Sensor Planning in 3D Object Search

by

Yiming Ye and John K. Tsotsos
Department of Computer Science
University of Toronto

Abstract

The task of object search is formulated and a mechanism of sensor planning for this task is proposed. The searcher is assumed to be a mobile platform equipped with a camera and a method of calculating depth, like stereo or a laser range finder. The search space is characterized by the probability distribution of the presence of the target. The goal is to reliably find the desired object with minimum effort. The control of the sensing parameters depends on the current state of the search space and the detecting ability of the recognition algorithm. In order to represent the surrounding environment of the camera and to efficiently determine the sensing parameters over time, a concept called the sensed sphere is proposed and its construction, using a laser range finder, is derived. The result of the sensing operation is used to update the status of the search space and this updated status is used to direct the next operation.

Contents

1 Introduction

Object search is the task of finding a given three-dimensional object in a given three-dimensional environment. It is clear that exhaustive, brute-force blind search will 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 how to select the sensing parameters so as to bring the target into the field of view of the sensor. This is actually the sensor planning problem for object search, which is the main concern of this document. The second is how to manipulate 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 ([Tso92a],[Baj85], [WT92]). The third is how to search for the target within the image. Actually this is the object recognition and localization problem, which attracts a lot of attention within the computer vision community (see ([CD86]) for a review).

Sensor planning for object search is very important if a robot wants to interact intelligently and effectively with its environment. For example, for a robot in its workspace, it often needs to use sensory information to determine the position of the objects in order to manipulate them. Normally speaking, 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 [Wil94] for further discussion). Also, for a robot to navigate in an industrial environment, in order to estimate its own position, it often needs to search the environment for known landmarks. 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 there is little research within the computer vision community ([Tso92a], [Wix92], [Gar76], [MB90], [RB92]).

Connell et al. [Connell89] constructed a robot that roams an area searching for and collecting soda cans. The robot is designed only to look for cans at a fixed height since soda cans are usually on tables. However, this may not be very practical as the cans may be present in other places. Wixson [Wixson92] analyzes the efficiency of indirect search over a wide range of situations that vary the spatial structure of the domain as well as recognition performances, and concludes that indirect search can improve efficiency in many situations. But his analysis is mainly $2D$ instead of $3D$. Rimey and Brown [RB92] have studied the problem of using the composite Bayes net and utility decision rule to decide where to point the sensor with respect to a table setting in their task-oriented system TEA. An expected area net is used to model the geometric relation between objects, the expected area for a node is calculated by combining messages about the expected areas sent to it from its parent and from all its children. Based on the belief values on the expected area net and the task net, a utility function will answer the question of where to look next by moving the camera to the center of mass of the expected area for a certain object. It is interesting to note that the operational research community has done a lot of research on optimal object search. But their approach is too general and thus it is inappropriate to directly apply their results here.

The work described here can be taken as the integration and extension of the work of [Tso92a], [RB92] and [Wix92]. We argue that the most important factors affecting the object search task are **the probability distribution of the location of the target** and **the recognition performance of the search algorithms**. We formulate the problem of object search and propose a practical algorithm. We also propose the idea of a **detection function** for a recognition algorithm. This function can serve as an evaluation matrix for any recognition algorithm. By introducing a statistical distribution, it is clear that the task of object search is **different** from the task of robot exploration and navigation. In order to represent the free space around the robot, we propose the idea of a **sensed sphere**[1]. The sensed sphere representation is very useful in deciding where to move next (the motion planning of the robot) and where to look next (deciding the viewing parameters of the camera).

2 Problem 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. From this point of view, two aspects play the key role in object search, namely, the probability and the cost. Almost every operation involved in search has something to do with these. The process of search is actually a process of updating the knowledge of the probability distribution of the target in the region.

To make the discussion easier, we use three coordinate systems. They are the world coordinate system, head coordinate system and camera coordinate system (see appendix 1 for detail). Also, we assume that the method used to get the depth information is by laser range finder, not by stereo. 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 we do not know its internal configuration. There is a world coordinate system that 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 will 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 and to analyze the image to find out whether the target is present in the image. The camera configuration is determined by the camera position (x_c, y_c, z_c) (x_c, y_c, z_c is the coordinate of the camera center with respect to the world coordinate system), the viewing axis direction (p, t) (p, t are in radians, the direction of p, t is the direction of the viewing axis with respect to the camera coordinate system, see Appendix 1 for detail), and the width and height of the solid viewing angle (w, h) (Note: we assume a rectangular image here, and the unit of w and h is radian). For convenience, here we use (w, h) instead of 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 $\mathbf{f} = \mathbf{f}(x_c, y_c, z_c, p, t, w, h, a)$, where a is the recognition algorithm used to search the image. This operation will return true if the target is found and false if the target is not found. The cost function $\mathbf{t_1}(\mathbf{f})$ of an operation $\mathbf{f}$ gives the time required to finish this operation, which is actually the time needed to take the picture under current configuration and run the search algorithm. The transition cost function $\mathbf{t_2}(\mathbf{f_i}, \mathbf{f_j})$ gives the time required to change the status of the camera configuration from action $\mathbf{f_i}$ to the status of the camera configuration of action $\mathbf{f_j}$. This is actually the time needed to calculate the next best configuration from $\mathbf{f_i}$ and the time needed for the mechanical operation to change the camera configuration status from $\mathbf{f_i}$ to $\mathbf{f_j}$. 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 t, $\mathbf{p}'((x, y, z), t)$

[1]We assume throughout the document that the depth is calculated by using a laser range finder, not stereo

gives the probability that the center of the target is at (x, y, z) when the time is t. 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: $\mathbf{p}(c_i, t) = \int_{c_i} \mathbf{p}'((x, y, z), t)dxdydz$.

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 $\mathbf{b}(c_i, \mathbf{f}) = \int_{c_i} \mathbf{b}'((x, y, z), \mathbf{f})dxdydz$ as the conditional probability of detecting the target given that the center of the target is located within c_i and the operation is $\mathbf{f}$. 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 (see Appendix 3 for detail). It is obvious that the probability of detecting the target by applying this action is given by

$$\int_{\Omega} \mathbf{p}'((x, y, z), t_{\mathbf{f}})\mathbf{b}'((x, y, z), \mathbf{f})dxdydz \tag{1}$$

where $t_{\mathbf{f}}$ is the time just before $\mathbf{f}$ is applied. After discretization, the probability is given by

$$\sum_{i=1}^{n} \mathbf{p}(c_i, t_{\mathbf{f}})\mathbf{b}(c_i, \mathbf{f}) \tag{2}$$

Let $\mathbf{O}_{\mathbf{\Omega}}$ be the set of all possible operations that can be applied, 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 $\mathbf{f}_i \in \mathbf{O}_{\mathbf{\Omega}}$.

The problem of object search can thus be defined as following: Suppose $\mathbf{K}$ is the total time that can be allowed in the search, then for any allocation $\mathbf{F} = \{\mathbf{f}_1, \ldots, \mathbf{f}_k\}$, the probability of detecting the target by this allocation is:

$$\begin{aligned} P[\mathbf{F}] \quad = \quad & \sum_{i=1}^{n} \mathbf{p}(c_i, t_{\mathbf{f_1}})\mathbf{b}(c_i, \mathbf{f_1}) \\ & + [1 - \sum_{i=1}^{n} \mathbf{p}(c_i, t_{\mathbf{f_1}})\mathbf{b}(c_i, \mathbf{f_1})] \times [\sum_{i=1}^{n} \mathbf{p}(c_i, t_{\mathbf{f_2}})\mathbf{b}(c_i, \mathbf{f_2})] \\ & + \ldots \\ & + \{ \prod_{j=1}^{k-1} [1 - \sum_{i=1}^{n} \mathbf{p}(c_i, t_{\mathbf{f_j}})\mathbf{b}(c_i, \mathbf{f_j})] \} \times [\sum_{i=1}^{n} \mathbf{p}(c_i, t_{\mathbf{f_k}})\mathbf{b}(c_i, \mathbf{f_k})] \end{aligned} \tag{3}$$

and the total time for applying this allocation is (following [Tso92b]):

$$T[\mathbf{F}] = \sum_{\mathbf{f} \in \mathbf{F}} \mathbf{t_1}(\mathbf{f}) + \sum_{1 \le i \le k-1} \mathbf{t_2}(\mathbf{f}_i, \mathbf{f}_{i+1}) + \mathbf{t_3}(\mathbf{f}_1) \tag{4}$$

where $\mathbf{t_3}(\mathbf{f}_1)$ is the time needed to set up the first operation.

The **task of object search** can be defined as finding an allocation $\mathbf{F} \subset \mathbf{O}_{\mathbf{\Omega}}$, which satisfies $T(\mathbf{F}) \le K$ and maximizes $\mathbf{P}[\mathbf{F}]$.

So, object search task is actually an optimization problem: to find the "best" sequence of actions that will maximize the probability of detecting the target given the condition that the total cost is less than a constant. In order to make the problem simpler, we can change the problem to a "one step look ahead problem": decide only which is the very next action to execute, consider the effect and the cost of that action.

3 The Detection Function, the Environment and the Sensed Sphere

3.1 The Detection Function

The detection function gives the detecting ability of the recognizer used in the search. The performance of each recognizer is influenced by various factors, such as the imaging parameters of the camera, the lighting condition, the complexity of the environment, the position of the target relative to the camera, the possibility of accidental view, the target and the task etc. We can use experiment to derive the detection function for an object recognizer. Let (θ, δ, l) be the relative position of the center of the target to the camera (see Appendix 1 for detail), where θ, δ is the horizontal and vertical direction offset to the viewing axis of the camera, l is the length of the projection of the line segment, which connects the camera center and the target center, on the viewing axis. We then put the target at this position and perform experiments under various conditions, such as light intensity, occlusion and background situation, and the relative orientation of the target with respect to the camera center. The final value of the detection function at (θ, δ, l) will be the total number of successful recognitions divided by the total number of experiments. These values can be stored in a look up table and retrieved when needed. Sometimes we may approximate these values by analytic formulas. Note: (1) (x, y, z) is the coordinate of the point in the world coordinate system, (θ, δ, l) is the relative position of the center of the target calculated according to the camera coordinate system; (2) the influence caused by the rotation of the target image about the camera axis is included in the consideration of the detection function value; (3) the value of $\mathbf{b}'((x, y, z), \mathbf{f})$ is also influenced more or less by previous actions (see Appendix 3 for detail). So, we can represent it as $\mathbf{b_H}'((x, y, z), \mathbf{f})$, where H represents the history of actions been applied before this action. Usually we will omit the subscript H and write it as $\mathbf{b}'((x, y, z), \mathbf{f})$ when there is no confusion. In the following discussion of this section, we assume no action been applied before the current action.

It is not necessary to record the detection function values of all different sizes of the solid viewing angles. We only need the detection values of one camera angle, and those of other camera angles can be approximately transformed to those of the known camera angle. Suppose we know the detection function values for operation $\mathbf{f}_0$ whose viewing angle size is w_0, h_0. We want to find the detection function values for $\mathbf{f}$ with angle size w, h. Suppose the center of the target is at (θ, δ, l) for $\mathbf{f}$, we now want to find the values of $(\theta_0, \delta_0, l_0)$ for $\mathbf{f}_0$ that have approximately the same detection value. To have the same detection value, we mean that when we use a recognition algorithm to analyze the picture taken with $\mathbf{f}$ and the picture taken with $\mathbf{f}_0$, we should get almost the same result. To guarantee this, the images of the target object should be almost same for both $\mathbf{f}$ and $\mathbf{f}_0$. This means that the two pictures must satisfy at least two geometric factors, namely the scale factor and the position factor. The scale factor means that the size of the projection of the target object on the image plane should be almost 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 at the same position on the image plane for both $\mathbf{f}$ and $\mathbf{f}_0$.

First we use the scale factor to find the value of l_0 when l is given (Figure 1(a)). The sizes of the projection of the target object on the image planes for $\mathbf{f}$ and $\mathbf{f}_0$ are approximately determined by l and l_0 respectively. The scale factor tells us that for a target patch on the viewing axis, the area of its projection on the image plane for $\mathbf{f}$ with distance l should be same as the area of its projection on the image plane for $\mathbf{f}_0$ with distance l_0. Suppose that the width of the image plane is m and the height of the image plane is n. Since the size of the image plane remains constant for different focal length, m and n will be same for any focal length. Let the height of the object patch be $2H$, the width of the object patch be $2W$. Then the height and width of the projected image of the target for $\mathbf{f}$ are $\frac{2fH}{l}$ and $\frac{2fW}{l}$ respectively, where f is the focal length for operation $\mathbf{f}$. Thus

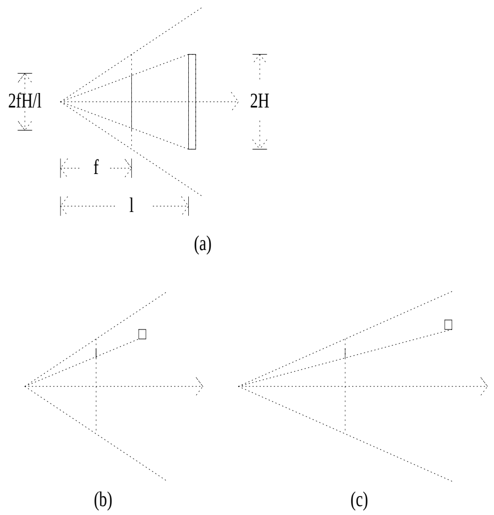


Figure 1: (a)The analyses of the scale factor. (b),(c) The position factor

the area of the target patch image is approximately

$$\frac{2fH}{l}\frac{2fW}{l}$$

Since

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

and

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

we get the area is

$$\frac{4f^2HW}{l^2} = \frac{4HW}{l^2}\frac{\frac{m}{2}}{tan(\frac{w}{2})}\frac{\frac{n}{2}}{tan(\frac{h}{2})} = \frac{HWmn}{l^2tan(\frac{w}{2})tan(\frac{h}{2})} \tag{5}$$

Similarly, the area of the image of the target patch for $\mathbf{f}_0$ with distance l_0 is

$$\frac{HWmn}{l_0^2tan(\frac{w_0}{2})tan(\frac{h_0}{2})} \tag{6}$$

To guarantee that the areas of the target patch be same for the two operations, we get (from (5), (6)):

$$l_0 = l\sqrt{\frac{tan(\frac{w}{2})tan(\frac{h}{2})}{tan(\frac{w_0}{2})tan(\frac{h_0}{2})}} \tag{7}$$

Second we use the position factor to find the value of θ_0, δ_0 when θ and δ are given (Figure 1(b)(c)). It is easy to know that for operation $\mathbf{f}$, the offset in the horizontal direction of the projection of the target center on the image plane to the image center is:

$$ftan(\theta) = \frac{\frac{m}{2}}{tan(\frac{w}{2})}tan(\theta) = \frac{m}{2}\frac{tan(\theta)}{tan(\frac{w}{2})} \tag{8}$$

Similarly, for operation $\mathbf{f}_0$ the horizontal offset is

$$\frac{m}{2}\frac{tan(\theta_0)}{tan(\frac{w_0}{2})} \tag{9}$$

To guarantee that this offset is same for two operations we get (from (8), (9))

$$\theta_0 = arctan[tan(\theta)\frac{tan(\frac{w_0}{2})}{tan(\frac{w}{2})}] \tag{10}$$

Similarly,

$$\delta_0 = arctan[tan(\delta)\frac{tan(\frac{h_0}{2})}{tan(\frac{h}{2})}] \tag{11}$$

So, when we want to find the detection function value of $\mathbf{f}$ with the expected target center at θ, δ, l, we can first find the corresponding θ_0, δ_0, l_0, then retrieve the detection value according to θ_0, δ_0, l_0 from the look up table of $\mathbf{f}_0$.

It is interesting to note that the above calculation is consistent with respect to the misfocus property of the two visual angles, see Appendix 2 for detail.

From above, we know that the general process of obtaining the detection function $\mathbf{b}'((x, y, z), \mathbf{f})$ is as follows. First, calculate the corresponding (θ, δ, l). Second, transform (θ, δ, l) into the corresponding $(\theta_0, \delta_0, l_0)$ of the standard operation $\mathbf{f}_0$. Third, retrieve detection value from the look up table, or get the detection value from a formula. The above calculation actually assumes that there is no other actions before the action $\mathbf{f}$. If there are actions before $\mathbf{f}$, then we will take the correlation between actions into our consideration. Please see Appendix 3 for a detailed discussion.

3.2 The Environment

The environment is assumed to be an office-like space or an industrial workspace. The robot knows the range 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 registered 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. 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 adaptation.

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_o be the event that the center of the target is outside the region, let β be the event that after applying a recognition action, the recognizer successfully detects the target. Since the above events $\alpha_1, \ldots, \alpha_n, \alpha_o$ are mutually complementary and exclusive, we can get the following formula (for $i = 1, \ldots, n, o$):

$$P(\alpha_i \mid \neg\beta) = \frac{P(\alpha_i)P(\neg\beta \mid \alpha_i)}{P(\alpha_o)P(\neg\beta \mid \alpha_o) + \sum_{j=1}^{n} P(\alpha_j)P(\neg\beta \mid \alpha_j)} \tag{12}$$

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. $P(\neg\beta \mid \alpha_o)$

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 $P(\neg\beta \mid \alpha_o) = 1$.

So, after each operation, the previous probability distribution $\mathbf{p}(c_i)$ and $\mathbf{p}(c_o)$ will be replaced by $P(\alpha_i \mid \neg\beta)$ and $P(\alpha_o \mid \neg\beta)$. Thus we have the following updating rule:

$$\mathbf{p}(c_i) \leftarrow \frac{\mathbf{p}(c_i)(1 - \mathbf{b}(c_i, \mathbf{f}))}{\mathbf{p}(c_o) + \sum_{j=1}^{n} \mathbf{p}(c_j)(1 - \mathbf{b}(c_j, \mathbf{f}))} \tag{13}$$

$$\mathbf{p}(c_o) \leftarrow \frac{\mathbf{p}(c_o)}{\mathbf{p}(c_o) + \sum_{j=1}^{n} \mathbf{p}(c_j)(1 - \mathbf{b}(c_j, \mathbf{f}))} \tag{14}$$

It is easy to see that after each failed detection operation, the probability of those cubes within the field of view of the camera will decrease and the probability of those outside the field of view of the camera will increase. The probability that the object is outside the search region will increase each time an operation is executed and the target is not found. This dynamic updating is very important for selecting the next action.

Figure 2 shows a $1D$ example of the environmental updating. 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 present in the region.

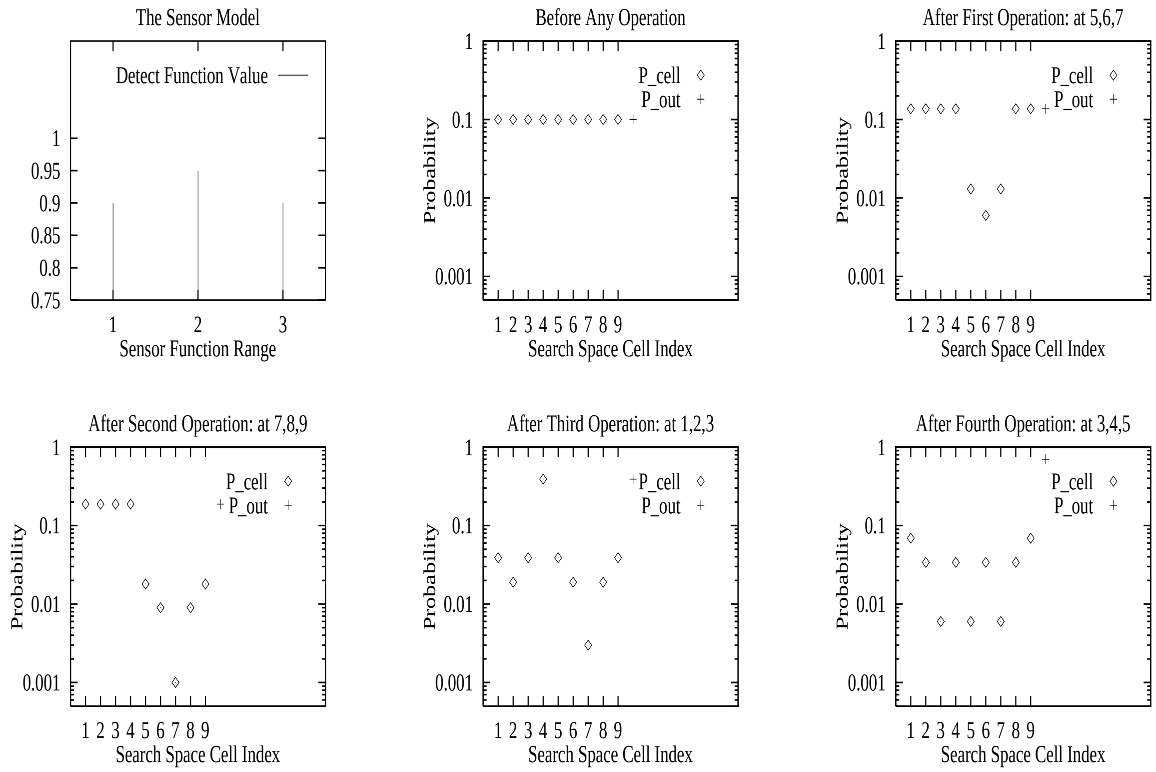


Figure 2: The sensor model and the probability update after four operations at: 5,6,7; 7,8,9; 1,2,3; and 3,4,5 respectively. Here we assume the independence between actions

3.3 Sensed sphere

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** [Tso92a]. We can use a laser to construct the sensed sphere. In order to do this, we need to **tessellate** the surface of the unit sphere so as to get a series of uniform or near uniform patches partitioning 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. But for our concern, it is not the best choice because the centers of those triangular facets are not regular, so we might often need to change the pan and tilt together in order to ping the next center. This is not efficient when the pan and tilt operations can not overlap in time, since the mechanical operation might waste time.

Figure 3(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-clockwisely 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 tessellating 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 (correspond to $tilt = \frac{\pi}{2}$) uniformly with angle α (so, for horizontal big circle, the value of pan can be 0, α, 2α,... see Figure 3(b)), then we divide the vertical big circle which corresponds to $pan = 0$ with the amount of *tilt* equal to α (so, for vertical big circle, *tilt* can be 0, α, 2α,..., see Figure 3(c)). The area of patches at the equator is approximately α^2. However, the area of each surface patch with this tessellation scheme will differ. In order to keep patch areas approximately equal, we adjust the amount of pan for each different *tilt*. In Figure 3(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. What we want to do is to find the value of *pan* for $tilt = \eta$. In order to make the area of the surface patch corresponding to $tilt = \eta$ approximately equal to the area of the surface patch corresponding to $tilt = \frac{\pi}{2}$, 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 3(d)). This follows $\mid AB \mid = \mid CD \mid$. After some calculations we can get the amount of pan corresponding to $tilt = \eta$ is

$$2arcsin\{\frac{sin(\frac{\alpha}{2})}{sin\eta}\} \tag{15}$$

Figure 4 shows an example of a tessellated sphere.

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} is the beginning and ending of *pan* and $tilt_{b_i}, tilt_{e_i}$ is the beginning and ending of *tilt* for the solid angle $SolidAngle_i$. $radius_i$ is the "radius" of this angle, which is the distance

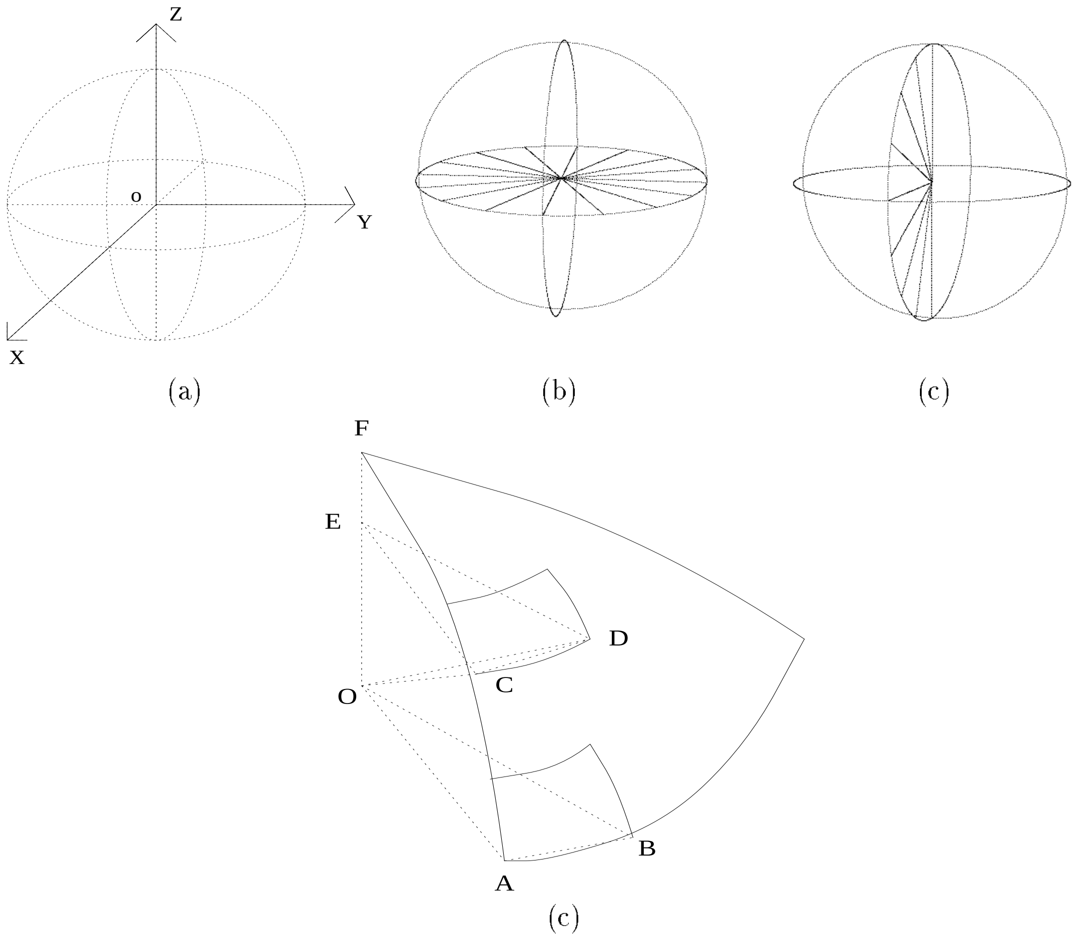
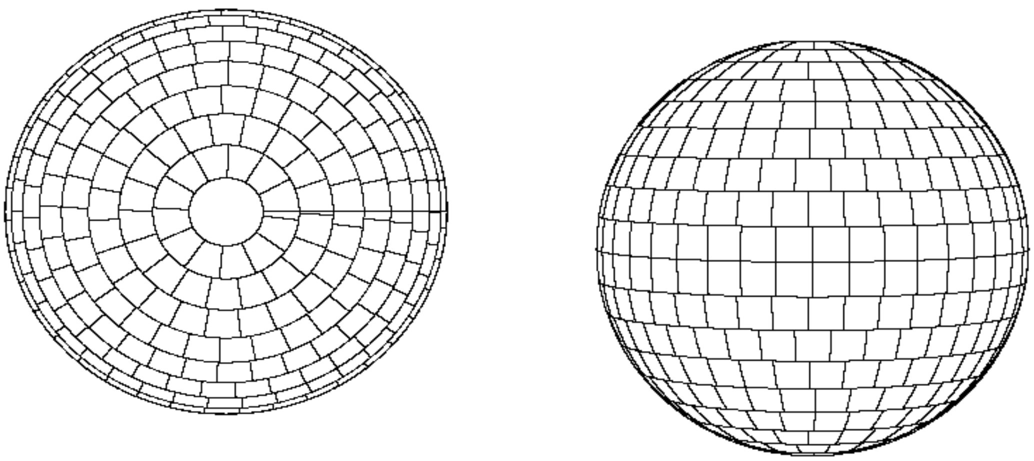


Figure 3: Tessellation of the unit sphere



(a) (b)

Figure 4: (a) The top view and (b) The side view of the tessellate sphere with $\alpha_0 = 0.157$ (9 degree)

measured using laser at $pan = \frac{pan_{b_i}+pan_{e_i}}{2}$ and $tilt = \frac{tilt_{b_i}+tilt_{e_i}}{2}$. $prob_i$ 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.

Sensed sphere can be used to register the solidity property of the search region. Since the sensed sphere is an approximation of the real situation surrounding the camera center, we have the following assumption during registration: for any part within a given solid angle, if its distance to the camera center is almost the same as the radius of the corresponding solid angle, then it is occupied by solid object. This is so because the laser, which emits from the center of the camera and along the central axis of the corresponding solid angle, hit a solid object with the distance equal to the radius of the solid angle during the construction of the sensed sphere. For each cube within the search space, we can first find the solid angle that the center of this cube belongs to. Then we calculate the distance between the center of this cube and the center of the camera. Lastly we compare this calculated distance with the radius of the corresponding solid angle and perform the registration. There are three situations to be considered during the registration. If the distance is almost the same as the corresponding radius, then the **solidity** of this cube is registered as *solid*. If the distance is less than or greater than the length of the radius by a certain amount, then the **solidity** of this cube is registered as *unknown*.

The next two sections will illustrate how to use sensed sphere to decide where to look next and where to move next.

4 Where to look next

As noted in section 2, 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 (3)) given the condition that the total cost $T[\mathbf{F}]$ (see formula (4)) is less than a constant. Since it seems that this task may be intractable in the general case (although proof not here, form is similar to Knapsack problem), we make this problem simpler by changing it to a "one step look ahead problem": decide only which is the very next action to execute and consider the effect and cost of that action. The approach described in the following is actually very similar to the approach of approximately solving Knapsack problem through linear programming method.

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 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

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

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 $\mathbf{t}(\mathbf{f})$.

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

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

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

Our strategy of 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

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

is fixed no matter what the 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

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

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

We further divide this problem into two sub-problems: the "where to look next" problem and the "where to move next" problem. The "where to look next" problem answers the question of how to select the field of view and the direction of the viewing axis of the camera when the robot position is fixed. The "where to move next" problem answers the question of how to select the next robot position when the probability of presence of the target within current sensed sphere is too small.

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, 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 space region of the viewing volume determined by p,t,w,h is V and the space region confined by the current sensed sphere is C. Let $\Psi = V\bigcap C$, then, of course, Ψ is those parts of Ω which can be "seen" by the current camera setting. So the probability of detecting the target by operation $\mathbf{f} = \mathbf{f}(p,t,w,h,a)$ is:

$$\int_{\Psi}\mathbf{p}'((x,y,z),t_{\mathbf{f}})\mathbf{b}'((x,y,z),\mathbf{f})dxdydz \tag{16}$$

After discretization, the probability will be

$$\sum_{c_i\in\Psi}\mathbf{p}(c_i,t_{\mathbf{f}})\mathbf{b}(c_i,\mathbf{f}) \tag{17}$$

If $\mathbf{f}$ is not the first action to be executed, the cost $\mathbf{t}(\mathbf{f})$ by applying operation $\mathbf{f}$ is $\mathbf{t_2}(\mathbf{f}^*,\mathbf{f})+\mathbf{t_1}(\mathbf{f})$, where $\mathbf{f}^*$ is the action just before $\mathbf{f}$ is applied. If $\mathbf{f}$ is the first action to be executed, the cost $\mathbf{t}(\mathbf{f})$ will be $\mathbf{t_1}(\mathbf{f})+\mathbf{t_3}(\mathbf{f})$ (see section 2). The utility function of operation $\mathbf{f}$ is

$$\mathbf{E}(\mathbf{f}) = \frac{\int_{\Psi}\mathbf{p}'((x,y,z),t_{\mathbf{f}})\mathbf{b}'((x,y,z),\mathbf{f})dxdydz}{\mathbf{t}(\mathbf{f})} \tag{18}$$

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 operations, 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 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 $\int_\Psi \mathbf{p}'((x,y,z), t_{\mathbf{f_a}})\mathbf{b}'((x,y,z),\mathbf{f_a})dxdydz$. After discretization, the problem becomes to select an operation $\mathbf{f_a}$ that maximizes

$$\sum_{c_i \in \Psi} \mathbf{p}(c_i, t_{\mathbf{f_a}})\mathbf{b}(c_i, \mathbf{f_a}) \tag{19}$$

The following sections will illustrate how to select the parameters p, t, w, h when the recognition algorithm is given.

4.1 The selection of 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 every possible viewing angle size one by one; what we are trying to do is 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, this may also cause the recognition algorithm fail to detect the target. When the target can be totally brought into the field of view of the camera and the features 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 operation. 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. 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 $[N_0, F_0]$. This means that N_0 is the beginning of the effective range for this viewing angle, and F_0 is the ending 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 $i \neq j$. So, 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 $N_1 = F_0$. To guarantee that the area of the image of the target patch of the first viewing angle $w_0 \times h_0$ at 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 (use equation 7)

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

Similarly, we can get

$$w_i = 2arctan[(\frac{N_0}{F_0})^i tan(\frac{w_0}{2})] \tag{20}$$

$$h_i = 2arctan[(\frac{N_0}{F_0})^i tan(\frac{h_0}{2})] \tag{21}$$

$$N_i = F_0(\frac{F_0}{N_0})^{i-1}; F_i = F_0(\frac{F_0}{N_0})^i \tag{22}$$

where w_i, h_i is the ith angle size we select, $[N_i, F_i]$ is the corresponding effective range.

Since $N_i \leq D$, we can get

$$i \leq \frac{ln(\frac{D}{F_0})}{ln(\frac{F_0}{N_0})} - 1 \tag{23}$$

Let

$$n_0 = \lfloor \frac{ln(\frac{D}{F_0})}{ln(\frac{F_0}{N_0})} - 1 \rfloor \tag{24}$$

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

Figure 5 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 5(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 5(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 5(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 5(d) shows that for a relatively large characteristic ratio, only a few sizes of viewing angle are enough for an object search task.

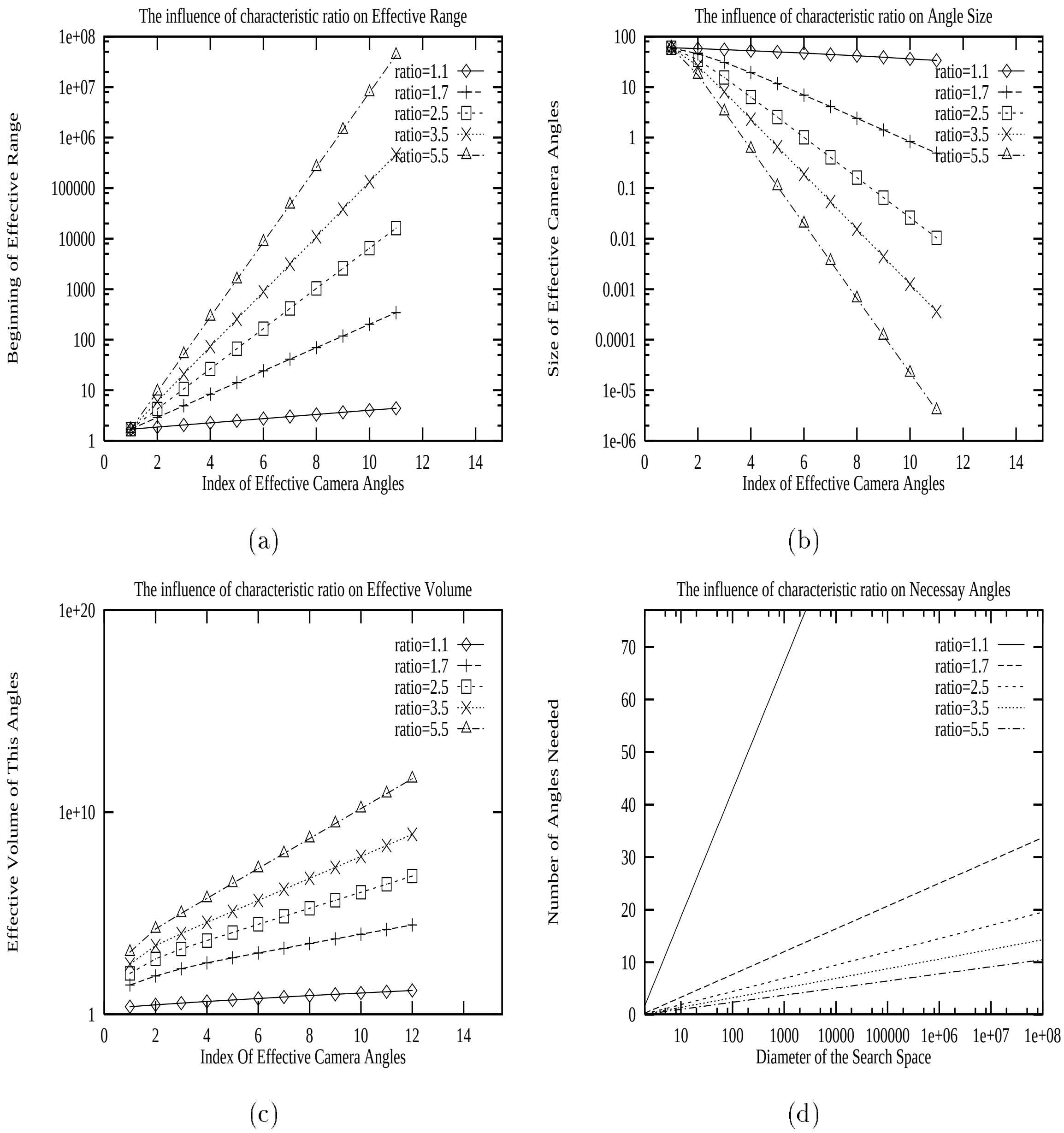


Figure 5: The Influence of Characteristic Ratio when $N_0 = 1.7$, $w_0 = h_0 = 1.046$ radian (actually 60^o degree)

4.2 The layered sensed sphere for a given recognition algorithm

For a given recognition algorithm, 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 $w_i \times h_i$ is the effective angle sizes determined by the method of section 4.1. Each layer corresponds to an effective viewing angle. It can be represented as follows:

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

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

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: (1) it is within $SolidAngle_i$ (2) the distance to the camera center is less than $radius_i$ (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 $F_{<w,h>}$.

4.3 The selection of the viewing direction for a given recognition algorithm and a given effective angle size

The layered sensed sphere (LSS) for the given recognition algorithm 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 and select the best one.

The selection of the candidate directions is the same as the tessellation of the sensed sphere. Our idea is that we only consider those directions whose viewing volumes will not overlap with each other and whose union of viewing volumes will cover the whole sphere. Thus we can use the same tessellation method as for the construction of the sensed sphere, with the tessellating angle size equals to $min\{w_k, h_k\}$. The central axis of each solid angle will be taken as the candidate direction.

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 $\mathbf{V}$ 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 $\mathbf{V}$.

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 $\frac{\pi}{2} - t$. So, in the new coordinate system, the region covered by this view volume can be approximately represented as $\frac{\pi}{2} - \frac{h_k}{2} \leq tilt \leq \frac{\pi}{2} + \frac{h_k}{2}$ and $-\frac{w_k}{2} \leq pan \leq \frac{w_k}{2}$. This means that in the new coordinate system, any radius whose $< tilt, pan >$ satisfies the above conditions will fall into the viewing volume $\mathbf{V}$.

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 $\frac{\pi}{2} - \frac{h_k}{2} \leq tilt_{new} \leq \frac{\pi}{2} + \frac{h_k}{2}$ and $-\frac{w_k}{2} \leq pan_{new} \leq \frac{w_k}{2}$, then the radius with direction $< t_i, p_i >$ belong to the viewing volume $\mathbf{V}$.

Since $prob_i^{<w_k,h_k>}$ gives the probability of presence of the target object within both the solid angle $SolidAngle_i$ 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 radius fall into the view volume V:

$$
\sum_{radius_i \in \mathbf{V}} prob_i^{<w_k,h_k>} \tag{25}
$$

Thus we can perform calculations (according to formula (25)) for each candidate direction t, p and select the direction with the maximum value as the best direction $< p_k, t_k >$ for the corresponding angle size $< w_k, h_k >$.

4.4 The selection of 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 4.1 (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 4.3. 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 (19) 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 (18). 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 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, we will want the robot to change position.

5 Where to move next

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

As described in section 2, 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. So, there are only a limited number of next candidate positions to be considered.

These next positions can be further reduced by considering whether the robot can reach these positions from its current position. 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.

Now we will select the next robot position among these reachable candidate positions (Figure 6). For each reachable candidate position of the robot, we can **calculate** its corresponding **expected sensed sphere** SS_e as follows. First, we construct the expected sensed sphere by just taking the boundary of the search region Ω (this is assumed to be known) as solid and **assume** that we ping laser around at this candidate position (Figure 6(d)), thus we can get the radius for each solid angle of SS_e. Then we will update these radii according to the current registration of the search space Ω. For each solid cube and its 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 will replace the radius of the solid angle by this distance (Figure 6(e)). Otherwise, the radius of the solid angle will not be updated. In this way, we constructed the expected sensed sphere SS_e for the current candidate position. Each candidate position will associate with an expected sensed sphere constructed by the above method. For each expected sensed sphere, 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 $\sum_{c_i \in SS_e} \mathbf{p}(c_i)$. So, for each candidate position, we can get its corresponding probability of presence of the target. We will then select the position whose corresponding expected sensed sphere has the maximum probability of presence of the target as our next position.

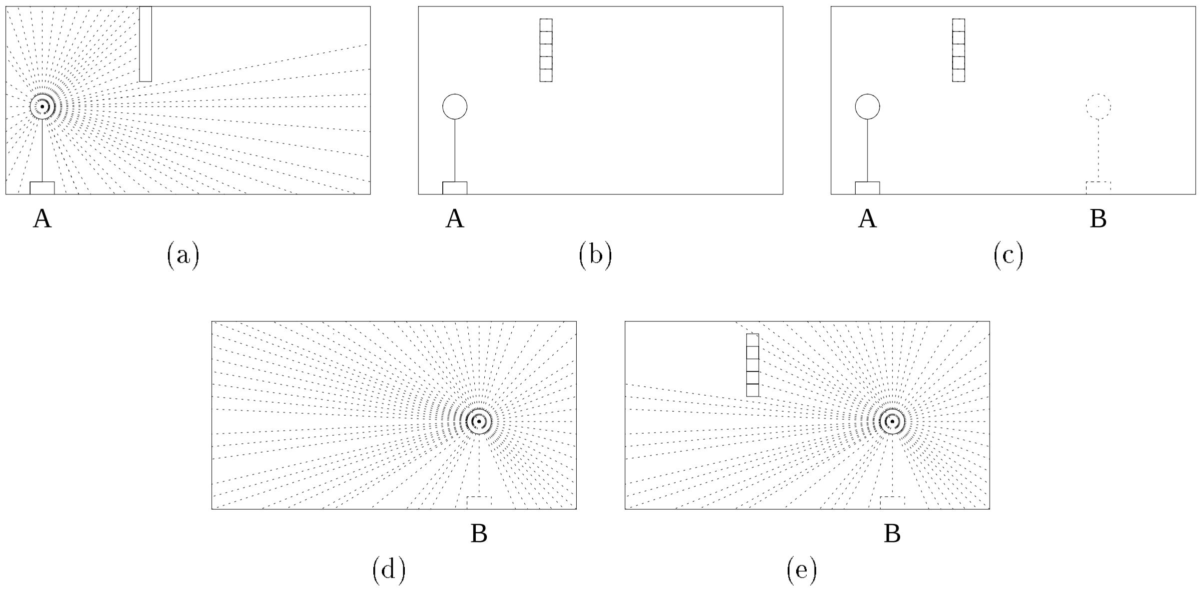


Figure 6: The construction of the expected sensed sphere. (a) The sensed sphere constructed at the current robot position A. (b) The solid cubes with respect to current sensed sphere. Those solid cubes near the boundary of the search region will not be considered. (c) B is one of the reachable candidate positions. (d) The expected sensed sphere by taking the boundary of the search space as solid and assuming we ping laser around at position B. (e) The final expected sensed sphere with respect to the reachable candidate position B after updating using the solid cubes.

After we find the next position, we drive the robot to this position and begin our new search. Since the property of the cube may not correctly represent the real situation, the robot may stop at the middle of the journey. If it cannot reach the next position, then it will begin the “where to look next” part at the place it stopped.

One thing need to note here is how to deal with the inaccuracy of the robot movement. We assume here that the error of the robot motion is within a certain small range. This can be realized by using landmarks etc.. When we move to a new place, the probability of presence of each cube will be the sum of the probability of its surrounding cubes within the error range divided by the number of these surrounding cubes. We need to normalize the result probability by making total probabilities equal to one.

5.2 The combination of the registration space

We know from section 3.3 that the sensed sphere can be used to register the search space (assigning the values of solidity to each cube). After we drive to the other place, we will use 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 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. So there is no spatial registration problem for cubes. We only need to assign the solidity property to each registered cube when the robot move from one position to another.

5.3 When to stop searching

The stopping criteria is determined by the benefit that we can get 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 maximum probability of presence of the next robot position (section 5.1) is less then a certain threshold, then we will stop the search process.

6 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. This model is based on the robot ARK (Autonomous Robot for a Known environment) project which 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 ARK robot is composed of a 3 wheeled Cybermotion K2A platform and a head. The platform is equipped with a small number of sonar transducers. The head is an active vision sensor combining a color video camera and zoom and focus controlled lens with a laser range finder pointing in the same direction (see Figure 7). 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 [NJM+93] 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 with 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.

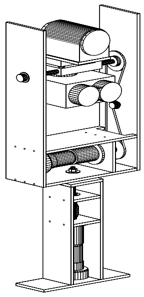


Figure 7: The ARK robot head. At the top is the zoom and focus controlled lens, and at the bottom is the Optech laser range finder. The pan-tilt unit is operated by two DC servocontrolled motors from Micromo Electronics, equipped with gearboxes and optical encoders

The algorithm we used in the simulation is following:

(1) Determine the necessary angle sizes according to the diameter of the search region (see section 4.1)
(2) Generate the layered sensed sphere corresponding to those angles in (1) (see section 4.2)
(3) For each angle size, find its best direction (section 4.3)
(4) Use formula (19) 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 3.2). Exit when find the target
(6) When the probability within current sensed sphere is **not** small enough (see section 4.4), goto (2);
(7) When the probability that the target is within the search region is small enough (see section 5.3), stop search
(8) Determine the next candidate positions, and calculate the probability of those expected sensed spheres (see section 5.1)
(9) Select the best position (section 5.1) and drive the robot to that position
(10) Combine the search region of the two positions (section 5.2).
(11) Go to (2) and begin the search at new position

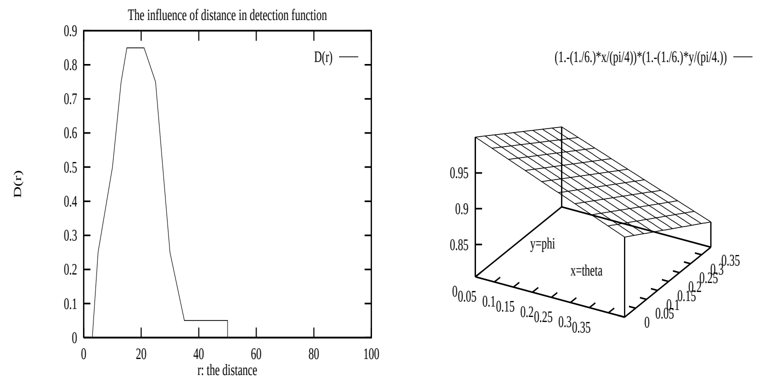
In our simulation experiments, the biggest angle for the sensor is $\pi/4$, and we assume only one recognition algorithm is available. The detection function for $\pi/4$ is:

$$b(c, f_{\pi/4}) = b(r, \varphi, \theta) = D(r)(1 - \frac{1}{6}\frac{\varphi}{\frac{\pi}{4}})(1 - \frac{1}{6}\frac{\theta}{\frac{\pi}{4}}) \tag{26}$$

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, 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 8. The effective region for $\pi/4$ is $[11, 27]$.



(a) (b)

Figure 8: Left: the influence of the distance r. Right: the influence of the angle φ, θ

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

In one experiment, the search region is a room with size $50 \times 50 \times 25$. The horizontal vertices of the room is $(0, 0)$, $(50, 0)$, $(50, 50)$, $(0, 50)$ respectively. There is one big obstacle with size $5 \times 37 \times 19$ present in the space that can cause occlusion. The horizontal vertices of the obstacle is $(36, 10)$, $(41, 10)$, $(41, 47)$, $(36, 47)$ respectively. The height of the robot is 15, and it is at $(45, 10)$ before the search. See Figure 9 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 it is within the room or outside the room. So 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).

The robot will first construct the sensed sphere (Fig. 9).

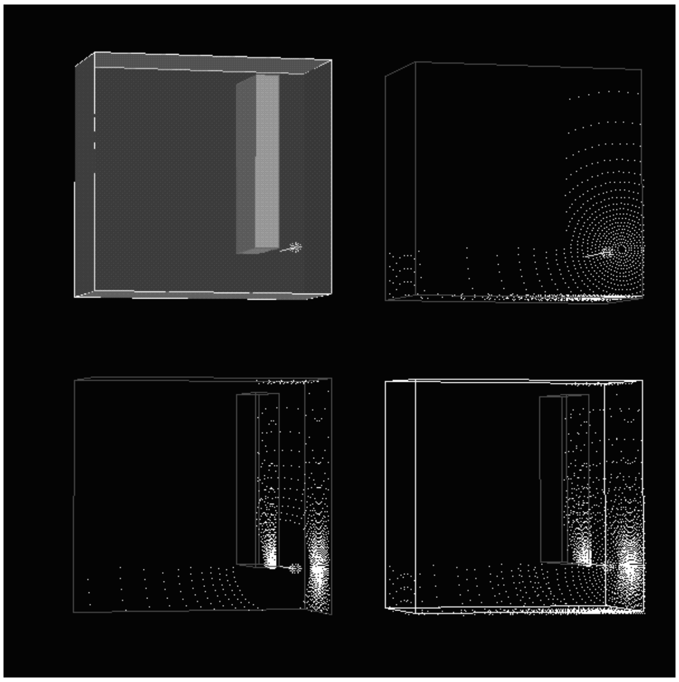
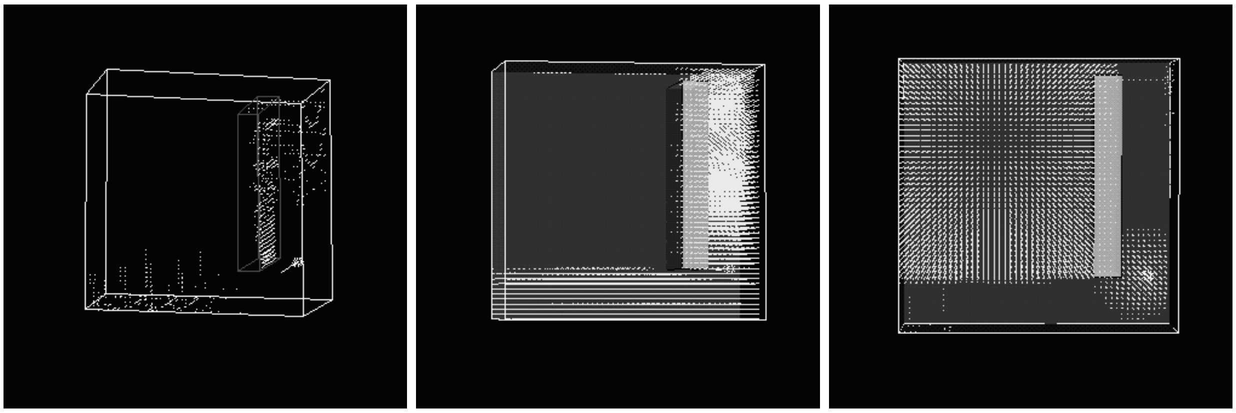


Figure 9: Top left: the environment for experiment 1, there is a big obstacle within a room, robot is at $(45, 10)$; Top right: a part of sensed sphere constructed at $(45, 10)$; Bottom Left: another part of sensed sphere constructed at $(45, 10)$; Bottom right: the complete sensed sphere constructed at $(45, 10)$. 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 this two figures combined together, we will get the lower right figure.

And then it will register the space (Figure 10). There are 832 solid cubes.



(a) (b) (c)

Figure 10: (a) the solid cube; (b) cubes without occlusion with respect to camera position ; (c) cubes with occlusion with respect to camera position

For each angle size, the robot will first select 5 actions based on the layered statistics sensed sphere, it will then use the detection function to compare these 15 actions and select the top 5 among them. Then it will execute those actions that belong to the top 5 and have not been executed before. After executing each action, the search region will be updated. This process is continued until the probability of current sensed sphere with respect to the probability of the search space is less then 0.16. The actual executed actions is shown in the following table and the top 7 actions are illustrated in Figure 11. We can see that these actions are correctly selected.

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

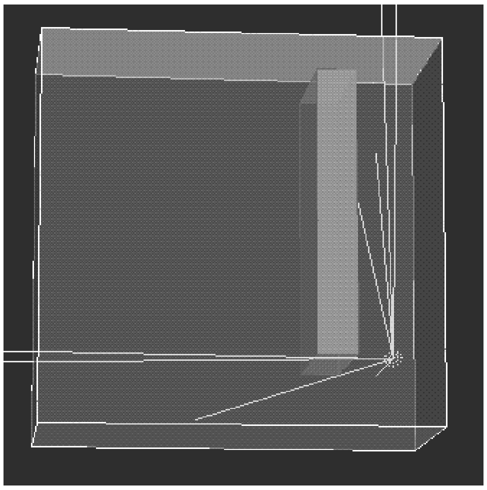


Figure 11: The top 7 actions selected. The long line correspond to the viewing axis of angle size 22.166346^o. The short line correspond to the viewing axis of angle size 45^o

Figure 12 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. We can see that the outside probability is increasing and the current sensed sphere probability and its percentage within the search space is decreasing, after each phase of actions.

Figure 13 shows the probability of detection from the layered sensed sphere and the real calculated probability 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

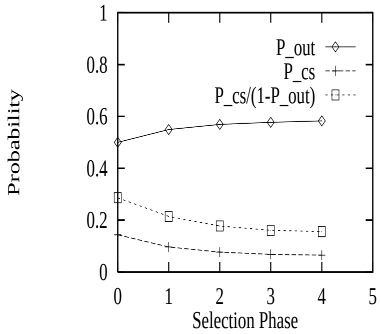


Figure 12: The probability change after each phase

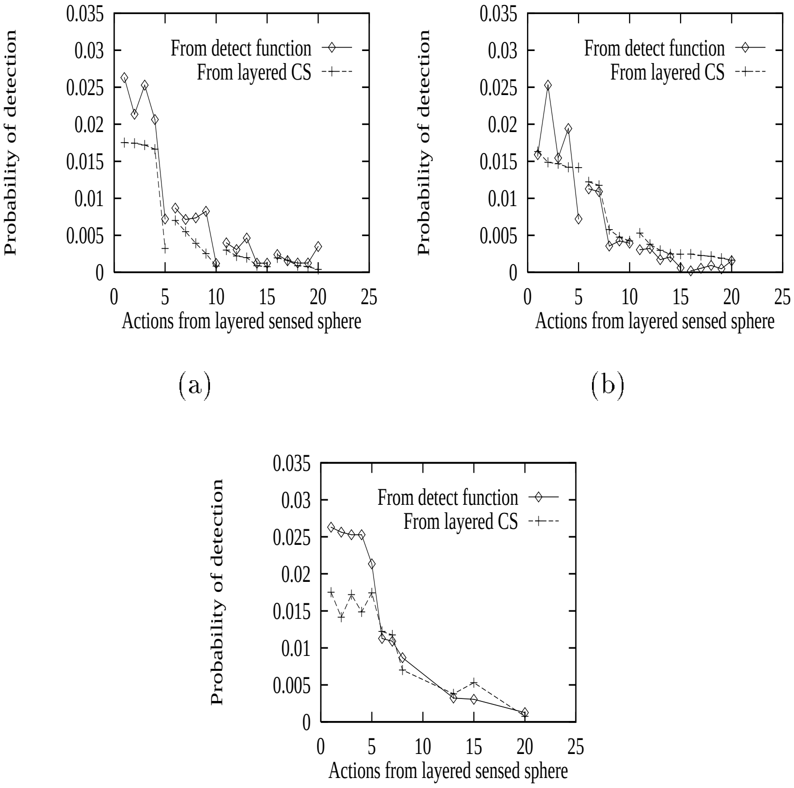


Figure 13: Comparison of the probability from layered sensed sphere and those from detection function. Upper left: For angle 0.785398. Upper right: For angle 0.386876. Bottom: For real executed actions

sphere is very useful in selecting the candidate actions. We can also see that the more actions being executed, the less benefit we can get.

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 position when $x = 1, 6, 11, \ldots, 46$ while $y = 1, 2, \ldots, 50$. Figure 14 shows this probability for each candidate position. Among them, the position $(36, 1)$ has the maximum probability 0.371333. So the robot will move to $(36, 1)$ and begin the new search.

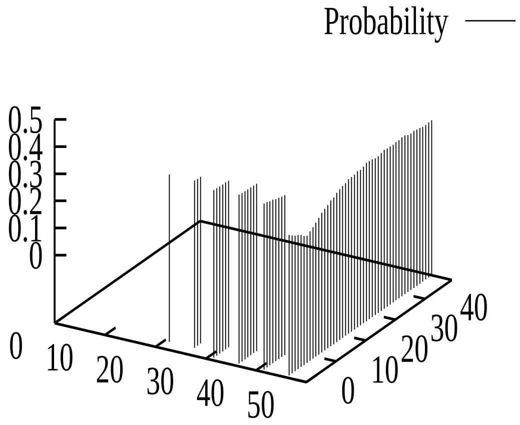


Figure 14: The probability of presence for the expected sensed sphere of each candidate position

Figure 15 shows the expected sensed sphere before the constrain using the solid cube.

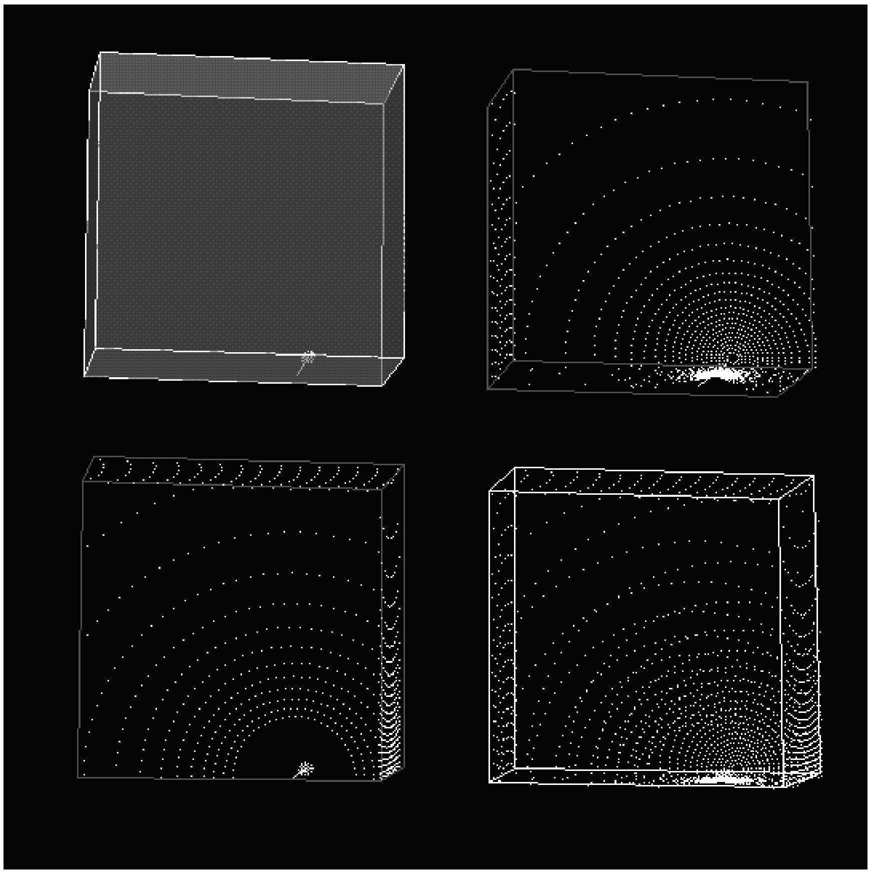


Figure 15: Top left: the expected environment for search space; Top right: a part of expected sensed sphere at $(36, 1)$ before constraining using the solid cube; Bottom Left: another part of expected sensed sphere at $(36, 1)$ before constraining using the solid cube; Bottom right: the complete expected sensed sphere at $(36, 1)$ before constraining using the solid cube; Among the figures above, only 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 this two figures combined together, we will get the lower right figure.

Figure 16 shows the expected sensed sphere after the constrain using the solid cube.

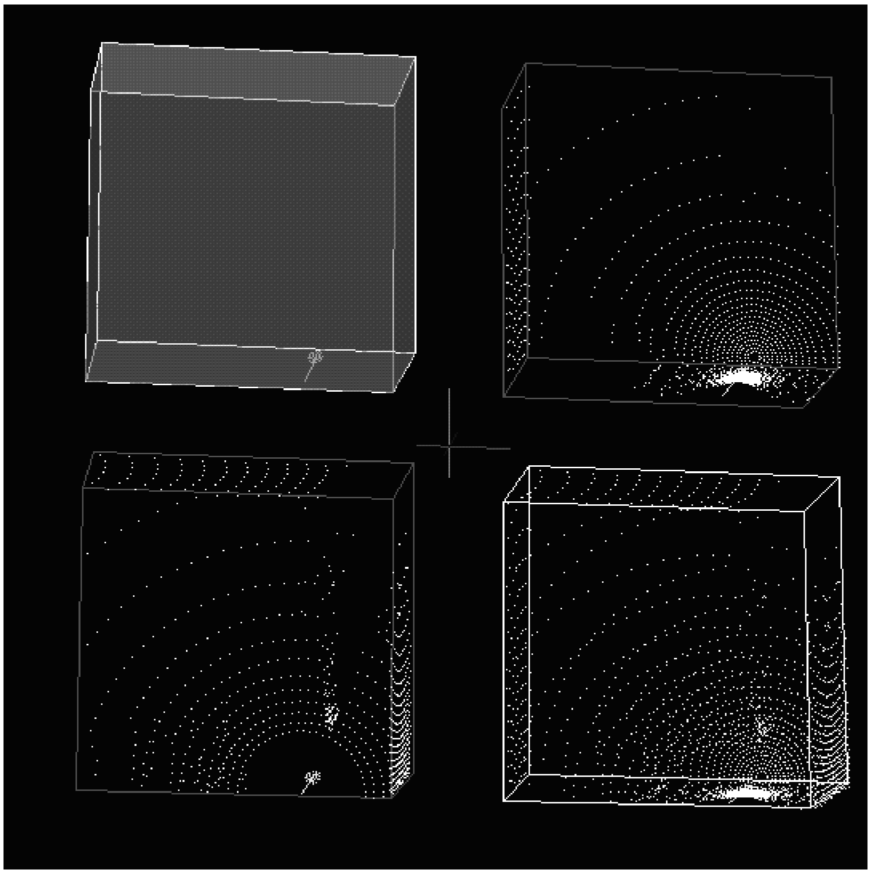


Figure 16: Top left: the expected environment for search space Top right: a part of expected sensed sphere at $(36, 1)$ after constraining using the solid cube; Bottom Left: another part of expected sensed sphere at $(36, 1)$ after constraining using the solid cube; Bottom right: the complete expected 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 just shows a part of the sensed sphere, there is no overlap between them. When this two figures combined together, we will get the lower right figure.

Figure 17 shows the environment and the sensed sphere when the robot drives to $(36, 1)$. Figure 18 shows the solid cubes, the cubes without occlusion and cubes with occlusion when the robot drives to $(36, 1)$.

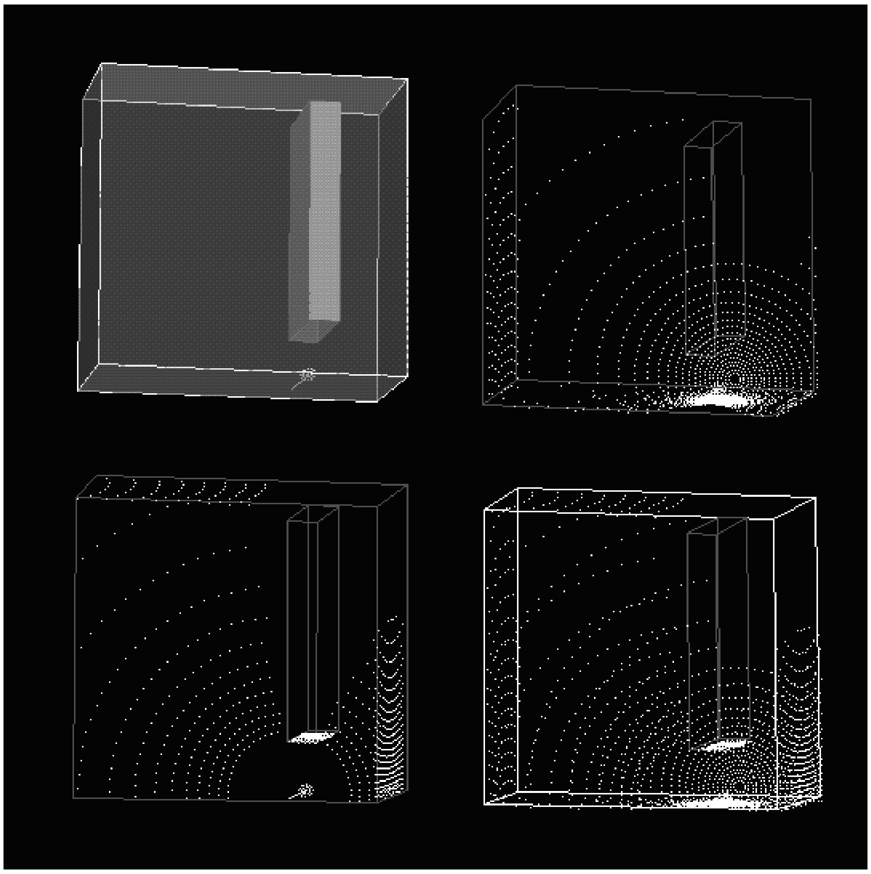
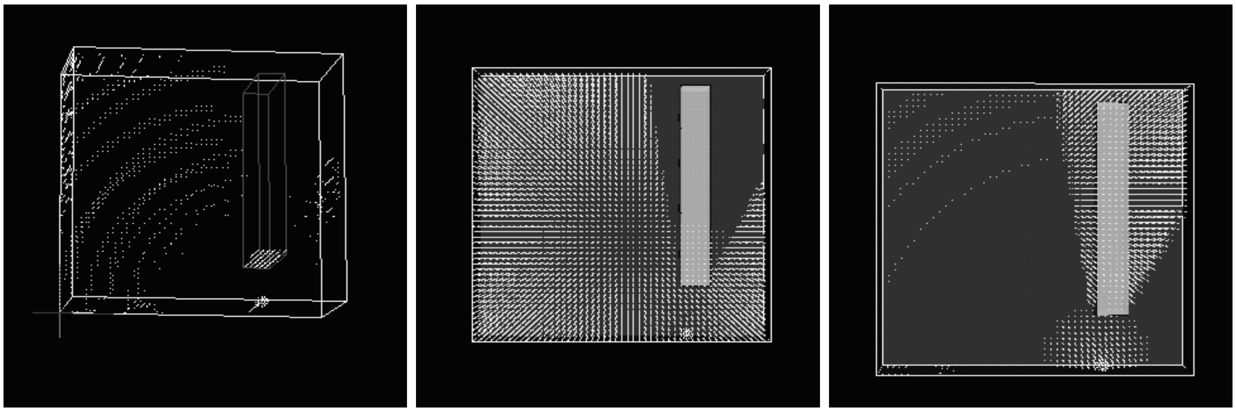


Figure 17: Top left: the environment; Top right: a part of sensed sphere at $(36, 1)$. Bottom Left: another part of sensed sphere at $(36, 1)$. Bottom right: the complete sensed sphere at $(36, 1)$. Among the figures above, only bottom right shows the complete sensed sphere together with the environment. 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 this two figures combined together, we will get the lower right figure.

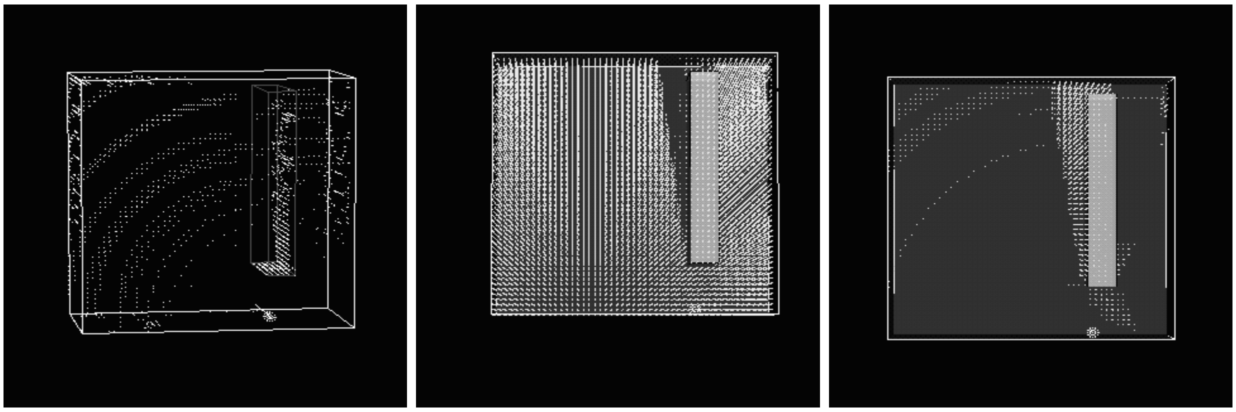
Figure 18 shows the solid cubes, the cubes without occlusion and cubes with occlusion when the robot drives to $(36, 1)$.



(a) (b) (c)

Figure 18: (a) the solid cube; (b) cubes not occluded with respect to the camera position; (c) cubes occluded with respect to the camera position;

Figure 19 shows the combined solid cubes and the combined occlusion situation of cubes.



(a) (b) (c)

Figure 19: (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.

Example 2 is designed to show that our scheme can effectively incorporate the previous knowledge into the search process. The search space is a room $119 \times 100 \times 41$. The horizontal vertices of the room is $(0,0)$, $(119,0)$, $(119,100)$, $(0,100)$ respectively. There is a long table $100 \times 10 \times 11$ present in the room. The horizontal vertices of the table is $(11,65)$, $(111,65)$, $(111,75)$, $(11,75)$ respectively. 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 20 shows the environment and the sensed sphere constructed at this position.

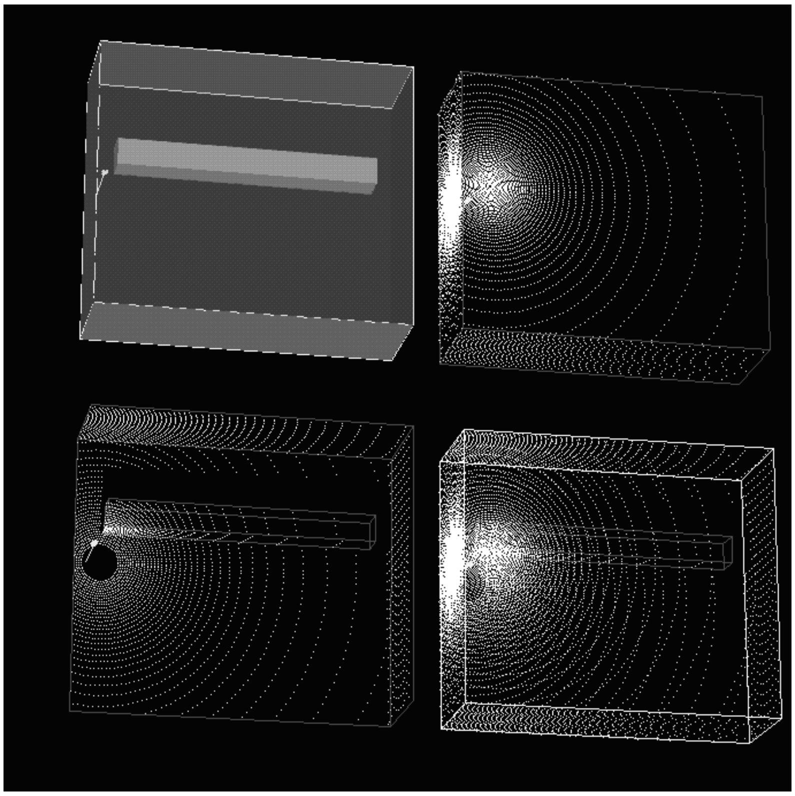
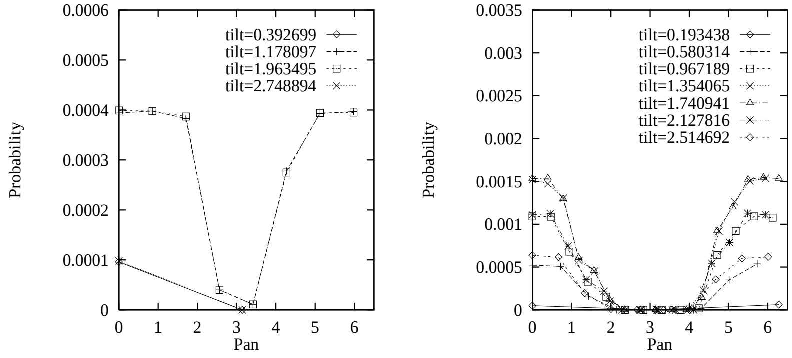


Figure 20: Top left: the environment for experiment 2, there is a big table in a room, robot is at $(11,55)$; Top right: a part of sensed sphere constructed at $(11,55)$; Bottom Left: another part of sensed sphere constructed at $(11,55)$; Bottom right: the complete sensed sphere constructed at $(11,55)$

Figure 21 and Figure 22 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 layered sensed sphere is listed below:



(a) (b)

Figure 21: The probability calculated from the layered sensed sphere for various candidate $(tilt, pan)$ (a) when $angle = 0.785398$; (b) when $angle = 0.386876$

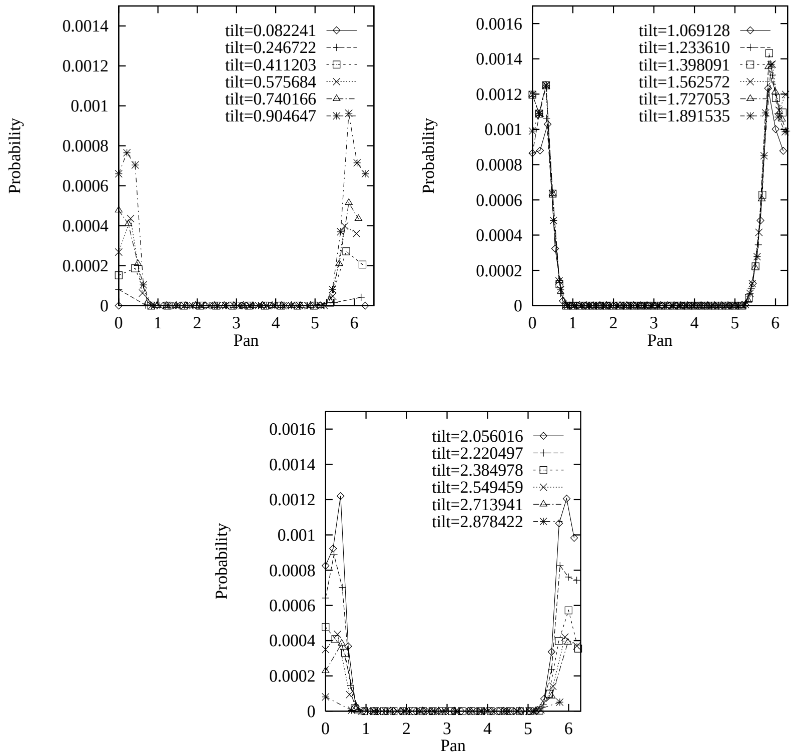


Figure 22: The probability calculated from the layered sensed sphere for various candidate $(tilt, pan)$ When $angle = 0.164481$

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 detection function among them is listed in the following table and is illustrated in Figure 23:

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

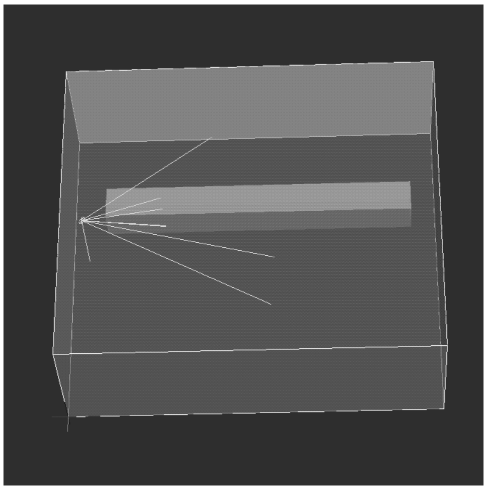


Figure 23: The top 7 actions selected for experiment 2. The long line correspond to the viewing axis of angle size 22.166346^o. The short line correspond to the viewing axis of angle size 45^o

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

7 Discussion

We formulated the object search scheme in this document. By introducing the concept of detection function for an object recognition algorithm and the concept of probability distribution, we argue that the object search problem is quite different from the navigation and exploration problem. The sensed sphere concept ([Tso92a]) described in this document has been proved to be a very powerful representation in selecting the imaging parameters for object search. We believe that both the concept of detection function and the concept of sensed sphere may find their applications in other vision and robotics fields.

The simulation experiments described here prove that our scheme is useful in the object search task. Open problems for future research might include (1) how to efficiently look for the target within an image. (2) how to manage the inaccuracy which occurs when the robot moves from one place to another, especially when there is no map. (3) how to formulate a cost function that explicitly quantifies the cost of camera motion, robot motion, and recognition. (4) how to get a better criteria for when to stop searching. (5) how to apply the mechanism here to other forms of hardware, for example, when the height of the camera center can be changed.

Acknowledgment: The second author is the CP-Unitel Fellow of the Canadian Institute for Advanced Research. The first author is grateful to Dr. Dave Wilkes and Dr. Piotr Jasiobedzki for their valuable comments, suggestions and generous help, to Victor Lee for introducing and help with the GL library.

8 Appendix

8.1 Appendix1: the coordinate systems

The coordinate systems used in this document are world coordinate system, head coordinate system and camera coordinate system. The world coordinate system is fixed, which 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 24 (a)). The head center can be moved by moving the robot platform. We assume the center of the head is the same as the nodal point of the camera. The viewing axis of the camera (Figure 24 (b)) is a half line that originated from the origin of the head coordinate system. The laser is assumed to be able to emit along the viewing axis of the camera. The direction of the viewing axis is specified by pan and tilt (p, t) (Figure 24 (c)). 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 system. The direction of the Z axis of the camera coordinate system is the same as the direction of the view axis (Figure 24 (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 24 (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 24 (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 is the horizontal and vertical direction offset to the viewing axis of the camera, l is the length of the projection of the line segment, which connects the camera center and the target center, on the viewing axis. Actually, considering the symmetry property of the camera viewing volume, we have:

$$\begin{array}{l} \theta = \mid arctan(\frac{x}{z}) \mid \\ \delta = \mid arctan(\frac{y}{z}) \mid \\ l = \mid z \mid \end{array}$$

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

$$\begin{array}{l} \theta \leq w \\ \delta \leq h \\ z > 0 \end{array}$$

8.2 Appendix 2: the consistence with misfocus property

It is interesting to note that the calculation in section 3.1 is consistent with respect to the misfocus property of the two visual angle sizes: the blur circle for $\mathbf{f}$ when the object distance is l is the same as the blur circle for $\mathbf{f}_0$ when the object distance is l_0. It is clear that from the *Gaussian lens law*, only points within one particular object plane are brought into focus on the image plane by an ideal lens. This distance is called *focusing distance*. A point which is near than the focusing distance is brought into focus to a place behind the image plane, while 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 which are at infinity are brought into focus. Those that are nearer than infinity are imaged as blur circles, each blur circle being formed by the intersection of the corresponding cone of light rays with the

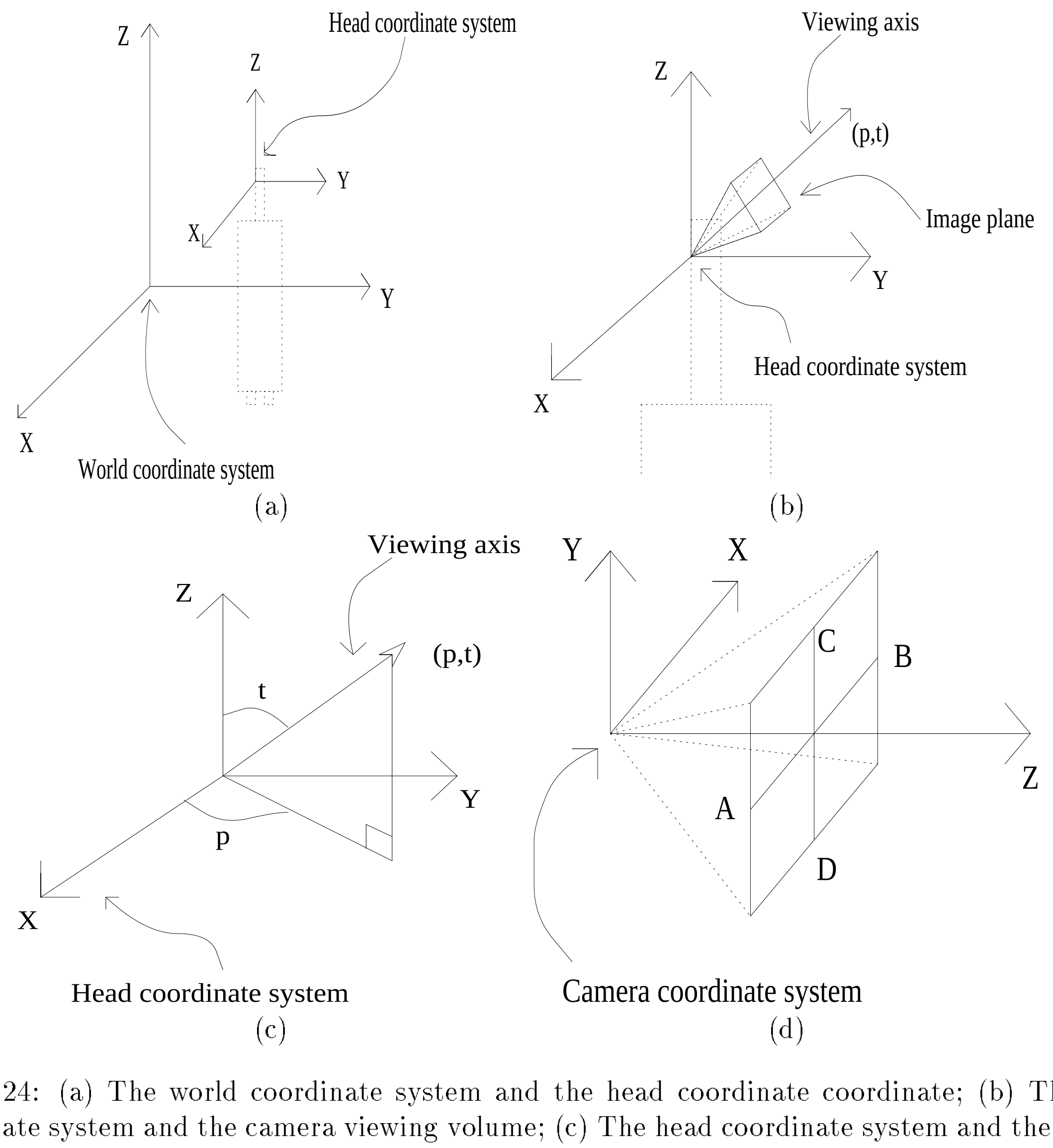


Figure 24: (a) The world coordinate system and the head coordinate coordinate; (b) The head coordinate system and the camera viewing volume; (c) The head coordinate system and the camera viewing axis direction (p, t); (d) The camera coordinate system and the image plane.

image plane. Suppose the diameter of the blur circle produced by $\mathbf{f}$ with distance l is c, the focus distance for l is v, the diameter of aperture is a. We have

$$\frac{c}{a} = \frac{v - f}{v} \tag{27}$$

$$\frac{1}{l} + \frac{1}{v} = \frac{1}{f} \tag{28}$$

So, $c = \frac{af}{l}$. Similarly we have $c_0 = \frac{af_0}{l_0}$, where c_0 is the diameter of the blur circle produced by $\mathbf{f_0}$ with distance l_0. Since

$$l_0 = l\sqrt{\frac{tan(\frac{w}{2})tan(\frac{h}{2})}{tan(\frac{w_0}{2})tan(\frac{h_0}{2})}} = l\frac{f_0}{f} \tag{29}$$

So we have $c = c_0$. Thus the above calculations is consistent with respect to the misfocus property.

Usually an object point is considered being well focused if the diameter of the blur circle size 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. So in our case, the depth of field will be from a distance, where the size of the blur circle is same as the size of the sensor pixel, to infinity.

8.3 Appendix 3: the detection function

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 $\mathbf{f}$, where $\mathbf{f} = \mathbf{f}(x_c, y_c, z_c, p, t, w, h, a)$, (x_c, y_c, z_c) are the position of the camera center in the world coordinate system, p, t is the direction of the camera viewing axis in the head coordinate system. The value of this function is not only dependent on the position (x, y, z) and action $\mathbf{f}$, but also on the previous actions. We will discuss the detection function in detail in this appendix.

8.3.1 The calculation of detection function when no previous action been applied

The detection function when there is no action applied before is denoted as $\mathbf{b}'((x, y, z), \mathbf{f}) = \mathbf{b_0}'((x, y, z), \mathbf{f})$. If β_0 is the event that there is no action applied before the current action $\mathbf{f}$ and β is the event that the current action succeeded in finding the target, we have: the value of the detection function is the probability that β is true under the condition β_0 is true

$$\mathbf{b_0}'((x, y, z), \mathbf{f}) = p(\beta|\beta_0)$$

To obtain $p(\beta|\beta_0)$, we can put the target at (x, y, z) and perform experiment under various conditions. The final value of the detection function will be the total number of successful recognitions divided by the total number of experiments.

The factors that influence the value of the detection functions are the following:

Position	The position of the target relative to the camera is specified by θ, δ, l. This position is uniquely determined by the position of the target (x, y, z) and the position and orientation (x_c, y_c, z_c, p, t) of the camera. $Pos =< \theta, \delta, l >$.

Orientation	The set of available relative orientations of the target with respect to the camera center is $\mathbf{O} = \{O_1, \ldots, O_{k_o}\}$.
Angle Size	$< w, h >$ is uniquely determined by the action.
Intensity	The set of available intensities around the target is $\mathbf{I} = \{I_1, \ldots, I_{k_i}\}$
Occlusion	The set of available occlusion and background complexity situations around the target is $\mathbf{C} = \{C_1, \ldots, C_{k_c}\}$

Since the position Pos and intrinsic factors $< w, h >$ are uniquely determined by the action and target position, the factors that can be changed when considering the detection function values are just orientation, intensity and occlusion. So, the set of situations need to be considered is $\Gamma_0 = \{< I_i, O_j, C_k >\mid I_i \in \mathbf{I}, O_j \in \mathbf{O}, C_k \in \mathbf{C}\}$.

To obtain the detection function value, we can perform experiments by putting the target at $Pos =< \theta, \delta, l >$ relative to the camera, and using $\mathbf{f}$ for every triple $< I_i, O_j, C_k > \in \Gamma_0$. Let $N_{success}$ be the total number of successful recognitions, N_{total} be the total number of experiments. Then $N_{total} =\mid \mathbf{I} \mid \times \mid \mathbf{O} \mid \times \mid \mathbf{C} \mid$.

$$\mathbf{b_0}'((x, y, z), \mathbf{f}) = \frac{N_{success}}{N_{total}} = \frac{N_{success}}{\mid \mathbf{I} \mid \times \mid \mathbf{O} \mid \times \mid \mathbf{C} \mid} \tag{30}$$

For a standard operation $\mathbf{f}_0$, first we can tessellate the camera viewing volume according to θ, δ, l. Second, for each tessellated unit, we can calculate the detection value by the above method. Third, we can put the result of the detection value into a database indexed by θ, δ, l. The resulting database will be used by the searcher during the search process (see section 3.1).

8.3.2 The calculation of detection function when there are one or more actions before

Suppose the first action is $\mathbf{f_1} = \mathbf{f}(x_{c1}, y_{c1}, z_{c1}, p_1, t_1, w_1, h_1, a)$, the current action is $\mathbf{f} = \mathbf{f}(x_c, y_c, z_c, p, t, w, h, a)$, β_1 is the event that the first action $\mathbf{f_1}$ succeeded in detecting the target, β is the event that the current action $\mathbf{f}$ succeeded in detecting the target. We have

$$\mathbf{b_1}'((x, y, z), \mathbf{f}) = p(\beta \mid \neg\beta_1) \tag{31}$$

In the following, we will compare the detection function value $\mathbf{b_1}'((x, y, z), \mathbf{f}) = p(\beta \mid \neg\beta_1)$ with the detection function value $\mathbf{b_0}'((x, y, z), \mathbf{f}) = p(\beta \mid \beta_0)$. To make the discussion easier, we first explain the symbols that will be used:

$\mathbf{I}, \mathbf{O}, \mathbf{C}, \Gamma_0$	The possible situations that can be considered when the action is the first action. These factors are determined by the search environment and the target. They are all the possible situations that can occur within the search environment. They have the same meaning as in 8.3.1.

N_{total}	The total number of situations needed to be considered when no actions been applied before. It has the same meaning as in 8.3.1.
(θ, δ, l)	The position of the target with respect to action **f**.
$(\theta_1, \delta_1, l_1)$	The position of the target with respect to action $\mathbf{f_1}$.
$\Gamma_{0-\mathbf{f_1}}$	The set of situations to be considered by $\mathbf{f_1}$.
$\Gamma_{0-\mathbf{f_1}-success}$	The set of situations that make $\mathbf{f_1}$ succeed.
$\Gamma_{0-\mathbf{f_1}-fail}$	The set of situations that make $\mathbf{f_1}$ fail.
$\Gamma_{0-\mathbf{f}}$	The set of situations to be considered by **f** when there is no action before **f**.
$\Gamma_{0-\mathbf{f}-success}$	The set of situations that make **f** succeed when there is no action before **f**.
$\Gamma_{0-\mathbf{f}-fail}$	The set of situations that make **f** fail when there is no action before **f**.
$\Gamma_{\neg\mathbf{f_1}-\mathbf{f}}$	The set of possible situations to be considered by **f** when the first action $\mathbf{f_1}$ failed.
$\Gamma_{\neg\mathbf{f_1}-\mathbf{f}-success}$	The set of situations that make **f**. succeed when there is an action $\mathbf{f_1}$ been applied before **f** and failed to detect the target.
$\Gamma_{\neg\mathbf{f_1}-\mathbf{f}-fail}$	The set of situations that make **f** fail when there is an action $\mathbf{f_1}$ been applied before **f** and failed to detect the target.

It is easy to know the following facts:

$$| \Gamma_0 |=| \mathbf{I} | \times | \mathbf{O} | \times | \mathbf{C} |$$

$$\Gamma_{0-\mathbf{f_1}} = \Gamma_0$$

$$\Gamma_{0-\mathbf{f_1}-success} \bigcup \Gamma_{0-\mathbf{f_1}-fail} = \Gamma_0$$

$$\Gamma_{0-\mathbf{f_1}-success} \bigcap \Gamma_{0-\mathbf{f_1}-fail} = \emptyset$$

$$\Gamma_{0-\mathbf{f_1}-success} \subseteq \Gamma_0$$

$$\Gamma_{0-\mathbf{f_1}-fail} \subseteq \Gamma_0$$

$$\Gamma_{0-\mathbf{f}} = \Gamma_0$$

$$\Gamma_{0-\mathbf{f}-success} \bigcup \Gamma_{0-\mathbf{f}-fail} = \Gamma_0$$

$$\Gamma_{0-\mathbf{f}-success} \bigcap \Gamma_{0-\mathbf{f}-fail} = \emptyset$$

$$\Gamma_{0-\mathbf{f}-success} \subseteq \Gamma_0$$

$$\Gamma_{0-\mathbf{f}-fail} \subseteq \Gamma_0$$

$$\Gamma_{\neg\mathbf{f_1}-\mathbf{f}-success} \bigcup \Gamma_{\neg\mathbf{f_1}-\mathbf{f}-fail} = \Gamma_{\neg\mathbf{f_1}-\mathbf{f}}$$

$$\Gamma_{\neg\mathbf{f_1}-\mathbf{f}-success} \bigcap \Gamma_{\neg\mathbf{f_1}-\mathbf{f}-fail} = \emptyset$$

$$\Gamma_{\neg\mathbf{f_1}-\mathbf{f}-success} \subseteq \Gamma_{\neg\mathbf{f_1}-\mathbf{f}-success}$$

$$\Gamma_{\neg\mathbf{f_1}-\mathbf{f}-fail} \subseteq \Gamma_{\neg\mathbf{f_1}-\mathbf{f}-success}$$

Since Γ_0 is the set of all possible situations, we also have:

$$\Gamma_{\neg\mathbf{f_1}-\mathbf{f}-success} \subseteq \Gamma_0$$

$$\Gamma_{\neg\mathbf{f_1}-\mathbf{f}-fail} \subseteq \Gamma_0$$

$$\Gamma_{\neg\mathbf{f_1}-\mathbf{f}} \subseteq \Gamma_0$$

The detection function values are calculated by:

$$\mathbf{b_0'}((x,y,z),\mathbf{f}) = p(\beta|\beta_0) = \frac{\mid \Gamma_{0-\mathbf{f}-success} \mid}{\mid \Gamma_0 \mid} = \frac{\mid \Gamma_{0-\mathbf{f}-success} \mid}{\mid \mathbf{I} \mid \times \mid \mathbf{O} \mid \times \mid \mathbf{C} \mid} \tag{32}$$

$$\mathbf{b_0'}((x,y,z),\mathbf{f_1}) = p(\beta_1|\beta_0) = \frac{\mid \Gamma_{0-\mathbf{f_1}-success} \mid}{\mid \Gamma_0 \mid} = \frac{\mid \Gamma_{0-\mathbf{f_1}-success} \mid}{\mid \mathbf{I} \mid \times \mid \mathbf{O} \mid \times \mid \mathbf{C} \mid} \tag{33}$$

$$\mathbf{b_1'}((x,y,z),\mathbf{f}) = p(\beta|\neg\beta_1) = \frac{\mid \Gamma_{\neg\mathbf{f_1}-\mathbf{f}-success} \mid}{\mid \Gamma_{\neg\mathbf{f_1}-\mathbf{f}} \mid} \tag{34}$$

Now, let's consider $\mathbf{b_1'}((x,y,z),\mathbf{f})$. Suppose $< I_{\neg\mathbf{f_1}}, O_{\neg\mathbf{f_1}}, C_{\neg\mathbf{f_1}} >$ is the real situation when we apply action $\mathbf{f_1}$ and fail to detect the target, we have $< I_{\neg\mathbf{f_1}}, O_{\neg\mathbf{f_1}}, C_{\neg\mathbf{f_1}} > \in \Gamma_{0-\mathbf{f_1}-fail}$. Suppose when we apply action $\mathbf{f}$, the situation becomes $< I_{\neg\mathbf{f_1}-\mathbf{f}}, O_{\neg\mathbf{f_1}-\mathbf{f}}, C_{\neg\mathbf{f_1}-\mathbf{f}} >$, 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 changes 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), 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 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 $(\theta_1, \delta_1, l_1)$ to (θ, δ, l), 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 relation:

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

and

$$\begin{aligned} T_r(\Gamma_{0-\mathbf{f_1}-fail}) &= T_r(\{< I_{\neg\mathbf{f_1}}, O_{\neg\mathbf{f_1}}, C_{\neg\mathbf{f_1}} >|< I_{\neg\mathbf{f_1}}, O_{\neg\mathbf{f_1}}, C_{\neg\mathbf{f_1}} >\in \Gamma_{0-\mathbf{f_1}-fail}\}) \\ &= \{T_r(< I_{\neg\mathbf{f_1}}, O_{\neg\mathbf{f_1}}, C_{\neg\mathbf{f_1}} >) \,|< I_{\neg\mathbf{f_1}}, O_{\neg\mathbf{f_1}}, C_{\neg\mathbf{f_1}} >\in \Gamma_{0-\mathbf{f_1}-fail}\} \\ &= \{< I_{\neg\mathbf{f_1}-\mathbf{f}}, O_{\neg\mathbf{f_1}-\mathbf{f}}, C_{\neg\mathbf{f_1}-\mathbf{f}} >|< I_{\neg\mathbf{f_1}-\mathbf{f}}, O_{\neg\mathbf{f_1}-\mathbf{f}}, C_{\neg\mathbf{f_1}-\mathbf{f}} >\in \Gamma_{\neg\mathbf{f_1}-\mathbf{f}}\} \\ &= \Gamma_{\neg\mathbf{f_1}-\mathbf{f}} \end{aligned} \tag{35}$$

The following table lists the factors that influence the values of detection functions $\mathbf{b}_\mathbf{1}^{'}((x, y, z), \mathbf{f})$ and $\mathbf{b}_\mathbf{0}^{'}((x, y, z), \mathbf{f})$.

Detection Function	$\mathbf{b}_\mathbf{1}^{'}((x, y, z), \mathbf{f})$	$\mathbf{b}_\mathbf{0}^{'}((x, y, z), \mathbf{f})$
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 above, we can notice that when we calculate $\mathbf{b}_\mathbf{1}^{'}((x, y, z), \mathbf{f})$ and $\mathbf{b}_\mathbf{0}^{'}((x, y, z), \mathbf{f})$, 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 $\mathbf{b}_\mathbf{1}^{'}((x, y, z), \mathbf{f})$ is more constrained than the situations Γ_0 considered by $\mathbf{b}_\mathbf{0}^{'}((x, y, z), \mathbf{f})$. This is because $\Gamma_{\neg\mathbf{f_1}-\mathbf{f}} \subseteq \Gamma_0$. And this constrained situation is dependent on the previous action $\mathbf{f_1}$, which is actually a transformation of $\Gamma_{0-\mathbf{f_1}-fail}$. So, in general situations $\mathbf{b}_\mathbf{1}^{'}((x, y, z), \mathbf{f})$ and $\mathbf{b}_\mathbf{0}^{'}((x, y, z), \mathbf{f})$ may not always equal to each other. In general, 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. The layered sensed sphere is used to reduce the error caused by the correlation between actions. Figure 25 gives an explanation of the above analysis.

In real application, we can only get $\mathbf{b}_\mathbf{0}^{'}((x, y, z), \mathbf{f})$ by retrieving from the database according to the relative position of the target with respect to the camera center. For $\mathbf{b}_\mathbf{H}^{'}((x, y, z), \mathbf{f})$ (see section 3.1), we will use approximate values based on $\mathbf{b}_\mathbf{0}^{'}((x, y, z), \mathbf{f})$. In the following, we will analyze the relation between $\mathbf{b}_\mathbf{1}^{'}((x, y, z), \mathbf{f})$ and $\mathbf{b}_\mathbf{0}^{'}((x, y, z), \mathbf{f})$ for several cases.

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 identical transformation. So $(\theta_1, \delta_1, l_1) = (\theta, \delta, l)$. And the situation are also same $< I_{\neg\mathbf{f_1}}, O_{\neg\mathbf{f_1}}, C_{\neg\mathbf{f_1}} >=< I_{\neg\mathbf{f_1}-\mathbf{f}}, O_{\neg\mathbf{f_1}-\mathbf{f}}, C_{\neg\mathbf{f_1}-\mathbf{f}} >$. Since $< \theta_1, \delta_1, l_1, I_{\neg\mathbf{f_1}}, O_{\neg\mathbf{f_1}}, C_{\neg\mathbf{f_1}} >$ makes $\mathbf{f_1}$ fail and $\mathbf{f_1} = \mathbf{f}$, we have, $< \theta_1, \delta_1, l_1, I_{\neg\mathbf{f_1}}, O_{\neg\mathbf{f_1}}, C_{\neg\mathbf{f_1}} >$ also make $\mathbf{f}$ fail. This is actually that $< \theta, \delta, l, I_{\neg\mathbf{f_1}-\mathbf{f}}, O_{\neg\mathbf{f_1}-\mathbf{f}}, C_{\neg\mathbf{f_1}-\mathbf{f}} >$ makes $\mathbf{f}$ fail. So, in this situation, $\mathbf{b}_\mathbf{1}^{'}((x, y, z), \mathbf{f}) = 0$.

In real application, we should avoid the repeating operation. When an action is repeated, the environment is not updated.

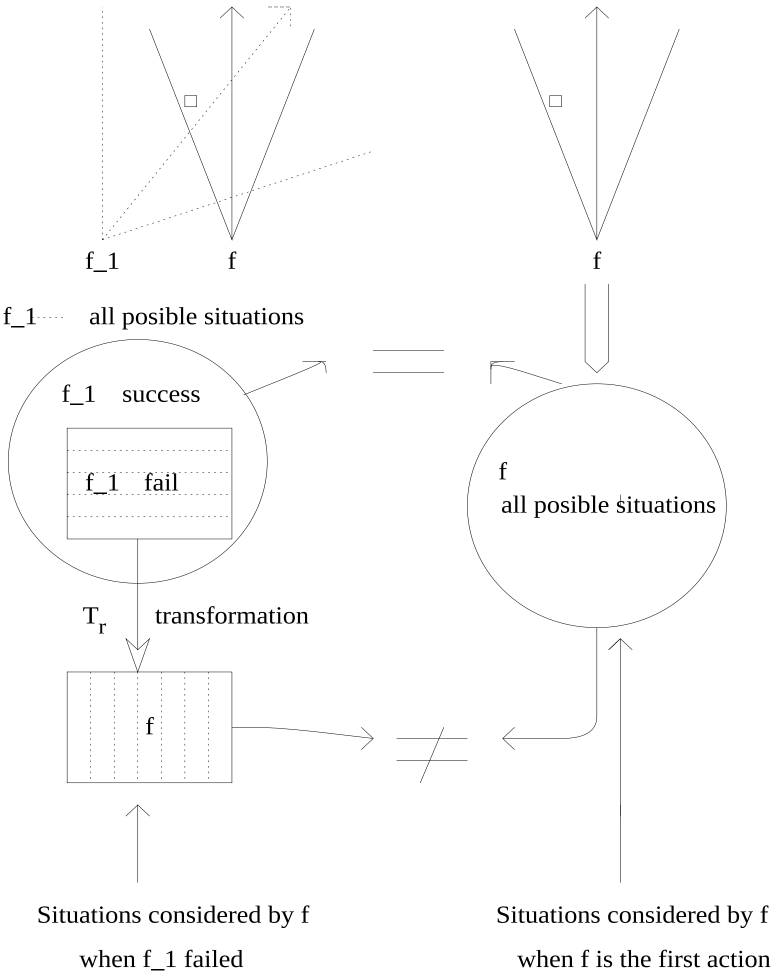


Figure 25: The correlation between actions

Case 2: the intrinsic parameters and the position of the two actions are 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 $(w_1, h_1) = (w, h)$, $(p_1, t_1) \neq (p, t)$. In this case, the position factor are different but the situation factors are 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 usually the camera angle is 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, thus the patterns of the target in the image should not change too much by just changing the viewing direction of the camera. Since the previous image make the recognition algorithm fail, the current image will probably also make the recognition algorithm fail. Thus in this case, we can approximate the detection function value as 0.

In 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 position of the two actions are same, but the intrinsic parameters are different

If the difference in direction and intrinsic parameters are small, we will approximate the detection function value of the second action as $\mathbf{b_1'}((x, y, z), \mathbf{f}) = 0$. Otherwise, we will assume the independence between the actions. Thus we have $\mathbf{b_1'}((x, y, z), \mathbf{f}) = \mathbf{b_0'}((x, y, z), \mathbf{f})$. This means that when we meet this situation, we will directly retrieve from the database for the value of $\mathbf{b_1'}((x, y, z), \mathbf{f})$.

Case 4: The position of the two actions are different

We will assume the independence between the actions. Thus we have $\mathbf{b_1'}((x, y, z), \mathbf{f}) = \mathbf{b_0'}((x, y, z), \mathbf{f})$ This means that when we meet this situation, we will directly retrieve from the database for the value of $\mathbf{b_1'}((x, y, z), \mathbf{f})$.

Case 5: The layered sensed sphere and the approximation of $\mathbf{b_H'}((x, y, z), \mathbf{f})$

One benefit of using the layered sensed sphere to select actions is to reduce the error caused by the dependence between actions. Each layer correspond to a given intrinsic parameters. The actions $\mathbf{f}$ with respect to a given layer has a high detection function values $\mathbf{b_0'}((x, y, z), \mathbf{f})$ when the target position (x, y, z) is within the layer of $\mathbf{f}$. Otherwise, $\mathbf{f}$ has a low detection function values $\mathbf{b_0'}((x, y, z), \mathbf{f})$ when the target position (x, y, z) does not belong to the layer of $\mathbf{f}$.

Each layer in the layered sensed sphere correspond to a given intrinsic parameters of action. The difference of the intrinsic parameters between different layers are large. We only select actions whose parameters appeared in the layered sensed sphere. So in the search process, if a cube has been examined more then once by actions belong to same layer, then the detection function value will be 0. This is reasonable according to case 1 and case 2. Otherwise, if a cube is first examined by an action belong to a layer, then the detection function value will be retrieved according to the database. According to case 3, this is reasonable because the difference of intrinsic parameters is high for actions between different layers.

With respect to different camera (robot) position, according to case 4, we assume independence between actions belonging to different positions.

Summary

The calculation of the detection functions in our algorithm is summarized as following:

(A) If the two actions examining the same cube have different camera position, we assume independence between them (the detection function of the later action will be retrieved from the database).

(B) If the two actions examining the same cube have the same camera position and they belong to different layers of the layered sensed sphere, we assume independence between them.

(C) If the two actions examining the same cube have the same camera position and they belong to the same layer of the layered sensed sphere, the detection function value of the second action will be zero.

(D) Repeating actions are avoided.

Note: the relationship of any two actions selected by our algorithm can only belong to (A), (B) or (C).

References

[Baj85] R. Bajcsy. Active perception vs. passive perception. In *Third IEEE Workshop on Vision*, pages 55–59, Bellaire, 1985.

[Baj88] R. Bajcsy. Perception with feedback. In *Image Understanding Workshop*, pages 279–288, 1988.

[CD86] R. Chin and C. Dyer. Model-based recognition in robot vision. *Computing Serveys*, 18(1):67–108, March 1986.

[Con89] Connel. *An Artificial Creature*. PhD thesis, AI Lab, MIT, 1989.

[Gar76] Thomas D. Garvey. Perceptual strategies for purposive vision. Technical Report Technical Note 117, SRI International, 1976.

[Gib50] J.J. Gibson. *The perception of the visual world*. Houghton-Mifflin, Boston, 1950.

[KAT92] Tarabanis K., P. Allen, and R. Tsai. A survey of sensor planning in computer vision. Technical report, Comp. Sci. Dept., Columbia University, 1992.

[Koo80] B. O. Koopman. *Search and Screen: general principles with historical applications*. Pergaman Press, Elmsford, N.Y, 1980.

[Kro87] Eric Krotkov. Focusing. *International Journal of Computer Vision*, 1:223–237, 1987.

[MB90] 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.

[NJM+93] 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.

[RB92] 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.

[Rim91] Raymond D. Rimey. Controlling eye movements with hidden markov models. *International Journal Computer Vision*, 7(1), 1991.

[RS92] D. Reece and S. Shafer. Using active vision to simplify perception for robot driving. Technical Report CMU-CS-91-199, Comp. Sci., Carnegie Mellon, 1992.

[SB91] Michael J. Swain and D. Ballard. Color indexing. *International Journal Computer Vision*, 7(1):11–32, 1991.

[Sto89] L.D Stone. *Theory of optimal search*. Military Applications Section, Operations Research Society of America, Arlington, Va, 1989.

[Tso92a] J.K. Tsotsos. 3d search strategy. *Internal ARK Working Paper, Department of Computer Science, Univrsity of Toronto*, 1992.

[Tso92b] J.K. Tsotsos. Active verses passive perception, which is more efficient? *IJCV*, 1992.

[Wil94] D. Wilkes. Active object recognition. Technical report, Department of Computer Science, University of Toronto, 1994.

[Wix92] L.E. Wixson. Exploiting world structure to efficiently search for objects. Technical Report Technical Report 434, Dept. Comp. Sci, Univ. of Rochester, 1992.

[WT92] D. Wilkes and J.K. Tsotsos. Active object recognition. In *Proceedings of Computer Vision and Pattern Recognition*, pages 136–141, Illinois, USA, June 1992.

[Baj85] [Baj88] [Con89] [MB90] [Gar76] [Tso92a] [Tso92b] [WT92] [Wix92] [Gib50] [Rim91] [SB91] [Kro87] [Sto89] [Koo80] [KAT92] [RS92]