



Visual search for an Object in a 3D Environment using a Mobile Robot

Ksenia Shubina

John K. Tsotsos

Technical Report CSE-2008-02

April 28, 2008

Department of Computer Science and Engineering  
4700 Keele Street Toronto, Ontario M3J 1P3 Canada

# Visual Search for an Object in a 3D Environment using a Mobile Robot

Ksenia Shubina and John K. Tsotsos

*Dept. of Computer Science and Engineering and Centre for Vision Research,  
York University, Toronto, Canada M3J 1P3*

---

## Abstract

Consider the problem of visually finding an object in a mostly unknown space with a mobile robot. It is clear that all possible views and images cannot be examined in a practical system. Visual attention is a complex phenomenon; we view it as a mechanism that optimizes the search processes inherent in vision. Here, we describe a particular example of a practical robotic vision system that employs some of these attentive processes. We cast this as an optimization problem, i.e., optimizing the probability of finding the target given a fixed cost limit in terms of total number of robotic actions required to find the visual target. Due to the inherent intractability of this problem, we present an approximate solution and investigate its performance and properties. We conclude that our approach, named the SYT algorithm, is sufficient to solve this problem and has additional desirable empirical characteristics.

*Key words:* visual search, sensor planning, search, mobile robot, active vision, active object recognition, visual attention

---

*Email addresses:* [ksenia@cse.yorku.ca](mailto:ksenia@cse.yorku.ca) (Ksenia Shubina),  
[tsotsos@cse.yorku.ca](mailto:tsotsos@cse.yorku.ca) (John K. Tsotsos).

## 1 Introduction

Attention is one of those visual phenomena that has been very easy to ignore in computer vision and robotics but seems to now be emerging as an important issue. Visual attention is a phenomenon that has been of interest to many disciplines for over a hundred years, with an enormous literature and thousands upon thousands of experiments investigating the vast range of its manifestations. Theoretical and computational models have been proposed since the 1950s in an attempt to explain how this phenomenon comes about and how it contributes to our perception of the real world (for a review see [1]). The first formal proof for the necessity of attentive processes appeared in [2] (see also [3, 4, 5, 6]). There, the problem of visual matching - the task of determining whether or not an instance of a particular model exists in a given image without the use of any knowledge whatsoever - was shown to be NP-Complete. It has exponential time complexity, in the size of the image, and further, the result is independent of implementation. In addition to other mechanisms, attention contributes to changing this problem into one with linear time complexity in both worst case and median case analyses [2, 7].

The breadth and variety of attentive phenomena as they relate to computer vision was described in [5]. There, a spectrum of problems requiring attention was laid out including: selection of objects, events or tasks relevant for a domain; selection of world model; selection of visual field; selection of detailed sub-regions for analysis; selection of spatial and feature dimensions of interest; and the selection of operating parameters for low level operations. Most computer vision research makes assumptions that reduce the combinatorial problems inherent in the above tasks, or better yet, eliminate the need for attention, using strategies such as:

- fixed camera systems negate the need for selection of visual field or selection of best viewpoints ;
- pre-segmentation eliminates the need to select a region of interest;
- ‘clean’ backgrounds ameliorate the segmentation problem;
- assumptions about relevant features and their values reduce their search ranges;
- knowledge of task domain negates the need to search a stored set of all domains;
- knowledge of objects appearing in scenes eliminates search of a stored set of objects;
- knowledge of which events are of interest eliminates search of a stored set of events.

In this way, the extent of the search space is seriously reduced before the visual processing takes place, and often even before the algorithms for solution are designed. However, it is clear that in everyday vision, and certainly in order to understand vision, these assumptions cannot be made. Real vision requirements, for humans as well as robots, are not so cooperative and attentive processes need to play a central role in all visual processes.

The example robotic system described in this paper looks at the viewpoint and selection of visual field issues in the context of search for a given, known object in an unknown 3D world. As such, it is an instance of the active vision approach [8]. Bajcsy argued that rather than simply analyzing a set of prerecorded images, the observer should actively control its image acquisition process so that the acquired images are relevant and useful for the task at hand. In the case of region segmentation problems, the camera could be moved to a viewpoint in which, for example, the projection of an object in the image

plane leads to a higher contrast region, or an object's edge projects to a stronger gradient in the image. If a particular view of an object (or one of its parts) is ambiguous, the camera can be moved to disambiguate the object. For example, Wilkes et al. [9] proposed a system that drives a camera to a standard viewpoint with respect to an unknown object. From such a viewpoint, the object recognition task is reduced to a two-dimensional pattern recognition problem. The authors choose to define a standard view as a position at which the lengths of two non-parallel object line segments are maximized, and the longer line has a specified length in the image. The standard view is achieved by moving the camera on the end of a robot arm. From a standard viewing position, the extracted line segments are used to index into the database to find a matching, stored object. In a different strategy for the same problem, Dickinson et al. [10] combine an attention mechanism and a viewpoint control strategy to perform active object recognition. Their representation scheme is called the aspect prediction graph. Given an ambiguous view of an object this representation can inform the algorithm if there is a more discriminating view of the object. If there is, the representation will indicate, in which direction the camera should be moved to capture that view. Finally, it specifies what visual events (appearance or disappearance of object features) one should encounter while moving the camera to the new viewpoint. In both cases, the image interpretation process is tightly coupled to the viewpoint selection and data acquisition process, as Bajcsy suggested. The success of these works lies in the fact that no assumptions about viewpoint were needed and attentive processes - selection processes - provided the reduction in the combinatorics of search that would cripple a brute-force, blind, search.

This paper focuses on the problem of visual search for an object in a 3D environment using a mobile robot, providing a description of the solution strategy

and an example of the robot's performance and an empirical performance evaluation.

## **2 A robot that searches: previous work**

Suppose one wishes a robot to search for and locate a particular object in a 3D world. A direct search certainly suffices for the solution. Assuming that the target may lie with equal probability at any location, the viewpoint selection problem is resolved by moving a camera to take images of the previously not viewed portions of the full 3D space. This kind of exhaustive, brute force approach can suffice for a solution; however, it is both computationally and mechanically prohibitive. As an alternative, Garvey [11] proposed the idea of indirect search for a target: first a sensor is directed to search for an intermediate object that commonly participates in a spatial relationship with the target. For example, if one wants to find a telephone in an image of an office, it is easier to first locate flat surfaces, e.g. table tops, on which the phone is most likely to rest. Then the sensor is directed to examine the restricted region specified by the relationship, i.e. the search for the phone is limited to the table tops. Indirect searches reduce the computationally expensive problem to a two-stage problem. In the first stage, one locates an intermediate object that typically participates in some spatial relationship with the target and which can be found with a lower resolution, i.e. with a wider field of view. In the second stage, the high-resolution search for the target is performed in the much smaller volume specified by the spatial relationship.

Wixson et al. [12] have elaborated the indirect search idea and have shown efficiency gains both theoretically and empirically, Other demonstrations of the

idea have also shown good performance (for example, [13]). The problem with indirect search is that the spatial relation between the target and intermediate object may not always exist. In addition, the detection of the intermediate object may not be easier than the detection of the target. In fact, search for an arbitrary object in a 3D space is provably NP-hard [14].

Searching for an object in a cluttered environment is often complicated by the fact that portions of the area are hidden from view. A different viewpoint is necessary to observe the target. This requirement is also characteristic for the task of scene reconstruction where multiple viewpoints must be selected to acquire a model or a map of the environment. As a consequence, viewpoint selection for search tasks seems similar to viewpoint selection for data acquisition of an unknown scene: new viewpoints are determined by yet unseen areas, e.g. [15], [16]. Kim et al. [17] have studied the problem of determining camera viewpoints for successive views looking for distinguishing features of an object. The distance of the camera to the object is determined by the size of the object and the size of the feature. Within this distance, the shape of the feature and presence of occluding objects determine the direction. An aspect graph with nodes assigned values representing the goodness of the view is suggested to guide the motion of the camera on the sphere. Cowan et al. [18] explicitly specify configurations of the camera's state parameters in order to perform a certain task. In these methods, the effectiveness of the system can largely be determined by the locations, types and configurations of the sensor used.

Wixson [19] argues that the two tasks of visual search and scene modeling are not that similar. The viewpoint selection problem for search tasks involves not only a choice of position and direction of the sensor, but also a solid angle

relative to the viewpoint. The author suggests that the difficult task of scene modeling that usually accompanies visual search is not necessary since the only requirement is that it brings otherwise hidden areas into view. Wixson proposes a model-free algorithm that first identifies an occluding edge and then rotates the sensor to a position where this edge becomes non-occluding. Yet it is unclear what is the criterion to abort the search and what is the strategy to decide on the next sensor position.

In recent literature, the majority of the search robots are intended for search and rescue applications. For the most part, they lack autonomy and are remotely controlled by humans. At the present time, most of the rescue teams only require teleoperated robots; exceptions are applications that claim autonomous robots for map building or exploration of environment [20].

Some researchers redefine the search task as one of exploration. The robot is given a set of locations that completely cover the environment and its task is to find the shortest path that visits all of these points [21]. While the robot moves along the path, it looks for objects. Tovar et al. [22] look for an optimal path in an unknown environment. Their robot uses lasers to construct a visibility tree that represents simply-connected planar environments. This tree is then dynamically updated to specify the optimal path. They try to avoid traditional problems such as complete map building and localization by constructing this minimal representation that is based entirely on critical events, such as crossing lines, in sensor measurements made by the robot.

Sarmiento et al. [23] specifies an optimal search path in such a way that the expected time to find the object is minimized. Later Sarmiento et al. [24] introduce a sampling scheme that generates the initial set of sensing locations for the robot. They propose a convex cover algorithm based on this sampling.



They use the resulting convex covering to generate a graph that captures the connectivity of the workspace. Then they search this graph to generate trajectories that minimize the expected time to find the object. The environment structure is known in advance. The probability of finding the target in the region is proportional to the size of the region. They do not propose any specific object recognition algorithm rather assume that the robot has one. Only simulation results are presented.

Lau [25]’s robot searches for multiple targets in a known building environment. The environment is divided into a set of distinct regions and an adjacency matrix is used to describe the connections between them. Travel and search costs of individual regions are specified. The algorithm uses the available target information: expected number of targets in a given region is set proportional to its size. This information is updated after each search operation performed by the robot. The task is to find the sequence of actions that minimizes expected time to find the target. The search strategy is then derived using dynamic programming. The problem becomes intractable for more than 14 regions. Only simulation results are presented.

Bourgault et al. [26] employed a Bayesian approach where the target probability density function is used as prior information. The target PDF is updated using the model of the sensor and expected target motion. The optimal search trajectory is defined as the one that maximizes the cumulative detection probability over a limited time period. The key assumptions in their strategy are that the PDF of the target locations is smooth and the search space is free from obstacles constraining the searcher’s motion, e.g. for rescue air vehicles.

Most of the above planning algorithms for search robots assume that the environment is known in advance. Based on this knowledge they derive the target

probability distribution: the probability of finding the target in the region is usually set proportional to the size of the region. The systems described above concentrate on generating the motion strategy through the environment. None of them describe recognition or detection methods used during the search process within specific regions. For the most part, only simulation results are presented.

Frintrop's VOCUS system (Visual Object detection with a CompUtational attention System, [27]) uses an attentional component to guide search. First, she finds salient image regions by using image contrasts and uniqueness of a feature generating a hypothesis for possible locations of the target. Then a classifier is applied to the regions to verify the hypothesis. The system demonstrates good detection results in still images. However, no attention is given to viewpoint control.

What we seek is a functional search robot that can find an arbitrary object in an unknown 3D environment. To date, this particular problem seems to have not received much attention as the above survey shows.

### **3 The object search problem**

Ye & Tsotsos define object search as a problem of maximizing the probability of detecting the target within a given cost constraint [28, 29]. Their formulation combines the influence of a search agent's initial knowledge and the influence of the performance of available recognition algorithms. For a practical search strategy, the search region is characterized by a probability distribution of the presence of the target. The control of the sensing parameters depends on the current state of the search region and the detection abilities of recognition

algorithms. In order to efficiently determine sensing actions over time, the huge space of possible actions is reduced to a finite set of actions that must be considered. The result of each sensing operation is used to update the status of the search space. In the following subsection, we introduce some of the key concepts (This material has been presented previously in [29] and the reader is referred there for further detail).

### 3.1 *Some concepts*

A search region  $\Omega$  is a 3D space to be searched. We assume that boundaries of  $\Omega$  are known exactly, but its internal configuration is not. The region  $\Omega$  is tessellated into a 3D grid of non-overlapping cubic elements  $c_i, i = 1 \dots n$ .

An operation  $\mathbf{f}$  on  $\Omega$  is an action of the search agent within the specified region. It consists of taking an image according to the camera configuration  $S(\tau)$  and analyzing it to find out whether the target is present. The camera configuration  $S(\tau)$  specifies its position  $(x_c, y_c, z_c)$ , direction of its viewing axis  $(p, t)$  and the width and height of its solid viewing angle  $(w, h)$  at time  $\tau$ . Actions are represented as  $\mathbf{f} = \mathbf{f}(S(\tau), a)$ , where  $a$  is an algorithm used to analyze the image.

The cost function  $\mathbf{t}(\mathbf{f})$  for an operation  $\mathbf{f}$  gives the time required for its execution. It includes moving a sensor from one configuration to another, acquiring an image, running a recognition algorithm and updating the agent's knowledge about the environment.

The target distribution is specified by the probability density function  $\mathbf{p}$ . This distribution is updated after each operation, therefore,  $\mathbf{p}$  is a function of both position and time.  $\mathbf{p}((x, y, z), \tau)$  gives the probability that the target is at

$(x, y, z)$  at time  $\tau$ .  $\mathbf{p}(c_{out}, \tau)$  gives the probability that the target is outside the search region  $\Omega$  at time  $\tau$ .

The detection function on  $\Omega$  is a function  $\mathbf{b}((x, y, z), \mathbf{f})$  that gives the conditional probability of detecting the target by applying action  $\mathbf{f}$  given that the target's center is at  $(x, y, z)$  - i.e., the target is centred at cube  $c_i$  whose centre is  $x, y, z$ .

If the center of a given cube  $c_i$  falls outside the current image,  $\mathbf{b}(c_i, \mathbf{f}) = 0$  for any operation  $\mathbf{f}$ . Similarly,  $\mathbf{b}(c_{out}, \mathbf{f}) = 0$  for any operation  $\mathbf{f}$ , because the target is outside the search region  $\Omega$ . If it is inside the image, the value of  $\mathbf{b}(c_i, \mathbf{f})$  is determined by various factors [28].

The influence range  $\Psi_{\mathbf{f}}$  of the action  $\mathbf{f}$  consists of those parts of  $\Omega$  that are “visible” to the search agent with the current camera's setting  $S(\tau)$ . Using the notation introduced above, the probability of detecting the target by operation  $\mathbf{f} = \mathbf{f}(S(\tau), a)$  becomes

$$\mathbf{P}_{\Psi_{\mathbf{f}}}(\mathbf{f}) = \sum_{c_i \in \Psi_{\mathbf{f}}} \mathbf{p}(c_i, \tau_{\mathbf{f}}) \mathbf{b}(c_i, \mathbf{f}) \quad (1)$$

where  $\tau_{\mathbf{f}}$  is the time just before  $\mathbf{f}$  is applied.

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

### 3.2 Problem statement

Ye & Tsotsos define the problem of object search as follows. Let  $\mathbf{K}$  be the total time available for search. Then for any effort allocation  $\mathbf{F} = \{\mathbf{f}_1, \dots, \mathbf{f}_k\}$ ,

the probability of detecting the target by this allocation is:

$$\begin{aligned}
 P[\mathbf{F}] = & \sum_{i=1}^n \mathbf{p}(c_i, \tau_{\mathbf{f}_1}) \mathbf{b}(c_i, \mathbf{f}_1) + [1 - \sum_{i=1}^n \mathbf{p}(c_i, \tau_{\mathbf{f}_1}) \mathbf{b}(c_i, \mathbf{f}_1)] * [\sum_{i=1}^n \mathbf{p}(c_i, \tau_{\mathbf{f}_2}) \mathbf{b}(c_i, \mathbf{f}_2)] \\
 & + \dots \\
 & + \{ \prod_{j=1}^{k-1} [1 - \sum_{i=1}^n \mathbf{p}(c_i, \tau_{\mathbf{f}_j}) \mathbf{b}(c_i, \mathbf{f}_j)] \} * [\sum_{i=1}^n \mathbf{p}(c_i, \tau_{\mathbf{f}_k}) \mathbf{b}(c_i, \mathbf{f}_k)]
 \end{aligned}$$

and the total time for applying this allocation is:

$$T[\mathbf{F}] = \sum_{\mathbf{f} \in \mathbf{F}} \mathbf{t}(\mathbf{f}) \tag{2}$$

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

Ye shows that this problem is NP-hard and concludes that a “greedy” algorithm is sufficient to obtain a good approximation to the solution [28].

### 3.3 The search strategy

Due to the inherent intractability, we take an approximation solution approach. We decompose the search space as well as the action space into manageable chunks as will be described. It is important to note that other decompositions or approximations may also provide a solution to the problem; ours is a satisficing solution but not a necessary one. The greedy algorithm works in stages considering one input image at a time. At each stage, considering the cost and effect of different actions, the next best action is selected and included into the partial solution developed so far. Ye simplifies the object search task further by dividing the problem in two subproblems: “where to look next” and “where to move next”. During the first stage the search agent’s

position is fixed and its task is to optimally select viewing configurations that give large probability of detecting the target. During the second stage the search agent determines the next optimal position.

There are two characteristics of the approach. The first is decomposition of the huge number of camera's parameters settings allowed by the hardware into a limited set of settings that must be considered. The spatial decomposition greatly reduces the complexity of the "where to look next" task.

The second is the emphasis on guidance using a priori knowledge. The advantage of this is that when initial knowledge is relatively good, the most promising actions tend to be selected first. On the other hand, as the quality of a priori knowledge degrades, so does the performance of the algorithm but only with respect to amount of time or number of actions required. Section 3.4 describes different kinds of a priori knowledge that can be used.

### *3.3.1 Representing the world and the actions*

In order to represent the surrounding environment of the camera and to efficiently determine the sensing parameters over time, Ye [28] introduces a concept called the sensed sphere. A sensed sphere discretizes space around the camera by dividing it into a series of solid angles. Each solid angle is specified as follows. Any line emitting from the center of the camera will hit a solid object in the environment and hence, can be described by its direction (*pan, tilt*) and length. Each solid angle is represented by the direction of the emitting line: its central axis - and length of the emitting line: its radius. A sensed sphere is a union of these angles. The sensed sphere can be constructed using any depth sensor.

The successful detection of the target depends on its appearance in the image. If the object is so close that it does not fit in the image or so far away that it appears only as a few pixels, most detection methods will fail. For each achievable camera zoom setting, the effective range of the algorithm can be specified that gives distances from which the target can be detected. This range depends on the size and appearance characteristics of the object. Thus, the sensed sphere can be divided in layers, each with an inner and outer radius, with each layer corresponding to an achievable camera zoom setting.

Our robot is equipped with the BumbleBee camera that does not have zoom capabilities. Therefore, the size of the target as it appears in the image depends on its distance from the camera. If there are no occluders within a solid angle, its outer radius is set equal to the largest distance from which the object can be detected using the available detection algorithm. The inner radius is set to the closest smallest distance from which the object can be detected.

Since the radius of any solid angle is set by detected occluders, a sensed sphere not only discretizes the space into viewing directions, but also allows us to take occluding situations into consideration when choosing the next optimal viewing direction or robot's position.

### 3.3.2 *“Where to look next”*

A ‘best-first’ strategy is used to examine all discretized camera's configurations one by one. However, when the number of settings is large or image analysis takes too much time, this strategy becomes inefficient. By applying the most promising settings first, the probability of detecting the target at an early stage increases and the time and effort spent on the search decreases.

Here, the knowledge about the potential target locations is encoded as a target probability distribution  $\mathbf{p}(c_i, \tau)$ . By combining the target distribution and detection function, the probability of detecting the target by operation  $\mathbf{f} = \mathbf{f}(p, t, w, h, a)$  can be calculated by Equation 1. Then, the utility of an operation  $\mathbf{f}$  is given by

$$\mathbf{E}_{\Psi_{\mathbf{f}}}(\mathbf{f}) = \frac{\sum_{c_i \in \Psi_{\mathbf{f}}} \mathbf{p}(c_i, \tau_{\mathbf{f}}) \mathbf{b}(c_i, \mathbf{f})}{\mathbf{t}(\mathbf{f})} \quad (3)$$

where  $\Psi_{\mathbf{f}}$  is the influence range of operation  $\mathbf{f}$  and  $\mathbf{t}(\mathbf{f})$  is the time action  $\mathbf{f}$  takes.

The goal is to select an operation with the highest utility value. Since the cost of each action is approximately the same if the robot is stationary, the next action is selected in such a way that it maximizes the numerator of Equation 3. This is the utility function that Ye [28] used in his experiments. In this paper, we explore several additional functions for the selection of the next action. These are detailed in Section 5.2.1.

After each action, target probabilities are updated using

$$\mathbf{p}(c_i, \tau_{\mathbf{f}+}) = \frac{\mathbf{p}(c_i, \tau_{\mathbf{f}})(1 - \mathbf{b}(c_i, \mathbf{f}))}{\mathbf{p}(c_{out}, \tau_{\mathbf{f}}) + \sum_{j=1}^n \mathbf{p}(c_j, \tau_{\mathbf{f}})(1 - \mathbf{b}(c_j, \mathbf{f}))}, i = 1, \dots, n, out \quad (4)$$

where  $\tau_{\mathbf{f}+}$  is the time after  $\mathbf{f}$  is applied and  $\mathbf{p}(c_{out}, \tau_{\mathbf{f}+})$  is the probability that the target is outside the search region  $\Omega$  at time  $\tau_{\mathbf{f}+}$ . In general, if action  $\mathbf{f}$  fails, the probability of any cube that is outside the influence range of  $\mathbf{f}$  increases and the probability of any cube that is inside the influence range of  $\mathbf{f}$  decreases. The next action is selected based on this updated information. In this way, the search process is guided by the target probability distribution.

The probability of the target being present within the sensed sphere of the cur-



rent position  $j$  is called the “covering probability” and is defined as  $Prob_{\Psi_j} = \sum_{c_i \in \Psi_j} \mathbf{p}(c_i)$ , where  $\Psi_j$  is the region within the sensed sphere at position  $j$ . This probability decreases every time an operation is applied with a negative result. When this probability decreases below some threshold  $\Theta_{move}$ , the robot moves to a different position where the probability of detecting the target is higher.

### 3.3.3 “Where to move next”

The best next position must satisfy two requirements: it must be reachable and have a high probability of detecting the target. The strategy here also is a ‘best-first’ one.

As we have mentioned before, the search space  $\Omega$  is tessellated into a 3D grid. This tessellation divides the horizontal plane of the search region into a 2D grid. Since the robot moves only horizontally and the height of its camera does not change in our implementation, only the vertices of the 2D grid are considered as the possible robot positions. There is no loss of generality here and the formulation extends to the third dimension (height) in the obvious manner, if the robot is so equipped.

For each possible position  $j$ , an expected sensed sphere  $\Psi_j$  is estimated. If the configuration of the environment is not known, the sensed sphere is calculated based on the knowledge obtained so far.  $Prob_{\Psi_j} = \sum_{c_i \in \Psi_j} \mathbf{p}(c_i)$  is the covering probability of position  $j$ . The robot then moves to an accessible position with the largest  $Prob_{\Psi_j}$ . If such position does not exist or the time constraint is exceeded, the algorithm fails.

### 3.4 *A priori search knowledge*

Initial probabilities can be set in a number of ways. Among the possibilities, individually or in combination, are:

- Type 1 No knowledge
- Type 2 Indirect search knowledge
- Type 3 Hints
- Type 4 Saliency knowledge
- Type 5 Predictive knowledge

Basically, the initial PDF highlights order of regions to try first. No knowledge leads to a uniform initial PDF distribution. Indirect Search Knowledge [11, 12] involves intermediate target objects in spatial proximity to target. Hints lead to higher probability values in specific regions and may include such information as past knowledge of the object’s location. Saliency knowledge [30, 27] takes into account distinctive target features in the search region. The PDF may be modified by saliency computation on each acquired image for use in determining the next viewpoint. Predictive Knowledge [31, 32] provides location possibilities due to pre-processing or spatiotemporal constraints. In this paper, Types 1 and 3 are explored and the remainder are left for future work.

## 4 **Implementation**

Basic requirements for the successful implementation of a search agent include having a method for determining depth, a method for detecting the target, and means to control sensor parameters and mobility.

The first implementation of the search algorithm [28] was based on the ARK robot [33], which is a mobile platform where depth was determined via laser ranger-finder. The laser is mounted on a robotic head with pan and tilt capabilities. It consists of a camera with a controllable focal length, a laser-range finder and a mirror. The mirror is used to ensure collinearity of effective optical axes of the camera lenses and range finder. With this mirror, the laser finder can measure the distance from the center of the camera to the object along the camera-viewing axis. The robot had a method for detection of a baseball by identifying a white round blob in the image.

Our search agent is implemented on a Pioneer 3 robot, a mobile four-wheel differentially steered drive ActiveMedia Robotics platform. The platform is equipped with a Point Grey Research Bumblebee camera mounted on a Directed Perception pan-tilt unit. It is a two lens stereo vision camera that is used for both target detection and environment data acquisition. To obtain depth information, the Bumblebee camera uses Triclops Stereo Vision Software Development Kit [34] that provides stereo processing capabilities. This library does stereo processing on the images obtained from the cameras. It establishes correspondence using the Sum of Absolute Differences correlation method. The TangentBug algorithm [35, 36] is used for navigation. Sections 4.1 and 4.2 describe the search robot's navigation using the stereo camera in more detail.

In general, the search agent may have several recognition algorithms available (*a* of Section 3.1). The planning module would identify which algorithm should be applied to what region of the environment in order to yield the best results. The examples shown in this paper employed two different detection methods, one based on normalized gray-scale correlation [37] and the other based on the

SIFT feature detection algorithm that is detailed in [38]. We use Rob Hess’s implementation [39] of the SIFT feature detector. The full search example in Section 5.1 uses the correlation-based detection algorithm. The rest of the examples and experiments use the SIFT method. The current implementation does not have the planning module that chooses which algorithm to use. Future implementations will include it.

#### 4.1 Sensed sphere from stereo

A sensed sphere can be constructed using any depth sensor. Our robot uses PointGrey BumbleBee stereo camera for the task. First, sparse depth sampling of the environment is obtained and then it is converted into a sensed sphere representation.

The camera comes with the Triclops SDK. Its stereo processing module uses the Sum of Absolute Difference algorithm. In addition, the SDK provides a number of validation techniques to reduce errors in stereo data, e.g. back and forth, texture, surface validation, etc. We use the texture validation technique [40] to reduce the number of errors that result from incorrect correspondence.

We combine the stereo data using the conventional inverse sensor model occupancy grid approach. The update equation is based on a Bayes filter. We use its logarithmic form, which is computationally advantageous and also avoids numerical instabilities that arise when probabilities are closer to zero:

$$\log \frac{p(m_i|z^t)}{1 - p(m_i|z^t)} = \log \frac{p(m_i|z_t)}{1 - p(m_i|z_t)} + \log \frac{1 - p(m_i)}{p(m_i)} + \log \frac{p(m_i|z^{t-1})}{1 - p(m_i|z^{t-1})} \quad (5)$$

where  $p(m_i|z^t)$  gives probability that cell  $c_i$  is occupied given information accumulated up to time  $t$ . The inverse sensor model  $p(m_i|z_t)$  specifies the

probability that a grid cell  $i$  is occupied based on a single sensor measurement  $z_t$ .  $p(m_i)$  gives the initial probability that cube  $c_i$  is occupied. In our current experiments, we do not include any prior knowledge about the solidity of cells, therefore,  $p(m_i) = p(-m_i) = 0.5$ . This makes  $\log \frac{1-p(m_i)}{p(m_i)} = \log 1 = 0$ .

Our inverse sensor model is largely based on the one proposed by [40]. For each 3D point derived from stereo data, an uncertainty region is defined as:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{B(u \pm 0.5)}{d \pm 0.5} \\ \frac{B(v \pm 0.5)}{d \pm 0.5} \\ \frac{fB}{d \pm 0.5} \end{bmatrix} \quad (6)$$

where  $(x, y, z)$  is 3D coordinates of a point as derived from the disparity image,  $v, u, d$  are row, column and disparity value in the disparity image.  $B$  is the baseline and  $f$  is the focal length of the system.

Let  $U$  be an uncertainty region of a point  $(x, y, z)^T$  as defined in Equation 6. Let  $C$  be a union of cubes  $c_i$  that overlap with the uncertainty region  $U$ , i.e.  $C = (\cup_{i=1}^n c_i) \cap U$ . We define

$$\log \frac{p(m_i|z_t)}{1 - p(m_i|z_t)} = \frac{1}{|C|} \quad (7)$$

where  $p(m_i|z_t)$  gives the probability that a grid cell  $i$  is occupied based on the single sensor measurement  $z_t$ .

If a grid element  $(x, y, z)^T$  is between the uncertainty region of the sensed obstacle and the robot, it is likely to be unoccupied and therefore we reduce

its probability:

$$\log \frac{p(m_i|z_t)}{1 - p(m_i|z_t)} = s \tag{8}$$

where  $s$  is a number that makes  $p(m_i|z_i) \ll p(\neg m_i|z_i)$ .

The following equation summarizes our inverse sensor model:

$$\log \frac{p(m_i|z^t)}{1 - p(m_i|z^t)} = \begin{cases} 1/|C| + \log \frac{p(m_i|z^{t-1})}{1 - p(m_i|z^{t-1})} & \text{if } c_i \in C \\ s + \log \frac{p(m_i|z^{t-1})}{1 - p(m_i|z^{t-1})} & \text{if } c_i \notin C \end{cases}$$

#### 4.2 Navigation

The search agent requires basic navigational skills to move around the room. We have adopted the TangentBug algorithm for this purpose [35] [36]. It uses range data to compute a locally shortest path, based on the structure termed the local tangent graph. The algorithm uses a graph for choosing the locally optimal direction while moving toward the target, and for making local short-cuts and testing a leaving condition while moving along an obstacle boundary. The transition between these two modes of motion is governed by a globally convergent criterion, which is based on the distance of the target from the robot. The authors of the algorithm showed using simulations that TangentBug produces paths that in simple environments approach the globally optimal path, as the sensor’s maximal detection-range increases.

In its original formulation, TangentBug assumes that a robot is equipped with a ring of range sensors that have a combined field of view of 360 degrees. These sensors provide distances to obstacles around the robot. Our search robot has only one stereo camera. A straightforward extension would be to require it to

pan around and collect range samples from all directions before each move. Our experiments have shown that such a strategy is not necessary. As the robot searches for the target object, it acquires information about the solidity of the environment. It uses this information to approximate the distances to obstacles around it. Thus, we require the robot to take range measurements only in the direction of its movement during navigation.

## 5 Experimental results

### 5.1 A Typical Full Search

Figure 1 shows the search region that is part of our laboratory. The region’s dimensions are 9m x 5m x 2.5m (width x length x height). We divide the floor plane into 1x1  $m^2$  grid. Each vertex of the grid is a potential robot’s location. The search region is also divided into  $5^3 cm^3$  voxels that hold target probability and solidity values of the environment. Figure 2 shows the target object used in this experiment. Its size is 23 cm in diameter. The robot does not have any prior knowledge about the target’s location, i.e. the target’s probability distribution is uniform. The detection algorithm used in this experiment is not viewpoint-independent. Therefore, we assume that the target faces the camera. The tilt range of the camera is  $(-42^\circ, 30^\circ)$ . The only tilt angles used in this example are  $0^\circ$  and  $30^\circ$ . Each time the search agent chooses from the total number of 17 (*pan, tilt*) angles. This example was summarized in [41] and is now described in full here.

Each stage of the robot’s performance is qualitatively detailed in Table 1. The



Fig. 1. The search region used in our experiments. It is a typical laboratory environment.



Fig. 2. The target object used in the full search example.

two columns of the Table are:

**Description** provides brief description of each action.

One of two possible Map types is shown in the second column:

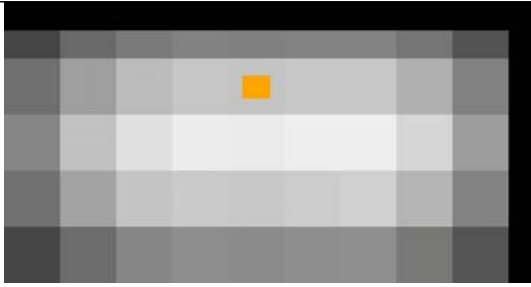
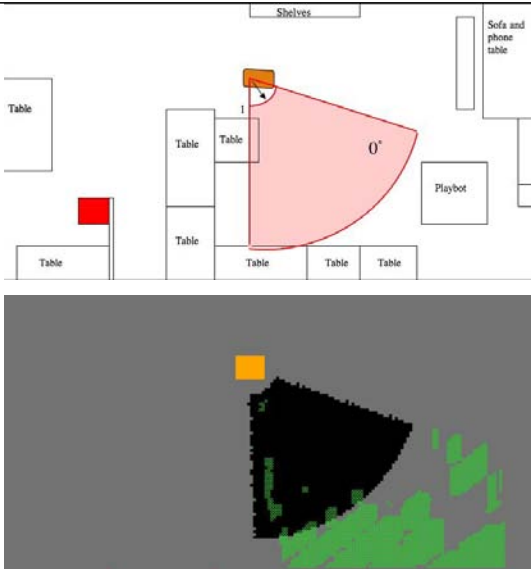
**Graphical Map (GM)** shows the progress of the search process schematically. The robot is represented by the orange rectangle. The path of the robot is shown with bold arrows. At each position, small arrows show the viewpoints inspected. The visual field for each viewpoint is shown as a colored region, with the depth of field for which recognition was possible delimiting the extent of the region. Elevation of the camera is  $0^\circ$  or  $30^\circ$  and is marked on each view. The target object is on the table, marked by a red rectangle in the lower left corner.

**Probability Map** shows probability maps of two kinds: “position probability” and “target probability”. “Target probability” maps (TPM) show how the agent’s knowledge of the environment changes as the search progresses.

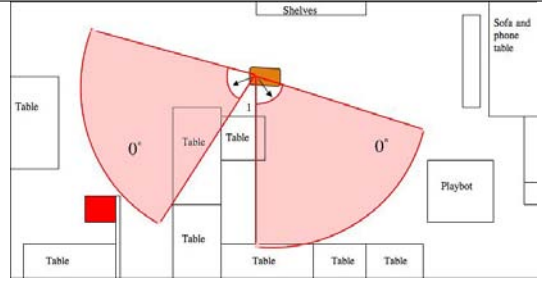


Red lines show the configuration of the room (it is not known to the search agent). The color of each pixel  $(x, y)$  corresponds to the sum of probabilities in the vertical column of the 3D environment with floor coordinates  $(x, y)$ . Lighter color corresponds to larger probability. Green crosses indicate locations of obstacles discovered so far. As mentioned above, the floor plane is divided into 1m x 1m squares, vertices of which are considered by the search agent as possible positions. “Position probability” maps (PPM) show the probability of detecting the target from each position. Each square corresponds to a possible position  $j$ , and its color - to the detection probability  $Prob_{\Psi_j}$  (as described in Section 3.3.3). Lighter color means higher probability.

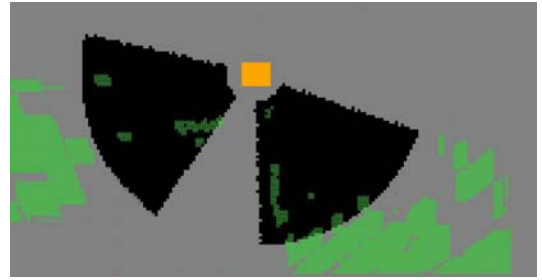
Table 1: A typical full search example.

Description	Maps
<p><b>Position 1.</b> “Position probability” map before the search starts. Since nothing is known about the environment yet, the largest probabilities are near the centre of the room: larger volume of the environment can be explored from these positions.</p>	 <p style="text-align: right;">PPM</p>
<p><b>Position 1, view 1.</b> As the robot knows nothing about solidity of the environment or target’s location, the first action is directed into the centre of the room. This viewing direction allows the robot to explore the largest volume “available” from its current position, and hence maximizes the probability of detecting the target.</p>	 <p style="text-align: right;">GM TPM</p>

**Position 1, view 2.** During each action, the robot updates its solidity map of the environment based on stereo reading. The green indicates the obstacles discovered so far.

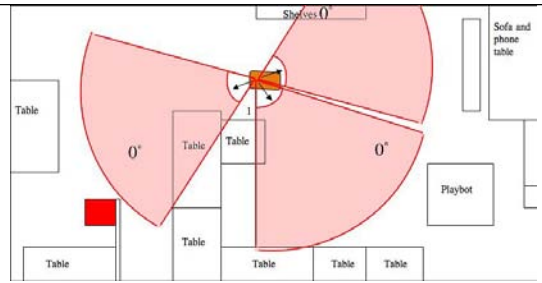


GM

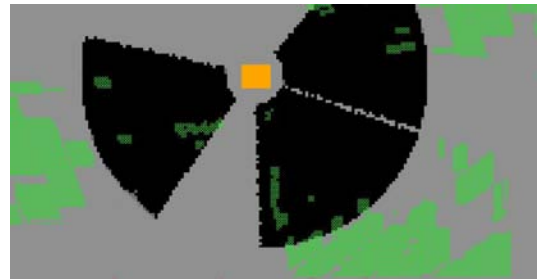


TPM

**Position 1, view 3.** As the search progresses, the color of still unexplored regions becomes lighter while the color of already “seen” regions becomes darker. This color change corresponds to the update of the target probability.

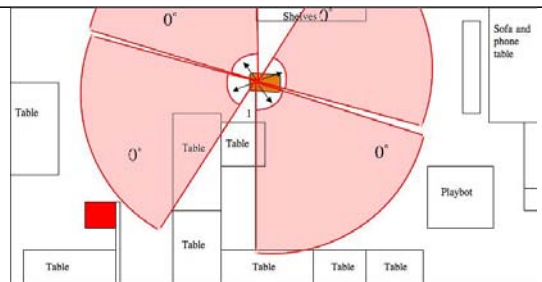


GM

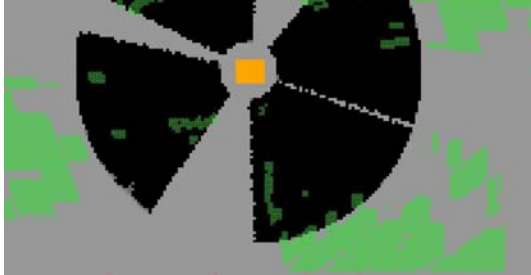
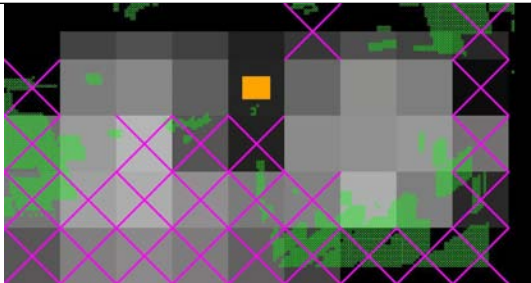
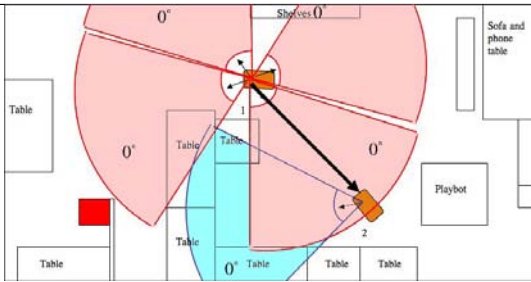
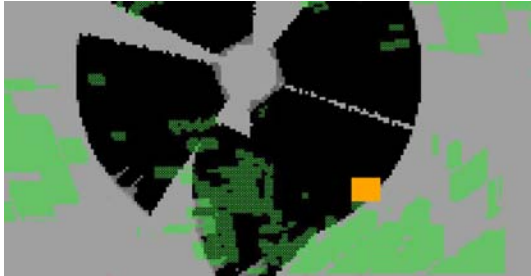
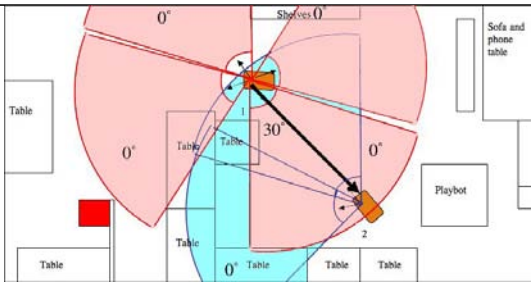


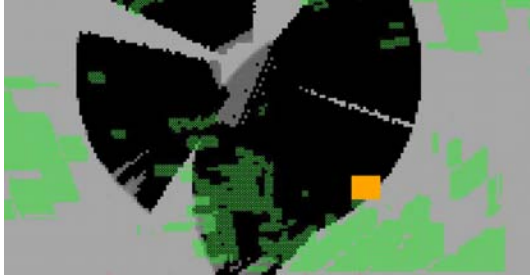
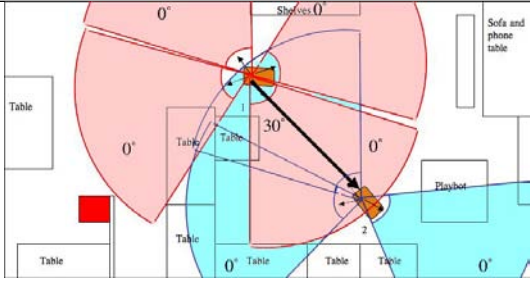
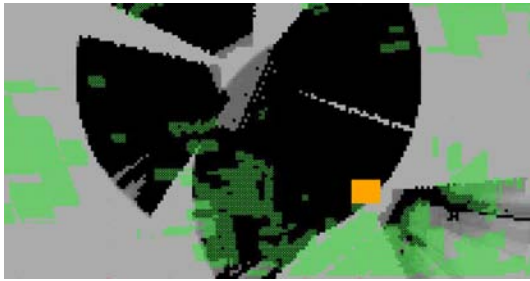
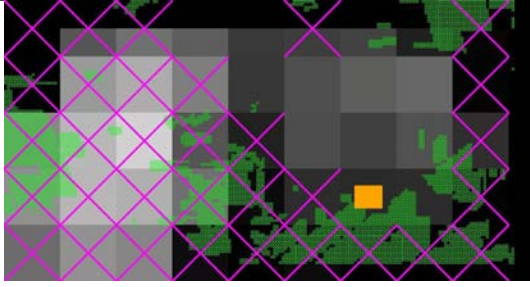
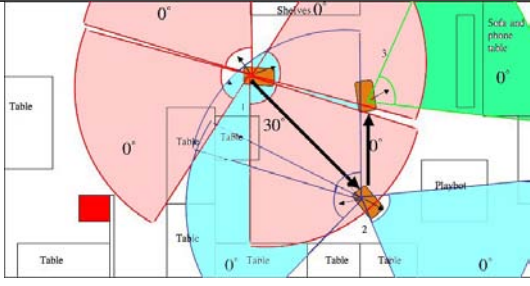
TPM

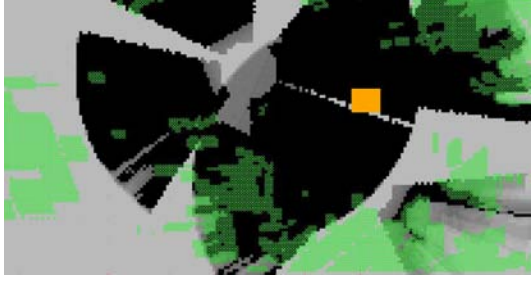
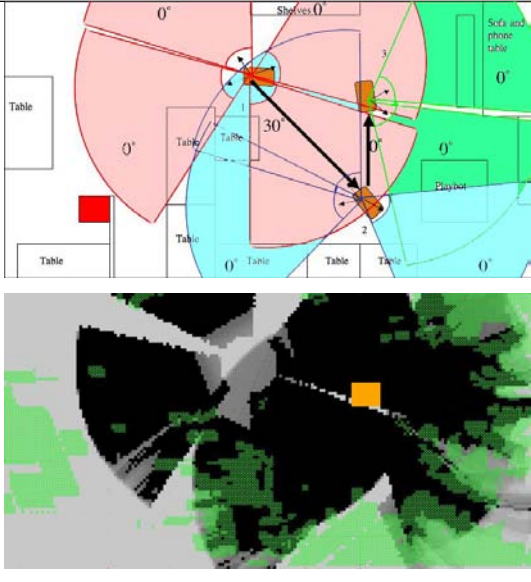
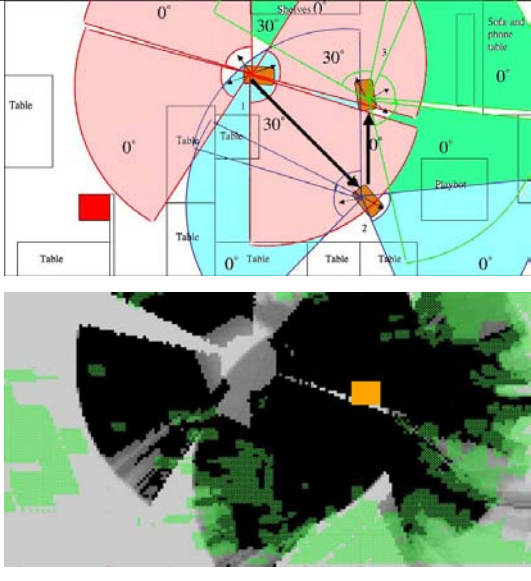
**Position 1, view 4.** The last viewing direction explored at the current position. Probability of detecting the target at the current position decreases below some threshold  $\Theta_{move}$ , the search agent looks for the next best position.



GM

		TPM	
<p><b>Choosing position 2.</b> “Position probability” map before the first move of the robot. In this example, the robot considers only the positions that are reachable by moving along a straight line. The crosses mark unreachable positions. The color of each square indicates the detection probability at that position.</p>		PPM	
<p><b>Position 2, view 1.</b> The robot moves to its next position and takes the next view (marked in blue). The move is showed by the black arrow.</p>	 	GM	TPM
<p><b>Position 2, view 2.</b> The number of degrees within each colored area gives the <i>tilt</i> of the camera during each action. In this action, the camera is tilted up 30 degrees, because the horizontal view was already performed and the next best viewing direction was above the horizontal.</p>		GM	

	 <p style="text-align: right;">TPM</p>
<p><b>Position 2, view 3.</b> The last viewing action at this position. The detection probability falls below <math>\Theta_{move}</math>.</p>	 <p style="text-align: right;">GM</p>  <p style="text-align: right;">TPM</p>
<p><b>Choosing Position 3.</b> “Position probability” map before the second move. Although the best spot is on the left, it is not reachable, so a relatively low probability location is chosen.</p>	 <p style="text-align: right;">PPM</p>
<p><b>Position 3, view 1.</b> The robot moves to the third position and takes the next view (in green).</p>	 <p style="text-align: right;">GM</p>

		TPM
<p><b>Position 3, view 2.</b></p>		GM TPM
<p><b>Position 3, view 3.</b> In this action, the camera is tilted up 30 degrees.</p>		GM TPM



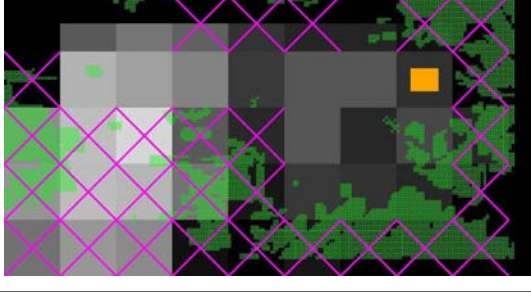
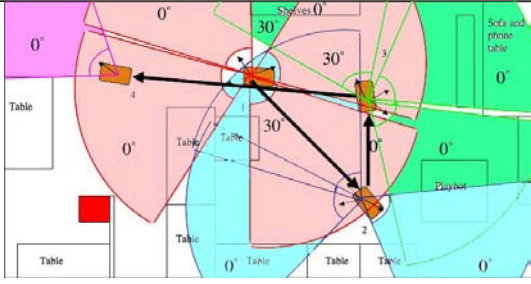
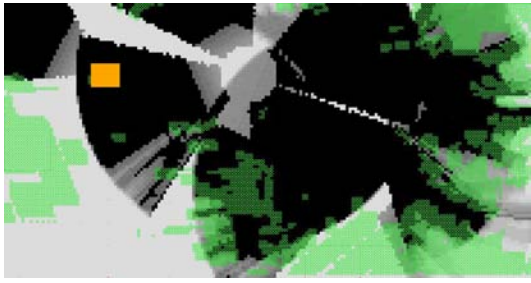
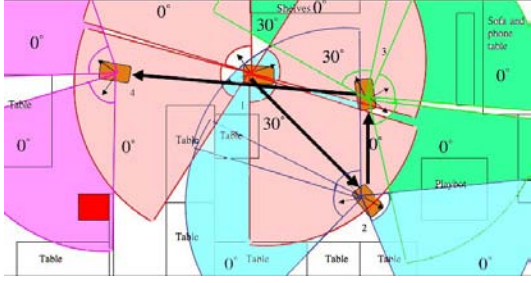
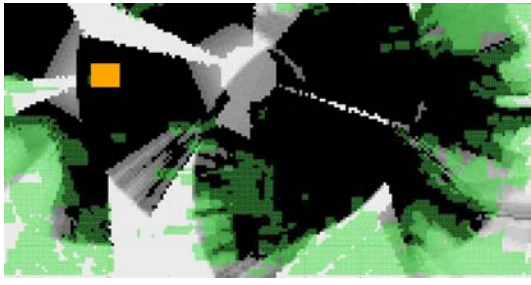
<p><b>Choosing position 4.</b>“Position probability” map before the third move.</p>		<p>PPM</p>
<p><b>Position 4, view 1.</b> The robot moves to the fourth position and takes the next view(in magenta). Because of a drawing quantization artifact, it may seem that the robot moves into a sensed obstacle. In reality, the robot is just close to it.</p>	 	<p>GM PPM</p>
<p><b>Position 4, view 2.</b>The target is found.</p>	 	<p>GM PPM</p>

Table 1: A typical full search example.

## 5.2 Testing different search utility functions

In this section, we present several additional example runs of the search robot as well as some quantitative results. These examples demonstrate the search process as well as the influence on it of such factors as a priori knowledge and action cost. In these experiments, we used the target object in Figure 3. The goal of this work and the experimental verification is not to study recognition methods per se; rather it is to study the viewpoint control strategy and to understand its properties and thus only a single object is employed in order to permit proper comparisons among the experimental setups and results. Naturally, in the real world the robot must be able to search for many different kinds of objects under the full variability of lighting conditions, for example; this is left for another experiment.

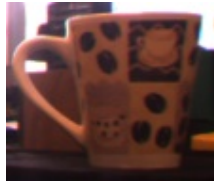


Fig. 3. The target object used in experiments of Section 5.2.

### 5.2.1 Cost

During the quantitative evaluation of the search algorithm, we look at the following cases:

- (1) Choose the action  $\mathbf{f}_{\tau_{\mathbf{f}+}}$  with the largest detection probability:

$$\mathbf{f}_{\tau_{\mathbf{f}+}} = \arg \max_{\mathbf{f}} \sum_{c_i \in \Psi_{\mathbf{f}}} \mathbf{p}(c_i, \tau_{\mathbf{f}}) \quad (9)$$

where  $\Psi_{\mathbf{f}}$  specifies the influence range of  $\mathbf{f}$  as defined in Section 3.1. Action  $\mathbf{f}$  is an arbitrary action that is specified by  $(x, y)$  location of the robot

and (*pan, tilt*) direction of the camera.

- (2) Explore the current position first. Choose the next position  $\mathbf{pos}_{\tau_{f+}}$  that maximizes the detection probability:

$$\mathbf{pos}_{\tau_{f+}} = \arg \max_{\mathbf{pos}} \sum_{c_i \in \Psi_{pos}} \mathbf{p}(c_i, \tau_{f+}) \quad (10)$$

where  $\Psi_{pos}$  specifies the influence range of  $\mathbf{pos}$  that consists of those parts of the search region that are “visible” to the search agent when it is at position  $\mathbf{pos}$ .

- (3) Explore the current position first. Choose the next position  $\mathbf{pos}_{\tau_{f+}}$  that maximizes the detection probability while minimizing the distance to the position:

$$\mathbf{pos}_{\tau_{f+}} = \arg \max_{\mathbf{pos}} \frac{\sum_{c_i \in \Psi_{pos}} \mathbf{p}(c_i, \tau_{f+})}{\mathbf{dist}(\mathbf{pos})} \quad (11)$$

where  $\mathbf{dist}(\mathbf{pos})$  is the distance from the robot’s current position to  $\mathbf{pos}$ .

- (4) Explore the current position first. The utility function for choosing the next position  $\mathbf{pos}_{\tau_{f+}}$  combines the utility functions in cases (2) and (3):

$$\mathbf{pos}_{\tau_{f+}} = \arg \max_{\mathbf{pos}} \sum_{c_i \in \Psi_{pos}} \mathbf{p}(c_i, \tau_{f+}) \left(1 + \frac{A}{\mathbf{dist}(\mathbf{pos})}\right) \quad (12)$$

Cost (1) is different from Ye [28]’s approach. It always chooses the action with the highest detection probability no matter how far the robot has to travel to reach the configuration required for the action. Costs (2)-(4) follow Ye’s two-stage (“where to look next” - “where to move next”) approach. First, the current position is inspected until some threshold  $\Theta_{move}$  is reached. Then the new position is selected based on one of the criteria described by Equations 10-12. In Cost (2), the next best position is the position that maximizes the detection probability. In Cost (3), the next best position should maximize the



detection probability while minimizing the distance travelled. Cost (4) is a combination of (2) and (3): it relaxes the requirement of distance minimization using the control  $A$ . In the experiments  $A = 1$ .

### 5.2.2 Search examples

Figure 4 shows the sequences of viewpoints that were inspected by the robot during 3 runs of the experiment. The brown rectangle represents the search robot. Its path is shown with bold arrows. At each position, small arrows show the viewpoints inspected. The visual field for each viewpoint is shown as a colored region, with the depth of field for which recognition was possible delimiting the extent of the region. The target object is on the table in the centre of the room. Its location is marked by the black cross.

No prior knowledge about the target’s location is available during the first two runs (Figure 4 (a) and (b)): the target probability distribution is uniform at the beginning of both runs. During the first run in Figure 4 (a), we use the utility function defined in Cost (2) in Section 5.2.1: the next position is the position with the largest detection probability. During the second run in Figure 4 (b), the utility function defined in Cost (3) in Section 5.2.1 is used: the next position is the position that maximizes the detection probability while minimizing the distance. One can see that in the second run the robot prefers to examine locations that are nearby before moving to the other side of the room.

In Section 3.4, we described different types of prior knowledge that can be used to speed up the search process. Figure 4 (c) shows an example run where the robot knows that the target object is on one the tables in the room and where these tables are located. The utility function defined in Cost (3) is used.



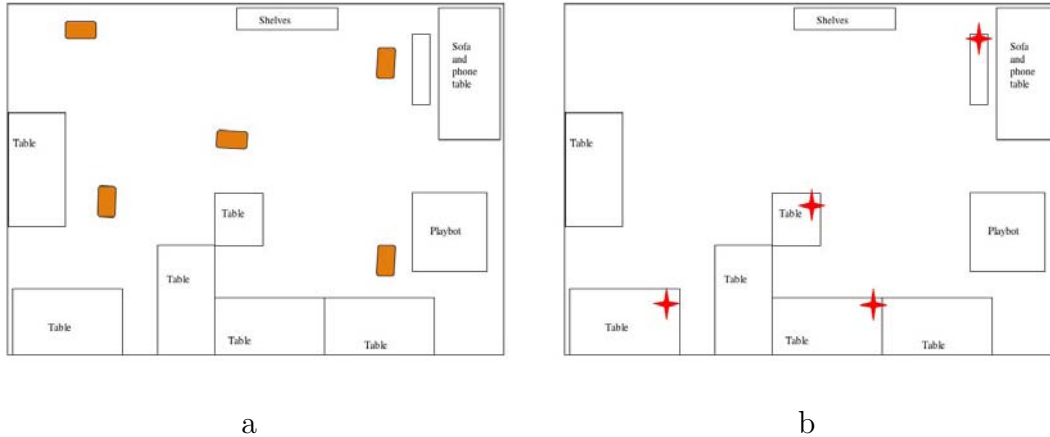


Fig. 5. Experimental setup. To test the influence of various action costs and a priori knowledge on the search process, we ran 20 experiments for each case. The robot was placed in one of the five locations denoted by brown rectangles in (a). The target was placed in one of the four locations denoted by red stars in (b).

the robot with some prior knowledge of the target’s location greatly improves the search process: minimizing the number of viewing actions and number of explored positions, and hence the time to find the object. These conclusions are supported by the quantitative results presented in Section 5.2.3.

### 5.2.3 Quantitative results

For each cost type of Section 5.2.1, we did two types of experiments:

- (1) No prior knowledge
- (2) The search agent knows that the target object is on one of the tables and knows the location of the tables in the room.

For each combination of prior knowledge and cost options, we ran 20 experiments. We varied the robot’s starting position by one of the locations denoted by brown rectangle in Figure 5 (a). For each robot’s starting position, we put the target object in one of the four locations marked by red stars in Figure 5 (b). Thus, we performed 160 experiments in total. 91% of these runs were

successful. The causes for failures include: lack of a localization algorithm and unreliability of the stereo depth sensor. The robot relies on dead reckoning for localization and therefore, gets lost with respect to the map if it moves a lot. The depth estimates from the stereo camera is not 100% reliable, and as a result the robot sometimes sees imaginary obstacles it cannot avoid. The search performance is measured in terms of total number of actions, total time of the search and the total distance travelled by the robot. Tables 2 and 3 show the influence of prior knowledge and various cost functions on the results.

<b>Performance of the search system</b>				
Average per run	Cost (1) +	Cost (2) +	Cost (3) +	Cost (4) +
	No knowledge	No knowledge	No knowledge	No knowledge
Number of actions	9.4375	8.722	8.45	8.733
Total time (min.)	16.1875	8.111	6.8	7.533
Distance traveled (m.)	22.385	8.77	3.85	8.0273

Table 2

Performance of the search system based on the experiments set described in Section 5.2.3. The search agent does not have any prior knowledge about the location of the target.

<b>Performance of the search system</b>				
Average per run	Cost (1) +	Cost (2) +	Cost (3) +	Cost (4) +
	Knowledge	Knowledge	Knowledge	Knowledge
Number of actions	5	5.5	4.9	4.933
Total time (min.)	10.4375	5.9	3.8	4.333
Distance traveled (m.)	13.4675	5.9815	3.4095	4.14133

Table 3

Performance of the search system based on the experiments set described in Section 5.2.3. The search agent knows that the target object is on one of the tables and knows the location of the tables in the room.

Several conclusions can be drawn:

- Prior knowledge about the target’s location can drastically accelerate the search process in terms of total time, total number of actions and distance travelled.
- Various cost functions of selecting the next best action do not have any significant effect on the total number of actions.
- The division of the search process in two stage: “where to look next” and “where to move next” -reduces distance travelled and, hence, the total time of the search significantly (any of Costs (2), (3) and (4)).
- To further reduce the distance travelled and the total time, the cost function that minimizes the distance to the next best position should be used (Costs (3) and (4)).
- The best results are achieved by the search agent possessing some prior knowledge and using the cost function with the strongest distance minimization(Cost (3)).
- The number of actions is best minimized by any of Costs (1), (3) and (4).

### 5.3 *Why not use a POMDP?*

At first glance, Partially Observable Markov Decision Processes (POMDP) seem to be an obvious strategy to approach our search problem. POMDP were introduced by Kaelbling, Littman and Cassandra [42] as a general procedure for solving the problem of choosing optimal actions in partially observable stochastic domains. Yet the standard value function approaches to finding polices are generally considered intractable for large modes. In [43], Roy et al. use dimensionality reduction approach to overcome this limitation. They argue that optimal polices are not always needed, and therefore, their approximation

algorithm suffices. They demonstrated the use of the algorithm on a robot localization problem with 20,230 states (238x85 grid cells).

We argue that the search problem cannot be easily solved using POMDP. First of all, our problem is much larger. In our experimental setup described in Section 5.1, there are 32 possible robot positions and 17 possible camera directions at any position. Thus, the total size of the robot's action set is 544. There is no restriction on the target's location in the room. Thus, it can occupy any of the 451,200 occupancy grid positions, each of which codes two values: target and solidity probability. In addition, the initial target probability distribution is not necessarily uniform and can be assigned based on the initial knowledge as described in Section 3.4. Lastly, our approximation algorithm is not intended for a generic problem solution, and therefore is more intuitive for the task.

### *5.3.1 System performance*

The search module runs on Dual 2GHz PowerPC G5 with 1GB RAM and sends requests to the robot, such as move, take an image, etc., through the wireless network. The code has not been optimized. Each action takes about 15 seconds and includes taking a stereo and raw images, sending them to the off-board computer, stereo analysis and application of the detection algorithm, update of the world representation with the stereo and detection results, choosing the next best action, sending request to the robot to setup and execute actions to prepare for the next search image. The run in Figure 4 (a) takes about 12 minutes, in Figure 4 (b) - 6 minutes, in Figure 4 (c) - 4 minutes.

## 6 Extending the Viewpoint Planning Method

### 6.1 *Object-Centred Viewpoint-Dependent Detectability Function*

Whether or not a camera system plus detection function can actually be in a position where detection is possible depends on a variety of interacting factors. Detectability depends on viewpoint, image size, distance, scale, rotation in 3D occlusion, let alone lighting conditions or surface reflectance. Here, the only dimensions of this problem that we address is those dependent of viewpoint selection and object pose. A target object may be at any pose in the environment; it may even be hidden. It is out of the scope of this work to deal with the more difficult situations. However, some progress can be made if one considers the performance of the chosen detection algorithm(s) with respect to object pose or degree of occlusion. Some of this empirical work can be seen in [44]. A detection accuracy function can be obtained by testing detection as pose changes, for example, or as the degree of occlusion increases. The use of an object-centred viewpoint-dependent detectability function can set the boundaries on detectability so that not all possible viewpoints need to be considered. In other words, this determines positions/viewpoints from where the target can be recognized. For example, if detection performance degrades to 85% with 20° object rotation in the image plane, and if this level of performance is acceptable for the task, then it means that viewpoint positions need be quantized only in 20° intervals in the image plane to deal with this rotation (see Figure 6). Similar constraints can be obtained for the other relevant dimensions described above.

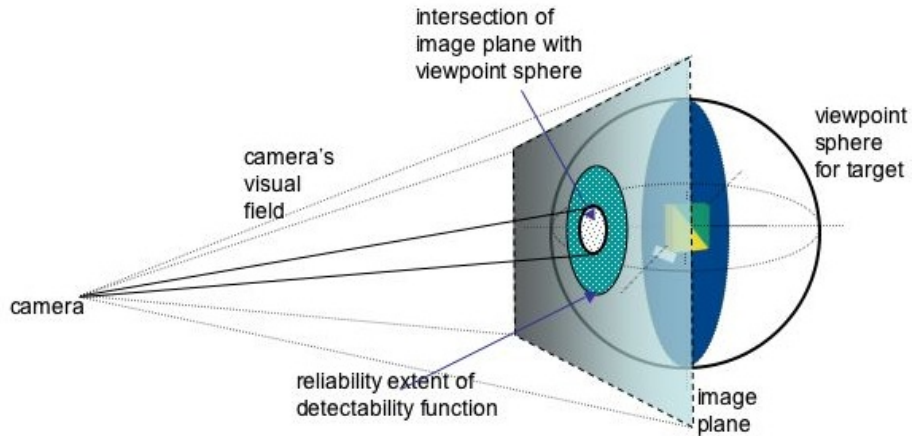


Fig. 6. Each fixation on a location at most tests a subset of viewpoint/pose relationships for the target. A location is not fully ruled until the full viewing sphere is inspected (up to detectability function constraints for the target).

## 6.2 Location-Centred Viewpoint-Dependent Visibility Function

Each fixation on a location at most tests a subset of viewpoint/pose relationships for the target. A location is not fully ruled until the full viewing sphere is inspected (up to detectability constraints for the target that are set by the object-centred viewpoint dependent detectability function). Future implementations will include these extensions.

## 7 Conclusions

The search for an object in a 3D space benefits greatly from attentive mechanisms that limit the search space in a principled manner. We presented a solution to this problem with an effective implementation using a robotic agent.



Our solution performs search for a target object in an unknown 3D environment. No assumptions are made about the configuration of the environment other than the locations of the exterior boundary nor the position of the target object. Since our search agent generates its path based on its current knowledge of the targets location encoded by the probability distribution computed during the search process, it can minimize the expected time to find the object, and does not simply determine a path that covers the entire environment.

The current implementation does not have the planning module that chooses between available detection and recognition algorithms. The detection algorithm used is viewpoint-dependent, however, our implementation assumes it is not. Therefore, the target object is assumed to face the camera. The viewpoint planning method discussed in the previous section is needed. Future work will include these enhancements. We also plan to explore the influence of the prior knowledge of Types 2, 4 and 3 on the search process.

We performed an extensive empirical evaluation of our implementation of the search algorithm. Several action cost functions were derived and their influence on the search process was analyzed. Although the cost functions that minimize the distance travelled significantly reduces the total search time, the best improvement in the number of actions and positions and the total search time comes with the use of prior knowledge.

*Acknowledgements* The authors wish to thank Dr. Ehud Rivlin for providing the code for the TangentBug algorithm. Research support was gratefully received from the Natural Sciences and Engineering Research Council of Canada, the Canada Foundation for Innovation, the Ontario Innovation Trust, and the Canada Research Chairs Program.

## References

- [1] A. L. Rothenstein, J. Tsotsos, Attention links sensing to recognition, *Image and Vis. Comput.*
- [2] J. Tsotsos, The complexity of perceptual search tasks, in: *Proc. of Int. Joint Conf. on A. I.*, 1989, pp. 1571–1577.
- [3] J. Tsotsos, A ‘complexity level’ analysis of vision, in: *Proc. of Int. Conf. on Computer Vision: Human and Machine Vision Workshop*, 1987.
- [4] J. Tsotsos, Analyzing vision at the complexity level, *Behavioral and Brain Sciences* 13 (3) (1990) 423–445.
- [5] J. Tsotsos, On the relative complexity of active vs. passive visual search, *Int. J. Comput. Vis.* 7 (2) (1992) 127–142.
- [6] R. Rensink, A new proof of the np-completeness of visual match, Tech. rep., Computer Science Department, University of British Columbia (1989).
- [7] P. Parodi, R. Lanciwicki, A. Vijh, J. Tsotsos, Empirically-derived estimates of the complexity of labeling line drawings of polyhedral scenes, *Artificial Intelligence* (1998) 47–75.
- [8] R. Bajcsy, Active perception vs. passive perception, in: *Proc. IEEE Workshop on Computer Vision: Representation and Control*, 1985, pp. 55–62.
- [9] D. Wilkes, J. K. Tsotsos, Active object recognition, in: *CVPR’92*, 1992, pp. 136–141.
- [10] S. J. Dickinson, H. I. Christensen, J. K. Tsotsos, G. Olofsson, Active object recognition integrating attention and viewpoint control, *Comput. Vis. and Image Understand.* 67 (3) (1997) 239–260.
- [11] T. D. Garvey, Perceptual strategies for purposive vision, Tech. rep., SRI International, 117 (1976).
- [12] L. Wixson, D. Ballard, Using intermediate object to improve efficiency of

- visual search, *Int. J. Comput. Vis.* 18 (3) (1994) 209–230.
- [13] D. A. Reece, Selective perception for robot driving, Tech. rep., Carnegie Mellon Computer Science (1992).
- [14] Y. Ye, J. Tsotsos, A complexity level analysis of the sensor planning task for object search, *Comput. Intelligence* 17 (4) (2001) 605–620.
- [15] C. I. Connolly, The determination of next best views, in: *IEEE Proc. Int. Conf. Robotics and Automation*, 1985, pp. 432–435.
- [16] J. Maver, R. Bajcsy, Occlusions as a guide for planning the next view, in: *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 15, 1993, pp. 417–433.
- [17] H. Kim, R. Jain, R. Volz, Object recognition using multiple views, in: *IEEE Proc. Int. Conf. Robotics and Automation*, 1985, p. 2833.
- [18] C. K. Cowan, P. D. Kovesi, Automatic sensor placement from vision task requirements, in: *IEEE Trans. on Pattern Anal. Mach. Intell.*, Vol. 10, 1988.
- [19] L. Wixson, Viewpoint selection for visual search, in: *CVPR*, 1994, pp. 800–805.
- [20] N. Ruangpayoongsak, H. Roth, J. Chudoba, Mobile robots for search and rescue, in: *IEEE Safety, Security and Rescue Robotics Workshop*, 2005, pp. 212–217.
- [21] Y. Fukazawa, C. Trevai, J. Ota, H. Yuasa, T. Arai, H. Asama, Controlling a mobile robot that searches for and rearranges objects with unknown locations and shapes, in: *Proc. of the 2003 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2003, pp. 1721–1726.
- [22] B. Tovar, S. LaValle, R. Murrieta, Optimal navigation and object finding without geometric maps or localization, in: *IEEE Proc. Int. Conf. Robotics and Automation*, 2003, pp. 464–470.
- [23] A. Sarmiento, R. Murrieta-Cid, S. Hutchinson, An efficient strategy for

- rapidly finding an object in a polygonal world, in: IEEE Proc. Int. Conf. Robotics and Automation, 2003, pp. 1153–1158.
- [24] A. Sarmiento, R. Murrieta-Cid, S. Hutchinson, A sample-based convex cover for rapidly finding an object in a 3d environment, in: IEEE Proc. Int. Conf. Robotics and Automation, 2005, pp. 3486–3491.
- [25] H. Lau, S. Huang, G. Dissanayake, Optimal search for multiple targets in a built environment, in: Proc. of the 2005 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2005, pp. 3740– 3745.
- [26] F. Bourgault, T. Furukawa, H. F. Durrant-Whyte, Coordinated decentralized search for a lost target in a bayesian world, in: IEEE/RSJ Proc. Int. Conf. Intelligent Robots and Systems, 2003, pp. 48–53.
- [27] S. Frintrop, *VOCUS: A Visual Attention System for Object Detection and Goal-Directed Search*, Springer, 2006.
- [28] Y. Ye, Sensor planning for object search, Ph.D. thesis, University of Toronto (1997).
- [29] Y. Ye, J. K. Tsotsos, Sensor planning for 3d object search, *Comput. Vis. Image Understand.* 73 (2) (1999) 145–168.
- [30] L. Itti, C. Koch, E. Niebur, A model of saliency-based visual attention for rapid scene analysis, in: *IEEE Trans. PAMI*, Vol. 20, 1998, pp. 1254–1259.
- [31] M. Kelly, Edge detection in pictures by computer using planning, *Machine Intelligence* 6 (1971) 397–409.
- [32] J. K. Tsotsos, A framework for visual motion understanding, Ph.D. thesis, University of Toronto (1980).
- [33] S. Nickerson, M. Jenkin, E. Millios, B. Down, P. Jasiobedzki, A. Jepson, D. Terzopoulos, J. K. Tsotsos, D. Wilkes, N. Baines, K. Tran, Ark: Autonomous navigation of a mobile robot in a known environment, *Autonomous Systems-3* (1993) 288–296.

- [34] Available: <http://www.ptgrey.com/products/triclopsSDK/index.asp>.
- [35] I. Kamon, E. Rimon, E. Rivlin, Tangentbug: a range-sensor-based navigation algorithm, *The International Journal of Robotics Research* 17 (9) (1998) 934–953.
- [36] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, S. Thrun, *Principles of Robot Motion*, The MIT Press, 2003, Ch. Bug algorithms.
- [37] W. MacLean, J. K. Tsotsos, Fast pattern recognition using gradient-descent search in an image pyramid, in: *Proc. of the 15 Int. Conf. on Pattern Recognition*, 2000, pp. 877–881.
- [38] D. Lowe, Object recognition from local scale-invariant features, in: *Int. Conf. on Computer Vision*, Vol. 2, 1999, pp. 1150–1158.
- [39] R. Hess, online, available:<http://web.engr.oregonstate.edu/hess/index.html>.
- [40] D. R. Murray, J. Little, Using real-time stereo vision for mobile robot navigation, *Autonomous Robots* 8 (2000) 161–171.
- [41] J. Tsotsos, K. Shubina, Attention and visual search: active robotic vision systems that search, in: *The 5th Int. Conf. on Computer Vision Systems*, 2007.
- [42] L. Kaelbling, M. Littman, A. Cassandra, Planning and acting in partially observable stochastic domains, *Artificial Intelligence* 101 (1998) 99–134.
- [43] N. Roy, G. Gordon, S. Thrun, Finding approximate pomdp solutions through belief compression, *Journal of Artificial Intelligence Research* 23 (2005) 1–40.
- [44] K. Shubina, Sensor planning for 3d object search, Master’s thesis, York University (2007).