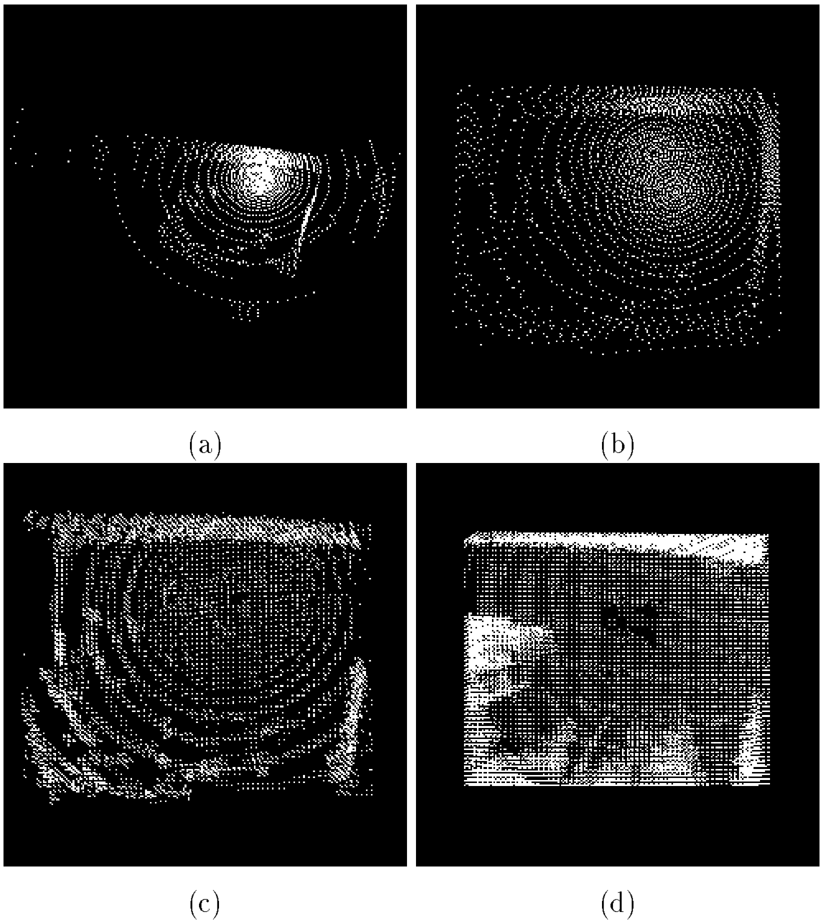
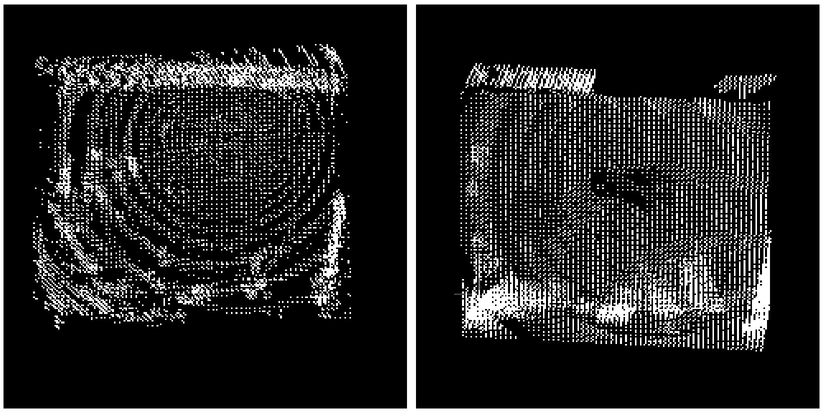


Figure 10.20: (a) The sensed sphere constructed by the real laser. (b) The sensed sphere by considering the boundary of the search region. (c) The solid cubes according to the sensed sphere. (d) The cubes that are occluded with respect to the current position.

10.2.5 Sensor Planning

At the new position, the robot will begin the search process again. The first action is: $tilt$ = 2.315700 and pan = 3.534601. This action finds the target. Figure 10.22 shows the image



(a) (b)

Figure 10.21: (a) The combined solid cubes. (b) The cubes that are occluded with respect to the previous and current position.

of this action and the image processing result. The results of this experiment tell us that our sensor planning strategy is able to select a new robot position which can avoid occlusion.

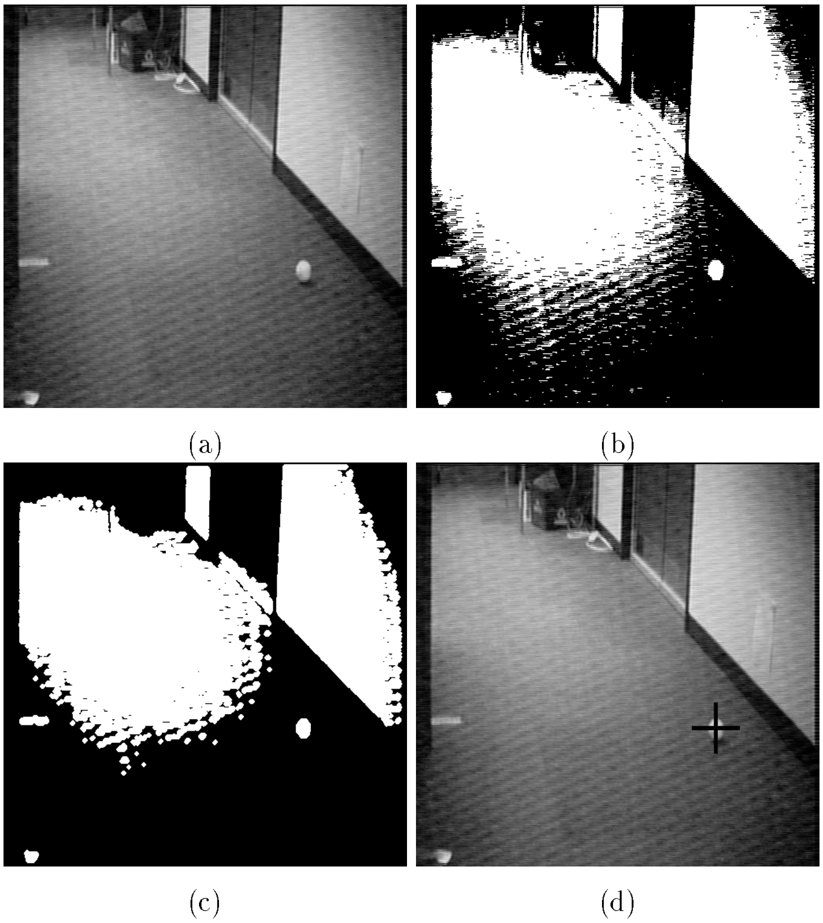


Figure 10.22: . (a) The image of the first action. (b) The generated binary image. (c) The image after the morphological operation. (d) The final result: the target is detected (blob size is 372, percentage is 0.9140).

Chapter 11

Conclusion

An interesting phenomenon of current computer vision research is that the purely theoretical side of computer vision is relatively more emphasized. As a result, some theoretical papers have been produced which may not be able to be applied to some real situations. These papers may start with unrealistic assumptions with little basis in real application. In order to solve this problem, it is beneficial to put more emphasis on the "task oriented vision" framework as an alternative to the classic reconstructionist paradigm. This framework should provide a set of solid assumptions and goals upon which each vision module should be developed and it should emphasize the engineering issues and promote the experimental side of computer vision. For this new framework, the vision system is just one component of an agent embedded in the world, and the vision system should provide only the minimum perceptual functionality the agent needs to perform whatever it is doing at the moment.

To do research within the "task oriented vision" framework, it is ideal to select a visual task that has the following characteristics:

- The agent needs to selectively analyze a scene by purposefully controlling the camera parameters.
- It is not necessary for the agent to know every detail about the environment to perform the task, it only looks for minimally sufficient evidence required to solve the task.

Object search has these characteristics and is therefore a suitable task for this framework. For this task, the vision module involved is object recognition, and this module is only one component of the agent's ability in performing the task. In order to find the target, the agent has to purposefully change the sensing parameters so as to bring the target into the field of view of the camera and to make the image of the target to be easily recognized by the available recognition algorithms. To locate the target, the agent is not required to reconstruct every detail of the environment before the search process. It only needs to direct the sensor to examine a certain part of the region until the target is found. The study of object search may lead to more insight into existing vision problems. For example, the problem of object recognition is less difficult when the image is intelligently grabbed by a controlled camera, and the image analysis is interpreted within specific contexts [87]. Of course, the simplicity obtained in the image analysis phase is at the cost of the complexity encountered in the camera controlling phase. Much work has been done on image analysis. It is now time to study the problems related to the control of the camera. This is one reason why the sensor planning task for object search should be studied, and it provides the prime motivation for this thesis.

In this thesis we systematically study the task of sensor planning for object search. We formulate the task and pinpoint many important properties. Guided by theoretical results, we present a practical sensor planning strategy and test this strategy by simulation experiments and real experiments.

11.1 Contributions

The contributions of this thesis are summarized as follows:

- We systematically study the task of $3D$ object search and identify three subtasks for object search: sensor planning, sensor manipulation, and object recognition.

- We point out two important factors that must be considered by an object search system. One is the agent's knowledge about the probability distribution of the target. The other is the capacity of the recognition algorithm. We argue that the task of $3D$ object search is different from the task of robot exploration. The goal of object search is not just to check every part of the environment to detect the target—–the sensor should be guided by the agent's knowledge about the target distribution.

- We formulate the task of sensor planning for object search as a problem of maximizing the probability of detecting the target within a given cost constraint. We point out that for a given effort allocation, the probability of detecting the target is not influenced by the order of application of the given set of actions. We also point out many other important properties of the object search task by analyzing the formulation.

- We prove that the sensor planning task for object search is NP-Complete in general and suggest a greedy strategy to solve this task. We prove that in certain situations the greedy strategy may lead to an optimal solution. For the case where the search agent is a mobile robot, we further divide the greedy strategy into two subtasks: "where to look next" and "where to move next".

- We propose the concept of detection function for a given recognition algorithm. This concept can be used as an evaluation matrix for comparing the performance of different recognition algorithms. We also propose the concept of a sensed sphere for a search agent within the environment. This allows representation of the unoccluded environment surrounding the search agent.

- We propose strategies for "where to look next" and "where to move next". We decompose the huge space of candidate actions into a finite set of actions that must be tried. We perform both simulation experiments and real experiments to test our strategies.

- We provide in this thesis a modeling of an autonomous agent - a $3D$ object search agent that decides for itself how to relate sensor data to motor commands in its efforts to achieve goals. Our work can be taken as a starting point to study various issues concerned with autonomous agents [102][100][97].

11.2 Future Directions

The task of $3D$ object search is a common task involving a vision module. Although this thesis formulates the task of sensor planning for object search and proposes a practical strategy for this task, it raises as many questions as it answers. We believe that the results presented here are a starting point for further study of "task oriented vision" in general and "$3D$ object search" in particular. Our research has opened a number of new avenues of research as listed below.

- **Dynamic $3D$ Object Search** A natural extension of our current research is the design of a sensing strategy for movable targets. There are generally two kinds of movements. One is that the target is allowed to move according to some general regulations, and the other is that the target is allowed to move intelligently. This extension requires the computer vision module to be able to solve a new and challenging problem: moving object recognition.

- **Re-evaluation of the Object Recognition Problem** Traditionally, research in object recognition has been aimed at building a perfect recognizer which has excellent performance over a wide range of conditions. This problem has proved to be very difficult. By treating the object recognition as just one component of a whole system within a specific environmental context, the requirement for recognition capacity may not be as high as before. The demands on object recognition are reduced in two ways. One is to constrain the situations to be considered according to the environment the agent is working on. The other is to design a good sensor planning strategy so as to shift some of the burden of the recognizer to the planner.

- **Partially Occluded Object Recognition** How does a robot recognize a partially occluded object? The design of a planning strategy such that the occluded image can be used to guide the camera to get a perfect view of the object is an important problem in sensor planning.

- **Initial Knowledge Acquisition** How do we obtain the initial target distribution? Future research must determine how to combine the knowledge from different sources so as to form a good initial distribution.

- **Planning** There are many results obtained by the AI community in the field of planning. Research is required to investigate how to apply their results to the sensor planning task for object search.

- **Performance evaluation** There are some polynomial algorithms which have been developed for the Knapsack problem which apply in special cases [76][27]. Although sensor planning for object search is not identical to the Knapsack problem, they are similar in certain aspects. So, it would be interesting to study the performance of the corresponding algorithms for the sensor planning task. Further, it would be interesting to study the performance of a greedy strategy with respect to various initial target distributions and various detection functions.

Appendix A

The Proof of the Properties

Property 1: *The total sum of the probabilities of all the cubes (including the outside probability) after each action is* 1*.*

$$\sum_{i=1}^{n,o} \mathbf{p}^{[k]}(c) = 1 \tag{A.1}$$

Proof

From the environment updating rule we have: for any given action $\mathbf{f}$,

$$\mathbf{p}(c_i, \tau_{\mathbf{f}+}) \leftarrow \frac{\mathbf{p}(c_i, \tau_{\mathbf{f}})(1 - \mathbf{b}(c_i, \mathbf{f}))}{\mathbf{p}(c_o, \tau_{\mathbf{f}}) + \sum_{j=1}^{n} \mathbf{p}(c_j, \tau_{\mathbf{f}})(1 - \mathbf{b}(c_j, \mathbf{f}))}, \quad i = 1, \ldots, n, o$$

Thus,

$$\begin{aligned}
\sum_{i=1}^{n,o} \mathbf{p}(c_i, \tau_{\mathbf{f}+}) &= \sum_{i=1}^{n,o} \left\{ \frac{\mathbf{p}(c_i, \tau_{\mathbf{f}})(1 - \mathbf{b}(c_i, \mathbf{f}))}{\mathbf{p}(c_o, \tau_{\mathbf{f}}) + \sum_{j=1}^{n} \mathbf{p}(c_j, \tau_{\mathbf{f}})(1 - \mathbf{b}(c_j, \mathbf{f}))} \right\} \\
&= \frac{\sum_{i=1}^{n,o} \mathbf{p}(c_i, \tau_{\mathbf{f}})(1 - \mathbf{b}(c_i, \mathbf{f}))}{\mathbf{p}(c_o, \tau_{\mathbf{f}}) + \sum_{j=1}^{n} \mathbf{p}(c_j, \tau_{\mathbf{f}})(1 - \mathbf{b}(c_j, \mathbf{f}))} \\
&= 1
\end{aligned}$$

$\square$.

Property 2 *For any given action* $\mathbf{f}$*, we have*

Part A

$$0 \le P(\mathbf{f}) \le 1 \tag{A.2}$$

Part B

$P(\mathbf{f}) = 0$ *if and only if* $\forall c$, $\mathbf{p}(c, \tau_{\mathbf{f}}) = 0$ *or* $\mathbf{b}(c, \mathbf{f}) = 0$.

Part C

$P(\mathbf{f}) = 1$ *if and only if* $\mathbf{p}(c_o, \tau_{\mathbf{f}}) = 0$ *and* $\forall c, \mathbf{p}(c, \tau_{\mathbf{f}}) \neq 0$, $\mathbf{b}(c, \mathbf{f}) = 1$.

Proof

Part A

Since $\mathbf{p}(c_i, \tau_\mathbf{f}) \geq 0$ and $\mathbf{b}(c_i, \mathbf{f}) \geq 0$, we have

$$P(\mathbf{f}) = \sum_{i=1}^{n} \mathbf{p}(c_i, \tau_\mathbf{f})\mathbf{b}(c_i, \mathbf{f}) \geq 0$$

Since $\mathbf{p}(c_i, \tau_\mathbf{f}) \leq 1$ and $\mathbf{b}(c_i, \mathbf{f}) \leq 1$, we have

$$\begin{aligned} P(\mathbf{f}) &= \sum_{i=1}^{n} \mathbf{p}(c_i, \tau_\mathbf{f})\mathbf{b}(c_i, \mathbf{f}) \\ &\leq \sum_{i=1}^{n} \mathbf{p}(c_i, \tau_\mathbf{f}) \\ &= 1 - \mathbf{p}(c_o, \tau_\mathbf{f}) \\ &\leq 1 \end{aligned}$$

Part B

It is obvious that if $\forall c$, $\mathbf{p}(c, \tau_\mathbf{f}) = 0$ or $\mathbf{b}(c, \mathbf{f}) = 0$, we have $P(\mathbf{f}) = \sum_{i=1}^{n} \mathbf{p}(c_i, \tau_\mathbf{f})\mathbf{b}(c_i, \mathbf{f}) = 0$.

Now we prove if $P(\mathbf{f}) = 0$, then $\forall c$, $\mathbf{p}(c, \tau_\mathbf{f}) = 0$ or $\mathbf{b}(c, \mathbf{f}) = 0$. We use the method of proof by contradiction.

If not, then $(\exists c)$, such that $\mathbf{p}(c, \tau_\mathbf{f}) > 0$ and $\mathbf{b}(c, \mathbf{f}) > 0$. Thus

$$\begin{aligned} P(\mathbf{f}) &= \sum_{i=1}^{n} \mathbf{p}(c_i, \tau_\mathbf{f})\mathbf{b}(c_i, \mathbf{f}) \\ &\geq \mathbf{p}(c, \tau_\mathbf{f})\mathbf{b}(c, \mathbf{f}) \\ &> 0 \end{aligned}$$

Contradiction.

Part C

If $\mathbf{p}(c_o, \tau_\mathbf{f}) = 0$ and $\forall c, c \neq c_o$, $\mathbf{b}(c, \mathbf{f}) = 1$, then

$$\begin{aligned} P(\mathbf{f}) &= \sum_{i=1}^{n} \mathbf{p}(c_i, \tau_\mathbf{f})\mathbf{b}(c_i, \mathbf{f}) \\ &= \sum_{i=1}^{n} \mathbf{p}(c_i, \tau_\mathbf{f}) \\ &= \sum_{i=1}^{n} \mathbf{p}(c_i, \tau_\mathbf{f}) + p(c_o, \tau_\mathbf{f}) \\ &= 1 \end{aligned}$$

Now we prove if $P(\mathbf{f}) = 1$, then $\mathbf{p}(c_o, \tau_\mathbf{f}) = 0$ and $\forall c, c \neq c_o$, $\mathbf{b}(c, \mathbf{f}) = 1$. We use the method of proof by contradiction.

If not, then either $\mathbf{p}(c_o, \tau_\mathbf{f}) > 0$ or $\exists c, c \neq c_o$ such that $\mathbf{b}(c, \mathbf{f}) < 1$.

If $\mathbf{p}(c_o, \tau_\mathbf{f}) > 0$, then

$$\begin{aligned} P(\mathbf{f}) &= \sum_{i=1}^{n} \mathbf{p}(c_i, \tau_\mathbf{f})\mathbf{b}(c_i, \mathbf{f}) \\ &\leq \sum_{i=1}^{n} \mathbf{p}(c_i, \tau_\mathbf{f}) \\ &= 1 - \mathbf{p}(c_o, \tau_\mathbf{f}) \\ &< 1 \end{aligned}$$

Contradiction.

If $\exists c, c \neq c_o$, $\mathbf{b}(c, \mathbf{f}) < 1$,

$$\begin{aligned} P(\mathbf{f}) &= \sum_{i=1}^{n} \mathbf{p}(c_i, \tau_\mathbf{f})\mathbf{b}(c_i, \mathbf{f}) \\ &< \sum_{i=1}^{n} \mathbf{p}(c_i, \tau_\mathbf{f}) \\ &= 1 - \mathbf{p}(c_o, \tau_\mathbf{f}) \\ &\leq 1 \end{aligned}$$

Contradiction.

Thus Part C is also proved.

□

Property 3: *After a failed action* $\mathbf{f}$*, the probability for any cube that is outside the influence range of* $\mathbf{f}$ *is increasing, that is:*

$$\mathbf{p}(c, \tau_{\mathbf{f}+}) \geq \mathbf{p}(c, \tau_\mathbf{f})$$

for $c \bar{\in} \Omega(\mathbf{f})$

Proof

Since $c \bar{\in} \Omega(\mathbf{f})$, we have $\mathbf{b}(c, \mathbf{f}) = 0$.

$$\begin{aligned} \mathbf{p}(c, \tau_{\mathbf{f}+}) &= \frac{\mathbf{p}(c, \tau_\mathbf{f})(1 - \mathbf{b}(c, \mathbf{f}))}{\mathbf{p}(c_o, \tau_\mathbf{f}) + \sum_{j=1}^{n} \mathbf{p}(c_j, \tau_\mathbf{f})(1 - \mathbf{b}(c_j, \mathbf{f}))} \\ &= \mathbf{p}(c, \tau_\mathbf{f}) \frac{(1 - \mathbf{b}(c, \mathbf{f}))}{\mathbf{p}(c_o, \tau_\mathbf{f}) + \sum_{j=1}^{n} \mathbf{p}(c_j, \tau_\mathbf{f})(1 - \mathbf{b}(c_j, \mathbf{f}))} \\ &= \mathbf{p}(c, \tau_\mathbf{f}) \frac{1}{\mathbf{p}(c_o, \tau_\mathbf{f}) + \sum_{j=1}^{n} \mathbf{p}(c_j, \tau_\mathbf{f})(1 - \mathbf{b}(c_j, \mathbf{f}))} \\ &= \mathbf{p}(c, \tau_\mathbf{f}) \frac{1}{\sum_{i=1}^{n,o} \mathbf{p}(c_i) - \sum_{i=1}^{n} \mathbf{p}(c_i)\mathbf{b}(c_i, \mathbf{f}))} \\ &= \mathbf{p}(c, \tau_\mathbf{f}) \frac{1}{1 - P(\mathbf{f})} \end{aligned}$$

Since action $\mathbf{f}$ failed, we have $0 \leq P(\mathbf{f}) < 1$. So $0 < 1 - P(\mathbf{f}) \leq 1$. Thus

$$\begin{aligned} \mathbf{p}(c, \tau_{\mathbf{f}+}) &= \mathbf{p}(c, \tau_{\mathbf{f}}) \frac{1}{1 - P(\mathbf{f})} \\ &\geq \mathbf{p}(c, \tau_{\mathbf{f}}) \end{aligned}$$

The equality holds when $P(\mathbf{f}) = 0$, this means $\forall c$, $\mathbf{p}(c, \tau_{\mathbf{f}}) \times \mathbf{b}(c, \mathbf{f}) = 0$. So, for each c, either $\mathbf{p}(c, \tau_{\mathbf{f}}) = 0$ or $\mathbf{b}(c, \mathbf{f}) = 0$.

$\square$

Property 4 *After a failed action* $\mathbf{f}$*, the sum of the probabilities of any cube that is outside the influence range of* $\mathbf{f}$ *is increasing, that is:*

$$\sum_{c \bar{\in} \Omega(\mathbf{f})} \mathbf{p}(c, \tau_{\mathbf{f}+}) \geq \sum_{c \bar{\in} \Omega(\mathbf{f})} \mathbf{p}(c, \tau_{\mathbf{f}})$$

Proof

From Property 3, we have $\forall c \bar{\in} \Omega(\mathbf{f})$, $\mathbf{p}(c, \tau_{\mathbf{f}+}) \geq \mathbf{p}(c, \tau_{\mathbf{f}})$. Thus, summing up the probabilities of all the cubes that are outside the influence range of action $\mathbf{f}$, we get

$$\sum_{c \bar{\in} \Omega(\mathbf{f})} \mathbf{p}(c, \tau_{\mathbf{f}}+) \geq \sum_{c \bar{\in} \Omega(\mathbf{f})} \mathbf{p}(c, \tau_{\mathbf{f}})$$

$\square$.

Property 5: *After a failed action* $\mathbf{f}$*, the sum of the probabilities of any cube that is inside the influence range of* $\mathbf{f}$ *is decreasing, that is:*

$$\sum_{c \in \Omega(\mathbf{f})} \mathbf{p}(c, \tau_{\mathbf{f}+}) \leq \sum_{c \in \Omega(\mathbf{f})} \mathbf{p}(c, \tau_{\mathbf{f}})$$

Proof

Since

$$\sum_{c \in \Omega(\mathbf{f})} \mathbf{p}(c, \tau_{\mathbf{f}}) + \sum_{c \bar{\in} \Omega(\mathbf{f})} \mathbf{p}(c, \tau_{\mathbf{f}}) = 1$$

and

$$\sum_{c \in \Omega(\mathbf{f})} \mathbf{p}(c, \tau_{\mathbf{f}+}) + \sum_{c \bar{\in} \Omega(\mathbf{f})} \mathbf{p}(c, \tau_{\mathbf{f}+}) = 1$$

We have

$$
\begin{aligned}
\sum_{c\in\Omega(\mathbf{f})} \mathbf{p}(c,\tau_{\mathbf{f}+}) &= 1 - \sum_{c\bar{\in}\Omega(\mathbf{f})} \mathbf{p}(c,\tau_{\mathbf{f}+}) \\
&< 1 - \sum_{c\bar{\in}\Omega(\mathbf{f})} \mathbf{p}(c,\tau_{\mathbf{f}}) \text{ (from Property 4)} \\
&= \sum_{c\in\Omega(\mathbf{f})} \mathbf{p}(c,\tau_{\mathbf{f}})
\end{aligned}
$$

□

Property 6 *Suppose two cubes c and c^* have the same probability value before an action $\mathbf{f}$ is applied, $\mathbf{p}(c,\tau_{\mathbf{f}}) = \mathbf{p}(c^*,\tau_{\mathbf{f}})$, where c belongs to the influence range of $\mathbf{f}$, $c \in \Omega(\mathbf{f})$, and c^* does not belong to the influence range of $\mathbf{f}$, $c^*\bar{\in}\Omega(\mathbf{f})$. Then after $\mathbf{f}$ is applied and fails, we have*

$$\mathbf{p}(c,\tau_{\mathbf{f}+}) < \mathbf{p}(c^*,\tau_{\mathbf{f}+})$$

Proof

Since c belongs to the influence range of $\mathbf{f}$, we have $0 < \mathbf{b}(c,\mathbf{f}) \leq 1$.

Since c^* does not belong to the influence range of $\mathbf{f}$, we have $\mathbf{b}(c^*,\mathbf{f}) = 0$.

So,

$$
\begin{aligned}
\mathbf{p}(c,\tau_{\mathbf{f}+}) &= \frac{\mathbf{p}(c,\tau_{\mathbf{f}})(1-\mathbf{b}(c,\mathbf{f}))}{\mathbf{p}(c_o,\tau_{\mathbf{f}}) + \sum_{j=1}^{n} \mathbf{p}(c_j,\tau_{\mathbf{f}})(1-\mathbf{b}(c_j,\mathbf{f}))}, \\
&< \frac{\mathbf{p}(c,\tau_{\mathbf{f}})}{\mathbf{p}(c_o,\tau_{\mathbf{f}}) + \sum_{j=1}^{n} \mathbf{p}(c_j,\tau_{\mathbf{f}})(1-\mathbf{b}(c_j,\mathbf{f}))}, \\
&= \frac{\mathbf{p}(c^*,\tau_{\mathbf{f}})}{\mathbf{p}(c_o,\tau_{\mathbf{f}}) + \sum_{j=1}^{n} \mathbf{p}(c_j,\tau_{\mathbf{f}})(1-\mathbf{b}(c_j,\mathbf{f}))}, \\
&= \mathbf{p}(c^*,\tau_{\mathbf{f}+})
\end{aligned}
$$

□

Property 7: *For allocation $\mathbf{F} = \{\mathbf{f}_1,\ldots,\mathbf{f}_q \mid \mathbf{f}_i \in \mathbf{O}_{\mathbf{\Omega}}\}$, we have*

$$\mathbf{p}^{[k]}(c) = \frac{\mathbf{p}^{[0]}(c)g(c,\mathbf{f}_1)g(c,\mathbf{f}_2)\ldots g(c,\mathbf{f}_k)}{(1-P(\mathbf{f}_1))(1-P(\mathbf{f}_2))\ldots,(1-P(\mathbf{f}_k))} \qquad k = 1,\ldots,q \tag{A.3}$$

where

$$g(c,\mathbf{f}) = \begin{cases} 1-\mathbf{b}(c,\mathbf{f}) & \textit{if } c \in \Omega(\mathbf{f}) \\ 1 & \textit{otherwise} \end{cases}$$

Proof

(1) When $k = 1$, from update rule 2.9 we have

$$\begin{aligned}
\mathbf{p}^{[1]}(c) &= \frac{\mathbf{p}^{[0]}(c)(1-\mathbf{b}(c,\mathbf{f}_1))}{\mathbf{p}^{[0]}(c_o)+\sum_{i=1}^{n}\mathbf{p}^{[0]}(c_i)(1-\mathbf{b}(c_i,\mathbf{f}))} \\
&= \frac{\mathbf{p}^{[0]}(c)(1-\mathbf{b}(c,\mathbf{f}_1))}{\sum_{i=1}^{n,o}\mathbf{p}^{[0]}(c_i)-\sum_{i=1}^{n}\mathbf{p}^{[0]}(c_i)\mathbf{b}(c_i,\mathbf{f}))} \\
&= \frac{\mathbf{p}^{[0]}(c)(1-\mathbf{b}(c,\mathbf{f}_1))}{1-P(\mathbf{f}_1)}
\end{aligned}$$

Since $\mathbf{b}(c,\mathbf{f}_1)=0$ when $c \in \mathbf{O}_{\boldsymbol{\Omega}} - \Omega(\mathbf{f}_1)$, we have

$$\mathbf{p}^{[1]}(c) = \frac{\mathbf{p}^{[0]}(c)g(c,\mathbf{f}_1)}{1-P(\mathbf{f}_1)}$$

So, the result is true when $k = 1$.

(2) Suppose when $k = r$, we have

$$\mathbf{p}^{[r]}(c) = \frac{\mathbf{p}^{[0]}(c)g(c,\mathbf{f}_1)g(c,\mathbf{f}_2)\ldots g(c,\mathbf{f}_r)}{(1-P(\mathbf{f}_1))(1-P(\mathbf{f}_2))\ldots,(1-P(\mathbf{f}_r))}$$

(3) When $k = r+1$,

$$\begin{aligned}
\mathbf{p}^{[r+1]}(c) &= \frac{\mathbf{p}^{[r]}(c)(1-\mathbf{b}(c,\mathbf{f}_{r+1}))}{\mathbf{p}^{[r]}(c_o)+\sum_{i=1}^{n}\mathbf{p}^{[r]}(c_i)(1-\mathbf{b}(c_i,\mathbf{f}_{r+1}))} \\
&= \frac{\mathbf{p}^{[r]}(c)(1-\mathbf{b}(c,\mathbf{f}_{r+1}))}{\sum_{i=1}^{n,o}\mathbf{p}^{[r]}(c_i)-\sum_{i=1}^{n}\mathbf{p}^{[r]}(c_i)\mathbf{b}(c_i,\mathbf{f}_{r+1}))} \\
&= \frac{\mathbf{p}^{[r]}(c)(1-\mathbf{b}(c,\mathbf{f}_{r+1}))}{1-P(\mathbf{f}_{r+1})} \\
&= \mathbf{p}^{[r]}(c)\frac{(1-\mathbf{b}(c,\mathbf{f}_{r+1}))}{1-P(\mathbf{f}_{r+1})} \\
&= \frac{\mathbf{p}^{[0]}(c)g(c,\mathbf{f}_1)g(c,\mathbf{f}_2)\ldots g(c,\mathbf{f}_r)}{(1-P(\mathbf{f}_1))(1-P(\mathbf{f}_2))\ldots(1-P(\mathbf{f}_r))}\frac{(1-\mathbf{b}(c,\mathbf{f}_{r+1}))}{(1-P(\mathbf{f}_{r+1}))} \\
&= \frac{\mathbf{p}^{[0]}(c)g(c,\mathbf{f}_1)g(c,\mathbf{f}_2)\ldots g(c,\mathbf{f}_r)(1-\mathbf{b}(c,\mathbf{f}_{r+1}))}{(1-P(\mathbf{f}_1))(1-P(\mathbf{f}_2))\ldots(1-P(\mathbf{f}_r))(1-P(\mathbf{f}_{r+1}))}
\end{aligned}$$

Since $\mathbf{b}(c,\mathbf{f}_{r+1})=0$ when $c \in \mathbf{O}_{\boldsymbol{\Omega}} - \Omega(\mathbf{f}_{r+1})$, we have

$$\mathbf{p}^{[r+1]}(c) = \frac{\mathbf{p}^{[0]}(c)g(c,\mathbf{f}_1)g(c,\mathbf{f}_2)\ldots g(c,\mathbf{f}_r)g(c,\mathbf{f}_{r+1})}{(1-P(\mathbf{f}_1))(1-P(\mathbf{f}_2))\ldots(1-P(\mathbf{f}_r))(1-P(\mathbf{f}_{r+1}))}$$

So, the property is true when $k = r+1$. Thus the conclusion is true for $k \geq 1$.

$\square$.

Definition 2 *For any group of actions* $\{\mathbf{f}_1,\ldots,\mathbf{f}_k\}$*, we define:*

$$\begin{aligned}
Q_k^{[0]}(\mathbf{f}_1\ldots\mathbf{f}_k) &= \Omega(\mathbf{f}_1)\bigcap\ldots\bigcap\Omega(\mathbf{f}_k)\\
&\ldots\\
Q_k^{[r]}(\mathbf{f}_1\ldots\overline{\mathbf{f}_{i_1}}\ldots\overline{\mathbf{f}_{i_2}}\ldots\overline{\mathbf{f}_{i_r}}\ldots\mathbf{f}_k) &= \Omega(\mathbf{f}_1)\bigcap\ldots\bigcap\overline{\Omega(\mathbf{f}_{i_1})}\bigcap\ldots\bigcap\overline{\Omega(\mathbf{f}_{i_2})}\bigcap\ldots\bigcap\overline{\Omega(\mathbf{f}_{i_r})}\bigcap\ldots\bigcap\Omega(\mathbf{f}_k);\\
&\ldots\\
Q_k^{[k]}(\overline{\mathbf{f}_1\mathbf{f}_2\ldots\mathbf{f}_k}) &= \overline{\Omega(\mathbf{f}_1)}\bigcap\ldots\bigcap\overline{\Omega(\mathbf{f}_k)};
\end{aligned}$$

where $Q_k^{[r]}$ $(r\le k)$ *means that there are* k *sets* $\Omega(\mathbf{f}_1)$, $\Omega(\mathbf{f}_2)$, $\ldots$, $\Omega(\mathbf{f}_k)$ *that are taken into consideration,* r *of which are in complement form.*

For a given k, it is easy to see that the intersection of any two different $Q_k^{[r]}$ is ϕ (Note: this property is used frequently in the proof of Property 8). The unification of all the possible $Q_k^{[r]}$ is Ω, as stated in **Lemma** 1.

Lemma 1 *For the above defined* $Q_k^{[r]}$*, we have:*

$$\bigcup_{r=0}^{k}\Big\{\bigcup_{1\le i_1<i_2<\ldots<i_r\le k} Q_k^{[r]}(\mathbf{f}_1\ldots\overline{\mathbf{f}_{i_1}}\ldots\overline{\mathbf{f}_{i_2}}\ldots\overline{\mathbf{f}_{i_r}}\ldots\mathbf{f}_k)\Big\}=\Omega$$

Proof:

(1) When $k=1$ we have

$$Q_1^{[0]}(\mathbf{f}_1)\bigcup Q_1^{[1]}(\overline{\mathbf{f}_1})=\Omega(\mathbf{f}_1)\bigcup\overline{\Omega(\mathbf{f}_1)}=\Omega$$

(2) Suppose when $k=v$,

$$\bigcup_{r=0}^{v}\Big\{\bigcup_{1\le i_1<i_2<\ldots<i_r\le v} Q_v^{[r]}(\mathbf{f}_1\ldots\overline{\mathbf{f}_{i_1}}\ldots\overline{\mathbf{f}_{i_2}}\ldots\overline{\mathbf{f}_{i_r}}\ldots\mathbf{f}_v)\Big\}=\Omega$$

(3) then when $k=v+1$, we have

$$\begin{aligned}
\bigcup_{r=0}^{v+1}\Big\{\bigcup_{1\le i_1<i_2<\ldots<i_r\le v+1} Q_{v+1}^{[r]}(\ldots)\Big\} &= \Big\{\bigcup_{r=0}^{v+1}\Big(\bigcup_{1\le i_1<i_2<\ldots<i_r\le v+1} Q_{v+1}^{[r]}(\ldots\mathbf{f}_{v+1})\Big)\Big\}\\
&\quad\bigcup\Big\{\bigcup_{r=0}^{v+1}\Big(\bigcup_{1\le i_1<i_2<\ldots<i_r\le v+1} Q_{v+1}^{[r]}(\ldots\overline{\mathbf{f}_{v+1}})\Big)\Big\}\\
&= \Big\{\Omega(\mathbf{f}_{v+1})\bigcap\Big\{\bigcup_{r=0}^{v+1}\Big(\bigcup_{1\le i_1<i_2<\ldots<i_r\le v+1} Q_{v+1}^{[r]}(\ldots\mathbf{f}_v)\Big)\Big\}\Big\}\\
&\quad\bigcup\Big\{\overline{\Omega(\mathbf{f}_{v+1})}\bigcap\Big\{\bigcup_{r=0}^{v+1}\Big(\bigcup_{1\le i_1<i_2<\ldots<i_r\le v+1} Q_{v+1}^{[r]}(\ldots\mathbf{f}_v)\Big)\Big\}\Big\}\\
&= \Big\{\Omega(\mathbf{f}_{v+1})\bigcap\Big\{\bigcup_{r=0}^{v}\Big(\bigcup_{1\le i_1<i_2<\ldots<i_r\le v} Q_v^{[r]}(\ldots)\Big)\Big\}\Big\}\\
&\quad\bigcup\Big\{\overline{\Omega(\mathbf{f}_{v+1})}\bigcap\Big\{\bigcup_{r=0}^{v}\Big(\bigcup_{1\le i_1<i_2<\ldots<i_r\le v} Q_v^{[r]}(\ldots)\Big)\Big\}\Big\}
\end{aligned}$$

$$
\begin{aligned}
&= \left\{\Omega(\mathbf{f}_{v+1}) \bigcap \Omega\right\} \bigcup \left\{\overline{\Omega(\mathbf{f}_{v+1})} \bigcap \Omega\right\} \\
&= \Omega(\mathbf{f}_{v+1}) \bigcup \overline{\Omega(\mathbf{f}_{v+1})} \\
&= \Omega
\end{aligned}
$$

So, the property is true.

□

Property 8. *For allocation* $\mathbf{F} = \{\mathbf{f}_1 \dots \mathbf{f}_q\}$, *we have*

$$
\begin{aligned}
P(\mathbf{f}_k) \;=\; & \frac{1}{(1-P(\mathbf{f}_1))\dots(1-P(\mathbf{f}_{k-1}))}\Bigg\{P^{[0]}(\mathbf{f}_k) \\
& +(-1)^1 \sum_{i_1=1}^{k-1}\Big(\sum_{c\in\Omega(\mathbf{f}_{i_1})\bigcap\Omega(\mathbf{f}_k)} \mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_{i_1})\mathbf{b}(c,\mathbf{f}_k)\Big) \\
& +(-1)^2 \sum_{1\le i_1<i_2\le k-1}\Big(\sum_{c\in\Omega(\mathbf{f}_{i_1})\bigcap\Omega(\mathbf{f}_{i_2})\bigcap\Omega(\mathbf{f}_k)} \mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_{i_1})\mathbf{b}(c,\mathbf{f}_{i_2})\mathbf{b}(c,\mathbf{f}_k)\Big) \\
& +(-1)^3 \sum_{1\le i_1<i_2<i_3\le k-1}\Big(\sum_{c\in\Omega(\mathbf{f}_{i_1})\bigcap\Omega(\mathbf{f}_{i_2})\bigcap\Omega(\mathbf{f}_{i_3})\bigcap\Omega(\mathbf{f}_k)} \mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_{i_1})\mathbf{b}(c,\mathbf{f}_{i_2})\mathbf{b}(c,\mathbf{f}_{i_3})\mathbf{b}(c,\mathbf{f}_k)\Big) \\
& +\dots \\
& +(-1)^{k-1} \sum_{c\in\Omega(\mathbf{f}_1)\bigcap\dots\bigcap\Omega(\mathbf{f}_{k-1})\bigcap\Omega(\mathbf{f}_k)} \mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_1)\mathbf{b}(c,\mathbf{f}_2)\dots\mathbf{b}(c,\mathbf{f}_{k-1})\mathbf{b}(c,\mathbf{f}_k)\Bigg\}
\end{aligned}
$$

for $k = 2, \dots, q$.

Proof:

$$
\begin{aligned}
P(\mathbf{f}_k) &= \sum_{j=1}^{n} \mathbf{p}^{[k-1]}(c_j)\mathbf{b}(c_j,\mathbf{f_k}) \\
&= \sum_{c\in\Omega(\mathbf{f}_k)} \mathbf{p}^{[k-1]}(c)\mathbf{b}(c,\mathbf{f_k}) \\
&= \sum_{c\in\Omega(\mathbf{f}_k)\bigcap\Omega} \mathbf{p}^{[k-1]}(c)\mathbf{b}(c,\mathbf{f_k}) \\
&= \sum_{c\in\bigcup_{r=0}^{k-1}\left\{\bigcup_{1\le i_1<i_2<\dots<i_r\le k-1} Q_{k-1}^{[r]}(\mathbf{f}_1\dots\overline{\mathbf{f}_{i_1}}\dots\overline{\mathbf{f}_{i_2}}\dots\overline{\mathbf{f}_{i_r}}\dots\mathbf{f}_{k-1})\bigcap\Omega(\mathbf{f}_k)\right\}} \mathbf{p}^{[k-1]}(c)\mathbf{b}(c,\mathbf{f_k})
\end{aligned}
$$

Let

$$
A(r) = \bigcup_{1\le i_1<i_2<\dots<i_r\le k-1} Q_{k-1}^{[r]}(\mathbf{f}_1\dots\overline{\mathbf{f}_{i_1}}\dots\overline{\mathbf{f}_{i_2}}\dots\overline{\mathbf{f}_{i_r}}\dots\mathbf{f}_{k-1})\bigcap\Omega(\mathbf{f}_k)
$$

Then, we have $A(i) \bigcap A(j) = \phi$ if $i \neq j$. So

$$
\begin{aligned}
P(\mathbf{f}_k) &= \sum_{r=0}^{k-1} \Big\{ \sum_{c \in \Big\{ \bigcup_{1 \le i_1 < i_2 < \ldots < i_r \le k-1} Q_{k-1}^{[r]}(\mathbf{f}_1 \ldots \overline{\mathbf{f}_{i_1}} \ldots \overline{\mathbf{f}_{i_2}} \ldots \overline{\mathbf{f}_{i_r}} \ldots \mathbf{f}_{k-1}) \bigcap \Omega(\mathbf{f}_k) \Big\}} \mathbf{p}^{[k-1]}(c)\mathbf{b}(c, \mathbf{f_k}) \Big\} \\
&= \frac{1}{(1-P(\mathbf{f}_1))(1-P(\mathbf{f}_2))\ldots,(1-P(\mathbf{f}_{k-1}))} \sum_{r=0}^{k-1} \Big\{ \\
&\sum_{c \in \Big\{ \bigcup_{1 \le i_1 < i_2 < \ldots < i_r \le k-1} Q_{k-1}^{[r]}(\mathbf{f}_1 \ldots \overline{\mathbf{f}_{i_1}} \ldots \overline{\mathbf{f}_{i_2}} \ldots \overline{\mathbf{f}_{i_r}} \ldots \mathbf{f}_{k-1}) \bigcap \Omega(\mathbf{f}_k) \Big\}} \mathbf{p}^{[0]}(c) g(c, \mathbf{f}_1) g(c, \mathbf{f}_2) \ldots g(c, \mathbf{f}_{k-1}) \mathbf{b}(c, \mathbf{f_k}) \Big\} \\
&= \frac{1}{(1-P(\mathbf{f}_1))(1-P(\mathbf{f}_2))\ldots,(1-P(\mathbf{f}_{k-1}))} \sum_{r=0}^{k-1} T_r \qquad \text{(A.4)}
\end{aligned}
$$

Where

$$
\begin{aligned}
T_0 &= \sum_{c \in \Big\{ Q_{k-1}^{[0]}(\mathbf{f}_1 \ldots \mathbf{f}_{k-1}) \bigcap \Omega(\mathbf{f}_k) \Big\}} \mathbf{p}^{[0]}(c) g(c, \mathbf{f}_1) g(c, \mathbf{f}_2) \ldots g(c, \mathbf{f}_{k-1}) \mathbf{b}(c, \mathbf{f_k}) \\
&= \sum_{c \in \Big\{ Q_{k-1}^{[0]}(\mathbf{f}_1 \ldots \mathbf{f}_{k-1}) \bigcap \Omega(\mathbf{f}_k) \Big\}} \mathbf{p}^{[0]}(c)[1-\mathbf{b}(c, \mathbf{f}_1)][1-\mathbf{b}(c, \mathbf{f}_2)] \ldots [1-\mathbf{b}(c, \mathbf{f}_{k-1})] \mathbf{b}(c, \mathbf{f_k})
\end{aligned}
$$

$$
\begin{aligned}
T_1 &= \sum_{c \in \Big\{ \bigcup_{1 \le i_1 \le k-1} Q_{k-1}^{[1]}(\mathbf{f}_1 \ldots \overline{\mathbf{f}_{i_1}} \ldots \mathbf{f}_{k-1}) \bigcap \Omega(\mathbf{f}_k) \Big\}} \mathbf{p}^{[0]}(c) g(c, \mathbf{f}_1) g(c, \mathbf{f}_2) \ldots g(c, \mathbf{f}_{k-1}) \mathbf{b}(c, \mathbf{f_k}) \\
&= \sum_{i_1=1}^{i_1=k-1} \Big\{ \sum_{c \in \Big\{ Q_{k-1}^{[1]}(\mathbf{f}_1 \ldots \overline{\mathbf{f}_{i_1}} \ldots \mathbf{f}_{k-1}) \bigcap \Omega(\mathbf{f}_k) \Big\}} \mathbf{p}^{[0]}(c) g(c, \mathbf{f}_1) g(c, \mathbf{f}_2) \ldots g(c, \mathbf{f}_{k-1}) \mathbf{b}(c, \mathbf{f_k}) \Big\} \\
&= \sum_{i_1=1}^{i_1=k-1} \Big\{ \sum_{c \in \Big\{ Q_{k-1}^{[1]}(\mathbf{f}_1 \ldots \overline{\mathbf{f}_{i_1}} \ldots \mathbf{f}_{k-1}) \bigcap \Omega(\mathbf{f}_k) \Big\}} \\
&\quad \mathbf{p}^{[0]}(c)[1-\mathbf{b}(c, \mathbf{f}_1)] \ldots [1-\mathbf{b}(c, \mathbf{f}_{i_1-1})][1-\mathbf{b}(c, \mathbf{f}_{i_1+1})] \ldots [1-\mathbf{b}(c, \mathbf{f}_{k-1})] \mathbf{b}(c, \mathbf{f_k}) \Big\}
\end{aligned}
$$

Note: $\Big\{ Q_{k-1}^{[1]}(\mathbf{f}_1 \ldots \overline{\mathbf{f}_i} \ldots \mathbf{f}_{k-1}) \bigcap \Omega(\mathbf{f}_k) \Big\} \bigcap \Big\{ Q_{k-1}^{[1]}(\mathbf{f}_1 \ldots \overline{\mathbf{f}_j} \ldots \mathbf{f}_{k-1}) \bigcap \Omega(\mathbf{f}_k) \Big\} = \phi$ if $i \neq j$. So the above transformation is true.

$\vdots$

$$
T_r = \sum_{c \in \Big\{ \bigcup_{1 \le i_1 < i_2 < \ldots < i_r \le k-1} Q_{k-1}^{[r]}(\mathbf{f}_1 \ldots \overline{\mathbf{f}_{i_1}} \ldots \overline{\mathbf{f}_{i_2}} \ldots \overline{\mathbf{f}_{i_r}} \ldots \mathbf{f}_{k-1}) \bigcap \Omega(\mathbf{f}_k) \Big\}} \mathbf{p}^{[0]}(c) g(c, \mathbf{f}_1) g(c, \mathbf{f}_2) \ldots g(c, \mathbf{f}_{k-1}) \mathbf{b}(c, \mathbf{f_k})
$$

$$= \sum_{1 \le i_1 < \ldots < i_r \le k-1} \left\{ \sum_{c \in \left\{ Q^{[r]}_{k-1}(\mathbf{f}_1 \ldots \overline{\mathbf{f}_{i_1}} \ldots \overline{\mathbf{f}_{i_2}} \ldots \overline{\mathbf{f}_{i_r}} \ldots \mathbf{f}_{k-1}) \bigcap \Omega(\mathbf{f}_k) \right\}} \mathbf{p}^{[0]}(c) g(c, \mathbf{f}_1) g(c, \mathbf{f}_2) \ldots g(c, \mathbf{f}_{k-1}) \mathbf{b}(c, \mathbf{f_k}) \right\}$$

$$= \sum_{1 \le i_1 < \ldots < i_r \le k-1} \left\{ \sum_{c \in \left\{ Q^{[r]}_{k-1}(\mathbf{f}_1 \ldots \overline{\mathbf{f}_{i_1}} \ldots \overline{\mathbf{f}_{i_2}} \ldots \overline{\mathbf{f}_{i_r}} \ldots \mathbf{f}_{k-1}) \bigcap \Omega(\mathbf{f}_k) \right\}} \mathbf{p}^{[0]}(c) \right.$$
$$[1 - \mathbf{b}(c, \mathbf{f}_1)] \ldots [1 - \mathbf{b}(c, \mathbf{f}_{i_1 - 1})][1 - \mathbf{b}(c, \mathbf{f}_{i_1+1})] \ldots [1 - \mathbf{b}(c, \mathbf{f}_{i_2-1})][1 - \mathbf{b}(c, \mathbf{f}_{i_2+1})] \ldots$$
$$\left. [1 - \mathbf{b}(c, \mathbf{f}_{i_r-1})][1 - \mathbf{b}(c, \mathbf{f}_{i_r+1})] \ldots [1 - \mathbf{b}(c, \mathbf{f}_{k-1})] \mathbf{b}(c, \mathbf{f_k}) \right\}$$

Note: when $\{i_1, \ldots, i_r\} \neq \{i_1^{'}, \ldots, i_r^{'}\}$

$$\left\{ Q^{[r]}_{k-1}(\mathbf{f}_1 \ldots \overline{\mathbf{f}_{i_1}} \ldots \overline{\mathbf{f}_{i_2}} \ldots \overline{\mathbf{f}_{i_r}} \ldots \mathbf{f}_{k-1}) \bigcap \Omega(\mathbf{f}_k) \right\} \bigcap \left\{ Q^{[r]}_{k-1}(\mathbf{f}_1 \ldots \overline{\mathbf{f}_{i_1^{'}}} \ldots \overline{\mathbf{f}_{i_2^{'}}} \ldots \overline{\mathbf{f}_{i_r^{'}}} \ldots \mathbf{f}_{k-1}) \bigcap \Omega(\mathbf{f}_k) \right\} = \phi$$

So, the above is true.

$\vdots$

$$\begin{aligned} T_{k-1} &= \sum_{c \in \left\{ Q^{[k-1]}_{k-1}(\overline{\mathbf{f}_1} \ldots \overline{\mathbf{f}_{k-1}}) \bigcap \Omega(\mathbf{f}_k) \right\}} \mathbf{p}^{[0]}(c) g(c, \mathbf{f}_1) g(c, \mathbf{f}_2) \ldots g(c, \mathbf{f}_{k-1}) \mathbf{b}(c, \mathbf{f_k}) \\ &= \sum_{c \in \left\{ Q^{[k-1]}_{k-1}(\overline{\mathbf{f}_1} \ldots \overline{\mathbf{f}_{k-1}}) \bigcap \Omega(\mathbf{f}_k) \right\}} \mathbf{p}^{[0]}(c) \mathbf{b}(c, \mathbf{f_k}) \end{aligned}$$

It is easy to show that only terms of the form $\mathbf{p}^{[0]}(c)[\mathbf{b}(c, \mathbf{f_{i_1}}) \ldots \mathbf{b}(c, \mathbf{f_{i_r}})]\mathbf{b}(c, \mathbf{f_k})$ (where $0 \le r \le k-1$) appear in the sum $\sum_{i=1}^{k-1} T_i$. We will now find the final representation for these forms.

When $r = 0$, the terms concerned with $\mathbf{p}^{[0]}(c)\mathbf{b}(c, \mathbf{f_k})$ are

$$\begin{aligned} & \sum_{c \in \left\{ Q^{[0]}_{k-1}(\mathbf{f}_1 \ldots \mathbf{f}_{k-1}) \bigcap \Omega(\mathbf{f}_k) \right\}} \mathbf{p}^{[0]}(c) \mathbf{b}(c, \mathbf{f_k}) \\ & + \sum_{c \in \left\{ \bigcup_{1 \le i_1 \le k-1} Q^{[1]}_{k-1}(\mathbf{f}_1 \ldots \overline{\mathbf{f}_{i_1}} \ldots \mathbf{f}_{k-1}) \bigcap \Omega(\mathbf{f}_k) \right\}} \mathbf{p}^{[0]}(c) \mathbf{b}(c, \mathbf{f_k}) \\ & + \ldots \\ & + \sum_{c \in \left\{ \bigcup_{1 \le i_1 < i_2 < \ldots < i_r \le k-1} Q^{[r]}_{k-1}(\mathbf{f}_1 \ldots \overline{\mathbf{f}_{i_1}} \ldots \overline{\mathbf{f}_{i_2}} \ldots \overline{\mathbf{f}_{i_r}} \ldots \mathbf{f}_{k-1}) \bigcap \Omega(\mathbf{f}_{k-1}) \right\}} \mathbf{p}^{[0]}(c) \mathbf{b}(c, \mathbf{f_k}) \Big\} \end{aligned}$$

$$
\begin{aligned}
&+\dots \\
&+\sum_{c\in\left\{Q_{k-1}^{[k-1]}(\overline{\mathbf{f}_1}\dots\overline{\mathbf{f}_{k-1}})\bigcap\Omega(\mathbf{f}_k)\right\}} \mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f_k}) \\
&=\sum_{c\in\Omega\bigcap\Omega(\mathbf{f}_k)} \mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f_k}) \\
&=\sum_{c\in\Omega(\mathbf{f}_k)} \mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f_k}) \\
&=P^{[0]}(\mathbf{f}_k)
\end{aligned}
$$

When $r=1$, we consider the terms concerned with $\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f_{i_1}})\mathbf{b}(c,\mathbf{f_k})$. As an illustration, we only calculate $\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f_1})\mathbf{b}(c,\mathbf{f_k})$. Others are quite similar.

The part contributed by T_0 is

$$
-\sum_{c\in\left\{Q_{k-1}^{[0]}(\mathbf{f}_1\dots\mathbf{f}_{k-1})\bigcap\Omega(\mathbf{f}_k)\right\}} \mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f_1})\mathbf{b}(c,\mathbf{f_k})
$$

The part contributed by T_1 is

$$
-\sum_{c\in\left\{\bigcup_{2\le i_1\le k-1} Q_{k-1}^{[1]}(\mathbf{f}_1\dots\overline{\mathbf{f}_{i_1}}\dots\mathbf{f}_{k-1})\bigcap\Omega(\mathbf{f}_k)\right\}} \mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f_1})\mathbf{b}(c,\mathbf{f_k})
$$

...

The part contributed by T_r is

$$
-\sum_{c\in\left\{\bigcup_{2\le i_1< i_2<\dots< i_r\le k-1} Q_{k-1}^{[r]}(\mathbf{f}_1\dots\overline{\mathbf{f}_{i_1}}\dots\overline{\mathbf{f}_{i_2}}\dots\overline{\mathbf{f}_{i_r}}\dots\mathbf{f}_{k-1})\bigcap\Omega(\mathbf{f}_k)\right\}} \mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f_1})\mathbf{b}(c,\mathbf{f_k})\Big\}
$$

...

The part contributed by T_{k-1} is 0.

The sum of the total contribution is

$$
\begin{aligned}
&-\sum_{c\in\left\{Q_{k-1}^{[0]}(\mathbf{f}_1\dots\mathbf{f}_{k-1})\bigcap\Omega(\mathbf{f}_k)\right\}} \mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f_1})\mathbf{b}(c,\mathbf{f_k}) \\
&-\sum_{c\in\left\{\bigcup_{2\le i_1\le k-1} Q_{k-1}^{[1]}(\mathbf{f}_1\dots\overline{\mathbf{f}_{i_1}}\dots\mathbf{f}_{k-1})\bigcap\Omega(\mathbf{f}_k)\right\}} \mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f_1})\mathbf{b}(c,\mathbf{f_k}) \\
&-\dots \\
&-\sum_{c\in\left\{\bigcup_{2\le i_1< i_2<\dots< i_r\le k-1} Q_q^{[r]}(\mathbf{f}_1\dots\overline{\mathbf{f}_{i_1}}\dots\overline{\mathbf{f}_{i_2}}\dots\overline{\mathbf{f}_{i_r}}\dots\mathbf{f}_{k-1})\bigcap\Omega(\mathbf{f}_k)\right\}} \mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f_1})\mathbf{b}(c,\mathbf{f_k})\Big\} \\
&-\sum_{c\in\left\{Q_{k-1}^{[k-2]}(\mathbf{f}_1\overline{\mathbf{f}_2}\dots\overline{\mathbf{f}_{k-1}})\bigcap\Omega(\mathbf{f}_k)\right\}} \mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f_1})\mathbf{b}(c,\mathbf{f_k}) \\
&=-\Big(\sum_{c\in\Omega(\mathbf{f}_1)\bigcap\Omega(\mathbf{f}_k)} \mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_1)\mathbf{b}(c,\mathbf{f}_k)\Big) \qquad (A.5)
\end{aligned}
$$

In general, lets consider the term for $\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f_{i_1}})\ldots\mathbf{b}(c,\mathbf{f_{i_r}})\mathbf{b}(c,\mathbf{f_k})$. We only calculate $\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f_1})\ldots\mathbf{b}(c,\mathbf{f_r})\mathbf{b}(c,\mathbf{f_k})$, others are similar.

The part contributed by T_0 is

$$(-1)^r \sum_{c\in\left\{Q^{[0]}_{k-1}(\mathbf{f}_1\ldots\mathbf{f}_r\mathbf{f}_{r+1}\ldots\mathbf{f}_{k-1})\bigcap\Omega(\mathbf{f}_k)\right\}} \left\{\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f_1})\ldots\mathbf{b}(c,\mathbf{f_r})\mathbf{b}(c,\mathbf{f_k})\right\}$$

The part contributed by T_1 is

$$(-1)^r \sum_{c\in\left\{\bigcup_{r+1\le i_1\le k-1} Q^{[1]}_{k-1}(\mathbf{f}_1\ldots\overline{\mathbf{f}_{i_1}}\ldots\mathbf{f}_{k-1})\bigcap\Omega(\mathbf{f}_k)\right\}} \left\{\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f_1})\ldots\mathbf{b}(c,\mathbf{f_r})\mathbf{b}(c,\mathbf{f_k})\right\}$$

$\vdots$

The part contributed by T_{k-r-1} is

$$(-1)^r \sum_{c\in\left\{Q^{[k-r-1]}_{k-1}(\mathbf{f}_1\ldots\mathbf{f}_r\overline{\mathbf{f}_{r+1}}\ldots\overline{\mathbf{f}_{k-1}})\bigcap\Omega(\mathbf{f}_k)\right\}} \left\{\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f_1})\ldots\mathbf{b}(c,\mathbf{f_r})\mathbf{b}(c,\mathbf{f_k})\right\}$$

The total contribution is:

$$\begin{aligned}
&(-1)^r \sum_{c\in\left\{Q^{[0]}_{k-1}(\mathbf{f}_1\ldots\mathbf{f}_r\mathbf{f}_{r+1}\ldots\mathbf{f}_{k-1})\bigcap\Omega(\mathbf{f}_k)\right\}} \left\{\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f_1})\ldots\mathbf{b}(c,\mathbf{f_r})\mathbf{b}(c,\mathbf{f_k})\right\}\\
&+\ldots+(-1)^r \sum_{c\in\left\{\bigcup_{r+1\le i_1\le k-1} Q^{[1]}_{k-1}(\mathbf{f}_1\ldots\overline{\mathbf{f}_{i_1}}\ldots\mathbf{f}_{k-1})\bigcap\Omega(\mathbf{f}_k)\right\}} \left\{\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f_1})\ldots\mathbf{b}(c,\mathbf{f_r})\mathbf{b}(c,\mathbf{f_k})\right\}\\
&+\ldots+(-1)^r \sum_{c\in\left\{Q^{[k-r-1]}_{k-1}(\mathbf{f}_1\ldots\mathbf{f}_r\overline{\mathbf{f}_{r+1}}\ldots\overline{\mathbf{f}_{k-1}})\bigcap\Omega(\mathbf{f}_k)\right\}} \left\{\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f_1})\ldots\mathbf{b}(c,\mathbf{f_r})\mathbf{b}(c,\mathbf{f_k})\right\}\\
&=(-1)^r \sum_{c\in\left\{\Omega(\mathbf{f}_1)\bigcap\ldots\bigcap\Omega(\mathbf{f}_r)\bigcap\Omega(\mathbf{f}_k)\right\}} \left\{\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f_1})\ldots\mathbf{b}(c,\mathbf{f_r})\mathbf{b}(c,\mathbf{f_k})\right\}
\end{aligned} \tag{A.6}$$

$\vdots$

Lastly, the form $\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_1)\ldots\mathbf{b}(c,\mathbf{f}_{k-1})\mathbf{b}(c,\mathbf{f}_k)$ is only generated by T_0. Obviously, it is

$$(-1)^{k-1}\sum_{c\in\Omega(\mathbf{f}_1)\ldots\Omega(\mathbf{f}_{k-1})} \mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_1)\mathbf{b}(c,\mathbf{f}_2)\ldots\mathbf{b}(c,\mathbf{f}_{k-1})$$

Thus, by summing everything together, we prove that **Property 8** is true.

□

Note: another way to prove **Property 8** is to use the following two properties:

$$\mathbf{p}^{[k-1]}(c) = \frac{\mathbf{p}^{[0]}(c)[1-\mathbf{b}(c,\mathbf{f}_1)]\ldots[1-\mathbf{b}(c,\mathbf{f}_{k-1})]}{[1-P(\mathbf{f}_1)]\ldots[1-P(\mathbf{f}_{k-1})]}$$

and

$$\sum_{c\in\Omega}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_{i_1})\ldots\mathbf{b}(c,\mathbf{f}_{i_j})\mathbf{b}(c,\mathbf{f}_k) = \sum_{c\in\Omega(\mathbf{f}_{i_1})\bigcap\cdots\bigcap\Omega(\mathbf{f}_{i_j})}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_{i_1})\ldots\mathbf{b}(c,\mathbf{f}_{i_j})\mathbf{b}(c,\mathbf{f}_k).$$

Details are omitted.

Property 9 *For the given allocation* $\mathbf{F} = \{\mathbf{f_1},\ldots,\mathbf{f_q}\}$, *we have*

$$\begin{aligned}
P(\mathbf{F}) &= \sum_{i_1=1}^{q} P^{[0]}(\mathbf{f}_i) \\
&\quad +(-1)^{2+1}\sum_{1\le i_1<i_2\le q}\Big(\sum_{c\in\Omega(\mathbf{f}_{i_1})\bigcap\Omega(\mathbf{f}_{i_2})}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_{i_1})\mathbf{b}(c,\mathbf{f}_{i_2})\Big) \\
&\quad \ldots \\
&\quad +(-1)^{r+1}\sum_{1\le i_1<i_2<\ldots<i_r\le q}\Big(\sum_{c\in\Omega(\mathbf{f}_{i_1})\bigcap\Omega(\mathbf{f}_{i_2})\bigcap\cdots\bigcap\Omega(\mathbf{f}_{i_r})}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_{i_1})\mathbf{b}(c,\mathbf{f}_{i_2})\ldots\mathbf{b}(c,\mathbf{f}_{i_r})\Big) \\
&\quad \ldots \\
&\quad +(-1)^{q+1}\Big(\sum_{c\in\Omega(\mathbf{f}_1)\bigcap\Omega(\mathbf{f}_2)\bigcap\cdots\bigcap\Omega(\mathbf{f}_q)}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_1)\ldots\mathbf{b}(c,\mathbf{f}_q)\Big) \qquad \text{(A.7)}
\end{aligned}$$

Proof:

(1) When $q = 1$, it is true.

(2) Suppose when $q = k$, it is true.

(3) Then when $q = k+1$, we have

$$\begin{aligned}
P[\mathbf{F}] &= \sum_{i=1}^{n}\mathbf{p}(c_i,\tau_{\mathbf{f_1}})\mathbf{b}(c_i,\mathbf{f_1}) \\
&\quad + [1-\sum_{i=1}^{n}\mathbf{p}(c_i,\tau_{\mathbf{f_1}})\mathbf{b}(c_i,\mathbf{f_1})]\times[\sum_{i=1}^{n}\mathbf{p}(c_i,\tau_{\mathbf{f_2}})\mathbf{b}(c_i,\mathbf{f_2})] \\
&\quad + \ldots \\
&\quad + \{\prod_{j=1}^{k}[1-\sum_{i=1}^{n}\mathbf{p}(c_i,\tau_{\mathbf{f_j}})\mathbf{b}(c_i,\mathbf{f_j})]\}\times[\sum_{i=1}^{n}\mathbf{p}(c_i,\tau_{\mathbf{f_k}})\mathbf{b}(c_i,\mathbf{f_{k+1}})] \\
&= P[\mathbf{F_k}] + \{\prod_{j=1}^{k}[1-P(\mathbf{f_j})]\}\times P(\mathbf{f_{k+1}}) \qquad \text{(A.8)}
\end{aligned}$$

From Property 8, we have

$$
\begin{aligned}
P(\mathbf{f_{k+1}}) &= \frac{1}{\prod_{j=1}^{k}[1-P(\mathbf{f_j})]}\Bigg\{P^{[0]}(\mathbf{f}_{k+1}) \\
&+(-1)^1\sum_{i_1=1}^{k}\Big(\sum_{c\in\Omega(\mathbf{f}_{i_1})\bigcap\Omega(\mathbf{f}_{k+1})}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_{i_1})\mathbf{b}(c,\mathbf{f}_{k+1})\Big) \\
&+(-1)^2\sum_{1\le i_1<i_2\le k}\Big(\sum_{c\in\Omega(\mathbf{f}_{i_1})\bigcap\Omega(\mathbf{f}_{i_2})\bigcap\Omega(\mathbf{f}_{k+1})}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_{i_1})\mathbf{b}(c,\mathbf{f}_{i_2})\mathbf{b}(c,\mathbf{f}_{k+1})\Big) \\
&+(-1)^3\sum_{1\le i_1<i_2<i_3\le k}\Big(\sum_{c\in\Omega(\mathbf{f}_{i_1})\bigcap\Omega(\mathbf{f}_{i_2})\bigcap\Omega(\mathbf{f}_{i_3})\bigcap\Omega(\mathbf{f}_{k+1})}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_{i_1})\mathbf{b}(c,\mathbf{f}_{i_2})\mathbf{b}(c,\mathbf{f}_{i_3})\mathbf{b}(c,\mathbf{f}_{k+1})\Big) \\
&+\dots \\
&+(-1)^k\sum_{c\in\Omega(\mathbf{f}_1)\bigcap\dots\bigcap\Omega(\mathbf{f}_k)\bigcap\Omega(\mathbf{f}_{k+1})}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_1)\mathbf{b}(c,\mathbf{f}_2)\dots\mathbf{b}(c,\mathbf{f}_k)\mathbf{b}(c,\mathbf{f}_{k+1})\Bigg\}
\end{aligned}
$$

So

$$
\begin{aligned}
P[\mathbf{F}] &= P[\mathbf{F_k}] \\
&+\Bigg\{P^{[0]}(\mathbf{f}_{k+1}) \\
&+(-1)^1\sum_{i_1=1}^{k}\Big(\sum_{c\in\Omega(\mathbf{f}_{i_1})\bigcap\Omega(\mathbf{f}_{k+1})}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_{i_1})\mathbf{b}(c,\mathbf{f}_{k+1})\Big) \\
&+(-1)^2\sum_{1\le i_1<i_2\le k}\Big(\sum_{c\in\Omega(\mathbf{f}_{i_1})\bigcap\Omega(\mathbf{f}_{i_2})\bigcap\Omega(\mathbf{f}_{k+1})}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_{i_1})\mathbf{b}(c,\mathbf{f}_{i_2})\mathbf{b}(c,\mathbf{f}_{k+1})\Big) \\
&+(-1)^3\sum_{1\le i_1<i_2<i_3\le k}\Big(\sum_{c\in\Omega(\mathbf{f}_{i_1})\bigcap\Omega(\mathbf{f}_{i_2})\bigcap\Omega(\mathbf{f}_{i_3})\bigcap\Omega(\mathbf{f}_{k+1})}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_{i_1})\mathbf{b}(c,\mathbf{f}_{i_2})\mathbf{b}(c,\mathbf{f}_{i_3})\mathbf{b}(c,\mathbf{f}_{k+1})\Big) \\
&+\dots \\
&+(-1)^k\sum_{c\in\Omega(\mathbf{f}_1)\bigcap\dots\bigcap\Omega(\mathbf{f}_k)\bigcap\Omega(\mathbf{f}_{k+1})}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_1)\mathbf{b}(c,\mathbf{f}_2)\dots\mathbf{b}(c,\mathbf{f}_k)\mathbf{b}(c,\mathbf{f}_{k+1})\Bigg\} \\
&= \Bigg\{\sum_{i_1=1}^{k}P^{[0]}(\mathbf{f}_i) \\
&+(-1)^{2+1}\sum_{1\le i_1<i_2\le k}\Big(\sum_{c\in\Omega(\mathbf{f}_{i_1})\bigcap\Omega(\mathbf{f}_{i_2})}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_{i_1})\mathbf{b}(c,\mathbf{f}_{i_2})\Big) \\
&\dots \\
&+(-1)^{r+1}\sum_{1\le i_1<i_2<\dots<i_r\le k}\Big(\sum_{c\in\Omega(\mathbf{f}_{i_1})\bigcap\Omega(\mathbf{f}_{i_2})\bigcap\dots\bigcap\Omega(\mathbf{f}_{i_r})}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_{i_1})\mathbf{b}(c,\mathbf{f}_{i_2})\dots\mathbf{b}(c,\mathbf{f}_{i_r})\Big) \\
&\dots \\
&+(-1)^{k+1}\Big(\sum_{c\in\Omega(\mathbf{f}_1)\bigcap\Omega(\mathbf{f}_2)\bigcap\dots\bigcap\Omega(\mathbf{f}_k)}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_1)\dots\mathbf{b}(c,\mathbf{f}_k)\Big)\Bigg\}
\end{aligned}
$$

$$
\begin{aligned}
&+\\
&\Big\{P^{[0]}(\mathbf{f}_{k+1})\\
&+(-1)^1\sum_{i_1=1}^{k}\Big(\sum_{c\in\Omega(\mathbf{f}_{i_1})\bigcap\Omega(\mathbf{f}_{k+1})}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_{i_1})\mathbf{b}(c,\mathbf{f}_{k+1})\Big)\\
&+(-1)^2\sum_{1\le i_1<i_2\le k}\Big(\sum_{c\in\Omega(\mathbf{f}_{i_1})\bigcap\Omega(\mathbf{f}_{i_2})\bigcap\Omega(\mathbf{f}_{k+1})}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_{i_1})\mathbf{b}(c,\mathbf{f}_{i_2})\mathbf{b}(c,\mathbf{f}_{k+1})\Big)\\
&+(-1)^3\sum_{1\le i_1<i_2<i_3\le k}\Big(\sum_{c\in\Omega(\mathbf{f}_{i_1})\bigcap\Omega(\mathbf{f}_{i_2})\bigcap\Omega(\mathbf{f}_{i_3})\bigcap\Omega(\mathbf{f}_{k+1})}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_{i_1})\mathbf{b}(c,\mathbf{f}_{i_2})\mathbf{b}(c,\mathbf{f}_{i_3})\mathbf{b}(c,\mathbf{f}_{k+1})\Big)\\
&+\dots\\
&+(-1)^k\sum_{c\in\Omega(\mathbf{f}_1)\bigcap\dots\bigcap\Omega(\mathbf{f}_k)\bigcap\Omega(\mathbf{f}_{k+1})}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_1)\mathbf{b}(c,\mathbf{f}_2)\dots\mathbf{b}(c,\mathbf{f}_k)\mathbf{b}(c,\mathbf{f}_{k+1})\Big\}\\
=&\sum_{i_1=1}^{k+1}P^{[0]}(\mathbf{f}_i)\\
&+(-1)^{2+1}\sum_{1\le i_1<i_2\le k+1}\Big(\sum_{c\in\Omega(\mathbf{f}_{i_1})\bigcap\Omega(\mathbf{f}_{i_2})}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_{i_1})\mathbf{b}(c,\mathbf{f}_{i_2})\Big)\\
&\dots\\
&+(-1)^{r+1}\sum_{1\le i_1<i_2<\dots<i_r\le k+1}\Big(\sum_{c\in\Omega(\mathbf{f}_{i_1})\bigcap\Omega(\mathbf{f}_{i_2})\bigcap\dots\bigcap\Omega(\mathbf{f}_{i_r})}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_{i_1})\mathbf{b}(c,\mathbf{f}_{i_2})\dots\mathbf{b}(c,\mathbf{f}_{i_r})\Big)\\
&\dots\\
&+(-1)^{k+1+1}\Big(\sum_{c\in\Omega(\mathbf{f}_1)\bigcap\Omega(\mathbf{f}_2)\bigcap\dots\bigcap\Omega(\mathbf{f}_{k+1})}\mathbf{p}^{[0]}(c)\mathbf{b}(c,\mathbf{f}_1)\dots\mathbf{b}(c,\mathbf{f}_{k+1})\Big) \qquad \text{(A.9)}
\end{aligned}
$$

Thus, it is true when $q = k+1$, and the property is true.

$\square$.

Property 11 *For an allocation $\mathbf{F} = \{\mathbf{f_1}, \dots, \mathbf{f_q}\}$, if we restrict the available actions such that there is no cube belonging to any two action ranges: $\Omega(\mathbf{f}_j)\bigcap\Omega(\mathbf{f}_i) = \phi$, if $i \neq j$, then $P[\mathbf{F}]$ can be calculated by the following:*

$$P(\mathbf{F}) = \sum_{i=1}^{q} P^{[0]}(\mathbf{f}_i)$$

Proof: By using Property 9, this result is easy to prove. Here, we give another method.

$$
\begin{aligned}
P(\mathbf{f}_k) &= \sum_{c\in\Omega(\mathbf{f}_k)}\mathbf{p}^{[k-1]}(c)\mathbf{b}(c,\mathbf{f}_k)\\
&= \sum_{c\in\Omega(\mathbf{f}_k)}\frac{\mathbf{p}^{[0]}(c)}{(1-P(\mathbf{f}_1))(1-P(\mathbf{f}_2))\dots(1-P(\mathbf{f}_{k-1}))}\mathbf{b}(c,\mathbf{f}_k)
\end{aligned}
$$

$$\begin{aligned} & \text{(Note: because } c \text{ does not belong to } \Omega(\mathbf{f}_1), \ldots, \Omega(\mathbf{f}_{k-1})) \\ = \; & \frac{P^{[0]}(\mathbf{f}_k)}{(1-P(\mathbf{f}_1))(1-P(\mathbf{f}_2))\ldots,(1-P(\mathbf{f}_{k-1}))} \end{aligned}$$

Thus,

$$[1-P(\mathbf{f}_1)][1-P(\mathbf{f}_2)]\ldots,[1-P(\mathbf{f}_{k-1})]P(\mathbf{f}_k) = P^{[0]}(\mathbf{f}_k)$$

So,

$$\begin{aligned} P(\mathbf{F}) &= P(\mathbf{f}_1) + [1-P(\mathbf{f}_1)]P(\mathbf{f}_2) + \ldots + [1-P(\mathbf{f}_1)][1-P(\mathbf{f}_2)]\ldots,[1-P(\mathbf{f}_{q-1})]P(\mathbf{f}_q) \\ &= \sum_{i=1}^{q} P^{[0]}(\mathbf{f}_i) \end{aligned} \tag{A.10}$$

□

Property 13 *For a group of actions* $\{\mathbf{f_1}, \ldots, \mathbf{f_q}\}$*, suppose that*

(1) There is no cube belonging to any two action ranges: $\Omega(\mathbf{f}_j) \bigcap \Omega(\mathbf{f}_i) = \phi$*, if* $i \neq j$*;*

(2) $\frac{P^{[0]}(\mathbf{f}_1)}{\mathbf{t}(\mathbf{f}_1)} > \frac{P^{[0]}(\mathbf{f}_2)}{\mathbf{t}(\mathbf{f}_2)} > \ldots > \frac{P^{[0]}(\mathbf{f}_k)}{\mathbf{t}(\mathbf{f}_k)}$.

Then the order of applying these actions by our one step look ahead strategy must be: $\mathbf{f}_1 \to \mathbf{f}_2 \to \ldots \to \mathbf{f}_k$.

Proof:

We prove it by mathematical induction on the number of actions selected.

(A) When $i = 1$, from condition (1) and object function used by the greedy algorithm, it is easy to show that $\mathbf{f}_1$ is selected.

(B) Suppose when $i = r$, the selected actions are $\mathbf{f}_1 \to \mathbf{f}_2 \to \ldots \to \mathbf{f}_r$.

(C) When $i = r + 1$, we need to select the next action. For any action $\mathbf{f}$ that belongs to $\{\mathbf{f}_{r+1}, \mathbf{f}_{r+2}, \ldots, \mathbf{f}_k\}$, we have:

$$\begin{aligned} P(\mathbf{f}) &= \sum_{j=1}^{n} \mathbf{p}^{[r]}(c_j)\mathbf{b}(c_j, \mathbf{f}) \\ &= \sum_{c \in \Omega(\mathbf{f})} \mathbf{p}^{[r]}(c)\mathbf{b}(c, \mathbf{f}) \end{aligned}$$

Since $c \in \Omega(\mathbf{f})$, thus $c \bar{\in} \Omega(\mathbf{f}_1)$, ..., $c \bar{\in} \Omega(\mathbf{f}_r)$, so

$$\begin{aligned} P(\mathbf{f}) &= \sum_{c \in \Omega(\mathbf{f})} \frac{\mathbf{p}^{[0]}(c) g(c, \mathbf{f}_1) g(c, \mathbf{f}_2) \ldots g(c, \mathbf{f}_r)}{(1 - P(\mathbf{f}_1))(1 - P(\mathbf{f}_2)) \ldots, (1 - P(\mathbf{f}_r))} \mathbf{b}(c, \mathbf{f}) \\ &= \sum_{c \in \Omega(\mathbf{f})} \frac{\mathbf{p}^{[0]}(c)}{(1 - P(\mathbf{f}_1))(1 - P(\mathbf{f}_2)) \ldots, (1 - P(\mathbf{f}_r))} \mathbf{b}(c, \mathbf{f}) \\ &= \frac{1}{(1 - P(\mathbf{f}_1))(1 - P(\mathbf{f}_2)) \ldots, (1 - P(\mathbf{f}_r))} \sum_{c \in \Omega(\mathbf{f})} [\mathbf{p}^{[0]}(c)\mathbf{b}(c, \mathbf{f})] \\ &= \frac{1}{(1 - P(\mathbf{f}_1))(1 - P(\mathbf{f}_2)) \ldots, (1 - P(\mathbf{f}_r))} P^{[0]}(\mathbf{f}) \end{aligned}$$

From condition (2), we know:

$$\frac{P(\mathbf{f}_{r+1})}{\mathbf{t}(\mathbf{f}_{r+1})} > \frac{P(\mathbf{f}_{r+2})}{\mathbf{t}(\mathbf{f}_{r+2})} > \ldots > \frac{P(\mathbf{f}_k)}{\mathbf{t}(\mathbf{f}_k)}$$

Thus, the next action selected by the greedy algorithm is $\mathbf{f}_{r+1}$.

From (A), (B), and (C), we conclude that the property is true.

Note: when equality occurs in condition (2), then the two actions with the equality relation can be switched.

□

Property 14 *For a group of actions $\{\mathbf{f}_1, \ldots, \mathbf{f}_q\}$, if there is no cube belonging to any two action ranges, i.e., $\Omega(\mathbf{f}_j) \bigcap \Omega(\mathbf{f}_i) = \phi$, for $i \neq j$, then the application order of the actions selected by our strategy minimize the expected time of detecting the target with respect to this group.*

Proof:

Without lose of generality, let's assume:

$$\frac{P^{[0]}(\mathbf{f}_1)}{\mathbf{t}(\mathbf{f}_1)} \geq \frac{P^{[0]}(\mathbf{f}_2)}{\mathbf{t}(\mathbf{f}_2)} \geq \ldots \geq \frac{P^{[0]}(\mathbf{f}_q)}{\mathbf{t}(\mathbf{f}_q)}$$

Then, from Property 13, the order of applied actions will be: $\mathbf{f}_1 \rightarrow \mathbf{f}_2 \rightarrow \ldots \rightarrow \mathbf{f}_q$ (Note: when equality occurs, the applying order might be different, but only those with equality can interchange, and this change will not influence the expected time of detecting the target. Therefore, we assume that the above is the application order selected by our algorithm).

Let $\mathbf{F} = \{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_q\}$. Suppose $\mathbf{F}' = \{\mathbf{f}_1', \mathbf{f}_2', \ldots, \mathbf{f}_q'\}$ is an effort allocation that generates the minimum expected time of detecting the target, we will prove in the following that $\Theta[\mathbf{F}] \leq \Theta[\mathbf{F}']$.

If $\mathbf{F}'$ is not the same as $\mathbf{F}$, then there must exist an integer r such that $\mathbf{f}_1 = \mathbf{f}_1'$, $\mathbf{f}_2 = \mathbf{f}_2'$, ..., $\mathbf{f}_{r-1} = \mathbf{f}_{r-1}'$, but $\mathbf{f}_r \neq \mathbf{f}_r'$.

Thus, the index of $\mathbf{f}_r$ in $\mathbf{F}'$ is bigger than r. From the greedy algorithm,

$$\frac{P^{[0]}(\mathbf{f}_r)}{\mathbf{t}(\mathbf{f}_r)} \geq \frac{P^{[0]}(\mathbf{f}_{r+i}')}{\mathbf{t}(\mathbf{f}_{r+i}')}$$

where $i \geq 0$.

Suppose the index of $\mathbf{f}_r$ in $\mathbf{F}'$ is $r + k$. Then,

$$\mathbf{F}' = \{\mathbf{f}_1, \ldots, \mathbf{f}_{r-1}, \mathbf{f}_r', \ldots, \mathbf{f}_{r+k-1}', \mathbf{f}_r, \ldots, \mathbf{f}_q'\}$$

Now let's consider another effort allocation $\mathbf{F}''$,

$$\mathbf{F}'' = \{\mathbf{f}_1, \ldots, \mathbf{f}_{r-1}, \mathbf{f}_r', \ldots, \mathbf{f}_r, \mathbf{f}_{r+k-1}', \ldots, \mathbf{f}_q'\}$$

$$\begin{aligned}
\Theta[\mathbf{F}'] &= \mathbf{t}(\mathbf{f}_1) \times P^{[0]}(\mathbf{f}_1) \\
&\quad + \Big(\mathbf{t}(\mathbf{f}_1) + \mathbf{t}(\mathbf{f}_2)\Big) \times P^{[0]}(\mathbf{f}_2) \\
&\quad + \ldots \\
&\quad + \Big(\mathbf{t}(\mathbf{f}_1) + \ldots + \mathbf{t}(\mathbf{f}_{r+k-1}')\Big) \times P^{[0]}(\mathbf{f}_{r+k-1}') \\
&\quad + \Big(\mathbf{t}(\mathbf{f}_1) + \ldots + \mathbf{t}(\mathbf{f}_{r+k-1}') + \mathbf{t}(\mathbf{f}_r)\Big) \times P^{[0]}(\mathbf{f}_r) \\
&\quad + \ldots \\
&\quad + \Big(\mathbf{t}(\mathbf{f}_1) + \ldots + \mathbf{t}(\mathbf{f}_q')\Big) \times P^{[0]}(\mathbf{f}_q')
\end{aligned} \tag{A.11}$$

$$\begin{aligned}
\Theta[\mathbf{F}''] &= \mathbf{t}(\mathbf{f}_1) \times P^{[0]}(\mathbf{f}_1) \\
&\quad + \Big(\mathbf{t}(\mathbf{f}_1) + \mathbf{t}(\mathbf{f}_2)\Big) \times P^{[0]}(\mathbf{f}_2) \\
&\quad + \ldots \\
&\quad + \Big(\mathbf{t}(\mathbf{f}_1) + \ldots + \mathbf{t}(\mathbf{f}_r)\Big) \times P^{[0]}(\mathbf{f}_r) \\
&\quad + \Big(\mathbf{t}(\mathbf{f}_1) + \ldots + \mathbf{t}(\mathbf{f}_r) + \mathbf{t}(\mathbf{f}_{r+k-1}')\Big) \times P^{[0]}(\mathbf{f}_{r+k-1}') \\
&\quad + \ldots \\
&\quad + \Big(\mathbf{t}(\mathbf{f}_1) + \ldots + \mathbf{t}(\mathbf{f}_q')\Big) \times P^{[0]}(\mathbf{f}_q')
\end{aligned} \tag{A.12}$$

Thus,

$$
\begin{aligned}
\Theta[\mathbf{F}^{''}] - \Theta[\mathbf{F}^{'}] &= \Big\{\Big(\mathbf{t}(\mathbf{f}_1) + \ldots + \mathbf{t}(\mathbf{f}_r)\Big) \times P^{[0]}(\mathbf{f}_r) \\
&\quad + \Big(\mathbf{t}(\mathbf{f}_1) + \ldots + \mathbf{t}(\mathbf{f}_r) + \mathbf{t}(\mathbf{f}^{'}_{r+k-1})\Big) \times P^{[0]}(\mathbf{f}^{'}_{r+k-1})\Big\} \\
&\quad - \Big\{\Big(\mathbf{t}(\mathbf{f}_1) + \ldots + \mathbf{t}(\mathbf{f}^{'}_{r+k-1})\Big) \times P^{[0]}(\mathbf{f}^{'}_{r+k-1}) \\
&\quad + \Big(\mathbf{t}(\mathbf{f}_1) + \ldots + \mathbf{t}(\mathbf{f}^{'}_{r+k-1}) + \mathbf{t}(\mathbf{f}_r)\Big) \times P^{[0]}(\mathbf{f}_r)\Big\} \\
&= \mathbf{t}(\mathbf{f}_r)P^{[0]}(\mathbf{f}^{'}_{r+k-1}) - \mathbf{t}(\mathbf{f}^{'}_{r+k-1})P^{[0]}(\mathbf{f}_r) \\
&= \mathbf{t}(\mathbf{f}_r)\mathbf{t}(\mathbf{f}^{'}_{r+k-1})\Big\{\frac{P^{[0]}(\mathbf{f}^{'}_{r+k-1})}{\mathbf{t}(\mathbf{f}^{'}_{r+k-1})} - \frac{P^{[0]}(\mathbf{f}_r)}{\mathbf{t}(\mathbf{f}_r)}\Big\} \\
&\leq 0
\end{aligned}
$$

Thus, $\Theta[\{\mathbf{f}_1, \ldots, \mathbf{f}_{r-1}, \mathbf{f}^{'}_r, \ldots, \mathbf{f}_r, \mathbf{f}^{'}_{r+k-1}, \ldots, \mathbf{f}_q\}] \leq \Theta[\{\mathbf{f}_1, \ldots, \mathbf{f}_{r-1}, \mathbf{f}^{'}_r, \ldots, \mathbf{f}^{'}_{r+k-1}, \mathbf{f}_r, \ldots, \mathbf{f}^{'}_q\}]$.

Continue the above switching process. We can finally move the action $\mathbf{f}_r$ forward until the index of $\mathbf{f}_r$ is r in $\mathbf{F}^{'}$, while at the same time, the expected detection time will not increase.

So, there exists another effort allocation whose first r actions are same as $\mathbf{F}$ and whose expected time of detecting the target is also minimum. Continue this process, we can finally prove that $\mathbf{F}$ also minimizes the expected time of detecting the target.

□.

Property 15 *Suppose the available operations* $\mathbf{O}_{\mathbf{\Omega}} = \{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_m\}$ *satisfy:*
(1) $\mathbf{t}(\mathbf{f}_i) = \mathbf{t}(\mathbf{f}_j)$, $1 \leq i, j \leq m$.
(2) $\Omega(\mathbf{f}_j) \bigcap \Omega(\mathbf{f}_i) = \phi$, $1 \leq i, j \leq m, i \neq j$.

Then, our simple strategy will generate the optimal answer.

Proof:

Because $\mathbf{t}(\mathbf{f}_i) = \mathbf{t}(\mathbf{f}_j)$ (for $1 \leq i, j \leq m$), it is obvious that the number of actions selected by the best allocation should be equal to that selected by our algorithm. Suppose the optimal allocation is $\mathbf{F}(\mathbf{f}_1, \ldots, \mathbf{f}_k)$. The actions selected by our algorithm are $\mathbf{F}^*(\mathbf{f}^*_1, \ldots, \mathbf{f}^*_k)$.

Suppose $\mathbf{F}^*$ is different from $\mathbf{F}$. Then, we must have $\mathbf{f}^*_i \in \mathbf{F}$, for $1 \leq i \leq r-1$ (where $r \leq k$), and $\mathbf{f}^*_r \notin \mathbf{F}$. From Property 10, we know that $P(\mathbf{F})$ is not influenced by the order of the operations applied. So we can assume that $\mathbf{F} = \{\mathbf{f}^*_1, \ldots, \mathbf{f}^*_{r-1}\mathbf{f}_r, \mathbf{f}_{r+1}, \ldots, \mathbf{f}_k\}$ and $\mathbf{f}_r \notin \mathbf{F}^*$

(Note, there must exist at least one operation among $\{\mathbf{f}_r, \ldots, \mathbf{f}_k\}$ that does not belong to $\mathbf{F}^*$ and we can assume this action is $\mathbf{f}_r$.

Now, consider the allocation $\mathbf{F}^{'}(\mathbf{f}_1^*, \ldots, \mathbf{f}_{r-1}^*, \mathbf{f}_r^*, \mathbf{f}_{r+1}, \ldots, \mathbf{f}_k)$.

It is easy to show that

$$\begin{aligned} P(\mathbf{F}^{'}) - P(\mathbf{F}) &= \sum_{i=1}^{r-1} P^{[0]}(\mathbf{f}_i^*) + P^{[0]}(\mathbf{f}_r^*) + \sum_{i=r+1}^{k} P^{[0]}(\mathbf{f}_i) - \sum_{i=1}^{k} P^{[0]}(\mathbf{f}_i) \\ &= P^{[0]}(\mathbf{f}_r^*) - P^{[0]}(\mathbf{f}_r) \end{aligned}$$

Since the probability distribution $\mathbf{p}$ is the same just before applying action $\mathbf{f}_r^*$ and action $\mathbf{f}_r$, from the simplified algorithm, we know

$$\frac{\sum_{i=1}^{n} \mathbf{p}(c_i)\mathbf{b}(c_i, \mathbf{f}_r^*)}{\mathbf{t}(\mathbf{f}_\mathbf{r}^*)} \geq \frac{\sum_{i=1}^{n} \mathbf{p}(c_i)\mathbf{b}(c_i, \mathbf{f}_r)}{\mathbf{t}(\mathbf{f}_\mathbf{r})}$$

Since $\mathbf{t}(\mathbf{f}_\mathbf{r}^*) = \mathbf{t}(\mathbf{f}_\mathbf{r})$, we simplify this inequality to

$$\sum_{i=1}^{n} \mathbf{p}^{[0]}(c_i)\mathbf{b}(c_i, \mathbf{f}_r^*) \geq \sum_{i=1}^{n} \mathbf{p}^{[0]}(c_i)\mathbf{b}(c_i, \mathbf{f}_r)$$

which can be written as

$$P^{[0]}(\mathbf{f}_r^*) \geq P^{[0]}(\mathbf{f}_r)$$

Thus, we have shown that $P(\mathbf{F}^{'}) \geq P(\mathbf{F})$. Since $\mathbf{F}$ is the optimal allocation, we have $P(\mathbf{F}^{'}) = P(\mathbf{F})$. If we repeat the above process by taking $\mathbf{F}^{'}$ into consideration, we can get another allocation $\mathbf{F}^{''}$ that satisfies $P(\mathbf{F}^{''}) = P(\mathbf{F})$ and which has more elements in common with $\mathbf{F}$ than $\mathbf{F}^{'}$. Continue to repeat the above process and finally we obtain $P(\mathbf{F}^*) = P(\mathbf{F})$.

Therefore, $\mathbf{F}^*$ is also an optimal solution, and the theorem is proved.

□

Appendix B

Object Search from the Point of View of Matrix Calculation

In order to get some insight of the problem, we can study the task of sensor planning for object search from the point of view of matrix calculations.

In order to do this, we define some matrixes and vectors.

The *distribution vector* $A_{n+1}^{[\tau]}$ gives the target distribution of the environment at time instant τ:

$$A_{n+1}^{[\tau]} = (a_0^{[\tau]}, a_1^{[\tau]}, \ldots, a_n^{[\tau]}) \tag{B.1}$$

where $a_i^{[\tau]} = \mathbf{p}(c_i, \tau)$.

For $A_{n+1}^{[\tau]}$, we define

$$diag(A_{n+1}^{[\tau]}) = diag(a_0^{[\tau]}, a_1^{[\tau]}, \ldots, a_n^{[\tau]}) = \begin{pmatrix} a_0^{[\tau]} & 0 & & \cdots & 0 \\ 0 & a_1^{[\tau]} & 0 & \cdots & 0 \\ & & & \vdots & \\ 0 & \cdots & & 0 & a_n^{[\tau]} \end{pmatrix} \tag{B.2}$$

The *detection function matrix* $B_{(n+1)\times m}$ gives the detection function value of all the possible actions with respect to all the cubes.

$$B_{(n+1)\times m} = (B_1, \ldots, B_m) = \begin{pmatrix} b_{01} & \ldots & b_{0m} \\ b_{11} & \ldots & b_{1m} \\ & \vdots & \\ b_{n1} & \ldots & b_{nm} \end{pmatrix} \tag{B.3}$$

where $b_{ij} = \mathbf{b}(c_i, \mathbf{f}_j^*)$, B_j gives the detection function values for action $\mathbf{f}_j^*$ with respect to all the cubes of the environment.

Suppose that the time just before the action $\mathbf{f}_j^*$ is applied is represented by $[j]$, the time just after the action $\mathbf{f}_j^*$ is applied is represented by $[j^+]$. Then, the probability of detecting the target by action $\mathbf{f}_j^*$ is given by $A_{n+1}^{[j]} \times B_j$.

The probability updating rule is given by

$$A_{n+1}^{[j+]} = \frac{diag(A_{n+1}^{[j]})[E_{k\times 1} - B_j]}{\mid A_{n+1}^{[j]}(E_{k\times 1} - B_j) \mid}, \tag{B.4}$$

where

$$E_{k\times 1} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}.$$

With the above updating rule, we can obtain the target distribution and its changes during the application for an effort allocation $\mathbf{F} = \{\mathbf{f}_{j_1}^*, \ldots, \mathbf{f}_{j_k}^*\}$. This can be represented by the *distribution matrix* $A_{k\times n}^{\mathbf{F}}$:

$$A_{k\times n}^{\mathbf{F}} = \begin{pmatrix} A_{n+1}^{[j_1]} \\ \vdots \\ A_{n+1}^{[j_k]} \end{pmatrix} = \begin{pmatrix} a_0^{[j_1]} & a_1^{[j_1]} & \ldots & a_n^{[j_1]} \\ & \vdots & & \\ a_0^{[j_k]} & a_1^{[j_k]} & \ldots & a_n^{[j_k]} \end{pmatrix} \tag{B.5}$$

The *detection probability vector* $D_k^{\mathbf{F}}$ for an effort allocation $\mathbf{F} = \{\mathbf{f}_{j_1}^*, \ldots, \mathbf{f}_{j_k}^*\}$ gives the probability of detecting the target for each action of $\mathbf{F}$.

$$D_k^{\mathbf{F}} = (A_{n+1}^{[j_1]} B_{j_1}, \ldots, A_{n+1}^{[j_k]} B_{j_k}) \tag{B.6}$$

For any vector $U = (u_1, \ldots, u_k)$, we define a function $\mathring{A}$ that maps this vector to a number:

$$\mathring{A}(U) = u_1 + (1 - u_1)u_2 + (1 - u_1)(1 - u_2)u_3 + \cdots + (1 - u_1)\cdots(1 - u_{k-1})u_k \tag{B.7}$$

Then, of course, $\mathring{A}(D_k^{\mathbf{F}})$ gives the probability of detecting the target by the effort allocation $\mathbf{F}$.

Thus, the task of object search is a task of finding an effort allocation $\mathbf{F}$ such that that value of $\mathring{A}(D_k^{\mathbf{F}})$ is maximized and the cost of $\mathbf{F}$ is within a given constraint.

Appendix C

The Selection of the Viewing Direction

In this appendix[1] , we discuss the strategy of selecting the camera's viewing direction for a fixed camera angle size during the search process. This strategy is similar to the tessellation of the sensed sphere, but not identical. The surface patterns cut by the camera's viewing volume are different from the surface patterns resulting from the tessellation of the sensed sphere. It is very difficult to select a set of camera viewing directions such that their union will cover the whole unit sphere without mutual overlap. Thus, our goal is to select a set of viewing directions such that their union can cover the whole unit sphere and that there is little overlap among them. Of course, there exist many ways to select the viewing directions. Here, we use a method similar to the method used to construct the sensed sphere. Suppose $\beta = min\{w_i, h_i\}$. We first simplify the problem by assuming that the width and the height of the viewing angle size are both equal to β.

We need to find the value of α used in the tessellation. The value of α should be a little smaller the β, such that when the target is at the boundary of two neighboring viewing angles determined by the tessellation using α, it can be totally covered by the real camera viewing volume. In Figure C.1, $\angle AOB = \alpha$, $\angle COD = \beta$, OG is the viewing direction. $\mid OG \mid = N_i$. Suppose the diameter of the target is d_{tar}, to guarantee that the target be totally covered by the real camera viewing volume when the target is at the boundary, EF must be equal or greater than d_{tar}. Thus, we get

$$\alpha = \beta - 2asin\{\frac{d_{tar}}{N_i}\} \tag{C.1}$$

The overall tessellation process is as follows (Figure C.2). We first divide the sphere into a series of slices. Each slice corresponds to a calculated tilt interval. Then we select the change of pan for each given tilt slice such that the selected viewing directions can cover the given slice.

[1]Because many symbols are required in this Chapter, the meanings of the symbols used here are independent of their meanings in other chapters of this thesis. For example, the w in this Chapter represents the width of the "cutting square", while the w in other chapters of this thesis represents the widths of the viewing angle size.

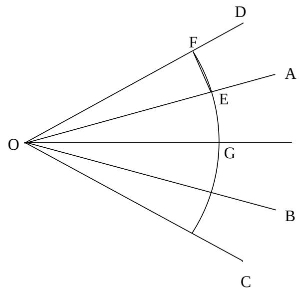


Figure C.1: The selection of α.

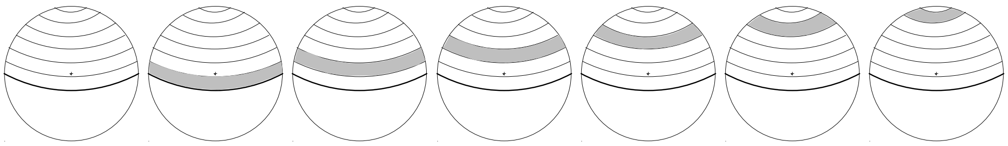


Figure C.2: The general strategy in selecting the viewing direction. (a) We need to divide the sphere into a series of slices, each slice corresponding to a tilt interval. (b) to (g) We need to select the pan value to cover each slice of tilt interval one by one.

The structure of the viewing volume is fixed. Suppose the radius of the sphere is r, the center of the sphere is O. In Figure C.3, of course we have $\mid OA \mid = \mid OB \mid = \mid OC \mid = \mid OD \mid = r$ and $\angle FOE = \frac{\alpha}{2}$. After some calculation, we obtain

$$\mid OE \mid = \frac{r}{\sqrt{1 + 2tan^2(\frac{\alpha}{2})}} \tag{C.2}$$

$$\mid FE \mid = \frac{rtan(\frac{\alpha}{2})}{\sqrt{1 + 2tan^2(\frac{\alpha}{2})}} \tag{C.3}$$

No matter what the view direction will be, the value of $\mid OE \mid$ and $\mid FE \mid$ will not change. Let $w = 2 \mid FE \mid$ and $h = \mid OE \mid$. We call w the width of the "cutting square" of the viewing volume and h the height of the "cutting square". The "cutting square" of the viewing volume refers to square $ABCD$.

Suppose the viewing direction of the camera (pan and tilt) is $< p, t >$. We want to determine the range of surface patch on the sphere that can be covered by the viewing volume of the camera. This surface patch should be specified by a tilt range $t_b \leq tilt \leq t_e$ and a pan range $p_b \leq pan \leq p_e$. Our purpose in the following is to find the values of t_b, t_e, p_b, and p_e, such that the corresponding surface patch can be covered by the camera viewing volume with direction (p, t).

We need to find the range of pan $\Delta_{pan} = p_e - p_b$ for the given viewing direction of the camera. From Figure C.4, we can see that any point J on arc $\widehat{AB}$ has a corresponding point E on arc $\widehat{CD}$ such that they have the same value of tilt. The smallest difference of pan for a

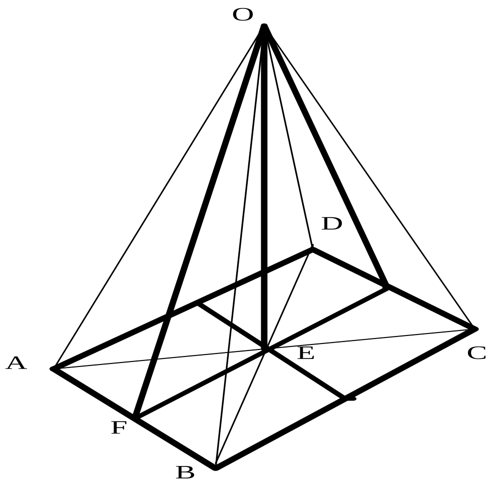


Figure C.3: The viewing volume formed by the given camera angle size. Where O is the camera center (also the center of the sphere). The viewing volume is formed by surface ABO, BCO, CDO, and DAO. A, B, C, D are the four points of the intersection of OA, OB, OC, OD with the surface of the sphere. OE is the camera's viewing direction.

point on $\widehat{AB}$ and its corresponding point on $\widehat{CD}$ occurs at point A and point D. We take the difference of the range of pan for points A and D as the value of Δ_{pan}.

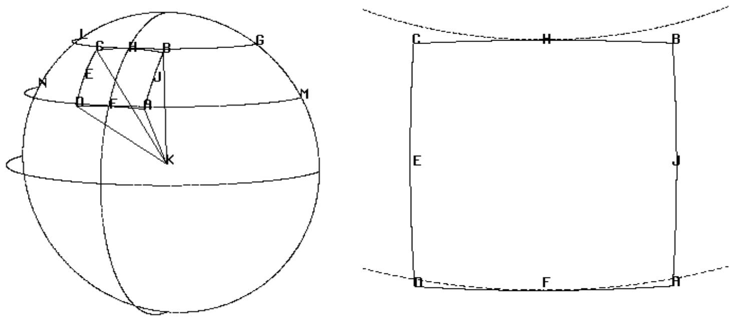
Without lose of generality, let us assume that the camera's viewing direction is $tilt = t$ and $pan = 0$ (Figure C.5(b)). Suppose when the viewing direction is $tilt = \frac{\pi}{2}$ and $pan = 0$ (Figure C.5(a)), the coordinates of A, B, C, D, I, J, and H are (x_A, y_A, z_A), (x_B, y_B, z_B), (x_C, y_C, z_C), (x_D, y_D, z_D), (x_I, y_I, z_I), (x_J, y_J, z_J), and (x_H, y_H, z_H). Suppose when the viewing direction is $tilt = t$ and $pan = 0$ (Figure C.5(b)), the coordinate of A, B, C, D, I, J, H are (x'_A, y'_A, z'_A), (x'_B, y'_B, z'_B), (x'_C, y'_C, z'_C), (x'_D, y'_D, z'_D), (x'_I, y'_I, z'_I), (x'_J, y'_J, z'_J), (x'_H, y'_H, z'_H).

From Figure C.6, we obtain

$$\begin{aligned} x'_J &= \sqrt{h^2 + (\frac{w}{2})^2} cos[\frac{\pi}{2} - (t + \frac{\alpha}{2})] \\ &= \sqrt{\left(\frac{r}{\sqrt{1 + 2tan^2(\frac{\alpha}{2})}}\right)^2 + \left(\frac{rtan(\frac{\alpha}{2})}{\sqrt{1 + 2tan^2(\frac{\alpha}{2})}}\right)^2} sin(t + \frac{\alpha}{2}) \\ &= r\frac{\sqrt{1 + tan^2(\frac{\alpha}{2})}}{\sqrt{1 + 2tan^2(\frac{\alpha}{2})}} sin(t + \frac{\alpha}{2}) \end{aligned} \qquad (C.4)$$

Similarly

$$z'_J = \sqrt{h^2 + (\frac{w}{2})^2} sin[\frac{\pi}{2} - (t + \frac{\alpha}{2})]$$



(a) (b)

Figure C.4: The surface patch cut by the view volume.

(a) The camera's viewing volume and the sphere. $KABCD$ is the camera's viewing volume, where K is the center of the camera, $\widehat{AB}$, $\widehat{BC}$, $\widehat{CD}$ and $\widehat{DA}$ are the big circle arcs cut by the bounding plane of the viewing volume. $\widehat{HF}$ is the big circle arc which is cut by the vertical plane of the viewing volume (YZ plane in Camera Coordinate System). H is at the middle of big circle arc $\widehat{BC}$, F is at the middle of big circle arc $\widehat{AD}$. $\widehat{NFM}$ is part of the locus of points on the sphere whose tilt is $t+\frac{\alpha}{2}$, and $\widehat{LHG}$ is part of the locus of points on the sphere whose tilt is $t-\frac{\alpha}{2}$. $\widehat{LHG}$ and $\widehat{BC}$ only have one common point H. $\widehat{NFM}$ and $\widehat{AD}$ only have one common point F. (b) A detailed view of the surface patch $ABCD$ that is contained within the camera's viewing volume.

$$= \quad r\frac{\sqrt{1+tan^2(\frac{\alpha}{2})}}{\sqrt{1+2tan^2(\frac{\alpha}{2})}}cos(t+\frac{\alpha}{2}) \tag{C.5}$$

Since $x_B^{'} = x_J^{'}$, $y_B^{'} = \frac{w}{2}$, and $z_B^{'} = z_J^{'}$, we can get the pan P_B of point B is:

$$\begin{aligned} P_B &= arctan\{\frac{y_B^{'}}{x_B^{'}}\} \\ &= arctan\{\frac{\frac{rtan(\frac{\alpha}{2})}{\sqrt{1+2tan^2(\frac{\alpha}{2})}}}{r\frac{\sqrt{1+tan^2(\frac{\alpha}{2})}}{\sqrt{1+2tan^2(\frac{\alpha}{2})}}sin(t+\frac{\alpha}{2})}\} \\ &= arctan\{\frac{sin(\frac{\alpha}{2})}{sin(t+\frac{\alpha}{2})}\} \end{aligned} \tag{C.6}$$

So, when the tilt value of the viewing direction of the camera is t, the change of pan should be

$$\Delta_{pan} = 2P_B = 2arctan\{\frac{sin(\frac{\alpha}{2})}{sin(t+\frac{\alpha}{2})}\} \tag{C.7}$$

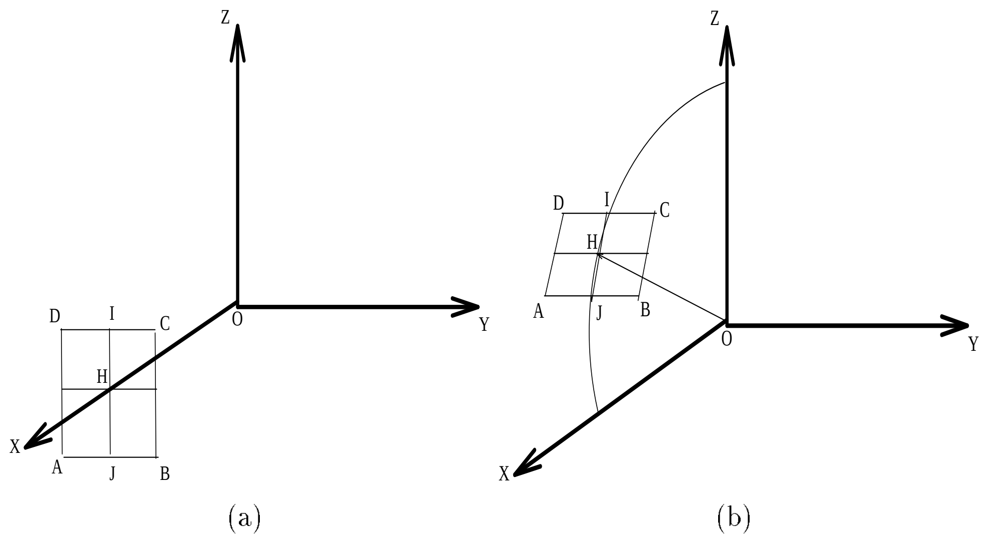


Figure C.5: (a) The view volume when the viewing direction is $pan = 0$ and $tilt = \frac{\pi}{2}$. (b) The view volume when the viewing direction is $pan = 0$ and $tilt < \frac{\pi}{2}$.

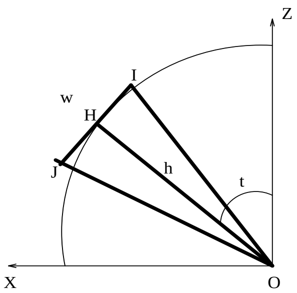


Figure C.6: The calculation of the coordinate of point J

But for this range of pan, the viewing volume cannot cover all the surface patch whose tilt value is between $t - \frac{\alpha}{2}$ and $t + \frac{\alpha}{2}$. From Figure C.4(b), we see that the big circle arc $\widehat{HB}$ gradually goes down (decreases in Z value) as the point moves from H to B. So, there is a part between big circle arc $\widehat{CHB}$ and arc $\widehat{LHG}$ that can not be covered by the camera's viewing volume when the intended pan range is Δ_{pan}. We need to determine the tilt value of a point as it approaches B from H along the big circle arc $\widehat{HB}$ with the change of pan equal to $\frac{\Delta_{pan}}{2}$.

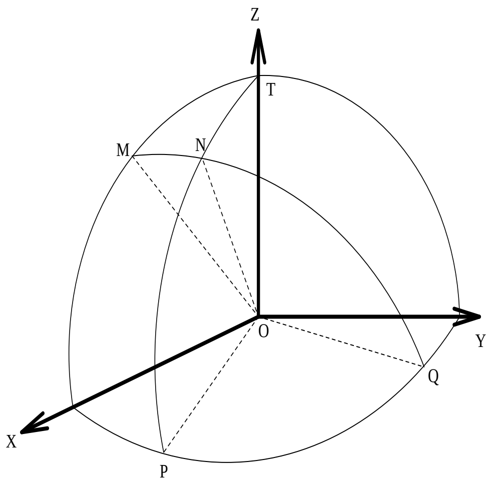


Figure C.7: The change of tilt for the upper side of the viewing volume for the given pan. $OPNT$ is the plane corresponding to $pan = \frac{\Delta_{pan}}{2}$. $\widehat{PNT}$ is the big circle arc cut by plane $OPNT$. $OMNQ$ is the plane corresponding to the upper side of the camera's viewing volume. ON is the intersection line of plane $OPNT$ and plane $OMNQ$.

Without lose of generality, let us assume again that the viewing direction is $tilt = t$ and $pan = 0$. As shown in Figure C.7, the tilt of line OM is $t - \frac{\alpha}{2}$. We want to find the tilt for line ON. From this we can find the change of tilt when a point moves from M to N.

It is easy to see that the direction of the normal to plane $OMNQ$ is $\{-rcos(t-\frac{\alpha}{2}), 0, rsin(t-\frac{\alpha}{2})\}$ and the direction of the normal to the plane $OTNP$ is $\{-rsin(P_B), rcos(P_B), 0\}$.

Thus, the equation for the line ON is

$$\begin{cases} -rcos(t-\frac{\alpha}{2})x + 0 + rsin(t-\frac{\alpha}{2})z = 0 \\ -rsin(P_B)x + rcos(P_B)y + 0 = 0 \end{cases}$$

Let

$$p = \begin{vmatrix} 0 & rsin(t-\frac{\alpha}{2}) \\ rcos(P_B) & 0 \end{vmatrix} = -rsin(t-\frac{\alpha}{2})rcos(P_B) = -r^2sin(t-\frac{\alpha}{2})cos(P_B)$$

$$q = -\begin{vmatrix} -rcos(t-\frac{\alpha}{2}) & rsin(t-\frac{\alpha}{2}) \\ -rsin(P_B) & 0 \end{vmatrix} = -rsin(t-\frac{\alpha}{2})rsin(P_B) = -r^2sin(t-\frac{\alpha}{2})sin(P_B)$$

$$\gamma = \begin{vmatrix} -rcos(t-\frac{\alpha}{2}) & 0 \\ -rsin(P_B) & rcos(P_B) \end{vmatrix} = -rcos(t-\frac{\alpha}{2})rcos(P_B) = -r^2cos(t-\frac{\alpha}{2})cos(P_B)$$

Suppose the tilt of ON is t_{ON}, then we have

$$\begin{aligned}
cos(t_{ON}) &= \frac{\mid \gamma \mid}{\sqrt{p^2+q^2+\gamma^2}} \\
&= \frac{cos(t-\frac{\alpha}{2})cos(P_B)}{\sqrt{sin^2(t-\frac{\alpha}{2})cos^2(P_B)+sin^2(t-\frac{\alpha}{2})sin^2(P_B)+cos^2(t-\frac{\alpha}{2})cos^2(P_B)}} \\
&= \frac{cos(t-\frac{\alpha}{2})cos(P_B)}{\sqrt{cos^2(P_B)+sin^2(t-\frac{\alpha}{2})sin^2(P_B)}} \\
&= \frac{cos(t-\frac{\alpha}{2})}{\sqrt{1+sin^2(t-\frac{\alpha}{2})tan^2(P_B)}} \\
&= \frac{cos(t-\frac{\alpha}{2})}{\sqrt{1+sin^2(t-\frac{\alpha}{2})\frac{sin^2(\frac{\alpha}{2})}{sin^2(t+\frac{\alpha}{2})}}} \\
&= \frac{cos(t-\frac{\alpha}{2})}{\sqrt{1+\frac{sin^2(\frac{\alpha}{2})sin^2(t-\frac{\alpha}{2})}{sin^2(t+\frac{\alpha}{2})}}} \qquad (C.8)
\end{aligned}$$

so,

$$t_{ON} = acos\{\frac{cos(t-\frac{\alpha}{2})}{\sqrt{1+\frac{sin^2(\frac{\alpha}{2})sin^2(t-\frac{\alpha}{2})}{sin^2(t+\frac{\alpha}{2})}}}\}$$

From above calculations, it is easy to show that the following surface patch of the sphere can be covered by the viewing volume with direction (p,t).

$$p - P_B \le pan \le p + P_B$$

$$t_{ON} \le tilt \le t + \frac{\alpha}{2}$$

or in more detail:

$$p - arctan\{\frac{sin(\frac{\alpha}{2})}{sin(t+\frac{\alpha}{2})}\} \le pan \le p + arctan\{\frac{sin(\frac{\alpha}{2})}{sin(t+\frac{\alpha}{2})}\} \qquad (C.9)$$

$$\frac{cos(t-\frac{\alpha}{2})}{\sqrt{1+\frac{sin^2(\frac{\alpha}{2})sin^2(t-\frac{\alpha}{2})}{sin^2(t+\frac{\alpha}{2})}}} \le tilt \le t + \frac{\alpha}{2} \qquad (C.10)$$

According to the above calculations, we obtain the following algorithm for enumerating the viewing directions to cover the whole sphere (Suppose the final viewing direction set is $S_{candidate}$).

1. $S_{candidate} = \phi$

2. $p \longleftarrow 0$, $t \longleftarrow 0$, $S_{candidate} = S_{candidate} \bigcup < p, t >$.

3. $t_e \longleftarrow \frac{\pi}{2}$

4. $t_b \longleftarrow acos\{\frac{cos((t_e-\frac{\alpha}{2})-\frac{\alpha}{2})}{\sqrt{1+\frac{sin^2(\frac{\alpha}{2})sin^2((t_e-\frac{\alpha}{2})-\frac{\alpha}{2})}{sin^2((t_e-\frac{\alpha}{2})+\frac{\alpha}{2})}}}\}$

5. Cover the slice on the sphere whose tilt is within the range of $[t_b, t_e]$ and the slice on the sphere whose tilt is within the range of $[\pi - t_e, \pi - t_b]$.

   (a) Let $t \longleftarrow t_e - \frac{\alpha}{2}$.

   (b) Let $\Delta_{pan} \longleftarrow 2arctan\{\frac{sin(\frac{\alpha}{2})}{sin((t_e-\frac{\alpha}{2})+\frac{\alpha}{2})}\}$

   (c) Use Δ_{pan} to divide $[0, 2\pi]$ for the given slice. We obtain a series of intervals $[p_b, p_e]$ as follows:. $[0, \Delta_{pan}]$, $[\Delta_{pan}, 2\Delta_{pan}]$, ..., $[k\Delta_{pan}, 2\pi]$. Note: the length of the last interval may not be Δ_{pan}.

   (d) For each division, let $p \longleftarrow \frac{p_b+p_e}{2}$. Then perform $S_{candidate} = S_{candidate} \bigcup < p, t >$ and $S_{candidate} = S_{candidate} \bigcup < p, \pi - t >$.

6. Let $t_e \longleftarrow t_b$

7. If $t_e \leq \alpha$, stop the process. Otherwise Goto 4.

One thing that need to note here is that if α is too large, for example $\alpha > \frac{\pi}{2}$, we should use a smaller angle size such as $\frac{\alpha}{2}$ to select the viewing angle direction. For a given viewing direction, there is a small region within the corresponding layer defined by the effective range that is not within the effective volume, although this region is within the camera's viewing volume. When the camera's viewing angle size is not too large, the influence of this region can be neglected, but for large α, the influence of this region should not be neglected. Thus, we use a smaller angle size to select the viewing direction. Since normally the camera's viewing angle size is not too large, we usually neglect the influence of this region. If we really want to cover this region, then the effective ranges and the viewing angle size should be selected according to the following formulas:

$$N_{i+1} = x = \frac{F_i}{\sqrt{1 + \tan^2\left(\frac{h_i}{2}\right) + \tan^2\left(\frac{w_i}{2}\right)}}$$

$$F_{i+1} = \frac{F_i^2}{N_i\sqrt{1 + \tan^2\left(\frac{h_i}{2}\right) + \tan^2\left(\frac{w_i}{2}\right)}}$$

$$w_{i+1} = 2arctan\left[\frac{N_i}{F_i}\tan\left(\frac{w_i}{2}\right)\sqrt{1 + \tan^2\left(\frac{h_i}{2}\right) + \tan^2\left(\frac{w_i}{2}\right)}\right]$$

$$h_{i+1} = 2arctan\left[\frac{N_i}{F_i}\tan\left(\frac{h_i}{2}\right)\sqrt{1+\tan^2\left(\frac{h_i}{2}\right)+\tan^2\left(\frac{w_i}{2}\right)}\right]$$

Bibliography

[1] A.L. Abbott. Active surface reconstruction by integrating focus,vergence, stereo, and camera calibration. In *Third International Conference on Computer Vision*, pages 489–491, Hilton Head, SC., December 1991.

[2] A.L. Abbott and N. Ahuja. Surface reconstruction by dynamic integration of focus, camera vergence, and stereo. In *Second International Conference on Computer Vision*, pages 532–543, Florida, USA, December 1988.

[3] A.L.Abbott. Selective fixation control for machine vision: A survey. In *IEEE Intervnational Conference on System, Man and Cybernetics*, pages 1–6, August 1991.

[4] Y. Aloimonous, I. Weiss, and A. Bandyopadhyay. Active vision. In *First International Conference on Computer Vision*, pages 35–54, London, England, June 1987.

[5] Firshid Arman. Model-based object recognition in dense-range images —– a review. *ACM Computing Surveys*, 25(1):5–45, 1993.

[6] R. Bajcsy. Active perception vs. passive perception. In *Third IEEE Workshop on Vision*, pages 55–59, Bellaire, 1985.

[7] R. Bajcsy. Perception with feedback. In *Image Understanding Workshop*, pages 279–288, 1988.

[8] R. Bajcsy and M. Campos. Active and exploratory perception. *CVGIP: Image Understanding*, 56(1):31–40, 1992.

[9] Andrew Balke and Alan Yuille. *Active Vision.* The MIT Press, Cambridge, Mass., 1992.

[10] D. H. Ballard and A. Ozcandarli. Eye fixation and early vision: Kinetic depth. In *Second International Conference on Computer Vision*, pages 524–531, Florida, USA, December 1988.

[11] Dana H. Ballard. Task frames in visuo-motor coordination. In *Proceedings IEEE Workshop on Computer Vision: Representation and Control*, pages 3–10, Bellaire MI., 1985.

[12] D.H. Ballard. Animate vision. *Artificial Intelligence*, 48:57–86, 1991.

[13] A. Bandyopadhyay, B. Chandra, and D. Ballard. Active navigation: Tracking an environmental point considered beneficial. In *Proceedings of IEEE Workshop on Motion: Representation and Analysis*, pages 23–28, Charleston,SC, May 1986.

[14] Paul J. Besl and Ramesh C. Jain. Three-dimensional object recognition. *ACM Computing Surveys*, 17(1):74–145, 1985.

[15] R.M. Bolle, A. Califano, and R. Kjeldsen. Data and model driven foveation. Technical Report RC 15096, IBM Research Division, T.J. Watson Research Center, NY, October 1989.

[16] M. Brand. An eye for design: Why, where and how to look for causal structure in visual scenes. *SPIE.Intelligent Robots and Computer Vision*, 1825(XI):180–188, 1992.

[17] Rodney Brooks. A layered intelligent control system for a mobile robot. *IEEE Journal Robotics and Automation*, (RA-2):14–23, April 1986.

[18] C.M. Brown. Issues in selective perception. In *11th International Conference on Pattern Recognition*, pages 21–30, 1992.

[19] R. Chin and C. Dyer. Model-based recognition in robot vision. *Computing Serveys*, 18(1):67–108, March 1986.

[20] Connel. *An Artificial Creature*. PhD thesis, AI Lab, MIT, 1989.

[21] Cregg K. Cowan and Peter D. Kovesi. Automatic sensor placement from vision task requirements. *IEEE Transactions on Pattern analysis and Machine Intelligence*, 10(3):407–416, May 1988.

[22] S.M. Culhane and J.K. Tsotsos. An attentional prototype for early vision. In *Second European Conference on Computer Vision*, pages 551–560, Santa Margherita Ligure, Italy, May 1992.

[23] S.M. Culhane and J.K. Tsotsos. A prototype for data-driven visual attention. In *11th International Conference on Pattern Recognition*, pages 36–40, 1992.

[24] T. Dean. Sequential decision making for active perception. In *Image Understanding Workshop*, pages 889–894, Pennsylvania, USA, September 1990.

[25] C. Durieu, H. Clergeot, and F. Monteil. Localization of a mobile robot with beacons taking erroneous data into account. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1062–1068, New York, U.S.A, 1989.

[26] A. Elfes. A sonar-based mapping and navigation system. In *IEEE International Conference on Robotics and Automation*, San Francisco, CA. U.S.A., April 1986.

[27] M. Garey and D. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. W.H. Freeman and Co., New York, 1979.

[28] T.D. Garvey. Perceptual strategies for purposive vision. Technical Report Tech. Note 117, SRI International, 1976.

[29] Thomas D. Garvey. Perceptual strategies for purposive vision. Technical Report Technical Note 117, SRI International, 1976.

[30] J.J. Gibson. *The perception of the visual world*. Houghton-Mifflin, Boston, 1950.

[31] J.J. Gibson. *The Ecological Approach to Visual Perception*. Houghton-Mifflin, Boston, 1979.

[32] W.E.L. Grimson. The combinatorics of local constraints in model-based recognition and localization from sparse data. *Journal of the Association for Computing Machinery*, 33(4):658–686, 1986.

[33] D.H. Hubel. *Eye, Brain and Vision*. Volume 22 of Scientific American Library W.H.Freeman and Company, 1988.

[34] K. Ikeuchi. Precompiling a geometrical model into an interpretation tree for object recognition in bin-picking tasks. In *Image Understanding Workshop*, pages 321–330, California, February 1987.

[35] Chris L. Jackins and Steven L. Tanimoto. Oct-trees and their use in representing three-dimensional objects. *Computer Graphics and Image Processing*, (14):249–270, 1980.

[36] R. Jain, S.L. Bartlett, and N. O'Brien. Motion stereo using ego-motion complex logarithmic mapping. *IEEE Transactions on Pattern analysis and Machine Intelligence*, PAMI-9(3):356–369, May 1987.

[37] Tarabanis K., P. Allen, and R. Tsai. A survey of sensor planning in computer vision. Technical report, Comp. Sci. Dept., Columbia University, 1992.

[38] J.R. Kender and Freudenstein. What is a "degenerate" view? In *Image Understanding Workshop*, pages 589–598, California, February 1987.

[39] H.S. Kim, R.C Jain, and R. A. Volz. Object recognition using multiple views. In *1985 IEEE Intervnational Conference on Robotics and Automation*, pages 28–33, July 1985.

[40] L. Kirousis and C. Papadimitriou. The complexity of recognizing polyhedral scenes. In *26 Annual Symposium on Foundations of Computer Science*, Portland, Ore., 1985.

[41] J. Koenderink and A. van Doorn. The internal representation of solid shape with respect to vision. *Biological Cybernetics*, 32:211–216, 1979.

[42] B. O. Koopman. *Search and Screen: general principles with historical applications.* Pergaman Press, Elmsford, N.Y, 1980.

[43] David J. Kriegman, Ernst Triendl, and Thomas O. Binford. Stereo vision and navigation in building for mobile robots. *IEEE Transactions on Robotics and Automation*, 5(6):792–803, December 1989.

[44] E. Krotkov. *Active computer vision by cooperative focus and stereo.* Springer-Verlag, New York, 1989.

[45] Eric Krotkov. Focusing. *International Journal of Computer Vision*, 1:223–237, 1987.

[46] K.N. Kutulakos and Dyer. Recovering shape by purposive viewpoint adjustment. In *Proceedings of Computer Vision and Pattern Recognition*, pages 16–22, Illinois,USA, June 1992.

[47] H.J. Levesque. What is planning in the presence of sensing? In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Portland, Oregon, August 1996.

[48] H.J. Levesque, R. Reiter, Y. Lespirance, F. Lin, and R. Scherl. Golog: A logic programming language for dynamic domains. *Journal of Logic Programming, Special issue on Reasoning about Action and Change*, to appear.

[49] S. Li. Realizing active vision by a mobile robot. In *IEEE Workshop on Visual Motion*, pages 205–209, 1991.

[50] D. Marr. *Vision.* W. H. Freeman, 1982.

[51] Massone, Sandini, and Tagliasco. “form-invariant” topological strategy for 2d shape recognition. *Computer Vision, Graphics and Image Processing*, 30:169–188, 1985.

[52] L. Mathies and S. A. Shafer. Error modeling in stereo navigation. *IEEE Journal of Robotics and Automation*, RA-3(3):239–248, 1987.

[53] J. Maver and R. Bajcsy. Occlusions and the next view planning. In *IEEE 1992 International Conference on Robotics and Automation*, pages 1806–1811, April 1992.

[54] Jasna Maver and Ruzena Bajcsy. How to decide from the first view where to look next. In *Proceedings of the DARPA Image Understanding Workshop*, September 1990.

[55] Merriam-Webster. *Webster's Ninth New Collegiate Dictionary.* Merriam-Webster, Springfield, Mass., 1985.

[56] Randal C. Nelson. Imtroduction: Vision as intelligent behavior: An introduction to machine vision at the university of rochester. *International Journal Computer Vision*, 7(1), 1991.

[57] S.B. Nickerson, M. Jenkin, E. Milios, B. Down, P. Jasiobedzki, A. Jepson, D. Terzopoulos, J. Tsotsos, D. Wilkes, N. Bains, and K. Tran. Ark: Autonomous navigation of a mobile robot in a known environment. In *Intelligent Autonomous Systems-3*, pages 288–296, Pennsylvania, USA, February 1993.

[58] W. H. Plantinga and C. R. Dyer. An algorithm for constructing the aspect graph. In *27 Annual Symposium on Foundations of Computer Science*, pages 123–131, Toronto, Ontario, 1986.

[59] Michael Potmesil. Generating octree models of 3d objects from their silhouettes in a sequence of images. *Computer Graphics and Image Processing*, (40):1–29, 1987.

[60] R.A.Brooks and J.H.Connell. Asynchronous distributed control system for a mobile robot. In *Mobile Robots:Proc. SPIE 727*, California, 1987.

[61] Raymond D. Rimey. Controlling eye movements with hidden markov models. *International Journal Computer Vision*, 7(1), 1991.

[62] R.D. Rimey. Task-specific utility in a general bayes net vision system. In *Proceedings of Computer Vision and Pattern Recognition*, pages 142–147, Illinois,USA, June 1992.

[63] R.D. Rimey and C.M. Brown. Where to look next using a bayes net: Incorporating geometric relations. In *Second European Conference on Computer Vision*, pages 542–550, Santa Margherita Ligure, Italy, May 1992.

[64] M.P. Robins. Free-ranging automatic guided-vehicle system. *GEC Reviews*, 2(2), February 1986.

[65] G. Sandini. Active tracking strategy for monocular depth inference over multiple frames. *IEEE trans. on PAMI*, 12:13–27, 1990.

[66] Bernt Schiele and James L. Crowley. Where to look next and what to look for. In *The 4th International Symposium on Intelligent Robotics Systems*, pages 139–145, Lisbon, July 1996.

[67] A. Shmuel and Werman. Active vision:3d from an image sequence. In *10th International Conference on Pattern Recognition*, pages 48–54, New Jersey, USA, June 1990.

[68] Paul Suetens. Computational strategies for object recognition. *ACM Computing Surveys*, 24(1):5–61, 1992.

[69] Michael J. Swain. Low resolution cues for guiding saccadic eye movements. In *Proceedings of Computer Vision and Pattern Recognition*, pages 737–740, Illinois, USA, June 1992.

[70] Michael J. Swain and D. Ballard. Color indexing. *International Journal Computer Vision*, 7(1):11–32, 1991.

[71] M.J. Swain and M. Stricker. Promising directions in active vision. Technical Report CS 91-27, Univ. Of Chicago Technical Report, 1991.

[72] Konstantinos Tarabanis, Roger Y. Tsai, and Peter K. Allen. Automatic sensor placement from vision task requirements. *CVGIP: Image Understanding*, 10.(3):340–358, May 1988.

[73] C. E. Thorpe, M. Hebert, T. Kanade, and S. Shafer. Vision and navigation for the carnegie-mellon navlab. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):362–373, May 1988.

[74] J.K Tsotsos. A complexity level analysis of immediate vision. *International Journal Computer Vision*, 4(1):303–320, 1988.

[75] J.K. Tsotsos. Analyzing vision at the complexity level. *The behavioral and brain science*, 13:423–469, 1990.

[76] J.K. Tsotsos. Active verses passive perception, which is more efficient? *Behavioral and brain sciences*, 14:4:770–773, 1991.

[77] J.K. Tsotsos. 3d search strategy. *Internal ARK Working Paper, Department of Computer Science, Univrsity of Toronto*, 1992.

[78] J.K. Tsotsos. Active verses passive perception, which is more efficient? *IJCV*, 1992.

[79] J.K. Tsotsos. On the relative complexity of active vs. passive visual search. *International Journal of Computer Vision*, 7:127–141, 1992.

[80] John K. Tsotsos, S. Dickinson, M. Jenkin, E. Milios, A. Jepson, B. Down, E. Amdur, S. Stevenson, M. Black, D. Metaxas, J. Cooperstock, S. Culhane, F. Nuflo, G. Verghese, W. Wai, D. Wilkes, and Yiming Ye. The playbot project. In *IJCAI95: Workshop for handicapped Children*, Montreal, Canada, 1995.

[81] M. A. Turk, D. G. Morgenthaler, K. D. Gremban, and M. Marra. Vits —– a vision system for autonomous land vehicle navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):362–373, May 1988.

[82] W.E.Grimson. Sensing strategies for disambiguating among multiple objects in known poses. *IEEE Journal of Robotics and Automation*, RA-2(4):196–213, December 1986.

[83] C.J. Westelius. Preattentive gaze control for robot vision. Technical Report Thesis No. 322, Linkoping Studies in Science and Technology, 1992.

[84] P. Whaite. Uncertain views. In *Proceedings of Computer Vision and Pattern Recognition*, pages 3–9, Illinois,USA, June 1992.

[85] P. Whaite and F. P. Ferrie. From uncertainty to visual exploratory. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):1308–1049, 1991.

[86] D. Wilkes. Active object recognition. Technical report, Department of Computer Science, University of Toronto, 1994.

[87] D. Wilkes and J.K. Tsotsos. Active object recognition. In *Proceedings of Computer Vision and Pattern Recognition*, pages 136–141, Illinois, USA, June 1992.

[88] Lambert Ernest Wixson. *Gaze Selection for Visual Search*. PhD thesis, Department of Computer Science, University of Rochester, 1994.

[89] L.E. Wixson. Exploiting world structure to efficiently search for objects. Technical Report Technical Report 434, Dept. Comp. Sci, Univ. of Rochester, 1992.

[90] L.E. Wixson. Looking near one object for another. *SPIE.Intelligent Robots and Computer Vision*, 1825(XI):159–167, 1992.

[91] L.E. Wixson and D. Ballard. Using intermediate object to improve the efficiency of visual search. *International Journal on Computer Vision*, 18(3):209–230, 1994.

[92] Yiming Ye and John K. Tsotsos. Tracking with pan, tilt, and zoomable camera. In *In preparation.*

[93] Yiming Ye and John K. Tsotsos. Sensor planning for object search. Technical Report RBCV-TR-94-47, Computer Science Department, University of Toronto, 1994.

[94] Yiming Ye and John K. Tsotsos. The detection function in object search. In *Proceedings of the fourth international conference for young computer scientists*, pages 868–873, Beijing, PRC, August 1995.

[95] Yiming Ye and John K. Tsotsos. Sensor planning in 3d object search: its formulation and complexity. In *The 4th International Symposium on Artificial Intelligence and Mathematics*, Florida, U.S.A., January 1995.

[96] Yiming Ye and John K. Tsotsos. Where to look next in 3d object search. In *IEEE International Symposium for Computer Vision*, pages 539–544, Florida, U.S.A., November 1995.

[97] Yiming Ye and John K. Tsotsos. On the collaborative object search team: a formulation. In *ICMAS-96 workshop on Learning, Interactions and Organizations in Multiagent Environment*, Kyoto, Japan, December 1996.

[98] Yiming Ye and John K. Tsotsos. A robot agent that can search. In *Proceedings of the sixth annual CAS conference*, IBM Canada Ltd. Laboratory Centre for Advanced Studie, Toronto, Ontario, Canada, November 1996.

[99] Yiming Ye and John K. Tsotsos. Sensor planning in 3d object search. In *The 4th INTERNATIONAL SYMPOSIUM on INTELLIGENT ROBOTIC SYSTEMS '96*, Lisbon, Portugal, July 1996.

[100] Yiming Ye and John K. Tsotsos. Knowledge difference and its influence on a search agent. In *First International Conference on Autonomous Agents*, Marina del Rey, California, February 1997.

[101] Yiming Ye and John K. Tsotsos. On the collaborative object search team: a formulation. In *Learning in DAI Systems, Gerhard Weiss edited*, Lecture Notes in Artificial Intelligence, Springer Verlag, 1997.

[102] Yiming Ye, John K. Tsotsos, and Karen Bennet. Live world wide web and its agents: the cas lwww project. In *Second International Conference on Multiagent Systems: ICMAS'96*, Kyoto, Japan, December 1996.

[103] Seungku Yi. Automatic sensor and light source positioning for machine vision. In *10th International Conference on Pattern Recognition*, pages 55–59, New Jersey, USA, June 1990.

[104] J.Y. Zheng, F. Kishino, Q. Chen, and S. Tsuji. Active camera controlling for manipulation. In *Proc. Computer Vision and Pattern Recognition*, Washington, 1991.

[9] [44] [33] [30] [31] [75] [79] [36] [74] [56] [61] [70] [51] [82] [45] [83] [71] [28] [15] [60] [53] [49] [104] [11] [46] [62] [69] [84] [87] [18] [23] [103] [67] [54] [38] [10] [24] [63] [22] [7] [6] [2] [1] [3] [39] [41] [8] [85] [16] [12] [37] [42] [57] [58] [77] [78] [75] [87] [86] [93] [80] [94] [95] [96] [99] [102] [98] [100] [97] [92] [47] [48] [101]