ATTENTION AND SENSOR PLANNING IN AUTONOMOUS ROBOTIC VISUAL SEARCH

Amir Rasouli

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF APPLIED SCIENCE

GRADUATE PROGRAM IN COMPUTER SCIENCE AND ENGINEERING

YORK UNIVERSITY

TORONTO, ONTARIO

February 2015

Abstract

An examination of saliency and sensor planning strategies in robotic visual search, using a practical robot, is presented. This thesis is concerned with the incorporation of saliency in visual search and the development of sensor planning strategies for search. The saliency model is a mixture of two schemes that extracts visual clues regarding the structure of the environment and object specific features. The sensor planning methods, namely Greedy Search with Constraint (GSC), Extended Greedy Search (EGS) and Dynamic Look Ahead Search (DLAS) are approximations to the optimal solution for the problem of object search, as extensions to the previous solutions of Ye and Shubina.

Experiments were conducted to evaluate the proposed methods and measure their performance with respect to variations in the size, configuration and setting of the environment. The experiments highlighted that by using saliency computation within visual search, a performance improvement up to 75% can be attained in terms of the number of actions taken to complete the search. Consequently, the time and energy consumption of the system is reduced significantly

As for the planning strategies, the GSC algorithm achieved the highest detection rate for the target object in various situations. It also had the best efficiency in the sense that it incurred the least cost to explore every percentage of the search environment.

Acknowledgements

I would like to thank Prof. John K. Tsotsos for his continuous support and mentorship throughout the research and completion of my thesis. I also want to thank Prof. Minas Spetsakis for his support and helpful comments on my thesis.

Special thanks to my dear friend and colleague Eugene Simine without whom the implementation and testing of my work on the robot platform would not be possible.

I also want to thank my colleague Yulia Kotseruba whose friendship and support always comforted and encouraged me during my work.

Finally, many thanks to my family for their patience, enthusiasm and love.

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Motivation

The ability to search for an object is a crucial part of any autonomous mobile robot whose tasks involve environment manipulation, item detection or social interactions. In a typical search environment, the configuration as well as the location of the target of interest that we are searching for, may vary. This means, it is not possible to memorize each environment setup and the potential target location, especially if the number of objects increases. Therefore, a framework is required to direct a robot within any unknown environment to search for a known object whose location is not known prior to the search.

While searching for an object within a cluttered environment, acquiring a single image of the environment, regardless of what viewpoint, usually does not suffice for detecting the target. In such an image, the target might be occluded by other objects or be too far distance to be recognized due to poor resolution, or be completely not in the image at all. These limitations necessitate a search agent to acquire multiple images from the environment and analyze the scene from different viewpoints in order to find the object.

The above discussion points to the need for an object search strategy which comprises the following two components: first, viewpoint selection that involves

selecting a location, navigating through the environment and configuring the sensory apparatus to capture an image, and second, analyzing the image to detect the target object.

Using a brute force approach to examine all possible viewpoints would certainly suffice for a solution, but it might not be computationally and mechanically feasible. Given the limitation of resources available to a practical robot, it is necessary to design a system to minimize the cost of search by an efficient selection of viewpoints while maximizing the chance of finding the target.

Analyzing the sensory information captured throughout the process is also crucial in object search. This data can simply be processed by applying a recognition algorithm to detect the object of interest or further be evaluated to guide the later stages of the search if the target is not found.

1.2 Previous Work

1.2.1 Object search strategies

The straightforward approach to object search is to look toward every possible view point within an environment. This is achieved by moving a camera to take images of parts of the environment that are not previously seen. Of course, such a brute force approach would suffice for a solution, but it is both computationally and mechanically prohibitive.

In an early version of visual search, Garvey [1] put forward the notion of indirect search. In this approach, a search domain is limited to those areas that have some form of spatial relationship with the object of interest. For instance, if the objective is to find a telephone, the locations of interest would be tables or desk surfaces, where there is a higher chance that the phone is placed on. Garvey divides the task of object search into two phases: first, the goal is to identify an intermediate object spatially related to the target, which typically can be detected with a lower resolution and a wider field of view; the search is then restricted only to those regions specified by the spatial relationship.

The idea of indirect search is put into practice by Aydemir *et al.* [2] in which they characterize an object's presence within an environment in the form of probability distributions. These distributions are controlled by a set of predefined spatial relationships between the target and intermediate objects. The task of search then is to choose an action sequence that maximizes the chance of detecting the target. For instance, a policy generated by the algorithm to search for a book looks like this: "Go to room 1, search for a small bookcase, search for the book on the bookcase, …".

Gobelbecker *et al.* [3] extend Aydemir *et al.*'s work by using a more generic representation of objects' relations within the search environment. In this work, the belief system of the target's presence with respect to intermediate objects encompasses two forms of knowledge: conceptual knowledge, which relates the category of one object to another, e.g. food items are found in kitchens, and instance knowledge, which

connects one specific object to another such as the cereal box is in room 2. This knowledge is updated after each instance of the search.

To further minimize a search space, Kunze and Hawes [4] use a more detailed description of objects' relations that they term as Qualitative Spatial Relations (QSRs). They categorize objects into groups of *static* and *dynamic* objects. Intuitively, the static objects are those with relatively fixed location (e.g. a desktop PC or printer) that are used as landmarks to locate dynamic objects (such as a keyboard or cup). Then, a directional relation is used to specify the configuration of a search region. For example, keyboard is "*in front of*" monitor, "*left of" cup.*

Indirect search algorithms, nevertheless, suffer from two common issues. The detection of intermediate objects is not necessarily easier than the actual target of interest. The search for an intermediate object is only viable if it is easier to be recognized or some forms of prior knowledge regarding its location are available to the search agent. More importantly, if the spatial relation between objects does not hold, indirect search fails to locate the target.

1.2.2 Active search for a 3D object

It is argued that an active vision approach is best suited for object search [5]. In contrast to simply receiving a series of prerecorded images, a search agent should actively control its sensory inputs and image acquisition process with respect to the task at hand. In the context of search, the target's presence might be ambiguous from a point of view

e.g. due to occlusion by another object. Thus, to reduce occlusion, the sensor can be moved to a different position to capture additional images from a different point of view.

In [6], Wilkes and Tsotsos introduce the concept of active object recognition in which a camera is mounted on a robot arm with a mobile base. The mobility of the system is used to position the camera at a standard viewpoint in regard to the object of interest. From this perspective, recognition takes place matching the target image with a two-dimensional pattern learned beforehand by an algorithm.

Dickinson *et al.* [7] propose a similar active approach. Given an ambiguous view of an object, their algorithm determines whether there is a more discriminating view of an object. If this is the case, it changes the direction of the camera to capture that view. At the end, it specifies the visual events such as appearance of object features that are encountered while moving the camera to the new viewpoint.

Alternatively, in [8] the pose of an unknown object is altered with respect to a camera using a humanoid robot arm. Each view of the object is characterized by a probability distribution indicating how much information that viewpoint contributes to recognizing the object. The task of recognition is then defined as selecting a sequence of viewpoints that minimizes the entropy of detecting the object.

1.2.3 Search for an object in 3D environment

Despite the previously described approaches, in practical applications, the task of object search heavily relies on human involvement. Search and rescue operations in

hazardous environments are examples of such applications in which the process of search is either fully controlled by a human operator [9] or autonomy is minimally involved for undertaking trivial tasks such as sensor adjustments or motion control [10, 11]. The exploration applications are no exception. In the well-known NASA robot, Curiosity, only the task of navigation is performed autonomously by the rover [12].

In [13], Fukazawa *et al.* define object search as the process of generating the shortest path that covers an entire environment. This path is followed by the robot to search for the target object. To create an exploration path, the environment is divided into a grid of potential locations. The distance between locations is based on the sensing area of the camera used in the search. A path then is produced to pass through the center of each cell in the shortest path.

Tovar *et al.* [14] employ a dynamic tree structure to model an unknown environment. Constructed by laser sensors, this visibility tree corresponds to a connected planar environment specifying an optimal path.

A multiple target search approach is introduced by Lau *et al.* [15] in which a robot searches for objects within a known environment. They divide the search space into distinct regions and use an adjacency matrix to portray the connections between them. The prior knowledge of the environment changes with the size of the environment or the number of the objects. Taking into considerations the cost associated with each

action, here, the task is to determine a sequence of operations to find the targets while minimizing the expected time of the search.

In the context of assistive robotics, Mehdi and Berns [16] use a probabilistic approach to characterize household environments to search for the elderly. An environment is divided into sub-regions each holding the probability distribution of the target's presence, defined in the form of *a-priori* knowledge. During the search process, the probability values are adjusted with respect to the cost of moving to each location from the current positon of the robot. Simulation results are presented to show the performance of the system using a Harr cascade classifier to detect the human subjects.

Ye [17] tackles the problem of object search in unknown environments. He describes the problem of search as maximizing the probability of detecting an object within a predefined cost constraint. In his work, only the exterior boundaries of the search place are known in advance and no assumption is made regarding the internal setting of the environment. He uses a uniform probability distribution defined on an occupancy grid to characterize the search environment corresponding to the likelihood of the target's presence at each location. The task of search is then to select the viewpoints that maximize the probability of detecting the target while minimizing the cost. After each unsuccessful detection of the target, the probability of regions observed by the robot are appropriately adjusted. Ye demonstrates the performance of this model using a mobile robot equipped with laser range finders and a monocular camera.

Shubina and Tsotsos [18] show an implementation of the above algorithm on a practical robot only using a stereo camera. At each point of the search, a pan-tilt unit is used to set the direction of the camera to a candidate viewpoint capturing an image of the environment. The image is analyzed by the application of a recognition algorithm to identify the object of interest. If the target is not found, the probability of that viewpoint, within the effective field of the recognition algorithm (the 3D spatial region, where the recognition algorithm can detect the target), is lowered to zero and redistributed to the remaining unseen regions. The robot continues the same process until the object is found.

Saidi *et al.* [19] improve the probability reallocation in Ye's search model by including the effect of occlusion within an environment. If there exists an obstacle that blocks the field of view, the probability distribution of the target's locations for the regions behind the obstacle is lowered as the chance of detecting the target beyond that point is smaller.

In the aforementioned search models, the scope of exploration is limited to the ability of a robot to detect the target within its effective field of view (the range within which a recognition algorithm is able to detect an object). This can be wasteful in the sense that any sensory information acquired from beyond the range of detection is discarded as they play no role in identifying the object of interest.

1.2.4 Saliency and visual search

Tsotsos and Shubina [20] argue that the use of attentive mechanisms optimizes the search processes inherent in vision. One factor that directs the attention of an agent to a particular point in a scenery is its visual saliency. Such visual saliency can be generic, corresponding to the areas that stand out with respect to their surroundings due to the possession of distinctive features [21]. Saliency also can be task driven meaning that the parts of a scene relating to specific features of interest are considered as salient [22].

In a social robotic application, Butko *et al.* [23] exploit the use of saliency to identify the motions corresponding to those of human subjects. This helps the social robot to orient its head toward the faces providing a natural feeling of interactions with them. They report that a simple use of saliency doubled the success rate of the camera to capture the images of people to 70% up from 35%.

Orabona *et al.* [24] use saliency as a means to recognize an object using a humanoid robot. They begin by transforming an input image into three separate opponent color channels, each in turn is used to identify the edges within the scene. The resulting edge maps are combined and quantized to form a conspicuity map of the environment. To recognize the object of interest, the saliency results are biased by calculating the Euclidean distance in the color opponent space, between the average color of the target and the salient locations. They only presented few examples of the algorithm performance without any validation in an actual search scenario.

In [25], saliency is used for the purpose of navigation and localization for a mobile robot in outdoor environments. A series of low level features are extracted from an input image consisting of color, intensity and orientation in three separate channels. Then, through the application of a cross-scale center surround difference procedure [26], salient locations are identified for each channel and combined forming a saliency map of the environment. To estimate the position of the robot, SIFT features are extracted from the salient locations and matched for consecutive images. In addition, the saliency results are compared against a trained database of the object's instances to identify the next destination for the robot to move to.

Robert *et al.* [27] propose the use of saliency for the fast detection of trees aiming to help an aerial robot to navigate its way through forested environments. A conspicuity map is built by estimating optical flow during the robot's transition and measuring the motion parallax. This helps the robot to distinguish the trees from their background regions and plan its trajectory.

1.2.5 Sensor planning strategy for object search

Ye and Tsotsos [28] comment on the tractability of object search and prove that it belongs to NP-hard class of problems. They introduce a greedy approach as an approximate solution to the problem. In this method, the search process is divided into two stages of "where to look next" and "where to move next". In the first step, the robot searches its surroundings until some threshold is reached indicating that a new location

should be searched. In the second step, the robot chooses a new destination that has the highest chance of detecting the target, and then, moves there to resume the search. This process continues until the object is found.

In [29], the authors employ a similar heuristic approach. The actions available to the search agent at any time are represented by their utilities given by dividing the probability values of performing each operation by the time of their applications. The algorithm greedily selects one or more actions at a time that yield the highest utilities.

In practice, the resources available to conduct a search are limited. This can be the robot's battery energy used in the search or the allowable time for conducting the search e.g. in search and rescue missions. The scarcity of resources imposes a constraint on object search processes. Solely relying on a greedy algorithm does not suffice for a solution to optimize a search process with respect to a cost constraint. The greedy approaches locally select the next best action with no look ahead and lack a global view of the entire process to determine the consequence of executing each action on the overall efficiency of search.

Aydemir *et al.* [2] use an exhaustive search method to select the sequence of policies that yield the lowest cost. The cost of each action depends on the motion of the robot to perform that task. The value of each action is defined by a non-uniform probability distribution at the start of the search and is updated after each instance of search. The authors propose the selection of 3 to 5 actions at a time and assume the

intermediate probability values are constant which significantly reduces the complexity of selecting the actions.

In this work, no actual recognition algorithm is presented and detection tasks are undertaken using simple Quick Response (QR) codes. Moreover, the authors do not specify the criterion for the termination of search and omit to mention the effect of cost constraints on the selection of policies.

Lau *et al.* [15] use a dynamic programming technique. They define operations that search particular locations within the environment, and the cost of each operation is the time it takes the robot to move to that location traveling along the shortest path available. The algorithm then selects the sequence of actions that yields the highest probability value in the shortest time, i.e. it minimizes the overall cost of the search regardless of any predefined constraints.

The authors simplified the task of search in the following ways: the environment is fully known and assumed obstacle free, and the probability values of each location are considered fixed, which reduces the complexity of the search significantly. In practice, there are both static and dynamic obstacles that need to be considered when selecting an action to perform. Lau *et al.* show that the processing time for generating an optimal sequence is around 20s for an environment with 14 regions and point out that for larger environments the problem is intractable. They also only present simulation results in which the environment is fully known.

1.3 Contributions

This thesis contributes two extensions to the work of Shubina [30] and the original object search formulation of Ye [17]. In these methods, the ability of a robot to find a target is limited to its recognition's effective field of view. If we identify clues regarding the target's location in ranges above the field of view, we can guide the robot to the locations of higher importance and as a consequence improve the process of the search.

Our first contribution is a novel use of saliency to spot the image regions that likely contain the object of interest and use them in the form of indirect search clues without the need for any prior knowledge of the environment or spatial relations between the objects. For this purpose, we combine two methods of saliency: the AIM algorithm [31] that identifies the interest points corresponding to the physical structure within the environment and Histogram Backprojection [32], which pinpoints the regions with the highest similarity to the target in terms of its RGB color distributions.

The saliency results generated by AIM generally correspond to those image regions with a higher chance of including the object such as tables or shelves that stand out within their surroundings. Then, through a top down approach, we distinguish between these structures by increasing the importance of the ones that also include similarities to the target. With respect to these saliency responses, the probability distributions of

the corresponding occupancy grid regions are enhanced, inducing the robot to search those areas earlier then otherwise.

The second contribution of this work is in the area of sensor planning strategies. In the original formulation of object search [17], Ye defines the task of search as maximizing the probability of detecting the target within a predefined cost constraint. However, due to the NP-hardness of the problem and the intractability of its exact solutions, Ye uses a greedy approach. In his approach, actions are selected one at a time and the overall constraint of the search is not considered. Hence, this question remains open: how should a robot select its operations to maximize the chance of detecting an object with respect to a predefined cost constraint?

To address this problem, we propose three sensor planning strategies, namely Greedy Search with Constraint (GSC), Extended Greedy Search (EGS), and Dynamic Look Ahead Search (DLAS). The first two approaches are similar to Ye's algorithm, with some modifications to take into account the overall cost constraint. The GSC algorithm relies on saliency information to select the best action at each point of the search. Once a percentage of the search constraint (e.g. time) is reached, it chooses actions with the highest chance of detecting the target regardless of their costs. EGS, on the other hand, generates a sequence of search operations blindly at the start of the search. To produce a sequence, it greedily selects the next best action within a given

cost constrain. The saliency information are taken into considerations only if there is a need to regenerate an action sequence during the search.

The DLAS algorithm uses a dynamic pruning technique to globally optimize the search. It performs a multi-step look ahead procedure and selects the arrangement of operations that maximizes the chance of detecting the target. The accuracy and processing time of DLAS can be changed by setting the maximum number of steps for the method to look ahead.

In order to evaluate the performance of the proposed methods, experiments are conducted within actual 3D environments of various sizes and configurations. The search agent is implemented on a Pioneer 3, a four-wheeled differential drive mobile robot. The source of sensory input is a Point Grey Research Bumblebee 2 stereo camera, which is used for estimating disparity in the environment, detecting and locating obstacles and to recognize the object of interest. The camera is mounted on a Directed Perception pan-tilt unit responsible for changing the gaze of the camera to desired directions.

The experiments are divided into two sets. First experiments are conducted to measure how much improvements can be achieved using saliency. To do so, two greedy search approaches with and without saliency are conducted in various environments with different configurations. In these experiments the robot and a target

are placed in random positions and the performance of each method is measured in terms of the number of actions performed and the time it takes to conclude the search.

The second set of experiments are aimed to compare the performance of proposed sensor planning techniques. The algorithms were evaluated using three cost functions including, the time of search, battery consumption of the system and the distance travelled by the robot.

1.4 Thesis outline

This thesis comprises six chapters. Chapter 1 discusses our motivation and reviews some previous related work. Chapter 2 revisits some of the methodologies and concepts proposed in the work of Ye on object search. Chapter 3 introduces the saliency mapping technique used in our work and its application to visual search. Chapter 4 describes the development of sensor planning strategies. Chapter 5 details the experimental results of the proposed work. Chapter 6 summarizes the thesis and recommends some future directions.

2 Visual Search in an unknown 3D Environment

As mentioned earlier, the starting point of this thesis is on the work of Yiming Ye [17] and the later extension by Shubina [30] on sensor planning and object search. As a result, in this chapter we briefly review some of the concepts introduced in their works with their mathematical formulations as a guide for the remainder of this thesis. Section 2.1 describes the formulation of object search introduced by Ye. Section 2.2 provides a sensor planning strategy to conduct the search.

2.1 Object Search in an Unknown Environment

2.1.1 Problem Statement

Assume we want to search a 3D environment Ω with known boundaries without any prior knowledge of its internal configuration. The search environment is tessellated into non-overlapping cubic elements, c_i, $i = 1,2, \dots, n$ forming an occupancy grid. The search agent action is defined by $\boldsymbol{f} = \boldsymbol{f}(S(\tau), a)$ in Ω, where $S(\tau)$ is the camera configuration. $S(\tau)$ is determined by the camera position (x_c, y_c, z_c), the direction of its viewing axis $(pan, tilt)$ in degrees of visual angle with respect to the origin, and solid viewing angle (w, h) represent the width and height of the camera's solid viewing angle in radians at time τ, and a is the recognition algorithm used to analyze the image.

The probability of the center of the target being located in cubic element c_i at time τ is $\boldsymbol{p}(c_i, \tau)$. The value within this distribution varies according to our prior knowledge of the target's presence. In the absence of data, when $\tau = 0$, a uniform probability distribution is considered as default. Similarly, the probability that the target is outside of the search environment at time τ is given by $\boldsymbol{p}(c_{out}, \tau)$, and,

$$\boldsymbol{p}(c_{out}, \tau) + \sum_{i=0}^{n} p(c_i, \tau) = 1. \tag{2.1}$$

The function on Ω is a function $\boldsymbol{b}(c_i, \boldsymbol{f})$ that gives the conditional probability of detecting the target, given that the target is centered at c_i, and $\boldsymbol{f}$ is a search operation. $\boldsymbol{b}(c_i, \boldsymbol{f})$ is equal to zero, if any of the following conditions occur: the center of cube c_i is outside of the image; the cube is occluded or too far or too near the camera; or the recognition algorithm, a, fails to detect the target. Excluding these conditions, the value of $\boldsymbol{b}(c_i, \boldsymbol{f})$ is determined based on the ability of the recognition algorithm to detect the target with respect to factors such as the object's orientation and the distance of the camera from c_i.

The probability of detecting the target by applying operation $\boldsymbol{f} = \boldsymbol{f}(S(\tau), a)$, given $S(\tau) = (x_c, y_c, z_c, pan, tilt, w, h)$, can be calculated by

$$\boldsymbol{P}_{\Psi_\mathbf{f}}(\boldsymbol{f}) = \sum_{\mathrm{c_i} \in \Psi_f} \boldsymbol{p}\big(c_i, \tau_{\boldsymbol{f}}\big)\boldsymbol{b}(c_i, \boldsymbol{f})\,, \tag{2.2}$$

where $\tau_{\boldsymbol{f}}$ denotes the time just before $\boldsymbol{f}$ is applied and $\Psi_{\boldsymbol{f}}$ is the influence range of action $\boldsymbol{f}$ corresponding to the cubic elements within the effective field of view of the camera, that are not occluded, i.e. regions within which the recognition algorithm can detect the target.

Let $\boldsymbol{O}_{\Omega}$ be the set of all possible operations on region Ω and $\boldsymbol{F} = \{\boldsymbol{f}_1, \boldsymbol{f}_2, \dots, \boldsymbol{f}_k\}$ be the ordered set of the operations (effort allocation) applied during the search given $\boldsymbol{f}_i \in \boldsymbol{O}_{\Omega}$. The probability of detecting the target by applying an effort allocation $\boldsymbol{F}$ is given by

$$P[\boldsymbol{F}] = P(\boldsymbol{f}_1) + [1 - P(\boldsymbol{f}_1)]P(\boldsymbol{f}_2) + \cdots + \left\{\prod_{j=1}^{k-1}\left[1 - P(\boldsymbol{f}_j)\right]\right\} P(\boldsymbol{f}_k), \tag{2.3}$$

where $P(\boldsymbol{f}_1)$ is the probability that first action detects the target and $[1 - P(\boldsymbol{f}_1)]P(\boldsymbol{f}_2)$ is the probability that first action fails to detect the target but the second action does, and so on.

The application of each operation incurs a cost, given by the total time or energy needed for altering the state of the hardware according to $\boldsymbol{f}$, capturing an image of the search environment, analyzing it by the recognition algorithm and updating the probability distribution values. Ye [17] defines the total cost of effort allocation $\boldsymbol{F}$ as follows:

$$T[\boldsymbol{F}] = \sum_{f \in F} \boldsymbol{t}(\boldsymbol{f}) \tag{2.4}$$

where $\boldsymbol{t}(\boldsymbol{f})$ is the cost of operation $\boldsymbol{f}$.

Suppose K is the total time (or energy) available to perform the search, then we can define the task of search as finding an effort allocation $\boldsymbol{F} \subset \boldsymbol{O}_{\Omega}$ that can satisfy $T(\boldsymbol{F}) \leq K$while maximizing $P[\boldsymbol{F}]$. In this manner, the actions are selected that yield the highest utility value (described in details in Section 2.2.1).

2.2 Conducting the Search

Based on the above formulation, Ye [17] proves that the sensor planning task for object search is NP-hard in terms of processing time. Due to the intractable nature of the problem, Ye proposes a heuristic greedy approach and argues that it would suffice as a good approximation to the solution.

Because of the fact that the cost of moving the robot during the search is usually significantly greater than the cost of changing the camera direction at any stationary position, Ye divides the task of search into two stages of "where to look next" and "where to move next".

2.2.1 Where to look next

At this stage, the robot is in a stationary position and the goal is to select an operation $\boldsymbol{f} = \boldsymbol{f}(p, t, w, h, a)$ that yields the highest utility given by

$$\boldsymbol{E}_{\Psi_f}(\boldsymbol{f}) = \frac{\sum_{c_i \in \Psi_f} \boldsymbol{p}(c_i, \tau_{\boldsymbol{f}}) \boldsymbol{b}(c_i, \boldsymbol{f})}{t(\boldsymbol{f})}, \tag{2.5}$$

where Ψ_f is the influence range of operation $\boldsymbol{f}$ and $t(\boldsymbol{f})$ is the cost of applying action $\boldsymbol{f}$. Given the similarity of actions' costs, Ye further simplifies the process at this stage by only considering the numerator part of (2.5).

If the target is not found after performing an operation, the probability distributions of the cubic elements are updated as follows:

$$\boldsymbol{p}(c_i, \tau_{\boldsymbol{f}+}) = \frac{\boldsymbol{p}(c_i, \tau_{\boldsymbol{f}}) \left(1 - \boldsymbol{b}(c_i, \tau_{\boldsymbol{f}})\right)}{\boldsymbol{p}(c_{out}, \tau_{\boldsymbol{f}}) + \sum_{j=1}^{n} \boldsymbol{p}(c_j, \tau_{\boldsymbol{f}}) \left(1 - \boldsymbol{b}(c_j, \tau_{\boldsymbol{f}})\right)}, \qquad i = 1, \ldots, n, out \tag{2.6}$$

where $\tau_{\boldsymbol{f}+}$ is the time after $\boldsymbol{f}$ is applied and $\boldsymbol{p}(c_{out}, \tau_{\boldsymbol{f}+})$ is the probability that the target is outside of the image at the time $\tau_{\boldsymbol{f}+}$. Intuitively, if operation $\boldsymbol{f}$ fails to detect the target, the probability of the influence range decreases while those of the other regions increase. The process of action selection and application continues until the "covering probability" of all remaining operations, $Prob_{\Psi_f} = \sum_{c_i \in \Psi_f} \boldsymbol{p}(c_i)$ where falls below

some threshold, Θ_{move} at the stage in which the robot considers to move to a new location. In this formulation $p(c_i)$ refers to the target probability represented at cubic element i.

2.2.2 Where to move next

The new destination of the robot is decided according to two criteria: the new location should be reachable and has a high probability of providing an appropriate viewpoint for detecting the target. Given that the height of the camera is fixed and the robot only moves horizontally, the new position is only within the vertices of the 2D grid. Alternatively if a robot is capable of changing its height of view, it would not be limited considering the search space is defined 3D.

The probability of each location j is calculated by $Prob_{\Psi_j} = \sum_{c_i \in \Psi_j} \boldsymbol{p}(c_i)$, where Ψ_j is the region within the union of all effective fields of view at position j. After selecting a new location, the robot moves there. If the robot detect an obstacle in the candidate location, it selects a new one.

Once at the current position, a similar procedure as before is repeated in which the robot selects and searches directions with the highest probability. The look next and move next processes continue until either the target is detected or it is not found within the environment.

3 Saliency in Visual Object Search

The range of a recognition algorithm is limited by factors such as the types of features used or the characteristics of the object of interest. This range is typically less than the range of stereo cameras within which they can measure disparity (see Figure 3.1), and varies according to the camera's baseline, resolution or sensor type. For instance, in the work of Shubina and Tsotsos [18], the recognition algorithm is capable of detecting the target (Figure 5.4) within the maximum range of 2.6 meters. The stereo camera used in their experiments, however, has at least twice as long a range of the recognition algorithm to detect disparity.

Discarding the information beyond the range of the recognition algorithm means a potential source of guidance is ignored. Such information can further be processed to identify clues regarding the target presence within the environment. A common approach in visual search applications for identifying regions of interest is the use of saliency algorithms. A saliency map can provide one with clues regarding a target's presence by highlighting the interest points, which in turn can be used to direct the attention of the search agent to the regions with a higher importance.

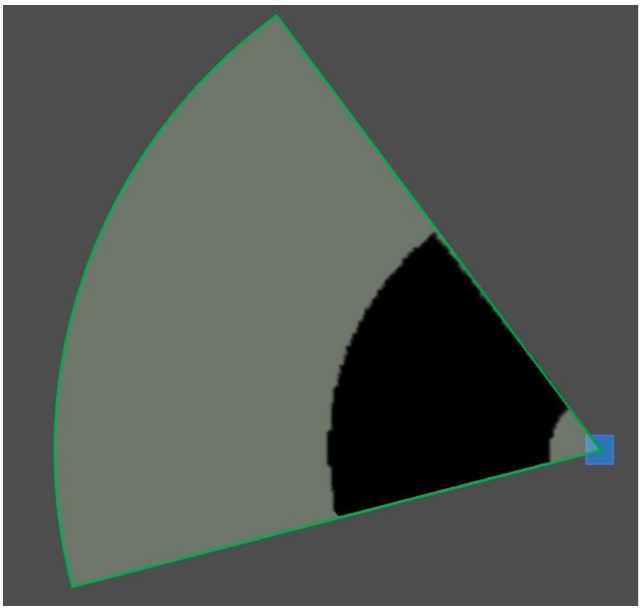


Figure 3.1: The application of an operation to the search environment. The gray background shows the uniform probability of the target presence, the black annulus denotes the range of the recognition algorithm and the green sector the disparity range of the stereo camera.

This chapter is organized as follows: Section 3.1 reviews vision literature to explore some of the common approaches to building a saliency map. Section 3.2 describes the AIM algorithm to construct a general saliency map. Section 3.3 explains Histogram Backprojection and its application to generate top down saliency. Section 3.4 shows the process of building the final saliency map. Section 3.5 concludes this chapter by demonstrating the application of saliency to visual search.

3.1 Saliency in Robotic Visual Search

In the computer vision literature, it is a common practice to identify points of interest using saliency mapping techniques for the purposes of attention, recognition, segmentation or navigation.

3.1.1 Bottom-up saliency

In general, there are two classes of algorithms to construct saliency maps [33]. One class of algorithms measure saliency without any prior knowledge of an object or a task (bottom-up approach). These models are designed to identify the portions of an image that stand out in comparison to the rest [21, 34, 35, 36, 37, 38, 39].

Bottom-up saliency in particular is useful to identify structures that stand out within an image. Such saliency may correspond to an object of interest (e.g. in the context of visual search) or other physical structures that are in some form of spatial relationship to that object e.g. tables or shelves. Hence, for the purpose of visual search, a bottom-up saliency can both directly and indirectly guide the attention of an agent to a target of interest.

There are two methods of building bottom-up saliency: object-based and space-based.

3.1.1.1 Object-based saliency

Object-based approaches in essence are similar to segmentation algorithms. However, instead of partitioning an image into regions of coherent properties, they only segment the salient object(s) from the background.

Goeferman *et al.* [40] introduce a model that they call context-aware saliency detection in which the objective is not only to detect the most prominent object but also to contain enough of the background that convey the context. To achieve this, they divide an image into patches of equal size. Saliency is generated by comparing the Euclidean distance between the patches in terms of their RGB colors. The more unique a patch is, i.e. more dissimilar to other patches, the more salient it becomes. This saliency is further modified by taking into consideration the positional distance between the similar patches. This is based on the idea that the background patches tend to have similarity with both patches in near or far distances whereas the salient patches tend to be grouped up in a close proximity of each other.

In addition, a comparison of patches is conducted in multiple scales (Gaussian pyramid levels). The authors believe that the background patches yield more similarities in the lower scales of pyramid, hence, if that is the case, the saliency value of those patches should be reduced accordingly. At the end, the generated saliency responses are adjusted by the application of a face detection algorithm. If a face is detected, the saliency of the corresponding pixels are increased.

Chang *et al.* [41] calculate saliency by minimizing an energy function comprising two forms of energy: the energy affected only by saliency and the energy affected by objectness. The saliency energy is obtained using the method in [40] with the difference of using superpixel segmentation instead of static patches.

As for the objectness, the energy is defined by how likely a portion of an image contains an object. Intuitively, an object is a standalone thing with a well-defined boundary and different from its surroundings.

Jiang *et al.* [42] identify saliency in two steps: saliency detection and shape extraction. In the first step, they over-segment an image by a superpixel operation at multiple scales. Then, the saliency of each region is computed by determining the difference between its color and surroundings. At the end, the results are averaged over all scales.

Next, the saliency map is processed to extract shapes by finding a closed contour covering the salient object. For this purpose, an edge detection is performed on the image followed by a line fitting operation. If the objective is to find multiple objects, this process is repeated more than once.

Object-based saliency methods have a number of shortcomings that makes them unsuitable for indoor search applications. First, these models heavily rely on segmentation of images, meaning that they are most effective for scenes where a limited number of objects are placed over a uniform background such as natural

images. Second, object-based saliency models are designed for identification of one or few objects, the number of which should be defined prior to the detection of saliency. In the context of visual search, the visually salient region does not necessarily correspond to the target of interest. In addition, it is hard to specify the number of salient objects for the algorithm to detect if there are many objects.

Moreover, the object-based models are generally optimized for particular objects or environments, e.g. [40] is optimized for human detection and [41] works best for natural images. Hence, they lack the generality to be used in different visual search contexts.

3.1.1.2 Space-based saliency

Space-based (fixation) models of saliency are more generic in the sense that they are designed to predict human eye fixations typically measured by subjective rankings of interesting and salient locations or eye movement [43].

Itti *et al.* [26] calculate a bottom-up saliency using low–level image characteristics including color, intensity and orientation (e.g. orientation is calculated by oriented difference of Gaussian (DOG)). Instead of the direct calculation of features, features are computed in a center-surround structure to derive the contrast of a feature to its surroundings. To do so, the differences between a fine and a coarse scale (Gaussian pyramid) for a given feature is computed. The same process repeats in eight scales for all features and at the end the resulting maps are cross-scale summed and normalized.

Hou and Zhang [44] use a sparse coding approach to decompose an image into a series of independent features. Then they measure the activity ratio of each feature and consider a feature to be salient if succeeding activations of that feature increases the entropy of the entire system.

In a different representation, Hou and Zhang [45] exploit a property of natural images to estimate saliency. This property indicates that natural images are scale invariant meaning that the amplitude $A(f)$ of their average Fourier spectrum obeys a $1/f$ distribution. Based on this characteristic, the authors estimated the saliency by identifying statistical singularities in the Fourier spectrum, i.e. the information that jumps out of the smooth curve corresponds to salient locations.

In [46] a graph-based approach is introduced. The image features are generated by the application of difference of offset Gaussian (DOOG) filters oriented toward six equally spaced directions. Next, the dissimilarity of each feature with its neighboring features is calculated by dividing their corresponding intensity values. This forms a graph in which each feature is considered as a node and connected to other nodes through edges, the value of which is determined by the normalized (0-1) dissimilarity between the corresponding nodes. To compute saliency, the authors define a Markov chain on the resulting graph, indicating that the equilibrium distribution at each node would naturally accumulate mass at nodes that have high dissimilarity with their surroundings.

In a different formulation of saliency, Bruce and Tsotsos [31] decompose an image into independent features using the Independent Component Analysis (ICA) algorithm. Then, they compute the joint distribution of features along the image at each pixel. The information measure of each pixel is calculated which is an indicator of saliency at that location, i.e. the less common features in the image (more salient) yield higher information responses.

3.1.1.3 Performance measure of space-based saliency models

Measuring and comparing the performance of saliency models is a challenging task. These models use different techniques of decomposing images into features and measuring their distributions. There have been a number of attempts to quantitatively analyze the performance of bottom-up saliency approaches using various scoring techniques and datasets [43, 47]. The common approach in all these schemes of performance measurement is comparing the saliency results of each model against ground truth images. The ground truth images are binary images produced by a human observer who identifies those locations to which humans are more likely to fixate.

Despite the fact that in the comparison studies there are models that consistently perform better than the rest, there are still minor differences in ranking of these models on being applied to different datasets. For instance, in [47] 5 datasets are used for comparison purposes. Along the 11 space-based models of saliency analyzed in this study, three models had the best performance on average in all datasets including

graph-based visual saliency (GBVS) [46], dynamic visual attention (DVA) [44] and attention based on information maximization (AIM) [31]. However, the performance of each model varies depending on the type of datasets. For example, GBVS has the best performance in the dataset with multiple number of objects while AIM achieves the best performance in the dataset with single dominant objects.

Given such variation in performance, we conducted our own evaluation of bottom-up saliency models using image samples collected from test environments used for our search experiments. We used the area under curve (ACU) scoring scheme similar to [48]. In this technique, points from human fixations (ground truth) are considered positive, and a number of points are sampled from images, which form the negative set. Then the saliency map is treated as a binary classifier which separates the positive from negative samples. These maps are thresholded with different percentile values ranges from 0% (no saliency responses) to 100% (all saliency responses considered). Next, a true positive rate vs. false positive rate is calculated and a receiver operating characteristic (ROC) [49] curve is plotted for each saliency map and is averaged for all the dataset images. By calculating the area underneath of each curve, we can predict the performance of a given saliency model. A score of 1 corresponds to perfect prediction whereas a score of 0.5 indicates chance level.

For evaluation purposes, we used 316 sample images with saliency maps thresholded by steps 5% apart, i.e. a total of 21 maps for each model. The saliency

models were selected based on their average performance measures in [47] namely, Itti [26], AIM [31], GBVS [46], DVA [44], spectral residual approach (SRA) [45] using fast fourier transform (FFT) and discrete cosine transform (DCT) techniques. Figures 3.2 and 3.3 show the ROC curves and their integral values respectively. Based on the ACU evaluation, AIM has the best performance measure with 0.7701.

3.1.2 Top-down saliency

As the name implies, the second class of saliency algorithms are those used to detect clues regarding specific objects (top-down approach). In this method of saliency, the objective is to identify portions of an image that corresponds to a specific object. Thus, using such methods requires some form of training prior to the process to learn features corresponding to those of the task at hand [22, 23, 50, 51].

Zhu *et al.* [50] introduce the notion of contextual pooling. In this model, top-down saliency is treated as a classification problem. For training purposes, random patches as well as their neighbouring patches are extracted from training data to take into account both the object specific features and the context of the samples. Then, SIFT features are obtained from each patch cluster and mapped to K-dimensional codes forming a code book through the use of a support vector machine (SVM) model. To measure the saliency in a test image, through the use of a Bayesian approach, the probability of each pixel belonging to the object is computed. Based on the distribution values, pixels are labeled as the background or the object.

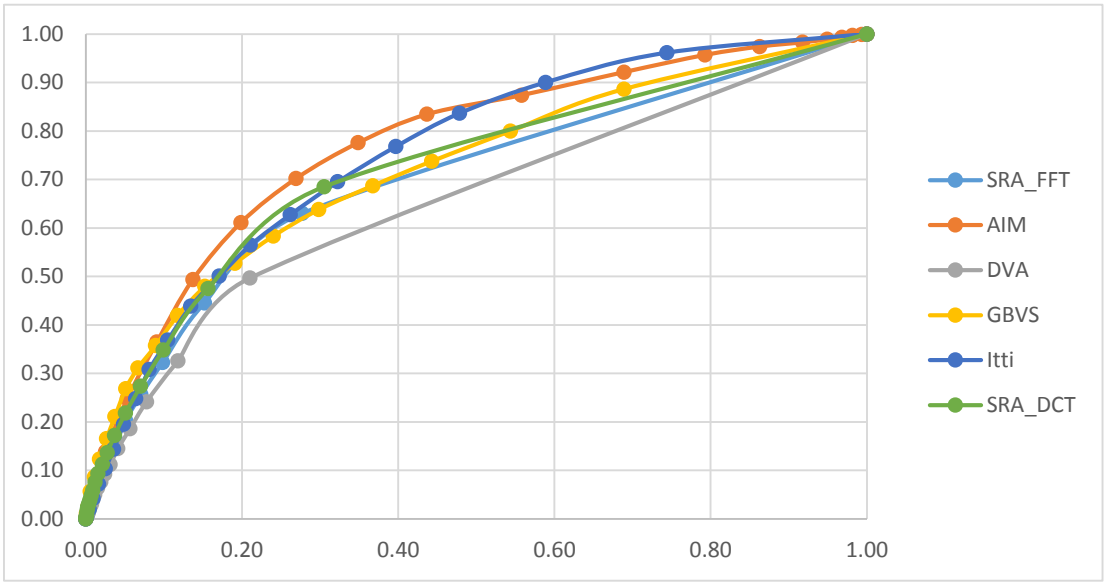


Figure 3.2: The ROC curves of the saliency models in ACU.

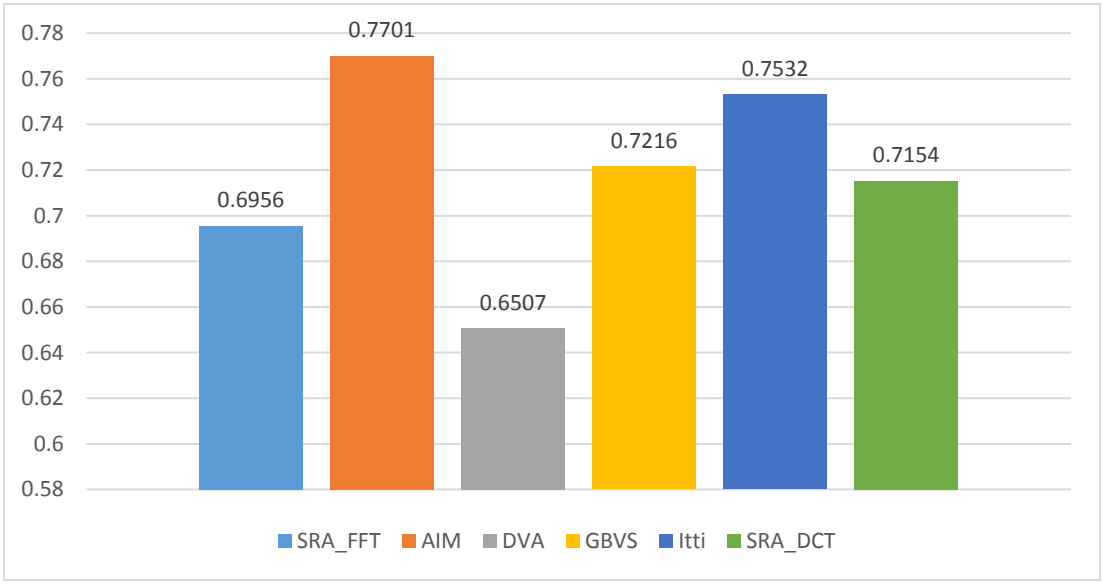


Figure 3.3: The integral values of ROC curves for each saliency model.

In [51], the authors use a similar classification approach to calculate top-down saliency. However, top-down saliency is combined with a bottom-up saliency map to generate more robust results. The bottom-up saliency is generated using low level filters such as Difference of Gaussian (DOG) and coded into a more compact format using the discrete cosine transform (DCT) technique.

To include contextual information, Yang and Yang [52] use a graphical model known as conditional random field (CRF). First, they train a dictionary of an object's features using SIFT descriptors. Then through the use of the CRF, they determine how a combination of these features and their neighboring regions contribute to the presence of an object. The result of the CRF is a probability distribution value indicating the saliency of a given feature in an image.

As it was seen in the above formulations of top-down saliency, these techniques are designed for classification and recognition purposes. Hence, they rely heavily on high level features (e.g. face) to separate an object from its background. In our visual search application, we intend to find object specific clues in distances above the effective range of our recognition algorithm, i.e. there is not enough features for recognition to succeed. Having this fact in mind, to generate top-down saliency results, we propose the use of histogram backprojection technique in which regions of interest are identified only based on low level features such as color.

3.2 Attention based on Information Maximization (AIM)

In order to identify the general structures of interest, we employ the work of Bruce and Tsotsos [31], commonly known as AIM. The reason behind this is AIM's superior performance in comparison to other state-of-the-art saliency methods in particular for natural images (refer to [43, 47] for more details). The AIM algorithm begins by decomposing an image into a series of distributions corresponding to independent features. These features are generated by the application of a basis function previously trained over a large number of natural samples using Independent Component Analysis (ICA) [53].

3.2.1 Independent Component Analysis (ICA)

In this section, some principles of ICA are briefly reviewed to better understand the process of feature generation in AIM. For more information regarding the operation and formulation of ICA, see [53, 54].

ICA is a common technique in signal processing applications where one intends to identify the individual sources of a mixed signal. A known application of ICA is in the so-called "cocktail-party problem". In this problem, n people are speaking simultaneously at a cocktail party and their voices are recorded by some microphones. The objective is to separate the independent voices chattering at the party.

Assume we have a random vector of n observations $\boldsymbol{x}(t) = [x_1(t), x_2(t), \dots, x_n(t)]$ at times $t \in (t_0, t_1, \dots, t_{\max})$ each is a mixture of n independent sources as follows:

$$x_j(t) = a_{j1}s_1(t) + a_{j2}s_2(t) + \cdots + a_{jn}s_n(t), \tag{3.1}$$

where $x_j(t)$ denotes the mixture, $s_1(t), \dots, s_n(t)$ and $a_{j1}, \dots, a_{jn}$ are the source elements and mixing coefficients respectively [55]. Putting the above notations into vector-matrix representation we get,

$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s} \tag{3.2}$$

where $\boldsymbol{A}$ is a $n\,x\,m$ matrix of mixing coefficients a_{ji} for $m \leq n$, $\boldsymbol{x}$ and $\boldsymbol{s}$ are the random vectors of the mixtures and sources respectively.

In the context of feature extraction, same principles are applied. Mixtures at each time interval t is represented by pixels of a sample image patch. For instance, a patch of size $10\,x\,10$ forms a vector of 100 mixtures.

The fundamental assumption of ICA is that the sources are non-Gaussian and statistically *independent*, i.e. information about the distribution of one source does not provide any information about other sources. Based on this assumption, the values of $\boldsymbol{A}$ and $\boldsymbol{s}$ are estimated. Once the mixing coefficients are calculated, one can obtain the independent components by multiplying the inverse of $\boldsymbol{A}$ to signal $\boldsymbol{X}$,

$$\hat{\boldsymbol{s}} = \boldsymbol{W}\boldsymbol{x} \tag{3.3}$$

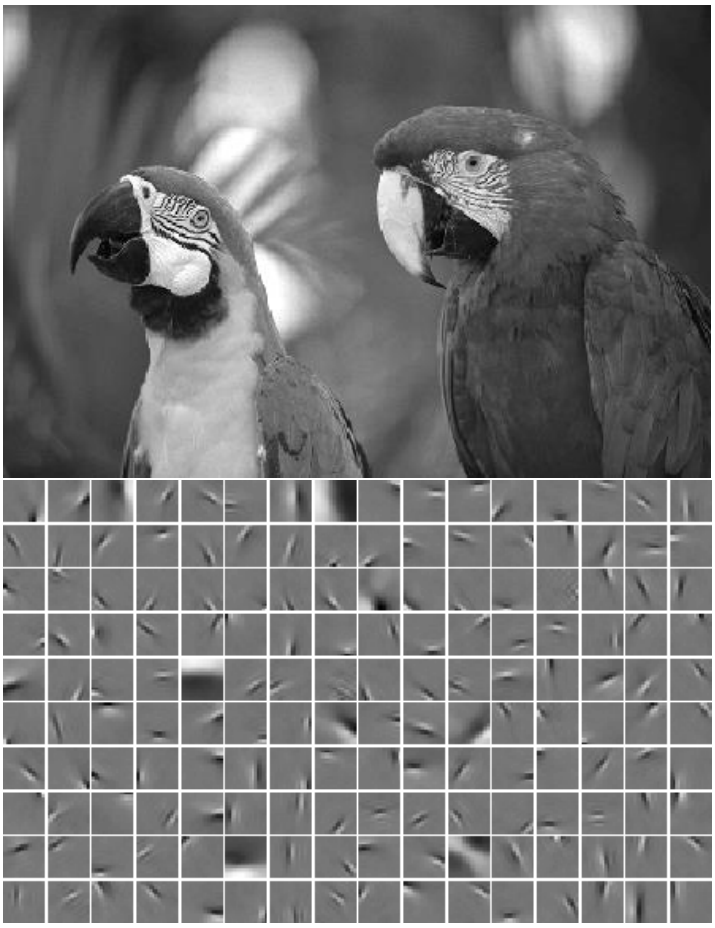


Figure 3.4: An example of decomposing a grayscale image (top) into independent features (bottom) using ICA algorithm [83].

where $\boldsymbol{W}$ can be thought as the inverse of matrix $\boldsymbol{A}$ and $\hat{\boldsymbol{s}}$ is an estimation of sources values.

There are numerous techniques proposed in the signal processing literature to estimate the values of $\boldsymbol{W}$ and **s** [56]. The one used for training the AIM basis matrices is commonly known as infomax [57], a method inspired by the Shannon's information and entropy measures [58].

In Shannon's theory, entropy is a measure of uncertainty which means the more information we have about a system, the lower the value of its entropy and consequently the lower the uncertainty. In this context, uncertainty corresponds to independence meaning that maximum entropy implies independent signals. Based on this assumption, infomax attempts to extract independent sources by estimating an un-mixing matrix that minimizes the mutual information of the sources, i.e. maximizes their entropy (see [57] for more details).

3.2.2 Distribution and Information Measures

The next stage in the AIM algorithm is the calculation of features' distributions generated by the application of the basis function (un-mixing matrix). Using a Gaussian kernel density function, we define the likelihood of the features by

$$p\left(w_{i,j,k} = v_{i,j,k}\right) = \frac{1}{\sigma\sqrt{2\pi}} \sum_{\forall r,t \in \Psi} \omega(r,t) e^{-\left(v_{i,j,k} - v_{i,s,t}\right)^2 / 2\sigma^2}, \qquad (3.4)$$

with $\sum_{\forall r,t \in \Psi} \omega(r,t) = 1$, where $\omega(r,t)$ denotes the degree to which the coefficient ω at coordinates r,t contributes to the probability estimates, $w_{i,j,k}$ is the set of independent coefficients based on neighborhood centered at j,k, $v_{i,j,k}$ is the local

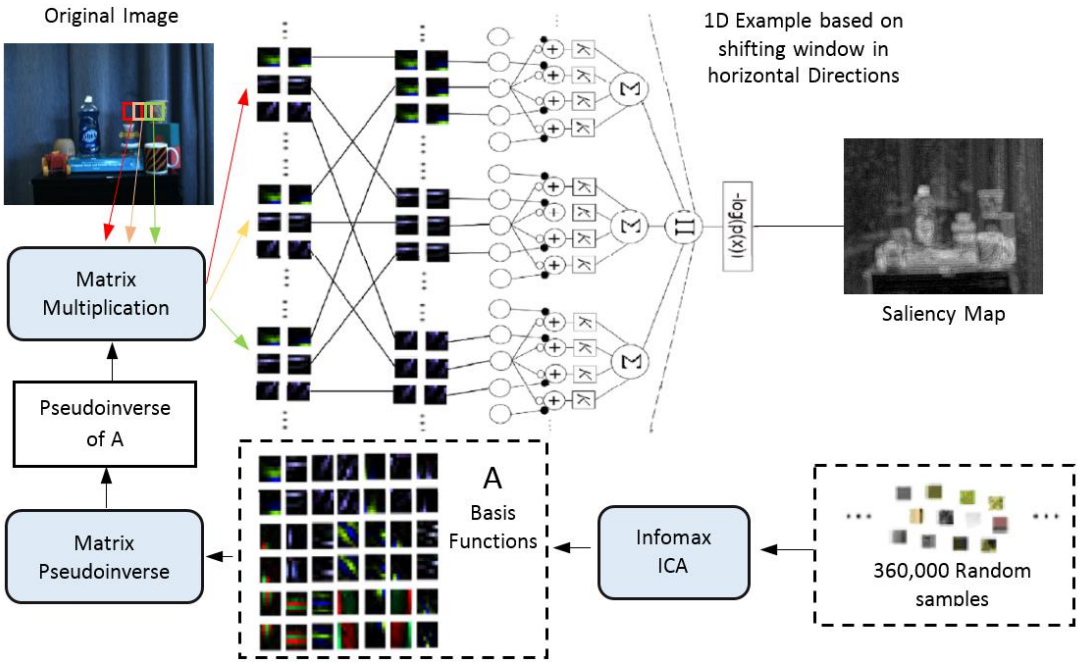


Figure 3.5: The framework of achieving information measures by application of AIM to a sample image using neural circuit to measure the distribution of features *[31]*.

statistic value and Ψ is the context which the probability estimate of the coefficients of ω is based on.

Since features generated by ICA are independent, the joint density function of the features is given by

$$p(w_1 = v_1, w_2 = v_2, \dots, w_n = v_n) = \prod_{i=1}^{n} p(w_i = v_i). \tag{3.5}$$

Using Shannon's self-information measure theory [58], the information of the resulting joint distribution is calculated by

$$I(W) = -\log(p(W)) \tag{3.6}$$

where $I(W)$ is the information measure of distribution $p(W)$. This information then serves as a means to detect saliency within an image. In this case, the regions with the highest information responses, i.e. the least common within the image, are identified as salient.

3.2.3 Parameter selection and performance

The saliency responses of the AIM information map highly fluctuate with respect to the changes in the kernel size used to generate local distributions, the number of features and environmental factors such as lighting condition.

3.2.3.1 Kernel size

Given that a basis matrix is multiplied by each local patch of an image to extract features, it is obvious that increasing the size of the kernel increases the dimension of the matrices multiplied together, and as a result, the processing time rises (Figure 3.6).

Moreover, altering the kernel size also has a direct impact on the saliency responses of AIM. Although the output of the conspicuity map highly depends on the content of the image, as a general rule, the bigger the kernel size be, the higher the chance that larger salient structures are detected within the image. Figure 3.7 demonstrates the effect of the kernel size on the saliency results using AIM. The algorithm is fully parallelized on a 12 core 2.1GHz Intel processor.

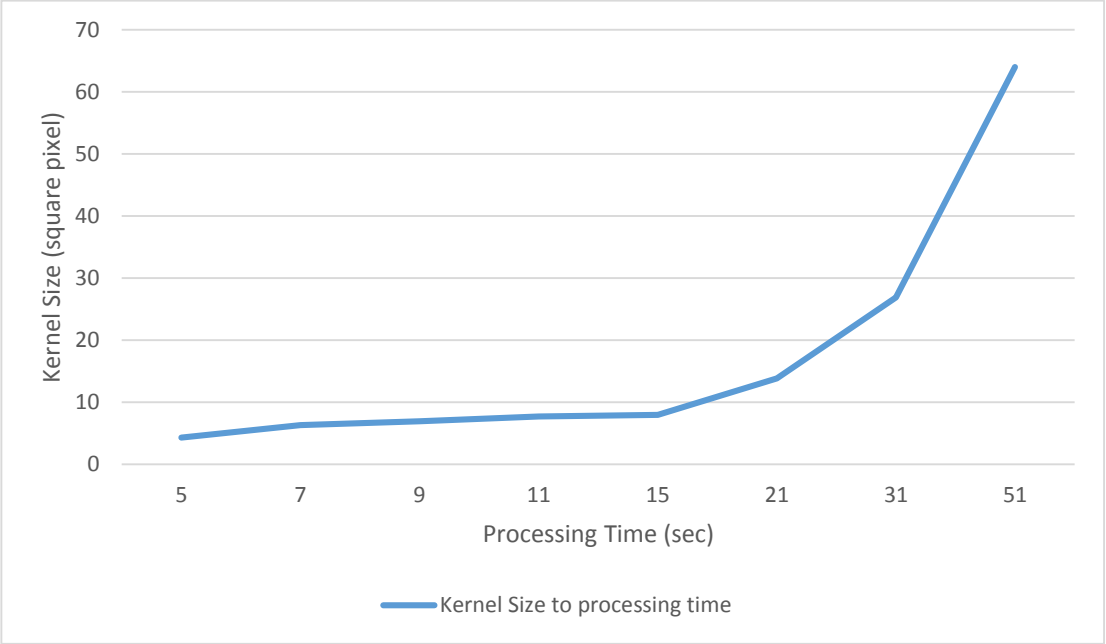


Figure 3.6: The relationship between the kernel size of a basis matrix and the processing time of building the AIM saliency map.

It is apparent that in the saliency responses generated by the small kernels of size 5^2 and 7^2 square pixels, the small components such as floor's texture have the highest intensity As the size of the kernel increases, the saliency drifts away from the smaller objects toward the bigger ones. For instance, in the saliency map with the kernel size of 31^2 square pixels, the stairs at the back of the room have the highest response while the smaller objects such as the colorful toys are much less salient.

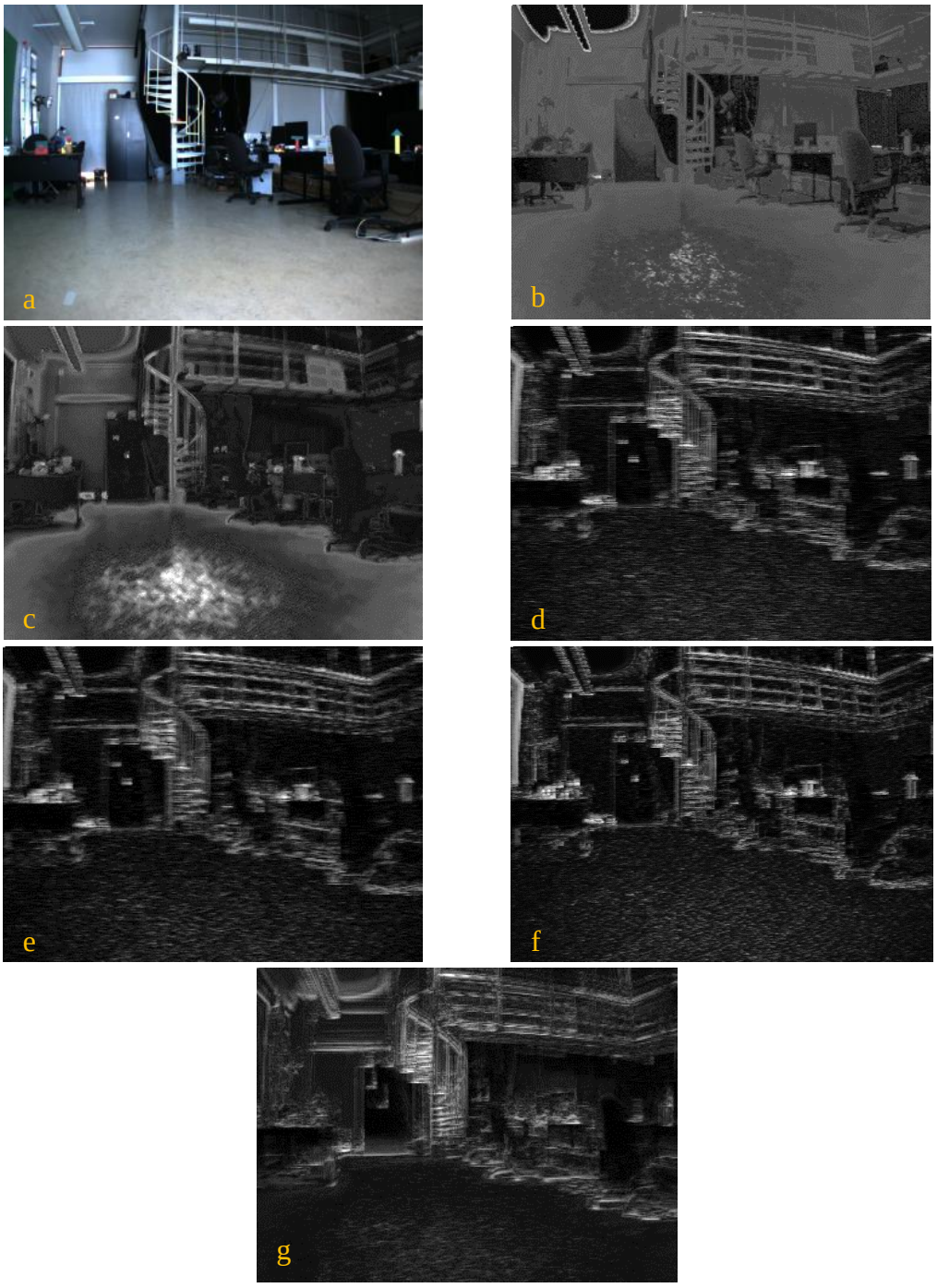


Figure 3.7: The application of AIM to a sample image using the basis matrices of various sizes. a) is the original image, and b-g) are information maps generated by AIM with the kernel sizes of 5, 7, 11,15, 21, and 31 square pixels respectively.

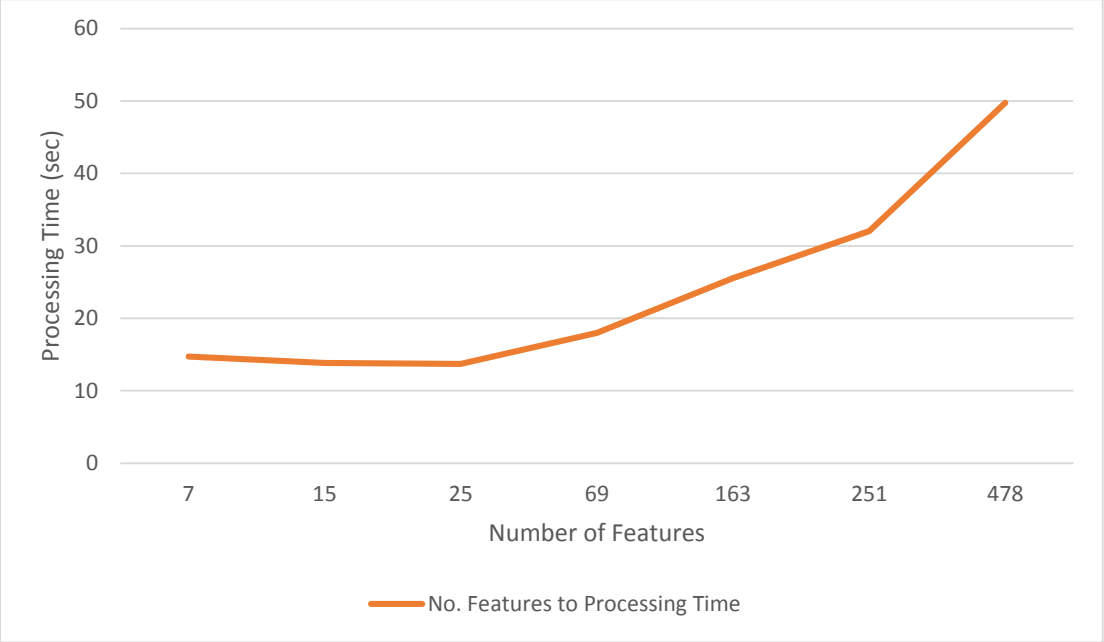


Figure 3.8: The relationship between the processing time of generating the AIM saliency map and the number of features used.

3.2.3.2 The number of features

The AIM saliency model also behaves differently by changing the number of features. Clearly, the processing time is directly correlated with the dimensionality of basis matrices because their number of rows are equal to the number of features. Figure 3.8 illustrates the relationship between the processing time and the number of features used to create an AIM saliency map. The same processor as before is used with the patch size of 21 square pixels. The number of features also influences the saliency responses of AIM. In fact, using ICA to generate the features of interest, each individual pixel is treated as an individual source of information. This means that using more features reduces the sensitivity of the model to correspond to meaningful shapes but rather to

individual pixel intensities or minor structures. Using a small number of features also is inefficient because there is a higher chance of accidental similarity among features for different objects, and consequently, generating similar saliency responses in spite of the variation in the scene.

Figure 3.9 shows the changes in the behavior of AIM using a varying number of features. In the case of using only 7 features, the resulting map is fairly generic as it shows uniform responses with low intensities on the majority of the image. By increasing the number of features, the saliency responses become more specific for individual objects until reaching a certain threshold at which they are drift away to textureless patterns within the image such as walls.

In addition to the type of the basis matrix used in AIM, the environmental factors such as lighting conditions or the objects' distances from the camera greatly alter the saliency results. Without any attempt to generalize the effect of each factor, Figure 3.9 demonstrates some of their impacts on the conspicuity outcomes.

3.2.3.3 Thresholding the saliency map

For the purpose of visual search, we are only interested in the highest saliency responses not the entire map. One way to focus on the highest responses is through static thresholding of the results in which the saliency responses below a fixed value (often set empirically) are discarded.

Figure 3.9: A sequence of the AIM saliency maps using a various number of features. a) is the original image, and b-g) are the saliency maps using 7, 25, 69, 163, 251, and 478 features respectively.

This method is inadequate because given the dependency of the saliency responses to elements such as the environment's configuration, the distance of objects to the camera and viewing angle, a fixed threshold value would not correctly reflect saliency.

A dynamic approach known as percentile thresholding is used to remove the low salient points. The pth percentile is a value below which p percentage of the data falls [59]. These authors compute an index of observation using n observations, $x_1, x_2, \dots, x_n$. First, the observations are sorted in an ascending format, thus x_1 has the lowest value and x_n the highest. Then they calculate the index of observation x_i for the percentile value p by

$$i = \frac{n.p}{100} + 0.5 \tag{3.7}$$

where i presents the index of observation x_i. If i is an integer, x_i is the pth percentile value, otherwise they interpolate as follows:

$$x_{int} = (1 - f)x_k + f x_{k+1} \tag{3.8}$$

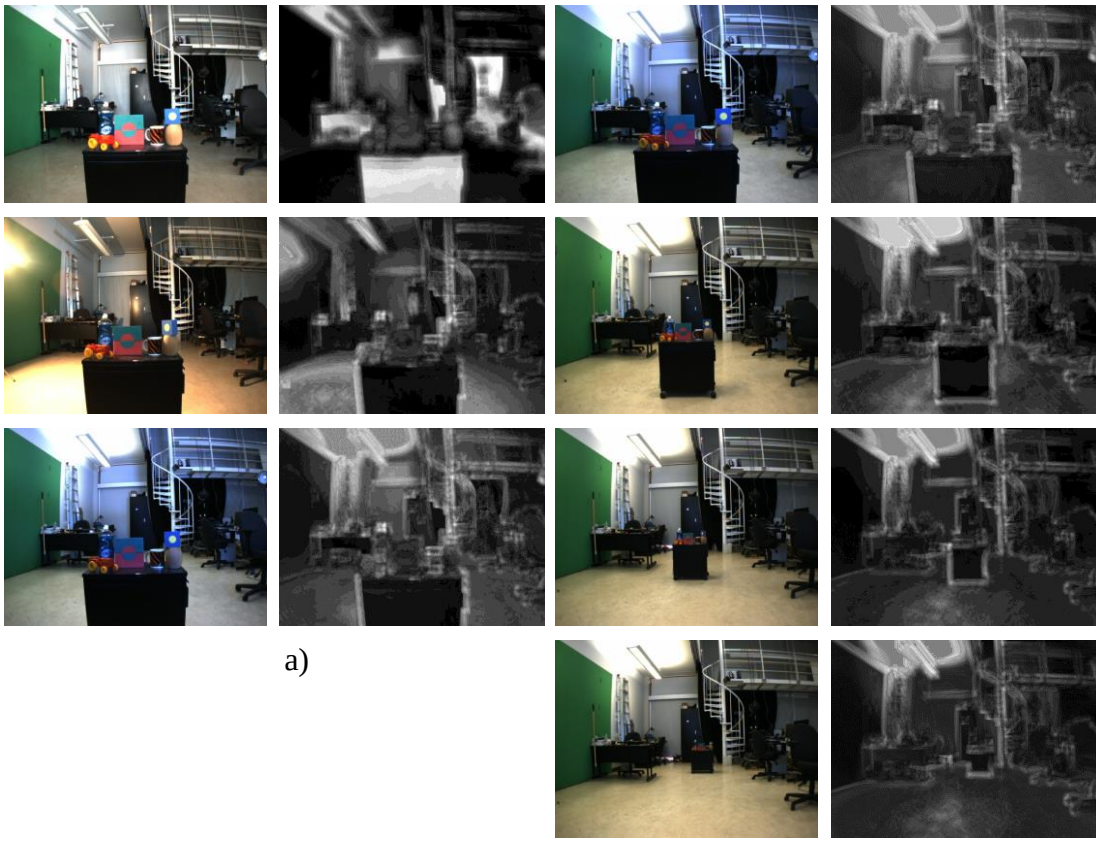
where x_{int} is the interpolated value, and k and f are the integer and fractional parts of x_i.

3.2.3.4 The limitations of AIM

The application of ICA to decompose the image of interest into independent features imposes a number of limitations on the performance of AIM. The basis functions trained by ICA do not account for the color distributions of objects, i.e. features generated for two identical objects with different colors might be the same. It is also challenging for ICA to learn the object specific features within natural environments due to the variations in scale, orientation and lighting conditions.

Moreover, ICA considers each pixel value of an image as a source of information. This means, it is not feasible to train the system over all the individual features of an object. For instance, in the case of a RGB patch of size 21^2 pixels, there will be a total of $21x21x3 = 1323$ features. Applying such a basis matrix to a typical image of size 640x480, we will have a feature space of $1323x620x460$ pixels. Using a smaller subset of features, however, creates similar basis functions for different objects, which makes it challenging to train a basis matrix for a specific object.

Despite such limitations, AIM perfectly satisfies our initial objective to identify general physical structures that likely correspond to regions such as shelves, tables or any other surfaces with a high chance of containing the object of interest.



a)

b)

Figure 3.10: The effects of the environmental factors on the saliency responses of AIM. Each AIM generated saliency uses a basis matrix with the kernel size of 21 square pixels with 25 features. a) Lighting changes within the search environment. On the left from the top to bottom, natural lighting, direct lighting and ceiling lighting and on the right, the saliency responses of each image. b) A drawer with random objects on the top with different distances from the camera. On the left, from the top to bottom, the original image with the drawer's distance of 1m, 2m, 3m and 5m. On the right side, the resulting saliency responses.

3.3 Histogram Backprojection (HB)

The saliency map presented in the previous section does not include clues relating to the target. There are variety of ways to achieve this based on the characteristics of an object such as shape, color or orientation. In this work, only the target's color distribution is considered.

To identify similarities of an object's colors within an environment, an algorithm commonly known as Histogram Backprojection (HB) [32] is employed. First, the HB method requires an object template in order to establish its color distributions.

3.3.1 Template extraction

In order to reduce false saliency responses, it is important to use an object template that has minimal background information. One can attain this by cropping the object from an image manually. This method is neither efficient in terms of timing nor suitable for online applications in which we intend to show an instance of the object that is not previously known to a system.

A clustering technique based on Gaussian Mixture Model (GMM) [60] is used to segment the target of interest from its background. In this method, the object and background colors are represented by a multivariate density function and the goal is to estimate the parameters of each distribution in the form of a GMM. In this manner,

care should be taken to use templates with a uniform background color, preferably distinguishable from those of the object.

A GMM probability distribution takes the form

$$p(x|\Theta) = \sum_{i=1}^{m} \alpha_i p_i(x|\theta_i), \tag{3.9}$$

where m is the number of mixtures and $\Theta = (\alpha_1, \dots, \alpha_m, \theta_1, \dots, \theta_m)$ are the parameters from which $\alpha_i \geq 0$ denotes the mixing coefficient (weight) of each mixture such that $\sum_{i=1}^{m} \alpha_i = 1$, and $\theta_i = (\mu_i, C_i)$ where μ_i and C_i refer to the mean and covariance of normal distribution p_i respectively. The distribution of each mixture is given in the form of a d dimensional Gaussian as follows:

$$p_i(x|\mu_i, \sigma_i) = \frac{1}{(2\pi)^{\frac{d}{2}} |C_i|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu_i)^T \mathrm{C}_i^{-1}(x-\mu_i)} . \tag{3.10}$$

Let x_i be an image RGB patch. The objective is to find the maximum likelihood estimate (MLE) of all the mixture parameters of Θ,

$$\log\big(L(\Theta|x)\big) = \log \prod_{i=1}^{N} p(x_i|\Theta) = \sum_{i=1}^{N} \log\left(\sum_{j=1}^{m} \alpha_j p_j\big(x_i|\theta_j\big)\right). \tag{3.11}$$

The Expectation Maximization (EM) algorithm is employed to estimate the parameters. EM consists of two steps, the Expectation or E-step and the Maximization

or M-step. In the E-step, $p_{i,j}$ is calculated, which corresponds to the probability of sample i belonging to mixture j using currently available parameters (initial values are set randomly)

$$p_{i,j} = \frac{\alpha_j p_j\left(x\middle|\mu_j, \mathrm{C}_j\right)}{\sum_{i=1}^{m} \alpha_k \, p_k(x|\mu_k, \mathrm{C}_k)}. \tag{3.12}$$

At the second step or M-step, the mixture parameters are refined by

$$\alpha_j = \frac{1}{N}\sum_{i=1}^{N} p_{i,j}\,, \qquad \mu_j = \frac{\sum_{i=1}^{N} p_{i,j} x_i}{\sum_{i=1}^{N} p_{i,j}},$$
$$\mathrm{C}_j = \frac{\sum_{i=1}^{N} p_{i,j}\left(x_i - \mu_j\right)\left(x_i - \mu_j\right)^T}{\sum_{i=1}^{N} p_{i,j}}. \tag{3.13}$$

Alternatively, the E-step and M-step can be applied in a reverse order depending on the availability of data at the time of calculation (see [60] for more details). After calculating the distribution of the background colors, their values are replaced by the color black (RGB value 0) (Figure 3.11).

3.3.2 Backprojection

A 3D RGB histogram of the object's template is created, ignoring the color black as it is used for the template background.

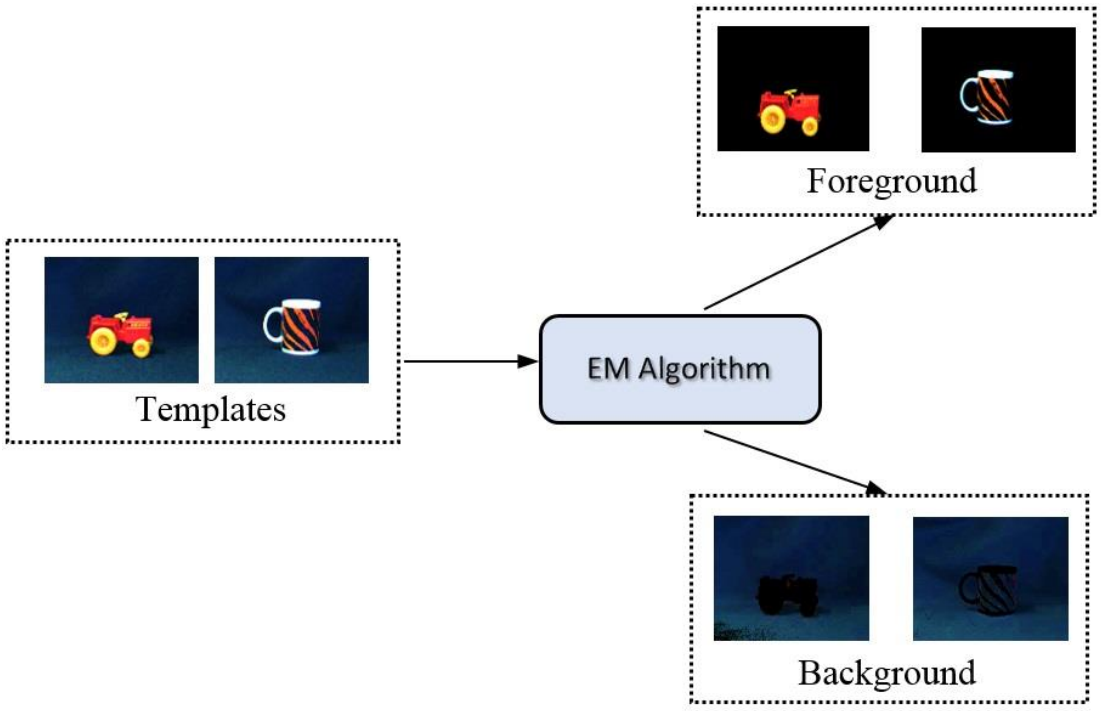


Figure 3.11: The application of the EM algorithm to separate the objects foreground from the background.

Let $h(CL)$ be the histogram function that maps color $CL = (R, G, B)$ to a bin of histogram $H(CL)$ generated based on object's template T_θ. We can perform backprojection of the object over an image as follows:

$$\forall\, x, y: b_{x,y} \coloneqq h(I'_{x,y,cl}), \tag{3.14}$$

where b is the grayscale backprojected image, and I' is normalized image I(see Figure 3.12).

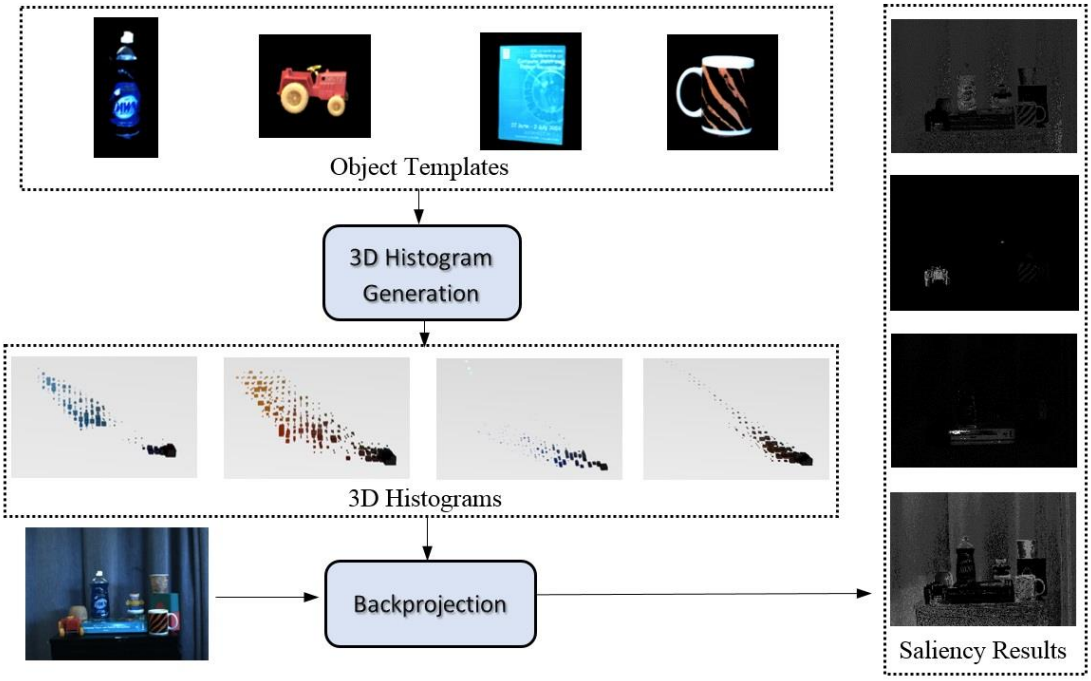


Figure 3.12: The Histogram Backprojection results of four samples. The saliency results from the top to bottom refer to the object templates from the left to right.

3.3.2.1 Image Normalization

The performance of HB is limited in applications where images are captured in variety of conditions. In practice, an object's template is created independent of the physical environment in which it will be searched. However, throughout the search process using a mobile robot, the object's colors might be perceived differently depending on the distance of the robot to the target, reflection, illumination changes and even in some cases the type of sensor used. As a result, a direct projection of the template's colors would fail to accurately identify the target in the majority of situations.

One way of addressing the issue of illumination changes, is to normalize the images of interest. A simple but effective technique is pixelwise normalization [61], in which every pixel's color values r_k, g_k and b_k are normalized by

$$r_k{}' = \frac{r_k}{s_k}\,, \qquad g_k{}' = \frac{g_k}{s_k}, \qquad b_k{}' = \frac{b_k}{s_k} \tag{3.15}$$

where r', g' and b' denote the normalized color values and $s_k = r_k + g_k + b_k$ is the intensity of each pixel. Normalization also can be achieved channelwise,

$$r_k{}' = \frac{r_k}{\sum_{i=1}^{n} r_i}\,, \qquad g_k{}' = \frac{g_k}{\sum_{i=1}^{n} g_i}, \qquad b_k{}' = \frac{b_k}{\sum_{i=1}^{n} b_i} \tag{3.16}$$

where n is the total number of pixels in each channel.

Swain and Ballard [32] proposed an alternative approach to normalization. They used what so called *the three opponent color* axes technique to isolate the intensity of an image in a separate channel, which in turn will be coarsely indexed to reduce the effect of lighting. In this model channels are defined as follows:

$$\begin{aligned} rg_k &= r_k - g_k, \\ by_k &= 2 * b_k - r_k - g_k, \\ wb_k &= r_k + g_k + b_k \end{aligned} \tag{3.17}$$

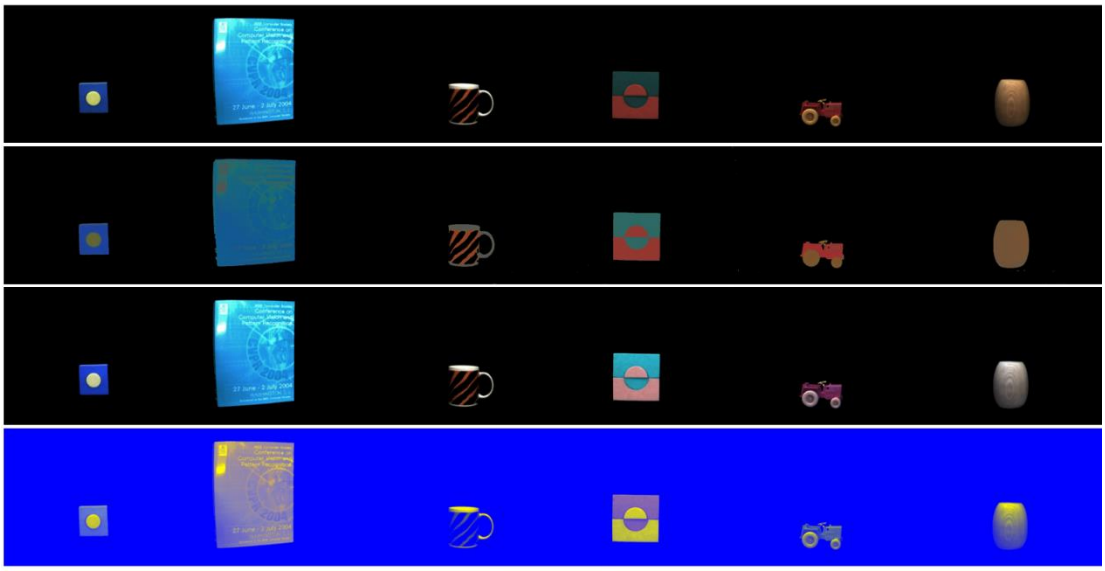


Figure 3.13: The normalization of the sample templates using different techniques. From the top to bottom: the original templates, pixelwise normalization, channelwise normalization, and the three opponent color axes.

where rg, by and wb are the three color axes. Figure 3.13 shows the result of each normalization technique on a number of sample templates.

3.3.2.2 Color Indexing

Although normalization reduces the effect of illumination changes, it certainly is not sufficient for backprojection. The color (hue) of an object also may change in regard to lighting's color, the surface reflection of other objects or shadow.

To resolve this issue, the histogram of templates ought to be indexed. i.e. a range of colors is considered for each bin other than using the specific number of colors (e.g. 0-255 in the case of 8 bit RGB). Here, care should be taken not to set the index values too high so that each bin includes too many colors, or too little that with the smallest

changes in the environment conditions such as lighting, HB fails to identify the object of interest.

Figure 3.14 shows the effects of the index size on the final output of the HB map. In this example, the images are pixelwise normalized and thresholded to only reflect the highest salient points. As can be seen, by increasing the size of the color index, the saliency becomes more specific up to a point that the algorithm fails to detect the target. These images also highlight the potential issues with merely relying on color distributions especially in the cases where there is a similarity between the object, e.g. the cup, and the environment's color distributions.

3.4 Building the Final Saliency Map

The final saliency map is produced by merging the AIM and HB conspicuity maps. For this purpose, a binary mask of the AIM saliency responses is created. This mask is then applied to the original image to extract the RGB values corresponding to the interest points identified by AIM,

$$\hat{I}_{\boldsymbol{\theta}} = I_{\theta} \times M(\boldsymbol{x}, \boldsymbol{y}),$$
$$\begin{cases} M(\boldsymbol{x}, \boldsymbol{y}) = 1 & info(\boldsymbol{x}, \boldsymbol{y}) > \boldsymbol{p} \\ M(\boldsymbol{x}, \boldsymbol{y}) = 0 & else, \end{cases} \tag{3.18}$$

where $info()$ refers to the information map generated by AIM.

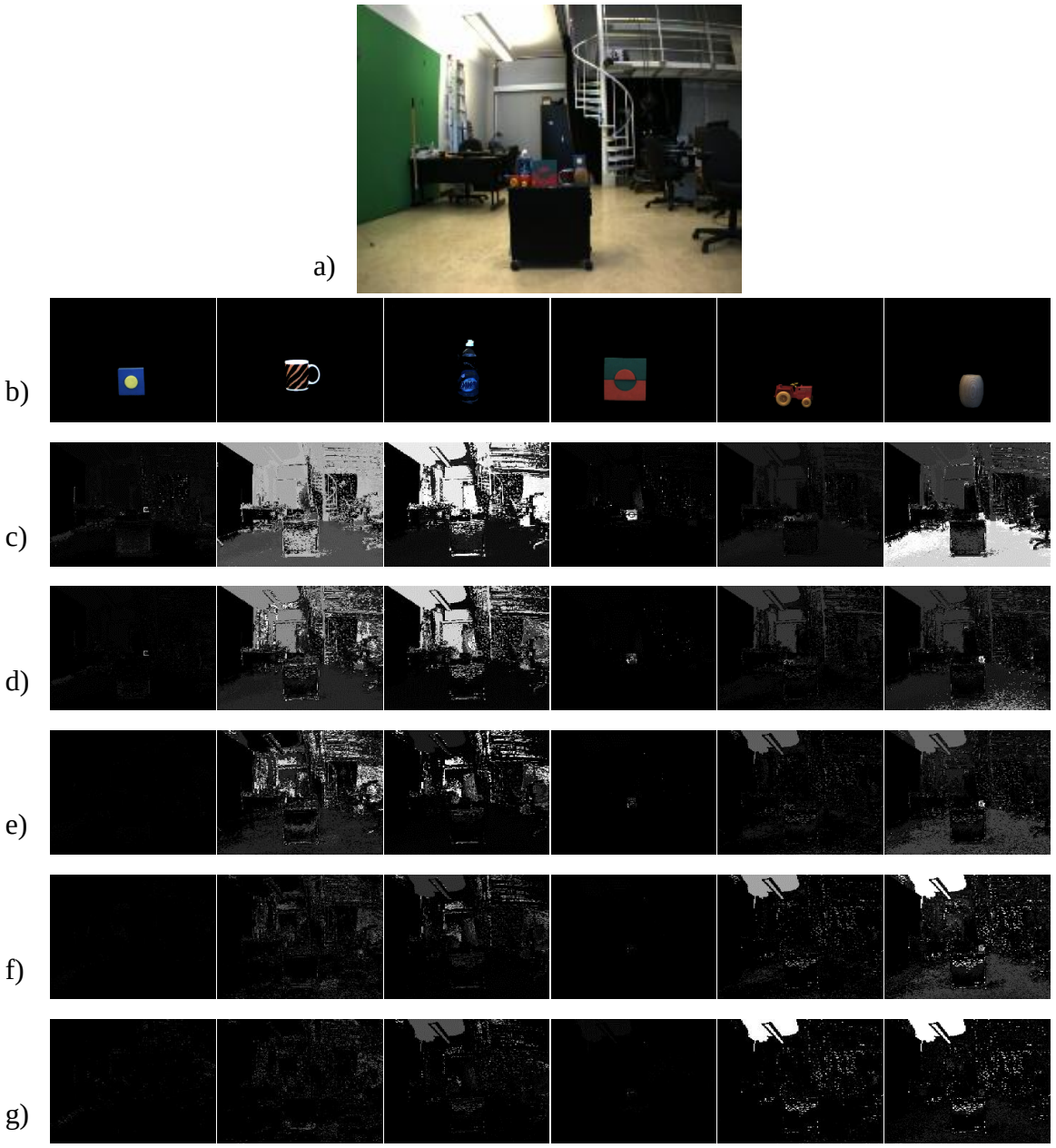


Figure 3.14: The HB saliency results using different index sizes. The images are pixelwise normalized and thresholded using a 8-bit value, 60. a) is the image of the environment, b) are the object templates and c-g) the HB generated saliencies with index sizes 16, 32, 64,128 and 256 respectively.

where I_θ is the original image captured through camera configuration θ , $info(\boldsymbol{x}, \boldsymbol{y})$ is the information map resulted from AIM, $M(\boldsymbol{x}, \boldsymbol{y})$ is the binary mask and $\boldsymbol{p}$ denotes the percentile threshold. Image $\hat{I}_\theta$, is used to produce the HB saliency through the projection of an object's template. Then, the result is integrated with the AIM conspicuity map, each contributing 60% and 40% to the final saliency results respectively. These values are set empirically to ensure that the saliency responses of each model are not exaggerated.

3.5 Applying Saliency to Visual Search

The resulting saliency map is used to increase the probability distribution of those regions that have a higher chance of detecting the target. By relying on a stereo camera, the depth of each point is calculated to determine its 3D coordinates within the environment. Those points that fall within the range of the recognition algorithm (the effective field of view) are discarded, otherwise based on their perceptual saliency, the probability distribution of the target's presence is increased. Figure 3.15, shows the process of generating and applying saliency to visual search.

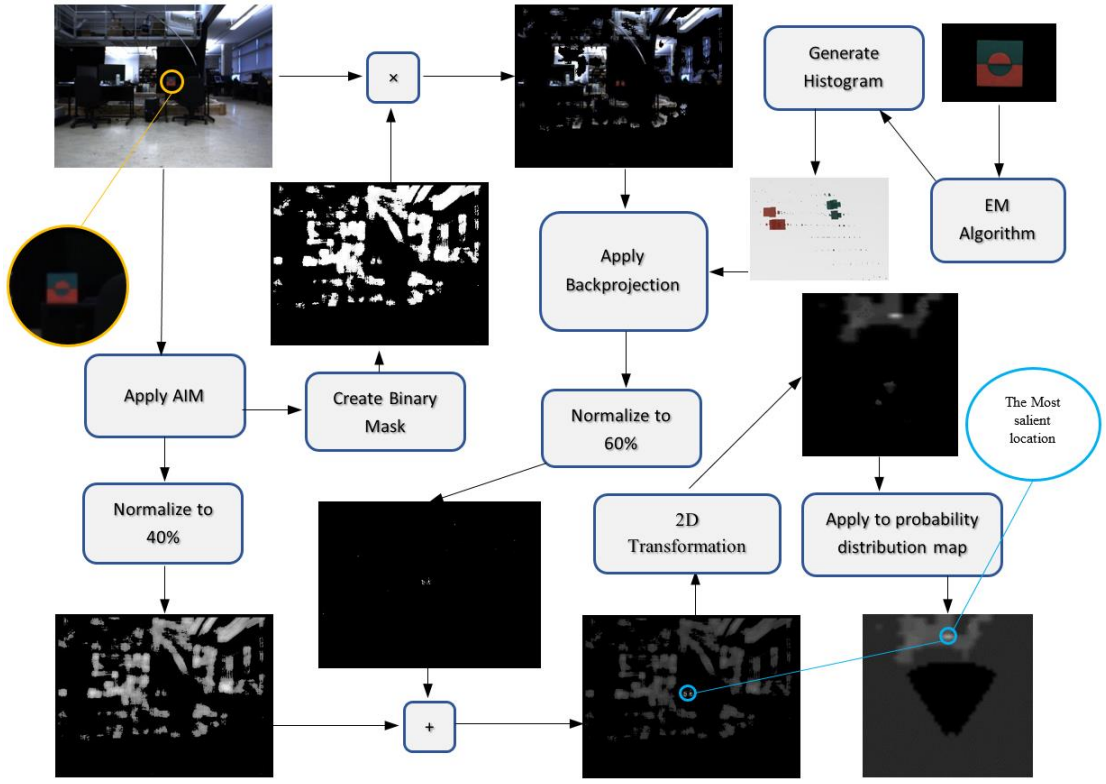


Figure 3.15: The process of applying saliency to the robotic visual search.

3.6 Summary

In this chapter, we explained two techniques for building saliency maps. The AIM algorithm is used to extract general interest points within an environment that have a higher chance of corresponding to structures that are in spatial relation to a target of interest. These points can play the role of indirect clues by guiding a search agent to the regions of a higher importance.

The second method is Histogram Backprojection (HB). HB is used to identify the interest points relating to the object of interest rather than the environment's structure. The combination of these two methods modifies the probability distributions of the target's presence, allowing the search agent to make more informed decisions when selecting an action.

4 Sensor Planning Strategies with Predefined Constraints

Recalling the concept of search from Chapter 2, the goal of object search is to find a sequence of operations that maximizes the probability of detecting the target within a predefined cost constraint. Ye [17] introduces an approximate solution to the problem using a greedy approach that maximizes the probability of detecting the target at any time when an operation is selected. Shubina and Tsotsos [18] extend this work by adding the cost of each operation in terms of the distance traveled by the robot. This cost is used to determine the utility of the next best location for the robot to move to.

In the above greedy approaches, the cost of each action is only considered locally, meaning that among all the possible operations, they determine what the next best operation that maximizes the value of an action. The cost is only included to compute the utility of an action. These methods do not consider the possibility of a larger scale optimization approach. This is particularly important to maximize the chance of finding a target within a given cost constraint.

Another aspect for potential improvement of Ye's sensor planning strategy concerns thresholding the utility of a location with a fixed value before moving to another one. If the robot perceives an area of a greater importance in another location, it is limited to finish searching its current location before deciding to move to the next,

i.e., searching its current location must continue until some threshold is reached regardless of what is present at other locations.

Setting the threshold itself is also challenging. In [18], the authors define the threshold value empirically based on the parameters of search such as the resolution of the search (the size of cubic elements holding the probability distributions), the recognition algorithm's effective field of view, or other factors that affect the search process. In practice, some of these elements may change prior to search, which then would require conducting evaluative experiments to estimate a reasonable threshold value. In addition, throughout the search there might be different methods of analysing an image to be used. For instance, an agent may be instructed to detect a table and then search for a cup on top of it. The algorithms to perform these tasks are not necessarily the same, each having its own characteristics. As a result having a fixed threshold to accommodate different tasks in this context may not suffice for a good search algorithm.

The aforementioned deficiencies in the previous sensor planning strategies point strongly to the need for a more general approach. First, such a technique first should have a global approach towards sensor planning by acting according to the constraints within which we intend to conduct the search such as available time or the battery energy of the robot. Second, the model should be free of any predefined parameters

that limit the performance of the search agent in employing different methods for detecting an object.

The contribution of the work presented in this chapter is twofold. First, we expand on the complexity of object search mentioned in [17], and then we formalize the problem in an attempt to define some global optimization technique to solve the problem with respect to a constraint. Section 4.1 shows the complexity of object search. Section 4.2 reviews a variety of the knapsack problems and their similarities to object search. Section 4.3 details the practical limitations of object search. Section 4.4 surveys some of the common techniques of solving the knapsack problem. Section 4.5 proposes some novel approaches to sensor planning for object search with a constraint.

4.1 The complexity of object search

Ye and Tsotsos [62, 28, 63] perform a complexity analysis of object search and prove that this problem belongs to NP class (later in this section, we comment on the $completeness$ of the problem).

It is difficult to extract any regularities in the problem of object search as described in Chapter 2, primarily due to the presence of intermediate probability distributions which are being changed after the application of each operation. To add further perspective to the problem of search, Ye further simplifies the search process at the

current location of the robot. He proposes to only update the probability of the regions within the sensed sphere (the region around the robot with a radius equal to the size of the recognition algorithm's effective field of view) at a time the robot is searching its current position. This means the probabilities of the remaining locations are updated when the robot moves to a new destination.

With respect to the above assumptions and restricting the actions in a way that no two actions share influence ranges (i.e. the recognition algorithm's effective field of view), $\Psi_{f_i} \cap \Psi_{f_j} = 0$ for $i \neq j, P[\boldsymbol{F}]$ can be redefined as follows:

` $$P[\boldsymbol{F}] = \sum_{i=0}^{n} p(\boldsymbol{f}_i, \tau_0). \quad (4.1)$$

According to (2.4) and constraint $T(\boldsymbol{F}) \leq K$, where K is the total time available to conduct the search, the problem of object search can be reduced to the KNAPSACK problem in which the objective is to maximize a value (probability distributions) while not exceeding a capacity (K). Based on this interpretation, the problem of object search is showed to be *NP-hard*.

Alternatively, one can transform the optimization problem into an equivalent decision problem by imposing a lower limit on $P[\boldsymbol{F}]$. In this case, the problem of object search is defined as finding an effort allocation $\boldsymbol{F}$ such that $T(\boldsymbol{F}) \leq K$ and $P[\boldsymbol{F}] \geq M$. By this definition, the problem belongs to $NP - complete$ class. In practice, however,

it is hard to set the M value, therefore, we treat object search as an optimization problem in this work.

4.2 The KNAPSACK problem

In order to formulate a sensor planning algorithm for object search with a cost constraint, we follow the same reasoning as Ye and reduce the object search problem to the KNAPSACK problem [64]. Here, however, we intend to generalize this idea to select a sequence of actions to maximize the probability of detecting an object with a given constraint.

Recalling the KNAPSACK problem, we have n items with $profits\ (p)$ and $weights(w)$ and we want to place a subset of items in a knapsack of $capacity\ (c)$. In this manner, we want to maximize the profit of the selected subset while its overall weight should not exceed the capacity of the knapsack.

Recalling from (2.3), let O_Ω be the set of all possible operations on Ω and $\boldsymbol{F} = \{f_1, f_2, \dots, f_k\}$ an ordered set of actions performed during the search. To fit into a similar description of KNAPSACK, we consider O_Ω as the set of all items available to pick from, and probability $p(f)$ and cost $t(f)$ as the profit and weight of each item respectively. Here, the capacity is defined in terms of the cost constraint $\boldsymbol{K}$ and maximization is constrained by $T[\boldsymbol{F}] \leq \boldsymbol{K}$ where $T[\boldsymbol{F}]$ is the total cost of subset $\boldsymbol{F}$.

4.2.1 Variation of KNAPSACK

In order to formulate a solution to our problem, we first need to know which category of the KNAPSACK problem, object search belongs to. Perhaps the most common case of the KNAPSACK problem is the ***0-1 knapsack problem***. This problem arises when the objective is to maximize the value of n objects while there is only one instance of each allowed,

$$maximize \sum_{j=1}^{n} p_j x_j \,,$$

$$subject\ to \sum_{j=1}^{n} w_j x_j \leq c,$$

$$x_j = 0\ or\ 1\,,\ \ j = 1, \dots, n. \tag{4.2}$$

A special case of this problem also exists where the goal is to select a subset of weights closest to the capacity of the knapsack [64].

The number of knapsacks may vary. In the ***0-1 Multiple Knapsack Problem*** [65], there are a total of m knapsacks with capacities $c_1, c_2, \dots, c_m$. Similar to the 0-1 knapsack problem, the aim is to maximize the value of items subject to each item selected must be put in all m knapsacks. Another special case of the problem is the ***bounded knapsack problem*** [66] that considers a limited number of each item. If there is no limit on the number of items to use, this problem transforms into an ***unbounded knapsack problem*** [67].

Maximizing the profit while having more than one weight (cost) is addressed in ***Multidimensional Knapsack Problem*** [68]. In the cases where x is not an integer, the problem becomes a ***Fractional Knapsack Problem*** [69].

There are other variation of the KNAPSACK problem, which are beyond of the scope of this thesis including the ***Temporal Knapsack Problem*** [70], ***Interactive Knapsack problem*** [71], ***Dynamic and Stochastic Knapsack Problem*** [72], ***Partially Ordered Knapsack*** [73], ***Static Stochastic Problem*** [74], and ***Change Making Problem*** [75].

In the object search problem, each operation is independent and unique. The process of action selection is binary, i.e. an operation is either selected or not. Each action is only selected once given the probability (profit) of the action is reduced to zero once selected. Given this definition, the problem of object search with a single constraint can be regarded as a 0-1 knapsack problem as follows:

$$maximize\ P[\boldsymbol{F}] = \sum_{i=0}^{n} p(\boldsymbol{f}_j)\, x_j,$$

$$subject\ to\ T[\boldsymbol{F}] = \sum_{i=0}^{n} t(\boldsymbol{f}_j)\, x_j \leq \boldsymbol{K}.$$

$$x_j \in \{0, 1\},\ j = 1, 2, \ldots, n \tag{4.3}$$

It is important to note that if we intend to use multiple constraints for the search (e.g. time and energy consumption), a multidimensional knapsack would be a more adequate choice.

4.3 The practical limitations of optimizing object search

Thus far, in every definition of the KNAPSACK problem, the profit and weight of each item are known in advance and constant, and the order of selecting the objects does not alter the final result. This is with the exception of the stochastic knapsack problem in which one component of the problem is not completely known in advance or may be subject to change.

In practice, only the constraint of search is known and constant throughout the process. Depending on the order of choosing actions, the value and cost of each operation changes. This implies that two sets of identical operations with different orders may result in dissimilar values and costs.

Assume we want to perform two operations $f_1 = (S_1, a)$ and $f_2 = (S_2, a)$ with the probabilities of detecting a target $p(f_1)$ and $p(f_2)$, where S_j is the camera's configuration parameters and f_j is the action for $j = 1, 2$. We assume that the number of cubic elements in f_j's effective field of view is given by ψ_j.

Lemma 1

$$p(f_1 = Successful) + p(f_2 = Successful \mid f_1 \neq Successful) =$$

$$p(f_2 = Successful) + p(f_1 = Successful \mid f_2 \neq Successful) \quad (4.4)$$

If $\psi_1 = \psi_2$ *and* $\psi_\Omega > \psi_1 + \psi_2$.

Proof

The probability of each action is given by

$$p(f) = \frac{\psi_f}{\psi_\Omega}$$

where Ω is the search space and $\psi_{f_1} \cap \psi_{f_2} = \emptyset$,

$$\frac{\psi_{f_1}}{\psi_\Omega} + \frac{\psi_{f_2}}{\psi_\Omega - \psi_{f_1}} = \frac{\psi_{f_2}}{\psi_\Omega} + \frac{\psi_{f_1}}{\psi_\Omega - \psi_{f_2}}$$

$$\frac{\psi_{f_1}(\psi_\Omega - \psi_{f_1}) + \psi_{f_2}\psi_\Omega}{\psi_\Omega(\psi_\Omega - \psi_{f_1})} = \frac{\psi_{f_2}(\psi_\Omega - \psi_{f_2}) + \psi_{f_1}\psi_\Omega}{\psi_\Omega(\psi_\Omega - \psi_{f_2})}$$

$$(\psi_\Omega^2 - \psi_{f_2}\psi_\Omega)(\psi_{f_1}\psi_\Omega - \psi_{f_1}^2 + \psi_{f_2}\psi_\Omega)$$

$$= (\psi_\Omega^2 - \psi_{f_1}\psi_\Omega)(\psi_{f_2}\psi_\Omega - \psi_{f_2}^2 + \psi_{f_1}\psi_\Omega)$$

$$\psi_{f_1}\psi_\Omega^3 - \psi_{f_1}^2\psi_\Omega^2 + \psi_{f_2}\psi_\Omega^3 - \psi_{f_1}\psi_{f_2}\psi_\Omega^2 + \psi_{f_1}^2\psi_{f_2}\psi_\Omega - \psi_{f_2}^2\psi_\Omega^2 =$$

$$\psi_{f_2}\psi_\Omega^3 - \psi_{f_2}^2\psi_\Omega^2 + \psi_{f_1}\psi_\Omega^3 - \psi_{f_1}\psi_{f_2}\psi_\Omega^2 + \psi_{f_1}\psi_{f_2}^2\psi_\Omega - \psi_{f_1}^2\psi_\Omega^2$$

$$\psi_{f_1}^2\psi_{f_2}\psi_\Omega = \psi_{f_1}\psi_{f_2}^2\psi_\Omega$$

$$\psi_{f_1} = \psi_{f_2} \quad (4.5)$$

QED.

Intuitively, the sum of probability of performing two actions f_1 and f_2 are equal if their influence ranges cover the same amount of space. This means if this equality does not hold the order of selecting the actions matter.

Similarly for the cost of operations, let $S_0 = \{\theta_0, \omega_0, \sigma\}$ be the current camera setting, where $\theta_0 = \{x_0, y_0, z_0\}$ is the camera position in 3D environment Ω, $\omega_0 = \{p_0, t_0\}$ the angles of pan and tilt, and $\sigma = \{w, h\}$ the width and height of the camera's field of view.

It is trivial to show that there are cases in which the order of selecting actions influences the overall cost of search (see Section 4.5.1). Hence, the value and cost of operation allocation $\boldsymbol{F}$ could be different depending on the order in which each action is applied. This points to the fact that in order to globally optimize a search, we are dealing with the permutation of operations other than combination. As a result the time complexity of solutions and the size of the search space is significantly increased.

4.4 Solutions to 0-1 knapsack problems

4.4.1 Exact solutions

4.4.1.1 Brute Force

The most straightforward approach to solve the KNAPSACK problem is to consider every possible combination of components and select the one that yields the highest

value. The time complexity of brute force for solving the object search knapsack is $O(n!)$ given the total number of possible permutations as follows:

$$\sum_{k=0}^{n} \frac{n!}{(n-k)!} \approx e * n! \tag{4.6}$$

where k is the size of permutation and n the total number of operations. To that extent, the brute force approach is extremely inefficient as a solution to object search. For example, assume we have a search application where the robot has a total of 20 discretized directions to look toward and 30 possible locations to perform the search from. The resulting number of operations is 600 to choose from. The time requirement of selecting a sequence of operations is given by, $600! * e \approx 3.46 * 10^{1408}$. This is neither feasible nor possible in any available practical system.

4.4.1.2 Dynamic Programming

One of the most common techniques of finding an exact solution to the KNAPSACK problem is dynamic programing [76]. Recalling (4.2), using dynamic programming, a maximum subset of the items can be calculated as follows: first, a 2-dimensinal array $f(k, y)$ is created, where k and y are integer and $0 \leq k \leq n$ and $0 \leq y \leq c$ where n and c are the number of items and capacity respectively. The $f(k, y)$ values are given by

$$f(0,y) = f(k,0) = 0$$

$$f(k,y) = \begin{cases} f(k-1,y) & if\ w_k > y \\ \max\{v_k + f(k-1,y-w_k), f(k-1,y)\} & if\ w_k \leq y\ and\ k > 0 \end{cases}. \quad (4.7)$$

This would lead to finding the maximum obtainable value from n items in $f(n,c)$. To estimate the value of x_j, we perform backtracking by

$$\text{repeat for } k = n-1, \ldots, 1\ ,$$

$$x_j = 1\ \ if\ f(k,y) \neq f(k-1,y),$$

$$x_j = 0\ if\ f(k,y) = f(k-1,y), \quad (4.8)$$

where the capacity for previous items is $y = y - w_k x_k$. The time complexity of dynamic programing is pseudo-polynomial with $O(cn)$.

Applying dynamic programing to the object search knapsack imposes a number of limitations. The dynamic programing splits constraint (capacity) into equal intervals. In object search, these intervals should be in real numbers in the order of thousands to accommodate variation in the probability values of operations. This significantly increases the size of sub problems. In addition, the main assumption of dynamic programing is that the profits and weights are constant throughout the process. Hence, this solution has no mechanism to address the changes in the probability values and costs of search operations.

4.4.1.3 Branch-and-Bound

A well-known branch-and-bound technique was introduced by Horowitz and Sahni [77]. This method consists of two operations: *forward move* in which the largest possible set is inserted into the current solution and *backtracking move* where the last inserted item is removed from the current solution. At any point when the next best item cannot be selected, an upper bound value U is calculated and compared to the best solution so far to realize whether a forward move could result in a better solution. If not, a backtracking move is performed. The algorithm terminates when no further backtracking is possible.

Despite the fact that branch-and-bound on average has a lower processing time than brute force due to the pruning of branches throughout the process, it has the worst case timing of $O(n!)$. In addition, for the instances of search with a large number of possible operations, the number of nodes and branches increases exponentially, which requires a significant amount of memory and time.

4.4.2 Approximate solutions

4.4.2.1 Greedy algorithm

The most immediate approach toward estimating a solution to the knapsack problem is through the use of a greedy algorithm [78]. Suppose there are a number of items sorted according to their utility values given by p/w. The greedy algorithm selects the items

in a descending order until a critical item is observed, i.e. the next best item does not fit into the knapsack.

The greedy algorithm can converge to an optimal solution and has the worst-case performance ratio of ½. The time complexity of the greedy algorithm is $O(nlogn) + O(n)$ and it only requires memory size of $S(n)$. In the case of object search, the greedy approach is slightly different. Because the values of remaining operations should be recalculated after selecting each action, we omit the sorting step and instead only select the action with the highest utility value each time. In this scenario, the processing time is increased to $O(n^2)$.

The major drawback of the greedy approach in object search is that it only picks actions one at a time and lacks the global view of consequence of each action on the overall cost constraint of the search.

4.4.2.2 Extended Greedy Algorithm

The extended greedy algorithm [79] is an improved version of the greedy algorithm in which the model continues selecting the next best items until the knapsack is full or it reaches the last item. It is trivial to show that the processing time of extended greedy algorithm to solve object search is also $O(n^2)$.

4.4.2.3 Polynomial-Time Approximation Schemes (PTAS)

Approximation schemes are a group of techniques that allow one to achieve any prefixed performance ratio at the expense of increasing the processing time. The most

common approaches in this category are Polynomial-Time Approximation Schemes (PTAS) [80] and Fully Polynomial Approximation Schemes (FPTAS) [81] from which PTAS will be discussed briefly.

Suppose items are sorted according to their utility values with profits p_j and weights w_j to be fit in a knapsack, subject to capacity constraint c. Let $z^h = 0$ be the highest value so far, to maximize the value of the knapsack using PTAS, the subsets of the items are calculated by

$$\forall\, m \;:\; 0 \leq m \leq k \;\;\&\;\; m \;\in \mathbb{Z},\; M = \binom{n}{m} : \sum_{j \in M} w_j \leq c \tag{4.9}$$

where M set of the subsets and k is a non-negative integer denoting the maximum size of the subsets. For each subset, starting from empty, $M = \{\}$, the corresponding items are fixed in the solution and the capacity is adjusted by

$$\hat{c} = c - \sum_{j \in M} w_j. \tag{4.10}$$

Then, the final value of the knapsack, considering $\hat{c} = \hat{c} - w_i$ after the selection of each new item i is given by

$$\begin{gathered} z = \sum_{j \in M} p_j + \sum_{i \notin M} p_i \;:\; w_i \leq \hat{c} \\ z^h = z \;\; if \; z > z^h \end{gathered} \tag{4.11}$$

$$i = 1, 2, \dots, n.$$

The processing time of PTAS is exponential with respect to the value of k given by $O(n^{k+1})$and it has the space complexity of $S(n)$. The worst case performance ratio, r, also is dependent on k, computed by $r = k/(k+1)$. This property of PTAS provides a flexibility to achieve different performance rates with respect to the allowable processing time of any given applications.

In the case of object search, the space complexity of PTAS remains the same as the maximum size of each set would not exceed the total number of items. However, the processing time significantly increases. After choosing an operation, the probability values and costs of the remaining ones have to be recalculated and sorted again, which add at least a factor of $O(nlogn)$ to each sub-problem. Permuting each subset M also increases the total number of sub-problems significantly, and as a result, the overall processing time. For instance, in the above problem using $k = 2$, there are at most a total of 16 subsets whereas 26 permutated ones.

4.5 Knapsack solution to object search

Object search, as mentioned earlier, is different in nature to the classical knapsack problems. It deals with the permutation of actions in which the values and costs of

operations change depending on the order of selections limiting the use of the solutions mentioned above in the context of search. This points to the need for a new formulation to accommodate the requirements of object search. In the following sub-sections, we propose three sensor planning strategies designed based on the solutions to the classical knapsack problem.

4.5.1 Cost function in object search

In [17], Ye considers the cost of actions only when a robot relocates to a new position. This is a reasonable approach, since he employs a greedy approach and the fact that the cost of each action is similar in a stationery position. For the purpose of a global optimization approach, however, we generalize this idea in two ways: first we consider all the costs associated with performing each action including changing the direction of the pan-tilt unit, applying the recognition algorithm, relocating the robot and any other processes associated with the algorithm (e.g. path planning). These costs have to be considered in order to estimate the remaining constraint in each stage of the search and act accordingly.

Second, we distinguish between the cost of each action even if both are performed from the same location. In practice, (as shown in lemma 2) the cost of a sequence depends on the order within which it is applied regardless of the location of each operation. For instance, assume a scenario in which the current pan angle of a camera

is 0. We have two operations f_1 and f_2 to perform from the same location each with pan angle 40 and 120 degrees respectively. If the order of application is $f_1 f_2$, the pan unit has to turn $40 + 80 = 120$ degrees. Applying the same actions in a reverse order, pan unit has to turn $120 + 80 = 200$ degrees. Given certain costs are associated with altering the pan unit's angle, such as time or energy it consumes, one can easily see why applying two actions may incur different costs.

For the purpose of this study, the cost of operations is measured in terms of the time they take to complete, energy a system (robot and processing unit) consumes to perform those operations and the distance travelled by the robot throughout the search. The first two cost functions are similar in the sense that every component of an action incurs them. The latter, however, is different because it is only applied when the robot moves to a new location. This means operations from same locations are identical in terms of the cost they incur, i.e. the cost of performing them is equal to 0.

4.5.2 Greedy Search with Constraint (GSC)

Perhaps the simplest approach to solving the problem of object search is the use of a greedy algorithm similar to the one introduced by Yiming Ye [17]. Due to the issues mentioned earlier, nevertheless, we need to modify this algorithm to first, take into account the overall constraint of the search and act accordingly and second, not to be dependent on any prefixed threshold.

Here, we propose a greedy algorithm with some modification to behave according to a constraint. The probabilities and costs of all operations are calculated and placed in two arrays of $P = \{p_1, p_2, \dots, p_n\}$ and $T = \{t_1, t_2, \dots, t_n\}$ respectively. The utility of actions are measured by

$$U(\tau) = \{\frac{p_1}{t_1}, \frac{p_2}{t_2}, \dots, \frac{p_n}{t_n}\} \tag{4.12}$$

where $U(\tau)$ is the set of all utilities at time τ. The next operation then is selected by

$$f(\tau) = \max\{U(\tau)\} \tag{4.13}$$

where $f(\tau)$ is the action chosen at time τ. Note that because the probabilities and costs of operations should be recalculated after each selection, instead of sorting the elements of set U, the algorithm simply picks the maximum value. This reduces the time complexity from $O(nlogn)$ to $O(n)$ at each stage.

The model behaves greedily by selecting the actions with the highest utilities until a percentage of the cost constraint, α, is reached. At this stage, the algorithm chooses the next action in regard to its probability distribution as follows:

$$\begin{aligned} U'(\tau) &= \{p_1, p_2, \dots, p_n\} \\ f'(\tau) &= \max\{U'(\tau)\} : \ t_j \leq \widehat{K}(\tau) \end{aligned} \tag{4.14}$$

where t_j denotes the cost of an action, and $\widehat{K}(\tau)$ the remaining search constraint at time τ. Intuitively, once percentage of constraint α is reached, the algorithm selects the next best action according to its probability value instead of its utility with respect to cost.

The major weakness of the greedy algorithm, in this context, is the locality of its scope, i.e. it lacks the ability to look ahead of its current action to determine how it affects the later stages of the search. This is an expected behavior because after applying each operation, the values of remaining actions change, which are not foreseen by the algorithm. To address this issue, Ye assigns a threshold to move the robot to a new location. By changing this value, one can determine how fast the robot should span the range of its search.

In the GSC algorithm the movement of the robot occurs naturally. Depending on how salient locations beyond the effective range of the camera look from the robot's perspective, it decides when to relocate. The α value also induces the robot to move to a new destination. The greater the value of α, the faster the robot tends to expand its scope of search and vice versa.

4.5.3 Extended Greedy Search (EGS)

Extended Greedy Search (EGS) is a direct adaptation of the extended greedy algorithm explained earlier in Section 4.4.2.2 with some modifications to account for the object

search characteristics. Similar to GSC, the utility of operations are calculated according to equation (4.15). Then, a sequence of actions to be performed during the search is given by

$$F(\tau) = \{ f_1, f_2, \dots, f_m \}, \quad m \leq n$$

$$f_1 = \max\{U(\tau)\} : t_1 \leq K \,,$$

$$f_j = \max\left\{U\left(\tau^{+}_{f_{j-1}}\right) - u_{j-1}\right\} : t \leq K - \sum_{m=1}^{j-1} t_m$$

$$j = 2, 3, \dots, n, \qquad u_{j-1} = \frac{p_{j-1}}{t_{j-1}} \tag{4.15}$$

where $F(\tau)$ denotes the sequence of actions f_j, $U(\tau)$ is the utility of the operations at time τ, $U\left(\tau^{+}_{f_{j-1}}\right)$ is the utility of operations at time τ just after the selection of action f_{j-1}, t_j is the cost of action f_j and K represents the overall constraint of the search. It is important to note that, the sequence of actions is formed before the application of any operation to the environment. Therefore, EGS does not have any information regarding the environment, e.g. the saliency clues, the first time it generates an action sequence.

The lack of knowledge regarding the environment implies a number of potential problems for the EGS algorithm. The most dominant one is increasing the chance of selecting an action that the robot is unable to perform due to the adjacency of the operation's location to an obstacle. In such a situation, the algorithm needs to generate an entire new sequence, which can add to the overall cost of the search.

4.5.4 Dynamic Look Ahead Search (DLAS)

We saw that in the previous sections designing an exact solution to the problem of object search is quite challenging. This is primarily due to the fact that if the order of choosing actions in a search changes, the overall probability value and cost of the search may change.

To find a near optimal solution to the problem of search, while reducing the complexity of the problem, both in terms of time and space, we propose some modifications to the brute force approach. We begin by creating the list of actions permutations in which instead of calculating all possible subsets, we incrementally create the list. This means, we start by permuting 2 actions as follows:

$$\{f_{11}(\tau_0)f_{21}(\tau^+_{f_{11}}), f_{11}(\tau_0)f_{20}(\tau^+_{f_{11}}), \dots, f_{ni}(\tau_0)\, f_{(n-1)i}(\tau^+_{f_{ni}})\}$$
$$i = 0, 1, \tag{4.16}$$

where $\left(\tau^+_{f_{ji}}\right)$ is the time after operation $f_{j1}$is applied and n is the total number of operations available to choose from.

At this stage, less optimal sets and the ones that exceed the search constraint are pruned. Determining the optimality of a sequence at this stage is difficult without having a prior knowledge of the search environment. Here, we use an estimate to select subsets that are near optimal as follows: if a subset has a higher cost in comparison to

others while yielding a lower probability value is considered as non-optimal. Based on this assumption, non-optimal subsets are removed from the list. The optimality measure is explained in details in Section 4.5.4.1.

The optimized sets are then combined with one more action to form subsets of three operations. In this manner, if adding one more action causes a subset to exceed the constraint, it is discarded. Once again after creating a new list, the subsets are optimized as before.

The process of synthesis and pruning continues until the maximum size of subsets defined by a user is reached. It is obvious to see that considering bigger subsets can eventually result in more optimal solutions but at the expense of a higher computation cost. However, after the list of possible action sequences is generated, the subset with the maximum probability value is selected as a candidate to be performed by the search agent. Note that an action sequence is generated without having any knowledge of the environment. So, if the robot is unable to perform an action due to its vicinity to an obstacle, an entire new action sequence is generated.

The time complexity of the proposed method is $O(n!/(n-k)!)$, where k is the maximum subset size defined by a user. In practice, the computation time of the algorithm is much less because the optimization of subsets significantly reduces the size of the list.

4.5.4.1 On optimality measure of subsets

Measuring the optimality of solutions calculated with DLAS is challenging. Mathematically it is difficult because the optimality of a solution is highly dependent on the context in which it is used. It is also challenging heuristically because the computation time of an exact solution such as brute force is not feasible for large instances of search. For instance, in the case of selecting 8 best operations out of 24 available ones, applying brute force takes 3.5 days!

In this subsection, we present a different empirical study to show how much improvement can be achieved using the DLAS algorithm compared to an extended greedy approach. The extended greedy algorithm is chosen because as mentioned before it can result in an optimal solution with the worst performance measure of 50% optimal.

For evaluation purposes, we conducted over 10000 experiments in simulated environments similar to those shown in Chapter 5. The search parameters also were selected similar to our practical experiments. The environments were discretized into voxels of size $100\ mm^3$and pan-tilt angles were discretized comprising a total of 15 possible directions. The number of locations available to the agent was approximately 30 in each environment creating more than 400 possible operations for

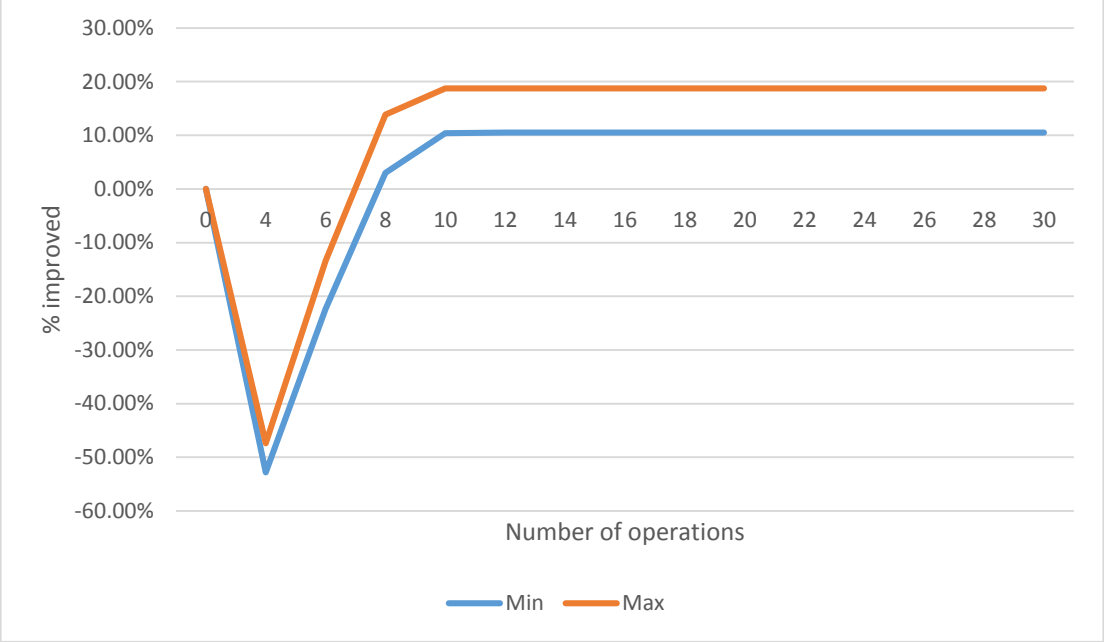


Figure 4.1: The performance improvement measure of using the DLAS algorithm in comparison to EGS.

the algorithms to choose from. The starting location and direction of the robot in each experiment was randomized. We only considered time constraint for these experiments ranging from 500 to 1500 seconds. For the DLAS algorithm, the maximum size of action sequences was set between 4-30 operations.

Figure 4.1 illustrates the results of our evaluations. As one would expect for small size of operation sequences, DLAS performs poorly. Of course after performing the initial sequence, if there is any remaining time, the algorithm would generate another sequence and continue the search. The performance of DLAS clearly is superior for instances where the algorithm selects 8 actions or more, where an improvement of at least 10% is observed.

4.6 Summary

In this chapter, three sensor planning algorithms were proposed to select appropriate search operations with respect to a cost constraint. The GSC algorithm is an approximate approach that greedily selects actions according to their utility values. Yet, once a percentage of the constraint is reached, it picks the operations with the highest probability values instead.

Another variation of greedy algorithms is the EGS method that greedily selects all the actions prior to the search. The disadvantage of this model is the lack of knowledge regarding the environment and a high chance of selecting the actions that the robot is unable to perform.

An improved approximate solution to object search, namely DLAS, was introduced that globally optimizes the search. This model achieves a near optimal solution at the expense of very low computation time, hence, makes it a suitable option to be used in robotic applications.

5 Experimental Evaluation

In this chapter, an empirical evaluation of the proposed methods is presented. Section 5.1 presents an evaluation of the use of saliency in visual search using Ye's sensor planning strategy to determine what performance improvements can be gained through the use of saliency. Section 5.2 shows runs of the proposed sensor planning strategies in Chapter 4 with different types of cost functions. Here the objective is to highlight the differences between each search strategy and to show which technique performs the best.

5.1 Saliency in visual search experiments

The saliency mapping technique used in our experiments follows the same procedure described in Section 3.4. The sensor planning strategy is the same as the one implemented by Shubina [30], which will be explained briefly in the next few subsections.

5.1.1 Sensor planning strategy

The search process is divided into two steps similar to that introduced in Chapter 2, namely "where to look next" and "where to move next". Prior to the search, the robot does not have any information regarding the environment except the dimensions and

locations of its external boundaries. Therefore, a uniform probability distribution is used to characterize the environment. Probability threshold Θ_{move} was empirically set to 0.05, meaning that if the probability of all available actions fall below that value, then the robot selects a new destination to explore.

5.1.2 Recognition algorithm

The detection model used in the experiments is based on normalized gray-scale correlation [82], as implemented by Shubina [30]. This algorithm is not view-independent, meaning that the target is only recognized when facing toward the camera up to some degree of transformation (in depth rotation). This algorithm reduces the task of 3D recognition to 2D recognition by relying on only one view of the object. In addition, the algorithm is scale and rotation (in the plane) invariant as long as the object stays within the detection range. Shubina [30] shows that this algorithm can handle up to 45 degrees of in depth rotation as long as the illumination of the target does not change significantly.

5.1.3 Navigation and localization

A stereo camera is used to supply sensory inputs for navigation purposes. The images captured through the camera are used to create a depth map of the environment using

OpenCV libraries. The depth information is transformed into a 2D grid of the search space used by the robot to select its path. The accuracy of the depth map highly depends on what the camera is pointing at but in general it is about 3 cm depth for the range of 3m. The depth error significantly increases beyond 10m to more than 30 cm.

The path planning is handled locally at each point of movement. The robot captures images from the direction of interest, builds an obstacle map and identifies the gaps that the robot can move through. Along the available paths, the robot chooses the first one it identifies by checking first the direction pointing toward its final destination and if occupied, checking the next best direction.

The localization of the robot in the environment depends upon the internal robot encoders and odometry information. Given the small size of our test environments and the fact that the robot does not relocate more than a few times throughout the search, the localization error is negligible.

5.1.4 Test environments

Four scenarios were used to conduct the experiments. Three were office environments furnished with desks, chairs and shelves and the forth was an outdoor terrain simulation of ground and rocks. Figures 5.1 and 5.2 show the images of the environments along with their top views.

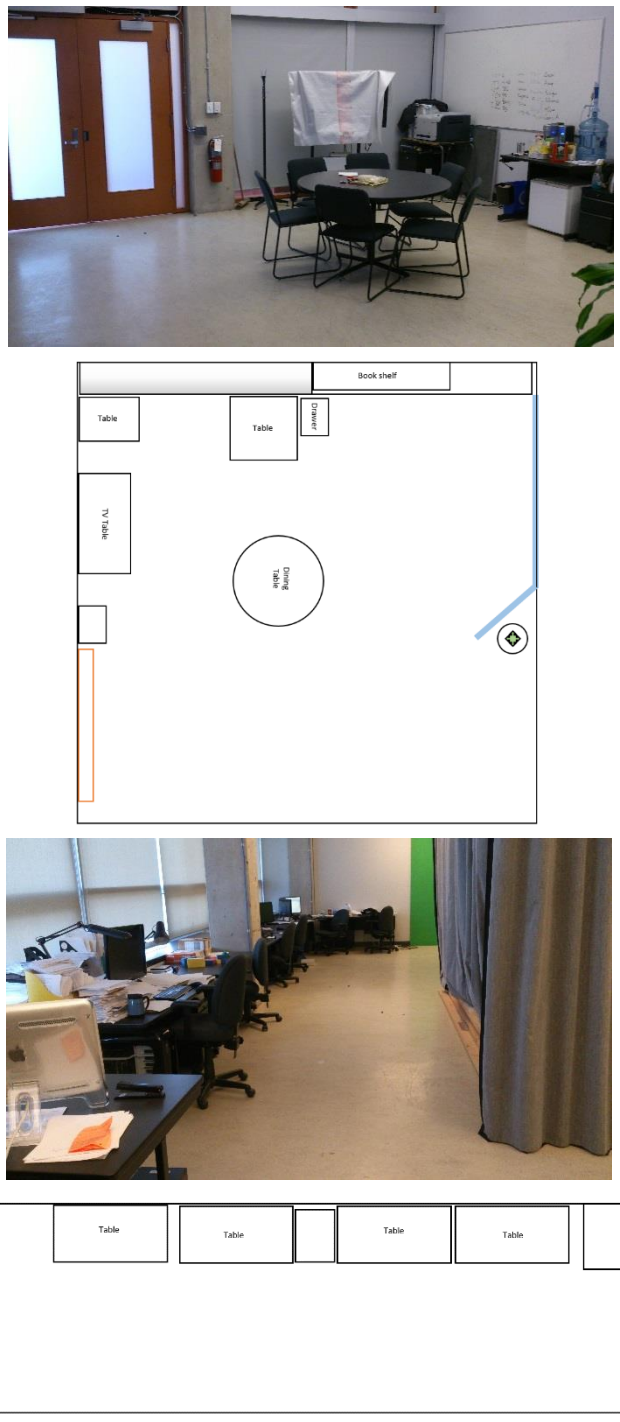
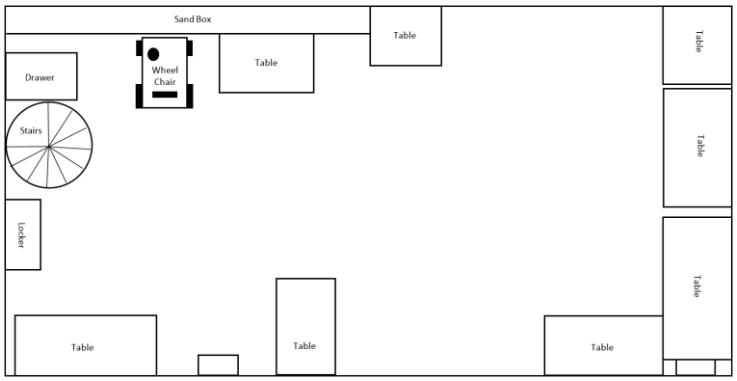
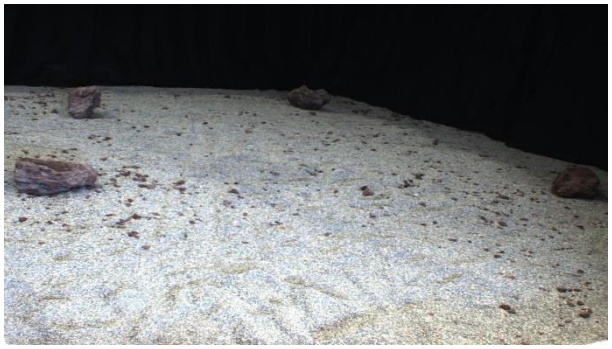


Figure 5.1: The environments where the experiments were conducted. The dimensions of the environments from the top to bottom are, 6.23 x 6.20 m and 2.8 x 11.5.





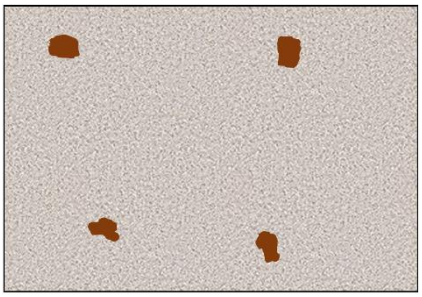


Figure 5.2: The environments where the experiments were conducted. The dimensions of the environments from the top to bottom are, 4.73 x 9.30 m and 5.50 x 3.80 m.

Figure 5.3: The robot used in the experiments.

5.1.5 Hardware

The search agent was implemented on a Pioneer 3 mobile robot (Figure 5.3) with four-wheel differential–drive. The robot was equipped with a Point Grey Bumblebee 2 stereo camera mounted on a Directed Perception pan-tilt unit. The robot contains an on-board computer with a Core Duo Intel CPU and 1GB RAM responsible for controlling the motors, capturing and transmitting images. The rest of the computation is handled by an off-board PC with a 2.67 GHz, 12 Core Intel Xeon CPU, 24 GB RAM and a Tesla C2050 graphic card.

5.1.6 Search parameters

The search environments were discretized into voxels of size $50\ mm^3$ each, and the target's probability values were represented for each. The maximum height to search was set to one meter and the interior configurations of the environments were unknown to the robot (e.g. layout of tables, chairs, etc.). Thus, a uniform probability distribution for target presence was considered for each environment given by $p = \frac{1}{totalNumberOfVoxels}$.

The pan and tilt angles of the camera were limited to $(-158^o, 158^o)$ and $(-20^o, 30^o)$ respectively. A subset of pan and tilt angles was used comprising a total of 142 possible directions. Threshold Θ_{move} was empirically set and remained constant for all the experiments.

The AIM saliency maps were generated using a kernel size of $21\ x\ 21\ pixels$ with 25 ICA features, trained over a large number of natural and indoor samples. A percentile value of 80% was empirically set to threshold the AIM saliency results. To minimize the computation time of AIM, it was implemented on the GPU, reducing the processing time from approx. 15 seconds on a fully parallelized code running on the CPU to less than 0.8 seconds.

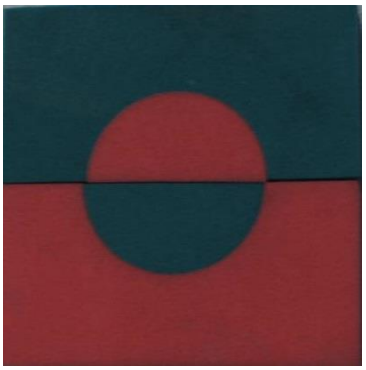


Figure 5.4: The object used in the experiments.

The HB algorithm was applied to pixelwise normalized images. An index size of 32 was used to create the 3D histograms of the images with a threshold value set empirically to minimize the number of outliers.

Figure 5-4 shows the object used in our experiments. This target is chosen because the detection algorithm in 5.1.2 shows a robust performance in recognizing this target in various lighting and view angles. In these experiments, the goal is to evaluate sensor planning strategies. Hence, having a robust algorithm to recognize an object is necessary to highlight the actual performance of each search method regardless of the detection errors that might be introduced by the recognition algorithm. We will speculate on the impact of an object characteristics on saliency results later in this chapter.

5.1.7 Experiments

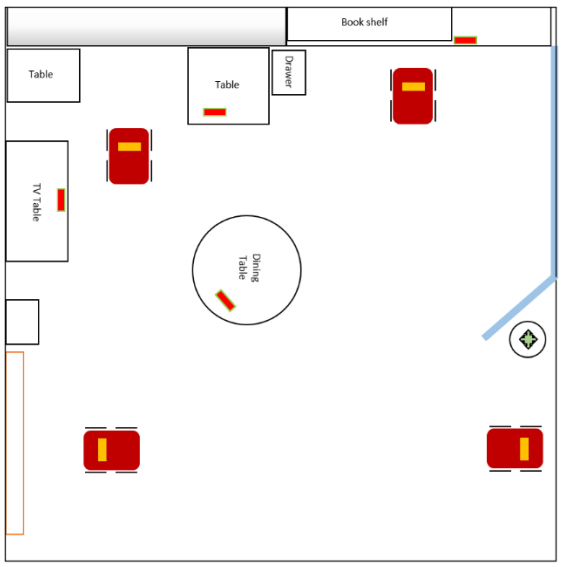
The primary objective of these experiments was to measure how much improvement can be achieved by the application of saliency to object search. Hence, we conducted

two sets of experiments, one with and one without the use of saliency. The performance of each method was measured in terms of the number of actions performed to detect the target, the time of search and the total distance travelled by the robot. In the remainder of this chapter, we refer to the search with saliency as "S_{sal}" and the search with no saliency as "S". It is important to note that except the use of saliency, S_{sal} and S are identical in every aspect.

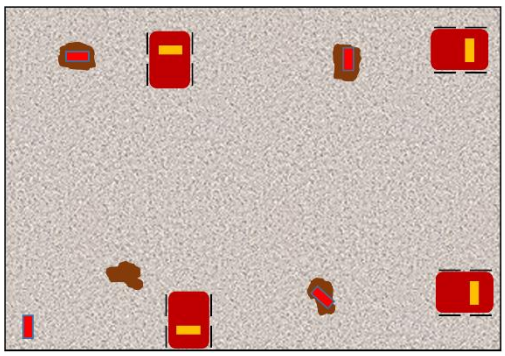
A total of 126 experiments were conducted by placing the robot and the target (Figure 5.4) in various locations. Figure 5.5 illustrates these combinations in each environment. The red and yellow rectangles represent the robot and the smaller red rectangle is the object of interest.

5.1.7.1 Search with no saliency (S)

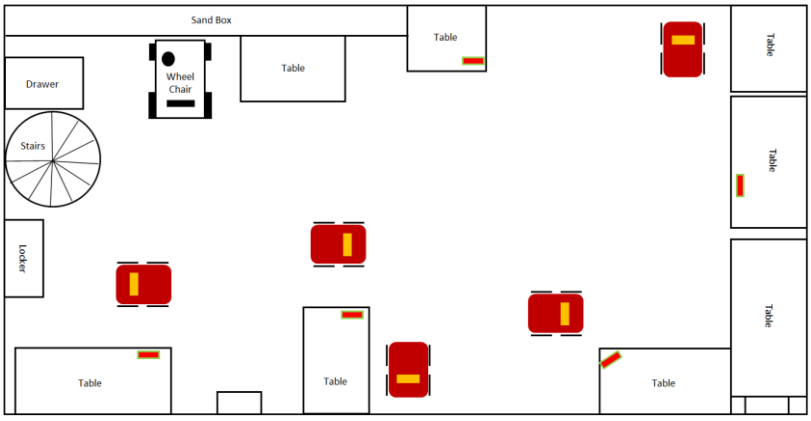
In this subsection, we describe an instance of S in environment 5.5c. Figures 5.6-10 illustrate the entire process of the search with the image captured through the camera, a 2D representation of the probability distribution map and the top view of the environment. In the probability maps, the black colored regions refer to areas searched by the robot, the green spots the obstacles and the grey background the probability of the target's location.



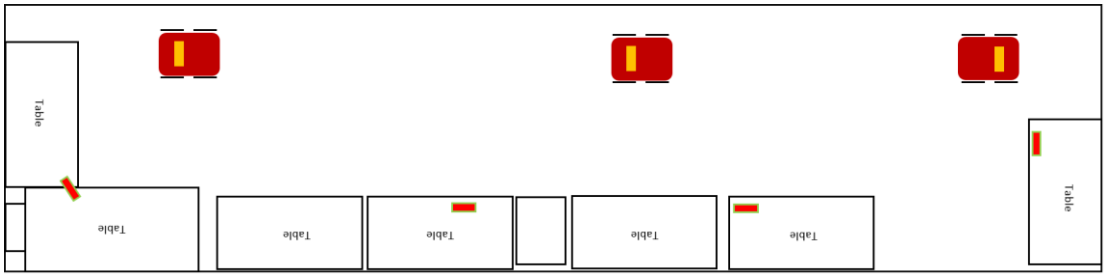
a)



d)



b)



c)

Figure 5.5: The placement of the robot (red-yellow rectangles) and target (small-red rectangles) in each environment. a-c) Refer to the office environments and d) the outdoor terrain environment.

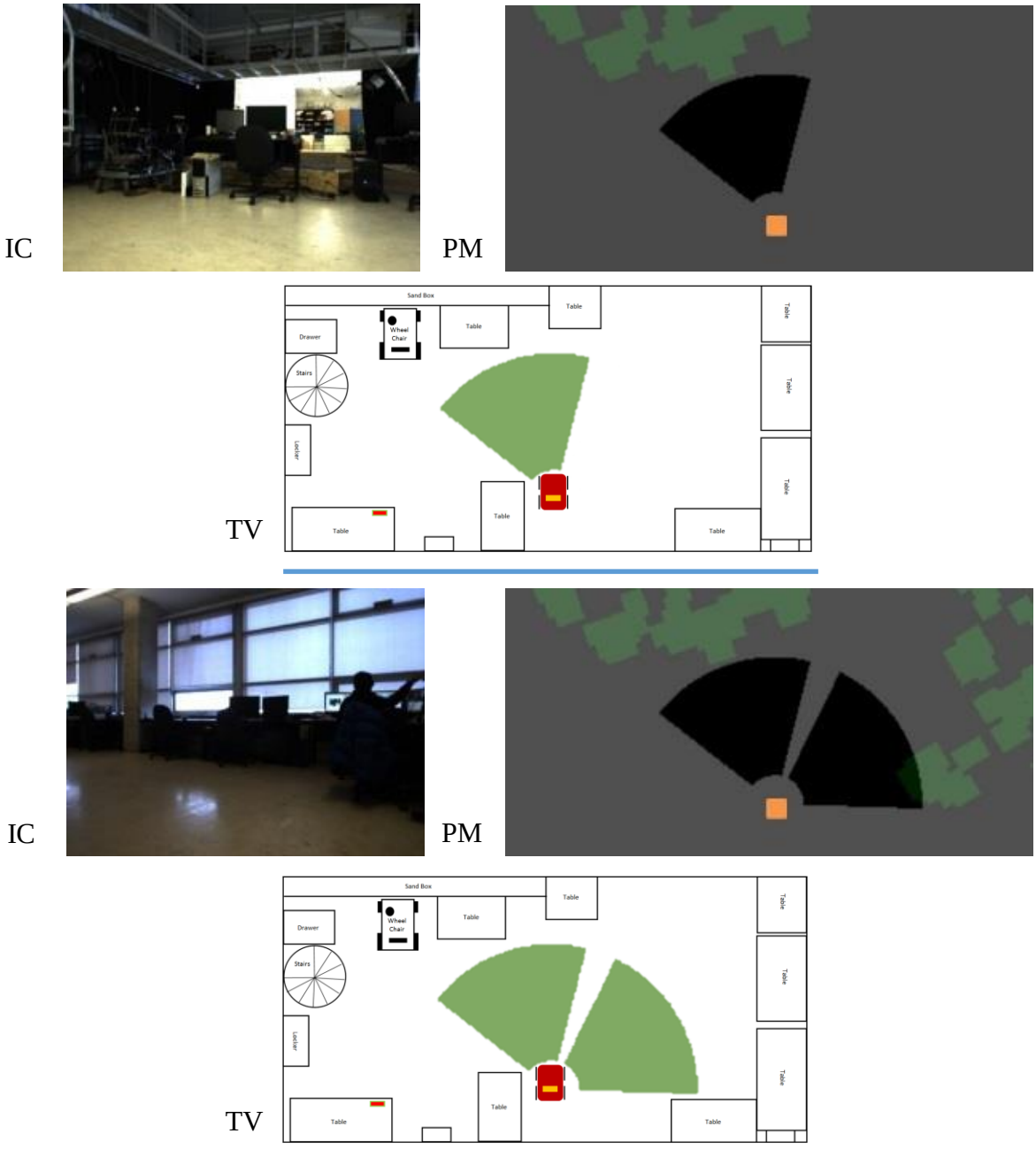


Figure 5.6: The search using *s*. Images are, the image captured by the camera (IC), a 2D representation of the probability distribution map (PM) and the top view of the environment (TV). The robot is at its initial position, searching the first two directions (pan, tilt) (20, 0) and (-60, 0) respectively.

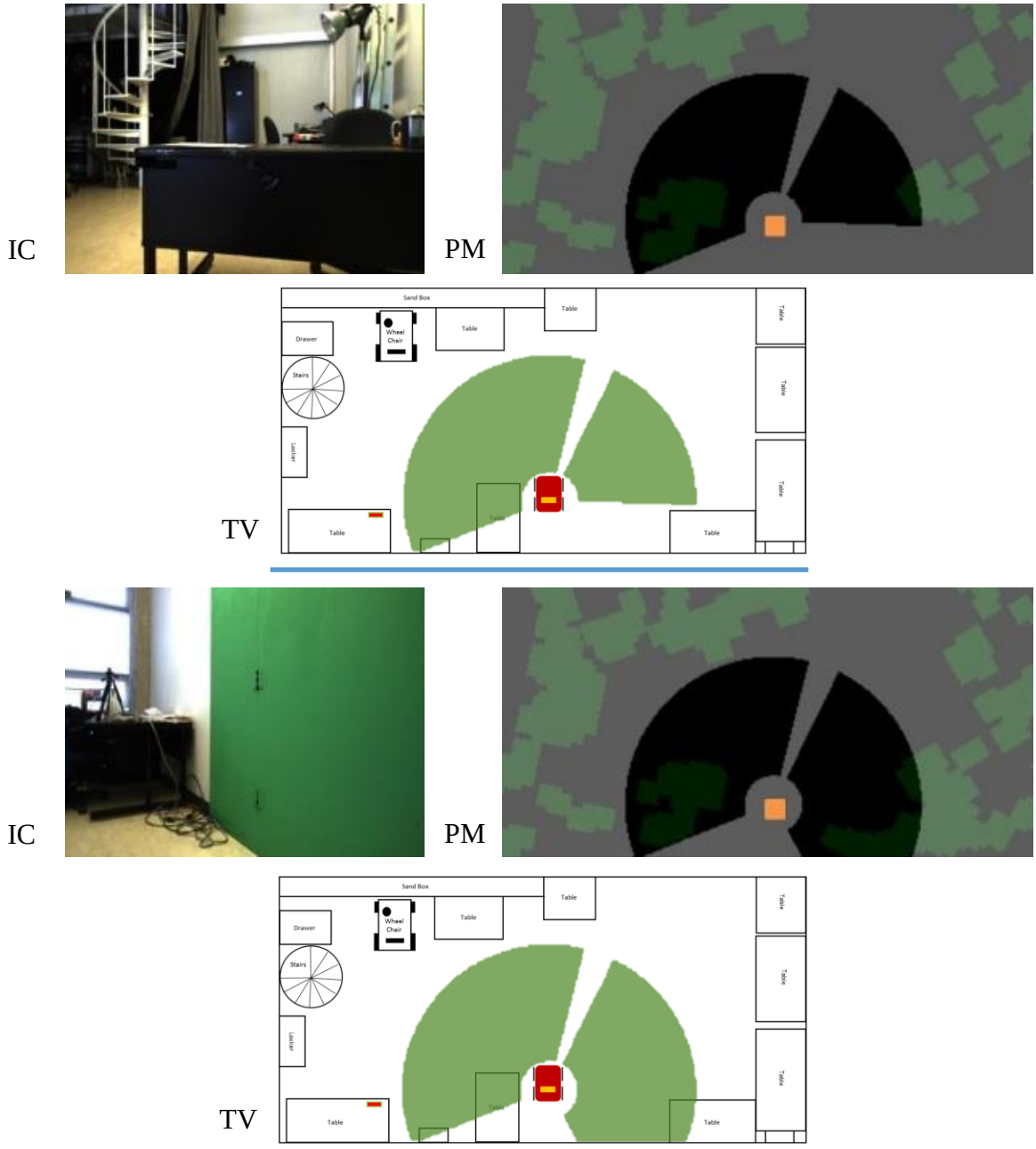


Figure 5.7: The search using *s*. Images are, the image captured by the camera (IC), a 2D representation of the probability distribution map (PM) and the top view of the environment (TV). The robot is at its initial position, searching the third and fourth directions (pan, tilt) (80, 0) and (-120, 0) respectively.

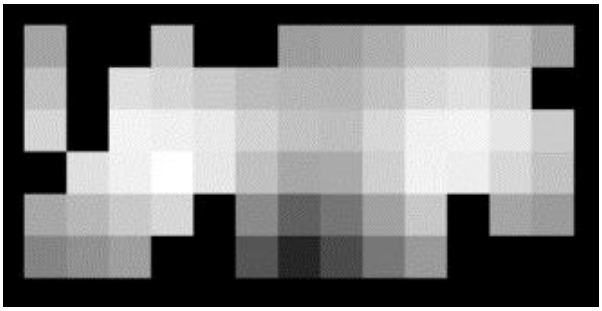


Figure 5.8: The grid indicating the possible locations for the robot to move to. The center of each cell is a potential destination and the intensity refers to the sum of the probabilities of all directions observable from that location. The regions with the color black are those falling over obstacles, therefore not reachable by the robot.

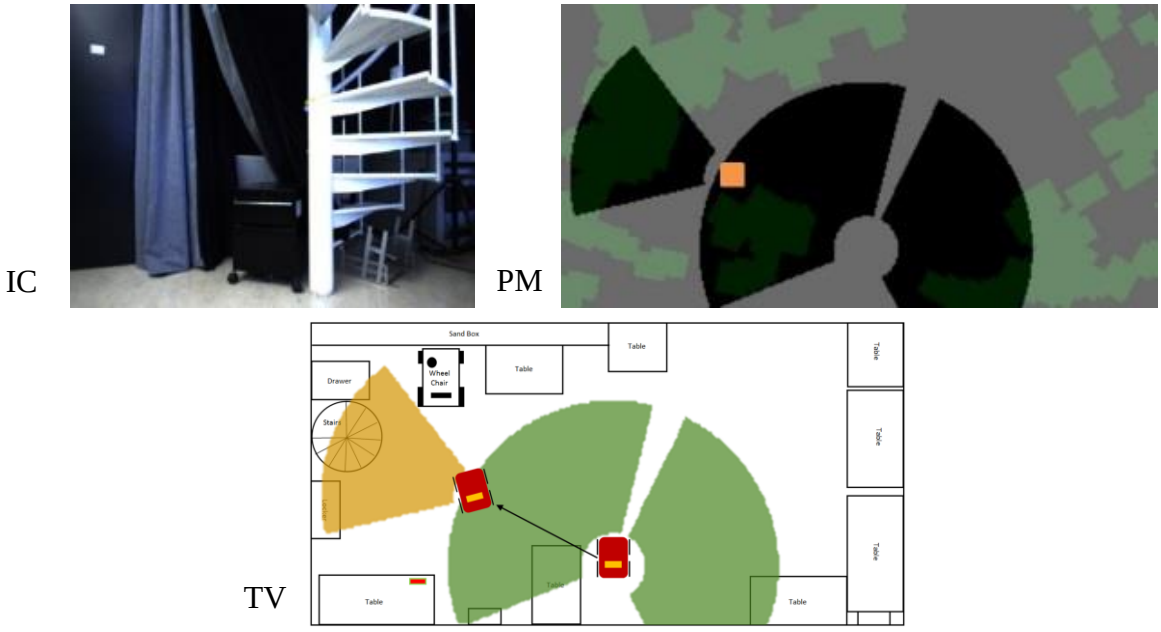


Figure 5.9: The search using *s*. Images are, the image captured by the camera (IC), a 2D representation of the probability distribution map (PM) and the top view of the environment (TV). The robot moves to the second position, and looks toward the fifth direction (pan, tilt) (20, 0).

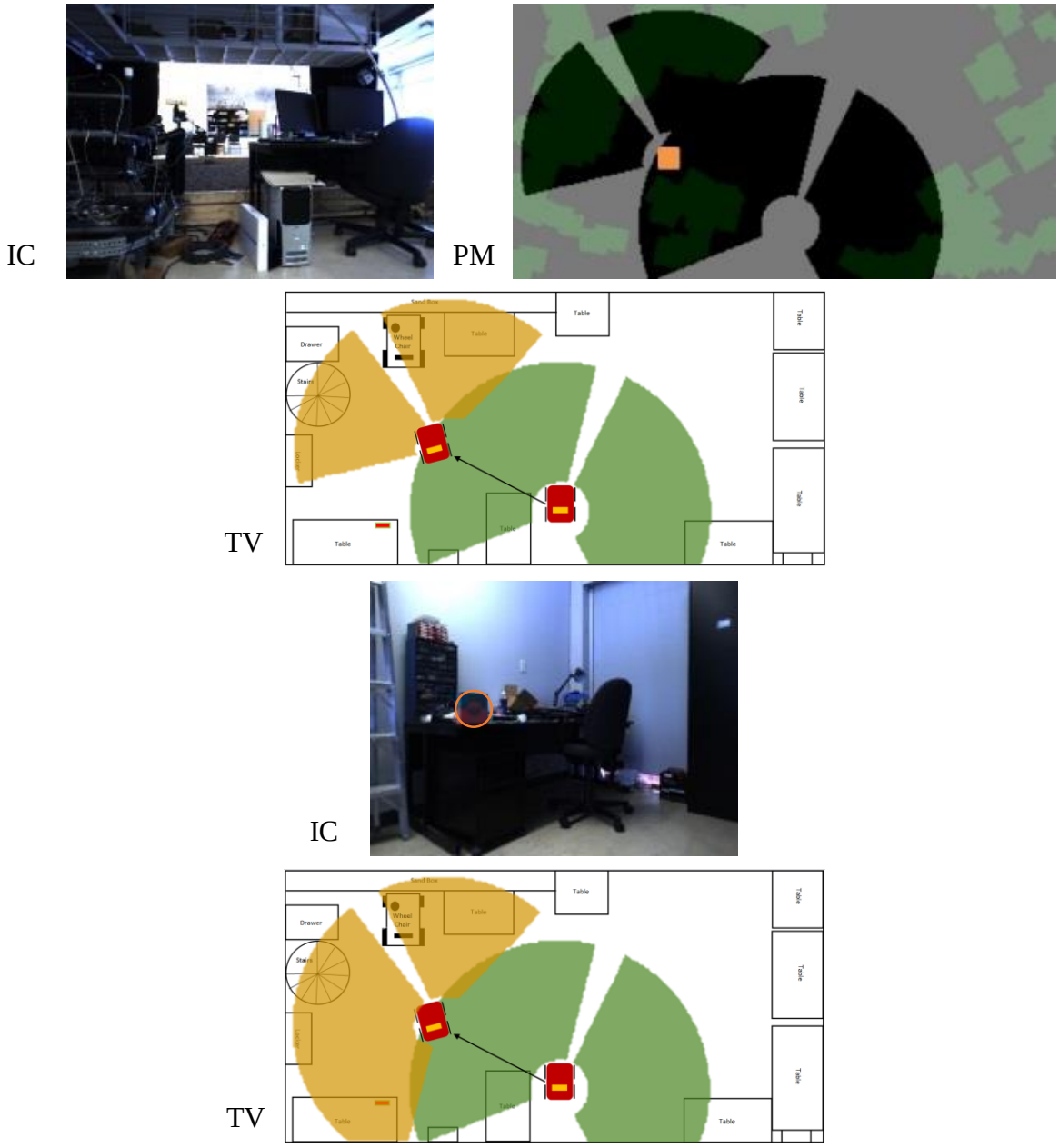


Figure 5.10: The search using *s*. Images are, the image captured by the camera (IC), a 2D representation of the probability distribution map (PM) and the top view of the environment (TV). The last two observations by the robot (pan, tilt) (-60, 0) and (80, 0). The target is found after performing the 7th action.

The search begins by calculating the probabilities of all possible directions by summing up the voxels' probability values within the effective field of view. Then, an image is captured and processed, which fails to detect the target. Therefore, the probability of the space within the effective field of view is lowered to zero and redistributed to the rest of the environment.

Three more directions are selected by the robot from its current location all of which fail to detect the target. At this point, the probabilities of the remaining actions fall below threshold Θ_{move}, forcing the robot to select a new position to explore. To do so, the environment is divided into potential locations forming a grid. Each grid cell (location) is the size of the robot and its value is determined by the sum of the probabilities of all directions observable from that location (see Figure 5.8). The robot chooses the location that yields the highest probability value and moves to its center. After arriving at the new position, the robot performs three more observations and detects the target after performing the 7th operation.

5.1.7.2 Search with saliency (S_{sal})

Figures 5.11-14 show the application of S_{sal} in practice with the same configuration as the above. The figures are represented as before with an additional illustration indicating the saliency responses in the environment. Moreover, the 2D probability maps are slightly different. They contain lighter grey spots pinpointing the location of saliency observed by the robot. These locations are estimated using the depth

information captured through the stereo camera. The intensity of these locations correspond to the strength of saliency observed by the robot, hence, their probability values are increased accordingly.

The search strategy is similar to S in which the direction with the maximum probability is selected and an image of the corresponding direction is taken. However, after failing to detect the target, a saliency map is developed to identify interest points beyond the effective range of the recognition camera. As it can be seen, there are high saliency responses over regions occupied by tables. Consequently, the probability of those regions are increased.

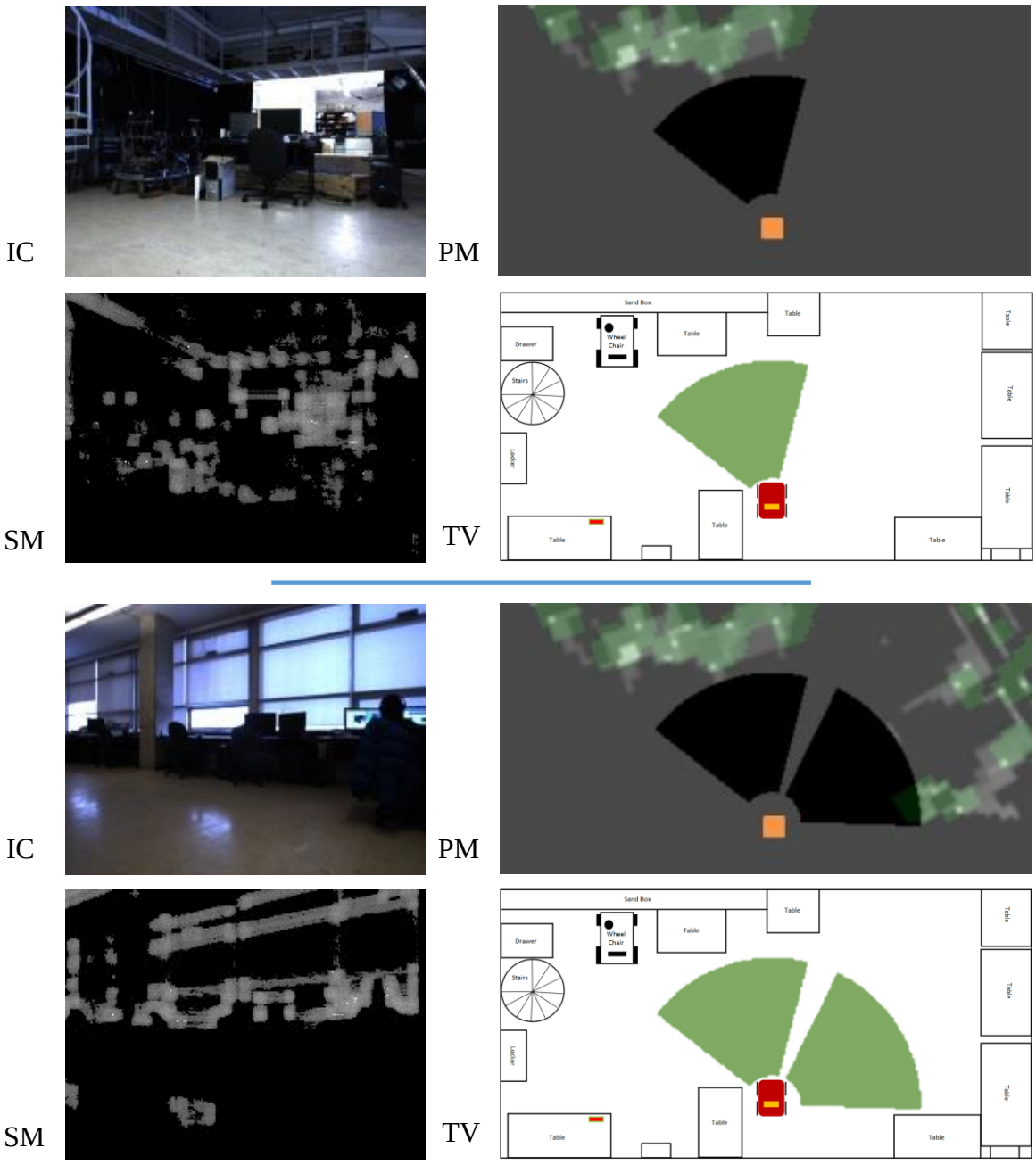


Figure 5.11: The search using S_{sal}. Images are, the image captured by the camera (IC), a 2D representation of the probability distribution map (PM), the saliency map of the image (SM) and the top view of the environment (TV). The robot is at its initial position, searching the first two directions (pan, tilt) (20, 0) and (-60, 0) respectively.

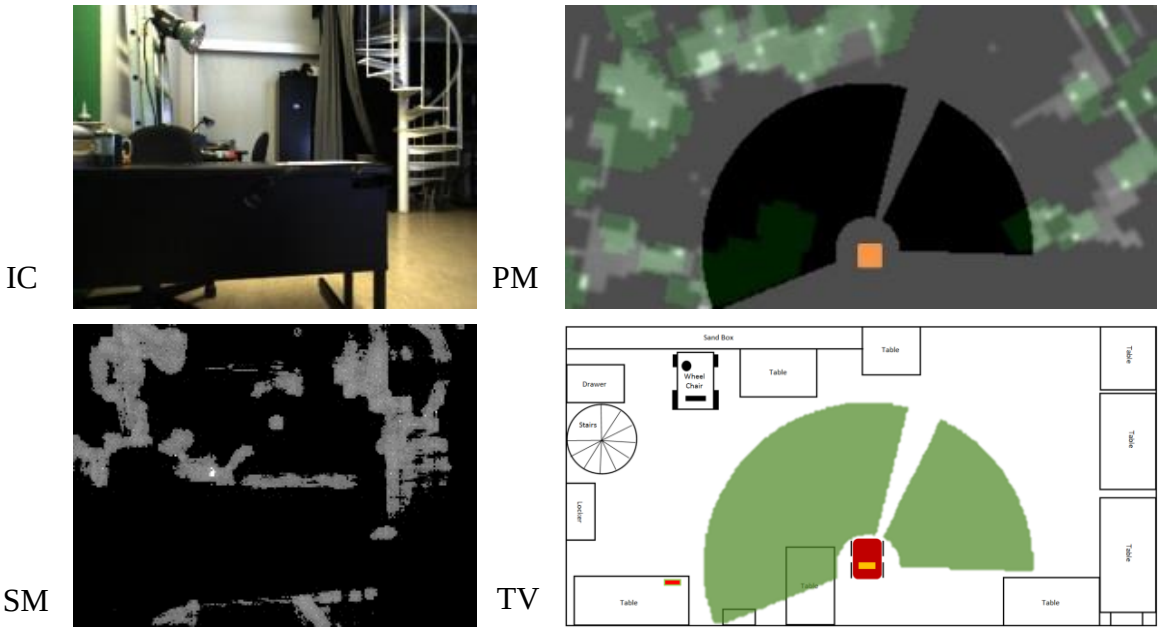


Figure 5.12: The search using S_{sal}. Images are, the image captured by the camera (IC), a 2D representation of the probability distribution map (PM), the saliency map of the image (SM) and the top view of the environment (TV). The robot is at its initial position, concluding its search at this point by looking toward direction (pan, tilt) (80, 0).

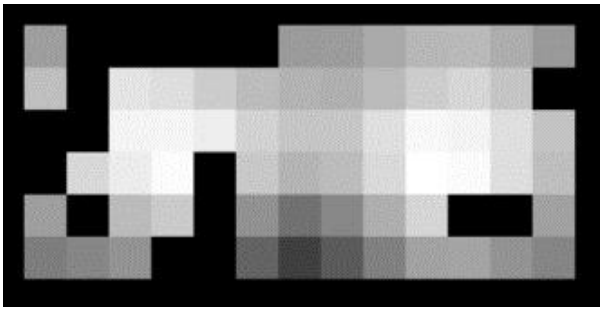


Figure 5.13: The grid indicating the possible locations for the robot to move to. The center of each cell is a potential destination and the intensity refers to the sum of the probabilities of all directions visible from that location. The regions with the color black are those falling over obstacles, therefore not reachable by the robot.

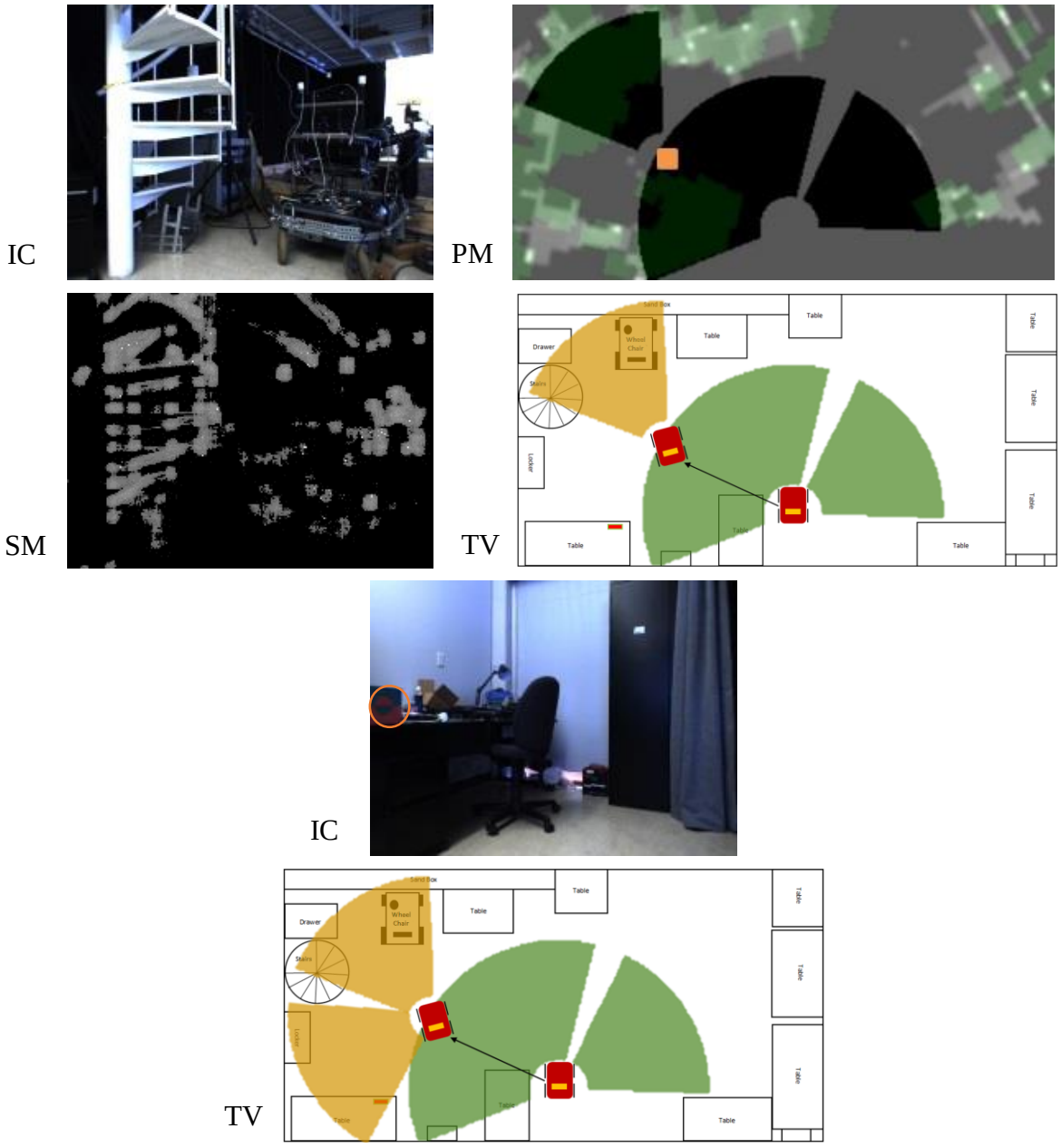


Figure 5.14: The search using S_{sal}. Images are, the image captured by the camera (IC), a 2D representation of the probability distribution map (PM), the saliency map of the image (SM) and the top view of the environment (TV). The robot is at the second position, where it completes the search by performing two actions (pan, tilt) (-40, 0) and (40, 0) respectively.

In this example, due to the high saliency responses in the distance, threshold Θ_{move} is reached after performing only three actions (as opposed to four actions in S). As a result, the robot moves to the next location and resumes the search from there. Relying on the saliency responses to select directions, the robot found the target with only two more operations, giving the S_{sal} method the advantage to detect the object with two fewer actions than S.

The AIM algorithm, as explained in Section 3.2, generates the saliency responses of each local neighbourhood with respect to its surrounding areas. This means depending on the distance of the camera from a scene or the direction of view with respect to the scene, different saliency responses would emerge. In practice, this is a common scenario for the robot to see a region multiple times from different angles or distances. To address this issue, the saliency responses of the locations are averaged, if they are observed more than once.

5.1.8 Quantitative results

In this section, we divide the results of the experiments into two groups of "Move" (M) and "No Move" (NM). The M group consists of the search scenarios in which the robot relocated at least once to detect the target. In such cases the object was placed far away from the initial location of the robot and was not detectable by the recognition algorithm. In the NM scenarios, the object was located within the effective range of the robot, hence, was found from its initial location. The reason behind this decision is that

the methods performed similarly in the NM situations, of course, with some minor disadvantage for S_{sal} in terms of the processing time of saliency. In each search strategy, the robot initially searches its surroundings before moving to a new location. Because the initial area is not seen beforehand, it has a uniform probability distribution (i.e. with no saliency information), which means similar directions are chosen regardless of the method of choice.

Table 5.2 summarizes the average outcomes of the experiments in each environment. In this table, the results are presented in two groups of S and S_{sal} each expressed in terms of the number of actions taken to complete the search, the duration of the search and the distance travelled by the robot to conclude the search.

Table 5.1: The results of the experiments conducted in the test environments. The results are expressed as the average performance in each category.

Office Location a							
S	NM	M	Overall	S_{sal}	NM	M	Overall
No. of Actions	1.7	11.61	9.75	No. of Actions	1.7	8.69	7.37
Duration of Search (s)	72.78	994.42	821.61	Duration of Search (s)	73.54	614.77	511.85
Distance Travelled (m)	0	16	13	Distance Travelled (m)	0	7.6	6.2
Office Location b							
S	NM	M	Overall	S_{sal}	NM	M	Overall
No. of Actions	2.8	10.69	7.84	No. of Actions	2.8	9.13	6.84
Duration of Search (s)	121.19	628.93	480.7	Duration of Search (s)	123.78	594.23	424.49
Distance Travelled (m)	0	8.3	5.3	Distance Travelled (m)	0	6.5	4.2
Office Location c							
S	NM	M	Overall	S_{sal}	NM	M	Overall
No. of Actions	1.7	9.2	7.3	No. of Actions	1.7	7.7	6.17

Duration of Search (s)	73.29	647.95	504.29	Duration of Search (s)	75.03	568.77	442.56
Distance Travelled (m)	0	11.3	8.5	Distance Travelled (m)	0	9.3	7
Mars Simulated Environment							
S	NM	M	Overall	S_{sal}	NM	M	Overall
No. of Actions	N/A	7.22	7.22	No. of Actions	N/A	5.9	5.9
Duration of Search (s)	N/A	402.22	402.22	Duration of Search (s)	N/A	332.78	332.78
Distance Travelled (m)	N/A	3.5	3.5	Distance Travelled (m)	N/A	3.28	3.28
Total							
S	NM	M	Overall	S_{sal}	NM	M	Overall
Avg. No. of Actions	2.06	9.68	8.03	No. of Actions	2.06	7.83	6.57
Duration of Search (s)	89.08	681.88	552.205	Duration of Search (s)	90.78	527.64	427.92
Distance Travelled (m)	0	9.76	7.56	Distance Travelled (m)	0	6.57	5.17

It is apparent that, both methods performed similarly in cases where the object was found from the initial location of the robot. However, the performance gap increased in favor of S_{sal} when the robot moved at least once to detect the target. As can be seen, on average the S_{sal} algorithm improved the search in the cases of Move by approximately 2 actions, 154 seconds and 3.2 meters travel distance.

A comparison between the methods is conducted in Table 5.3 to illustrate the percentage each method performed better in terms of the number of actions taken to conclude the search. Note that for the reasons mentioned earlier, only the cases of Move are considered in this evaluation.

Table 5.2: A comparison between S and S_{sal} methods in terms of the number of action taken to complete the search.

Method performed better	Office a	Office b	Office c	Mars	Total
S_{sal}	76.92%	68.75%	77.77%	55.55%	69.75%
S	7.69%	18.75%	11.11%	11.11%	12.17%
Similar performance	15.38%	12.5%	11.11%	33.33%	18.08%

The least performance improvement was achieved in the Mars environment, where the ratio of the environment size to the effective field of view is at the lowest. In this case, such performance is expected because there is a higher chance that S_{sal} selects similar operations as S.

The effectiveness of saliency also reduces as the number of distracters increases within the environment. For instance, office b is populated with a large amount of furniture, therefore it yielded a lower performance improvement rate comparing to the other office environments.

5.1.9 Effectiveness of saliency in search

In the experiments presented earlier, the basic assumption was that in a typical environment objects are more likely placed on surfaces such as tabletops which are possibly to be observed as salient from an agent point of view. However, if this is not the case, saliency clues can play an opposite role in visual search. Instead of guiding a

robot to the object of interest, they would distract the attention of the robot to regions away from the actual place of the target.

Moreover, in our experiments we only used one object due to the complication of developing a robust recognition algorithm. It is important to note that target characteristics play an important role in the efficiency of saliency information. The less distinctive the features of an object be, the less salient the object is perceived by an agent. This, in particular, is true for the top-down saliency model if the target's color is similar to its surroundings. The bottom-up saliency, however, is less affected. It still can be effective even though the object of interest does not stand out clearly from its surroundings. Given the main purpose of using bottom-up saliency is to produce indirect search clues, regardless of the characteristics of the target, the salient locations that are in spatial relation to the target still can be detected.

5.2 Sensor planning experiments

In this section, the sensor planning strategies with a predefined search constraint are evaluated. For this purpose, we used three different cost functions specifically the total time for the search, battery consumed by the system and the overall distance travelled by the robot. The same hardware and search parameters are used as before unless otherwise mentioned.

5.2.1 Operation cost calculation

The cost of each action is calculated based on the following components involved in the search: the cost of the robot to move to a new location, the pan and tilt angle changes, and the costs associated with the image processing and environment mapping. The last two components are constant for every given operation because the processing of each image is identical. The pan and tilt angle changes are calculated by measuring the difference between an operation's direction and the pan and tilt angles of the last action performed by the robot.

Perhaps, the most challenging aspect of the cost calculation is the estimation of the distance travelled by the robot. This is mainly due to the fact that the environment is either unknown or partially known by the robot at the time of selecting actions. In addition, the performance of stereo cameras is limited to detect disparity. This is typically limited to some ranges above which the estimation error increases significantly. Therefore, we consider an uncertainty cost to be added to estimated distances above the reliable range of our stereo camera. This means operations in far distances are even less likely to be selected by the algorithm due to a higher rate of ambiguity.

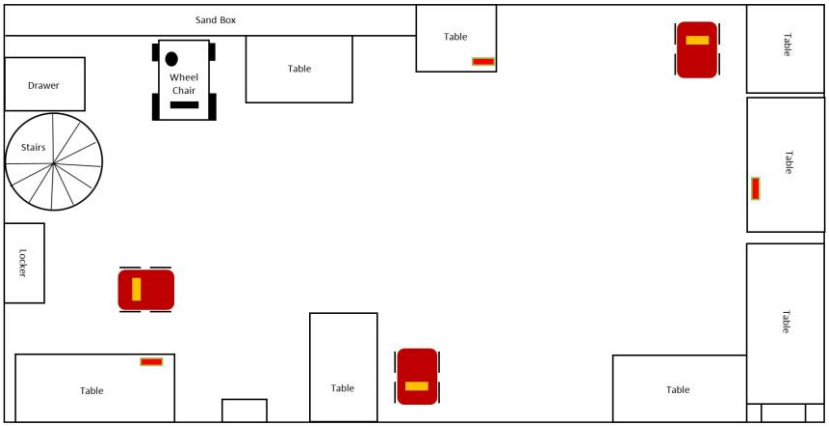
5.2.2 Experiments

Due to a larger volume of experiments, from the test environments shown in Figures 5.1-2, only the office locations were chosen with a sparser number of configurations as

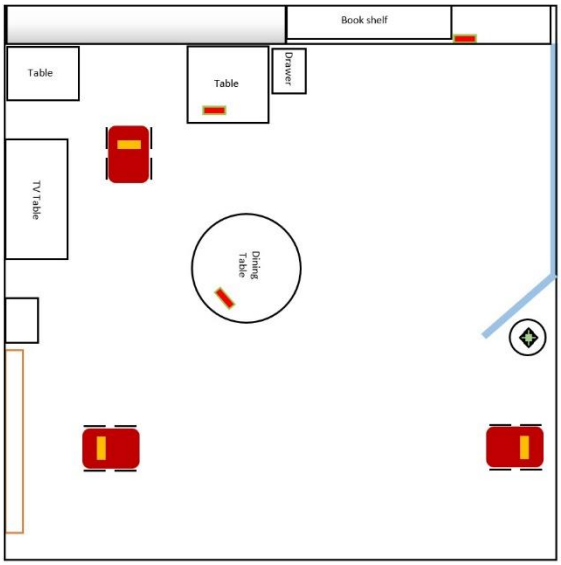
shown in Figure 5.15. The environments were discretized by dividing each into the voxels of size 100 mm^3. Each operation was defined in terms of the view direction and location of the robot. In these experiments, only 11 directions were considered by fixing the angle of the tilt and dividing the pan angles into 11 discrete portions. As for the possible locations of the robot, the environments were divided into the cells of size equal to the radius of the robot.

In the previous series of experiments, we established the benefit of saliency in visual search. Therefore, in the following experiments the saliency model is used as default in all algorithms. In the case of GSC algorithm, each time the algorithm selects an action it takes into account the saliency information. In the other two planning methods, if an action sequence is generated some time during the search, the saliency information is used.

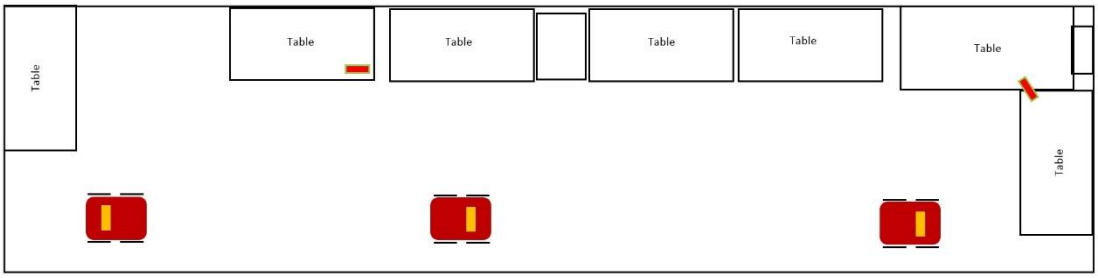
The cost constraint limits in each location were selected according to the average performance of the algorithms in the previous experiments. The reason for this decision is to make sure at least in some instances the target is found. This gives us a better insight on performance of each planning method. Table 5.4 shows the cost constraints of the search in each environment. Note that these parameters were the same for all the



a)



b)



c)

Figure 5.15: The robot and target configurations in each environment.

Table 5.3: The search constraints in each environment.

	Time (s)	Energy (kJ)	Distance (m)	Action Count
Office a	700	67.9	6	9
Office b	640	62.4	7	8
Office c	500	48.5	6.5	7

strategies with the exception of the maximum action sequence size, which was only applied to DLAS. In addition, the α value of the GSC algorithm (the value after which the algorithm selects actions based on their probability values) was set to 10%. It is worth mentioning that for evaluation purposes, the cost constraints can be set to any desirable value.

A total of 216 experiments were conducted using the strategies mentioned in Chapter 4, namely GSC, EGS and DLAS. Each method was tested in practice by applying different cost functions including time, energy consumption and distance. The following sub-sections demonstrate examples of each sensor planning technique using different types of cost constraints. The target in Figure 5-4 was used in these experiments.

5.2.2.1 Distance

Figures 5.16-17 show an instance of the GSC algorithm in office b, where the robot is positioned at the bottom left of the room and the target at the top right. In this example,

the cost of each action is calculated based on the distance traveled. Hence, at the initial location of the robot, the cost is equal to zero. This encourages the robot to fully search its initial position before considering a new one.

Once the current location is fully explored, the robot chooses its next destination toward the top due to the high saliency responses of this area. Despite this, the robot conservatively moves upward for only a short distance. Such behaviour is expected for two reasons: first, the tendency of the greedy algorithm to locally select an action, and second, larger motion increase the uncertainty and cost of the operations.

The robot searches the second location and repeats the same process in another 3 destinations until it detects the target after performing 24 actions. Note that due to the large number of operations, only a few critical stages are demonstrated in the Figures. In this particular scenario, value α is not triggered, therefore the search only behaves greedily.

One particular property of the GSC algorithm is its concentration on the areas of the environment populated with furniture. This increases the chance of detecting the target if it is placed on those furniture. Once again confirms the importance of saliency and how it can effectively guide the search agent to the regions of high importance.

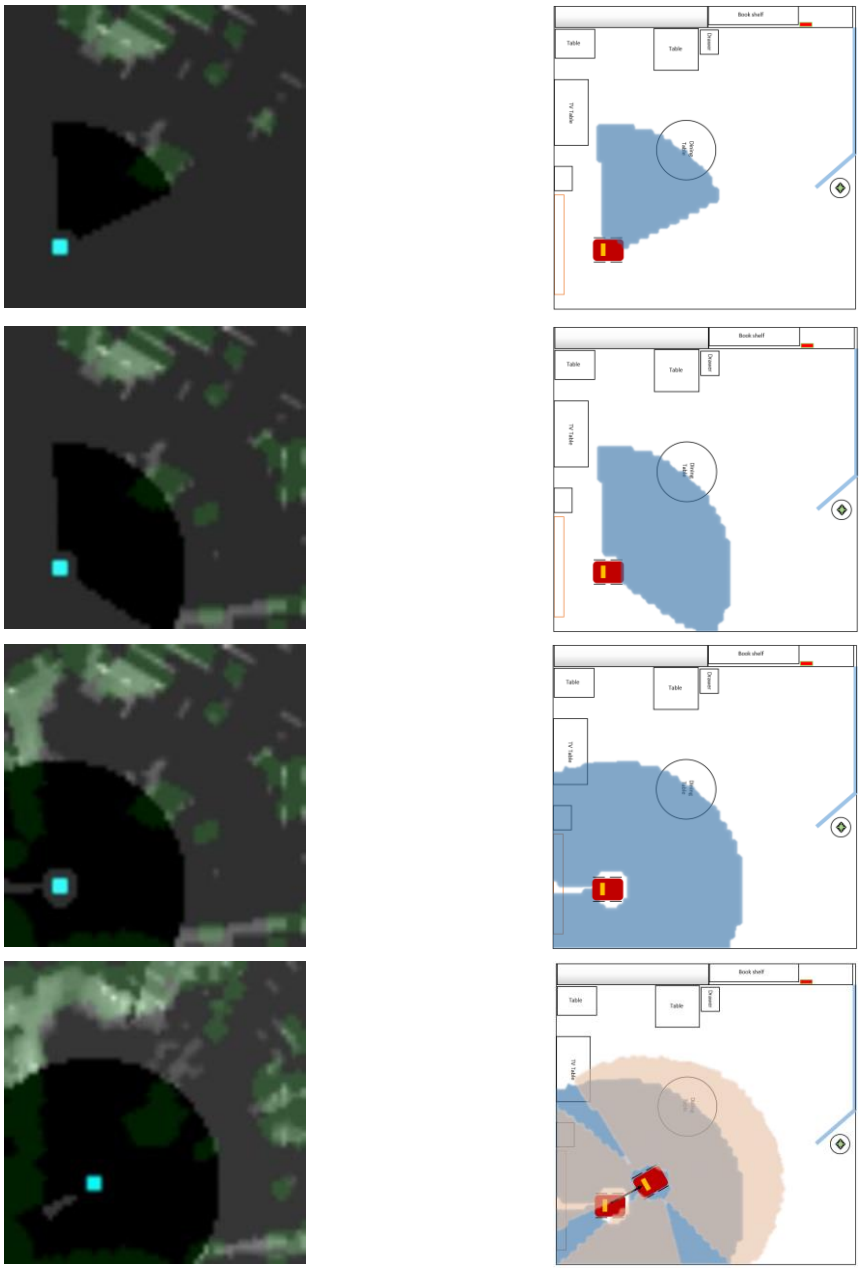


Figure 5.16: The search process using the GSC algorithm with the distance constraint. From the left to right, a 2D representation of the 3D probability distribution map and the top view of the environment. The robot explores two positions.

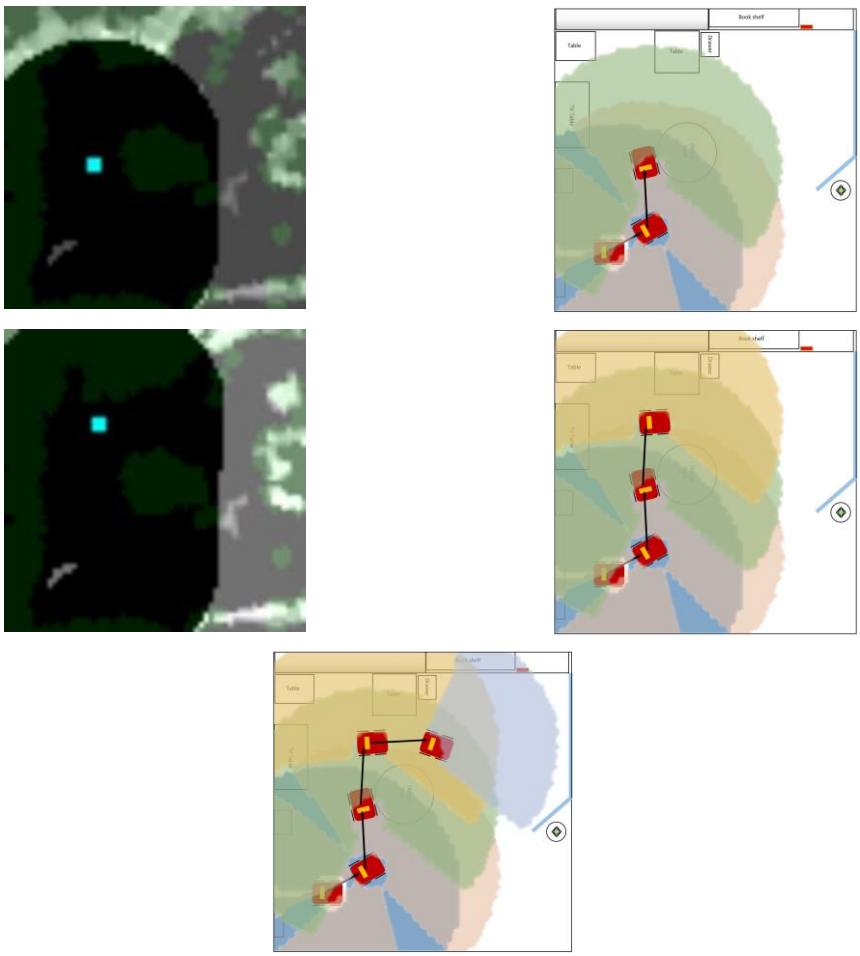


Figure 5.17: The search process using the GSC algorithm with the distance constraint. From the left to right, a 2D representation of the 3D probability distribution map and the top view of the environment. The robot detects the target after exploring two directions from the fifth position.

EGS follows a similar routine to conduct the search as can be seen in Figures 5.18-19. The algorithm first generates a sequence of operations to be performed in the search. Based on that, the robot begins by exploring its surroundings, and performs 7 actions. Since the target is not found, the robot moves to the next location that is located on the right side of the room. Here, the absence of saliency is apparent as the new destination is different from that was chosen in GSC.

After exploring the second and third locations, a new action sequence is produced by EGS because the next destination is unreachable due to its vicinity to an obstacle (the round table). At this point, the saliency clues take effect in the new action sequence, changing the search route toward the regions on the top.

Conforming to the new plan, the robot first inspects one more direction prior to moving to the next destination. Although this direction (shown by the yellow color at the bottom of the image) only covers an insignificant portion of the environment, its utility is still higher than the operations to be performed at the next spot primarily because of the large distance of the new location from the current position of the robot.

The robot concludes the search after performing 25 operations and fails to find the target within the given constraint.

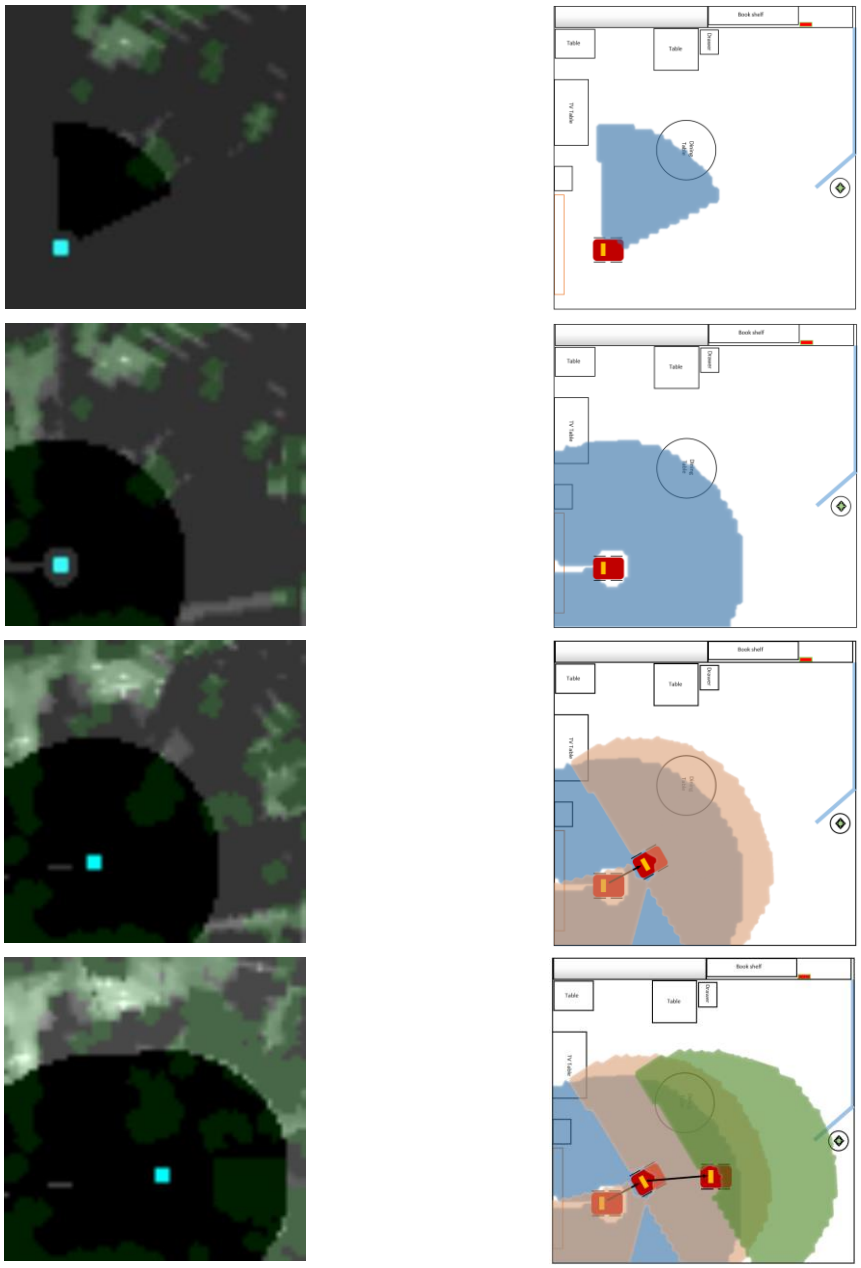


Figure 5.18: The search process using the EGS algorithm with the distance constraint. From the left to right, a 2D representation of the 3D probability distribution map and the top view of the environment. The robot completes searching three locations after performing 14 actions.

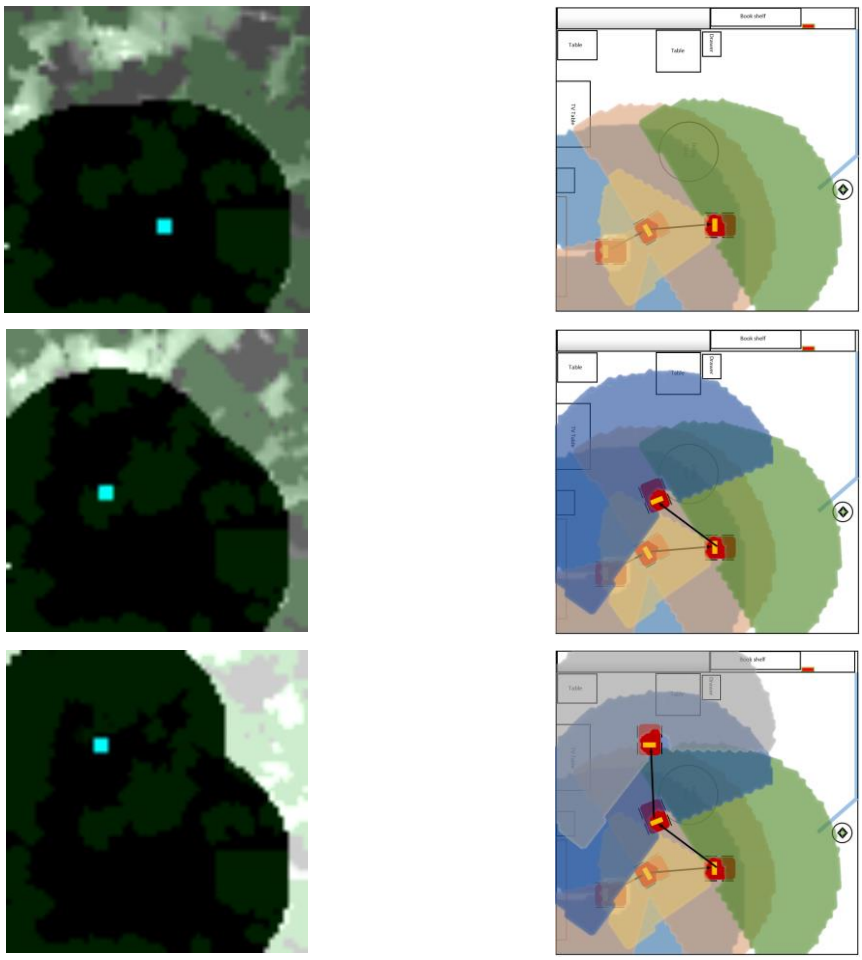


Figure 5.19: The search process using the EGS algorithm with the distance constraint. From the left to right, a 2D representation of the 3D probability distribution map and the top view of the environment. The robot concludes the search and fails to detect the target after exploring 5 locations.

The DLAS algorithm demonstrates a very different behaviour (see Figures 5.20-21). It first generates an action sequence to be performed during the search. Based on that, the robot only looks toward two directions and then moves to a new position. After searching the second location, the robot places itself at the third one and searches only one more direction. Thereafter, it attempts to reach the forth position but it fails to do so because the navigation cost exceeds the overall constraint.

This example reveals two potential problems with DLAS. The first problem arises from limiting the number of operations generated by DLAS (to save processing time as described before). Applying a distance constraint to the search means if the robot searches all the directions from a position, it does not incur any additional cost except the cost of moving to that location. For the same reason in instances of the GSC and EGS algorithms, the robot always fully searches its current location until the probability values of all available actions are zero.

DLAS on the other hand, attempts to maximize the probability by performing a limited number of actions in a given constraint. As a result, it prefers inspecting locations with larger unexplored surroundings. Hence, instead of fully searching its current location it moves to the next one. This is a potential problem for DLAS algorithm because it explores a lower percentage of the environment during the search in comparison to GSC and EGS.

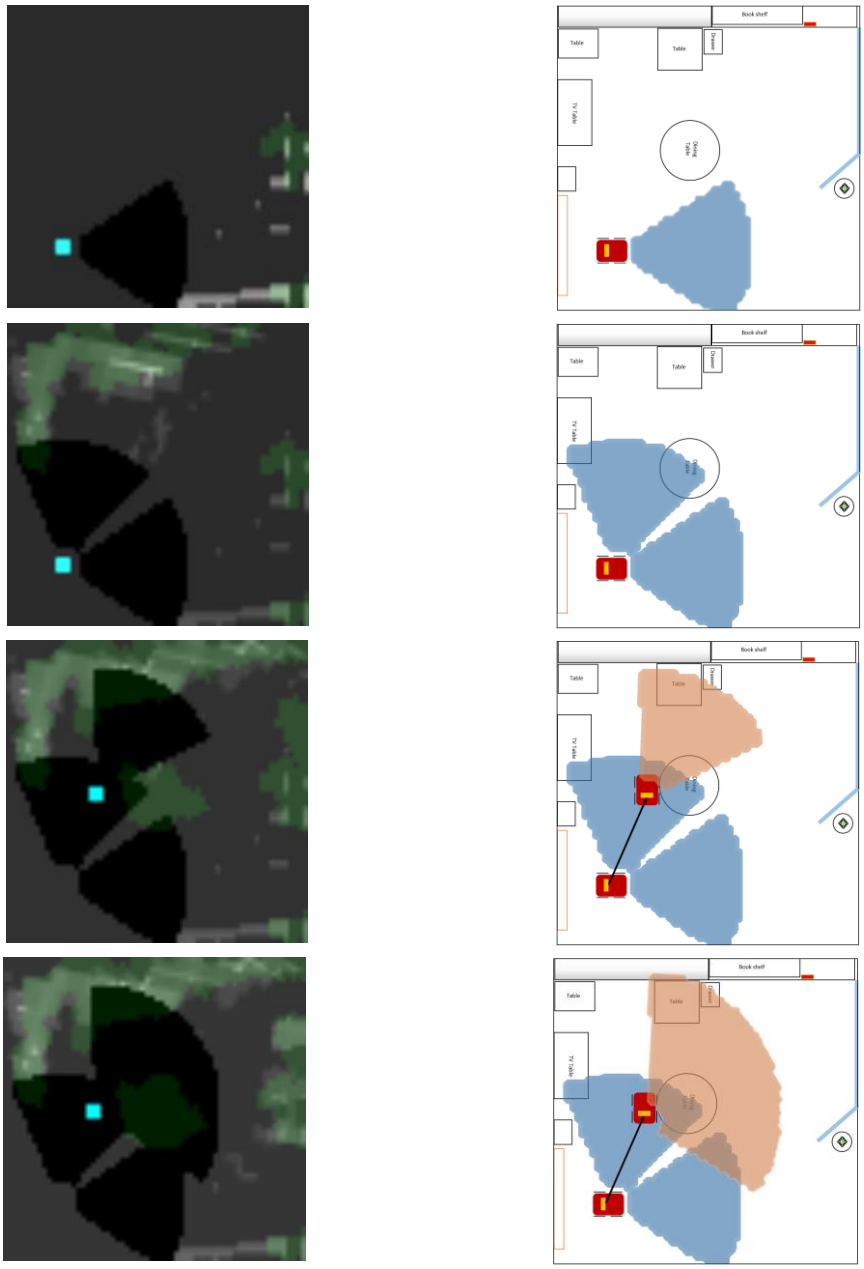


Figure 5.20: The search process using the DLAS algorithm with the distance constraint. From the left to right, a 2D representation of the 3D probability distribution map and the top view of the environment. The robot commences the search by performing the first four operations.

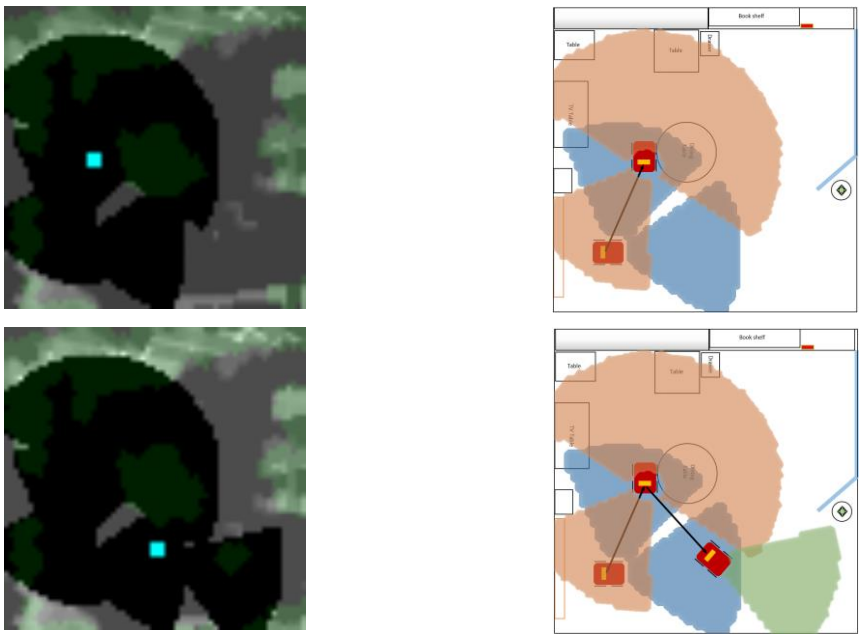


Figure 5.21: The search process using the DLAS algorithm with the distance constraint. From the left to right, a 2D representation of the 3D probability distribution map and the top view of the environment. The robot finishes searching the third location is unable to move any further.

Obviously, if time permits, one can calculate the complete solution using DLAS in which the target could be found. Figure 5.22 shows the 8-action approximate and full approximate solutions of DLAS to the search. The processing time of this solution is 359.2 seconds in comparison to 19.9 seconds for the 8 action approximation.

The second potential issue using DLAS is the difficulty to estimate the cost of each action (in this case distance) prior to the search when there is no information available regarding the environment. In the above example, DLAS produced an action sequence commanding the robot to move to four locations and look toward a total of nine

Figure 5.22: The comparison of the approximate solution using 8 actions on the left and the complete solution on the right using DLAS.

directions. However, from the position 2 to 3, the robot faces an obstacle, therefore it is must to travel a longer distance around the obstacle in order to arrive at the next location. This causes the actual distance travelled to exceed the one forecasted by the algorithm, therefore the robot fails to reach its final destination.

5.2.2.2 Energy consumption

The search examples with energy constraint were conducted in the same environment as before. Figures 5.23-24 illustrate the process of the GCS algorithm to find the target. Once again the search commences at the current location of the robot where it inspects three directions. Then the robot chooses its next destination on the top and continues the search from there.

One may notice that the number of actions performed at the start is significantly fewer than the search with the distance constraint. This can be explained by the fact

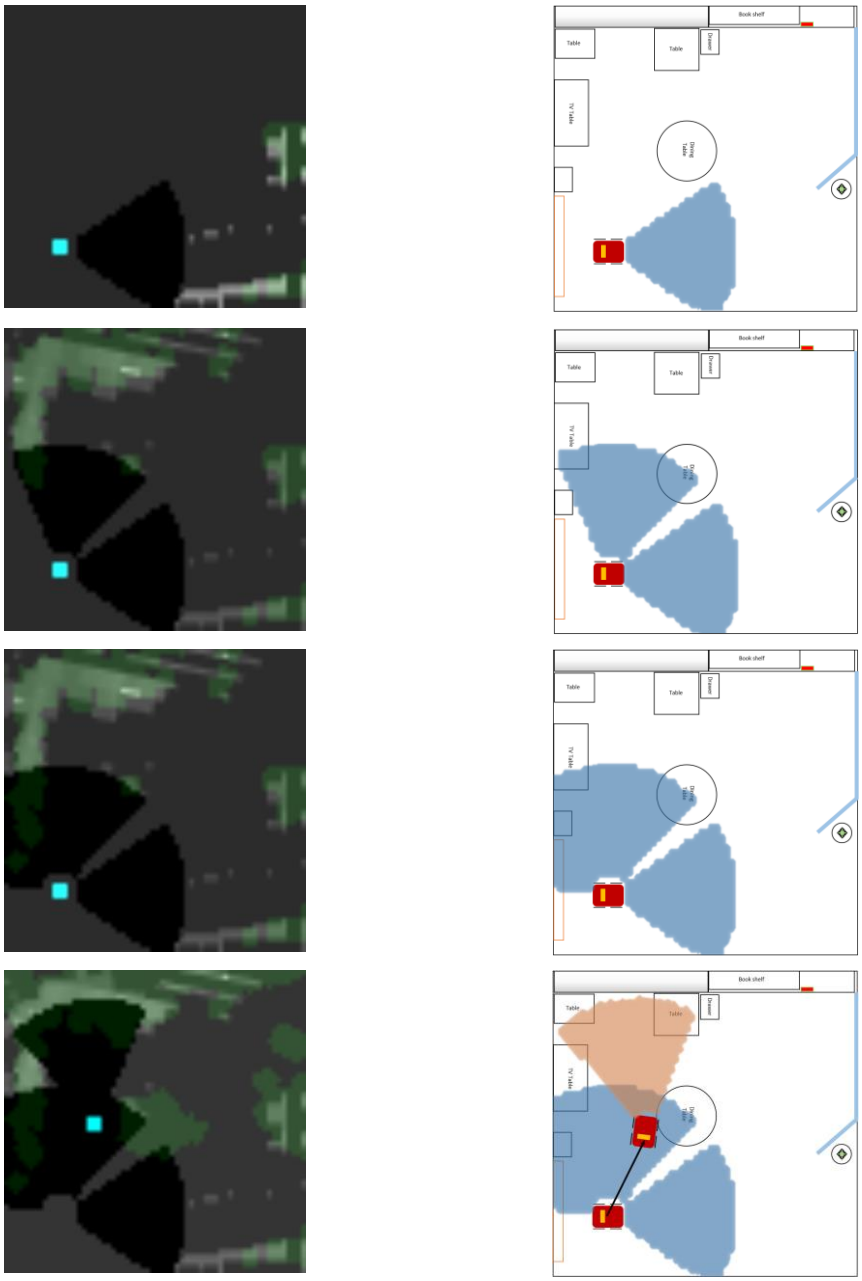


Figure 5.23: The search process using GSC with the energy constraint. From the left to right, a 2D representation of the 3D probability distribution map and the top view of the environment. The robot searches its initial location and moves to the second one.

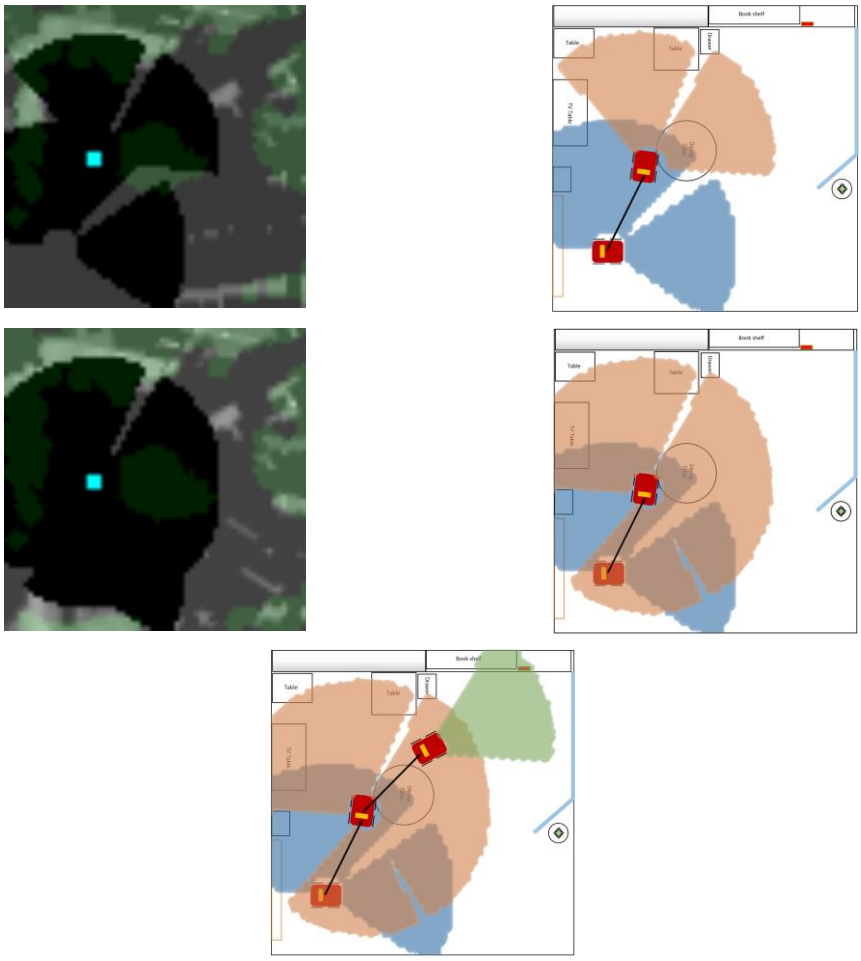


Figure 5.24: The search process using GSC with the energy constraint. From the left to right, a 2D representation of the 3D probability distribution map and the top view of the environment. The robot concludes the search and finds the target after looking toward the first direction from the third location.

that each operation now incurs a cost even from a stationary position. It causes the robot to expand the search faster to new locations in spite of inducing more costs.

After the robot fully searches its surroundings on the second position, it moves to the next one, where it detects the target after executing only one more action. This counts to a total of 9 actions to find the target using GSC.

The EGS algorithm (Figures 5.25-26) follows a similar route to GSC to perform the search. The major difference, however, is it does this more conservatively. The robot takes smaller steps to move forward each time it relocates to a new location. This makes EGS very inefficient in terms of timing or energy consumption even though it detects the target within the predefined energy constraint.

Using the EGS algorithm, the search begins by inspecting three directions. Then the robot moves to a new position from where it searches two more directions. Once more, it relocates to another position and performs three more actions after which it fails to reach the next destination. At this point, a new action sequence is generated by EGS defining a different route to be searched.

Following the new path and inspecting two more locations, the robot finally finds the target. The entire process is completed after performing 11 operations and relocating 4 times.

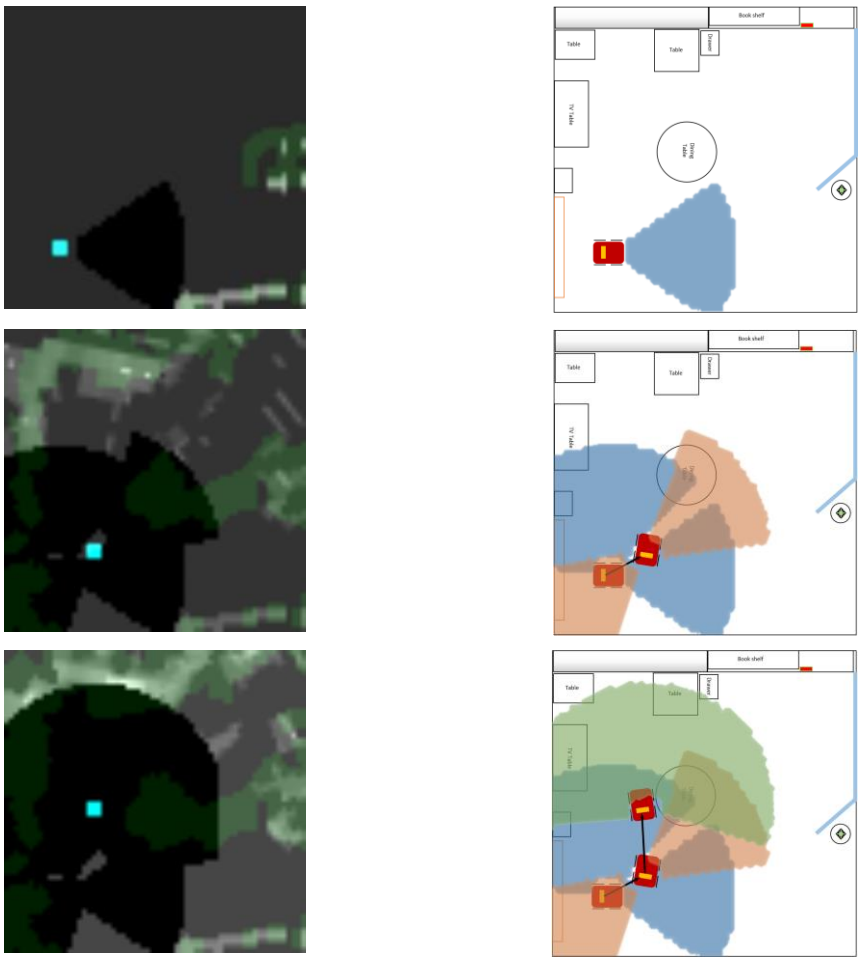


Figure 5.25: The search process using EGS with the energy constraint. From the left to right, a 2D representation of the 3D probability distribution map and the top view of the environment. The robot inspects the 3 locations and fails to move to the next one.

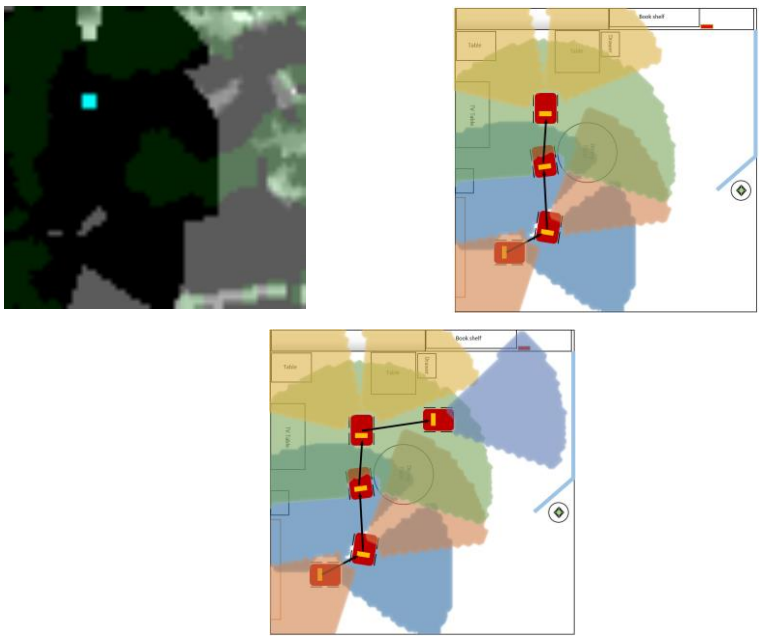


Figure 5.26: The search process using EGS with the energy constraint. From the left to right, a 2D representation of the 3D probability distribution map and the top view of the environment. The robot executes the new operation sequence and detects the target from the 5th location after inspecting one direction.

The DLAS algorithm also detects the target but after performing only 9 actions (see Figures 5.27-28). In this scenario, the robot explores two directions before deciding to relocate at which point it is unable to do so. Hence, DLAS generates a new action sequence through which the robot continues the search by exploring 4 directions at the second position and 2 more at the third one. At the final spot, the object is seen by the robot and the search is concluded.

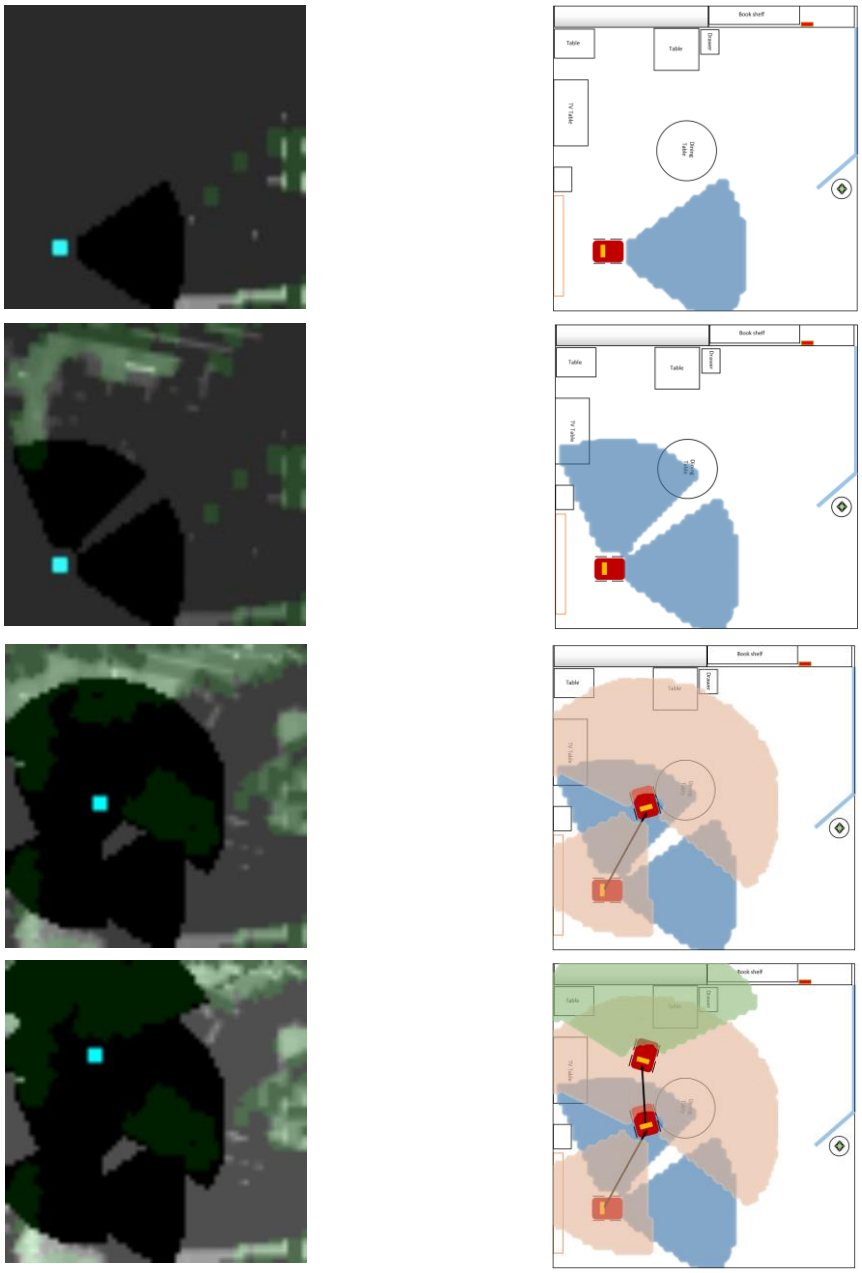


Figure 5.27: The search process using DLAS with the energy constraint. From the left to right, a 2D representation of the 3D probability distribution map and the top view of the environment. The first two operations performed by the robot.

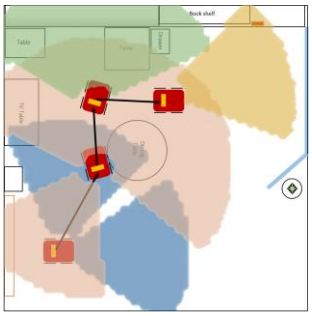


Figure 5-28. The search process using DLAS with the energy constraint. From the left to right, a 2D representation of the 3D probability distribution map and the top view of the environment. The robot concludes the search for the target by detecting it from the 4th location.

Figure 5-29. The 8 action approximate (on the left) and complete (on the right) solutions of DLAS.

It might be of interest to observe that how the 8-action approximate and complete solutions are in terms of efficiency and the processing time. Figure 5.29 shows both solutions from the left to right, the approximation using 8 actions and the complete solution with the processing times of 29 and 3033.8 seconds respectively. It is easy to see the superiority of the complete solution in terms of efficiency by observing the intensity of the background or the percentage of the environment forecasted to be

searched by the robot. However, such performance comes with a significant cost, increasing the processing time by more than 100 times, making the complete solution impractical to be used in the search.

5.2.2.3 Time

The last example in this series is illustrated by applying the time constraint to the search. We begin with an instance of the GSC algorithm in which the robot fails to detect the target (see Figures 5.30-31). It is worth mentioning that the path selection of the robot during the search is consistent with the previous examples, i.e. the robot always goes toward the regions populated with more furniture.

In this example, the robot searches its initial position by performing 2 operations followed by moving upward where it explores 5 additional directions. It continues by approaching the third location and inspecting two more directions after which the α threshold is triggered, thus, the robot changes behaviour and selects an operation with the highest probability distribution with a cost less than the remaining time constraint. As for the last operation, the robot moves downward to a position in the vicinity of the previous one and searches one additional direction after which the search is terminated.

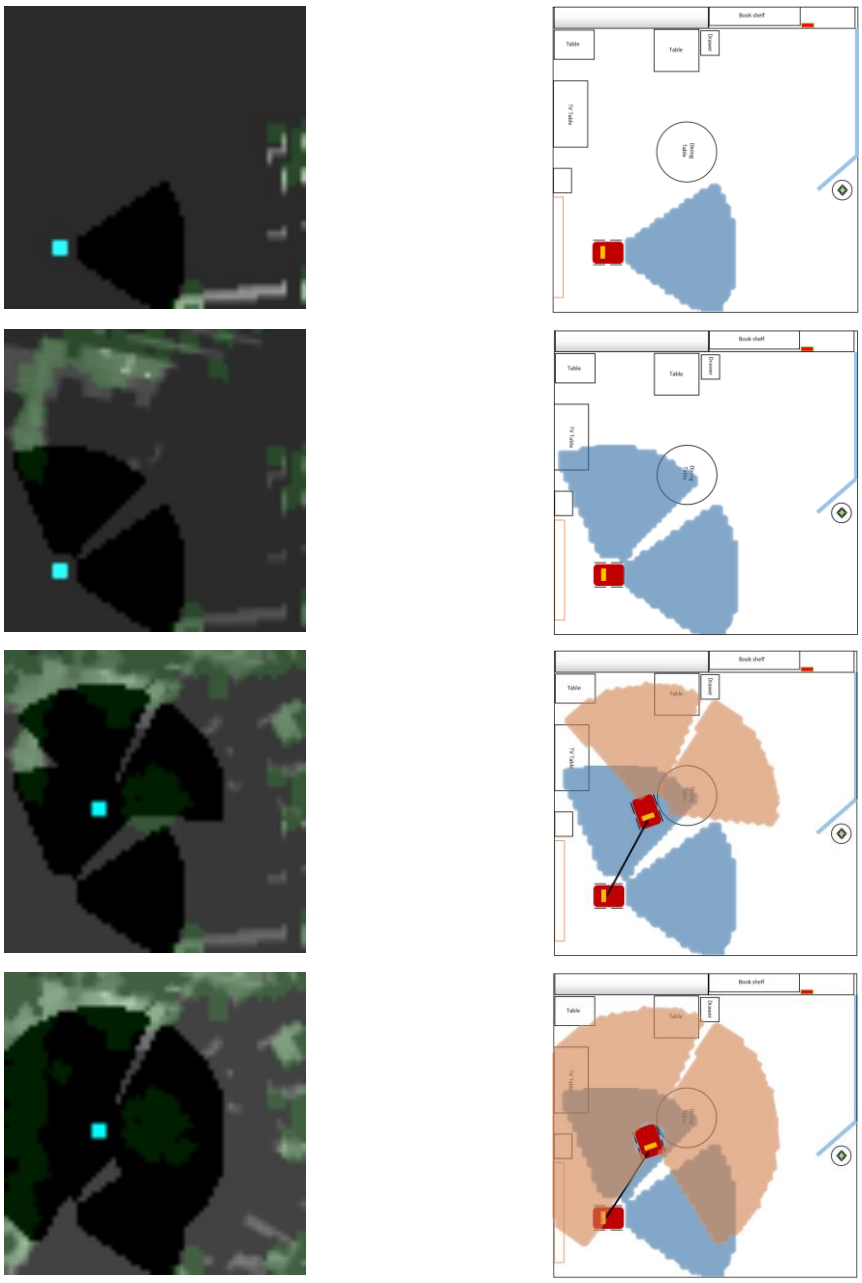


Figure 5.28: The search using the GSC algorithm with the time constraint. From the left to right, a 2D representation of the 3D probability distribution map and the top view of the environment. The robot inspects two locations.

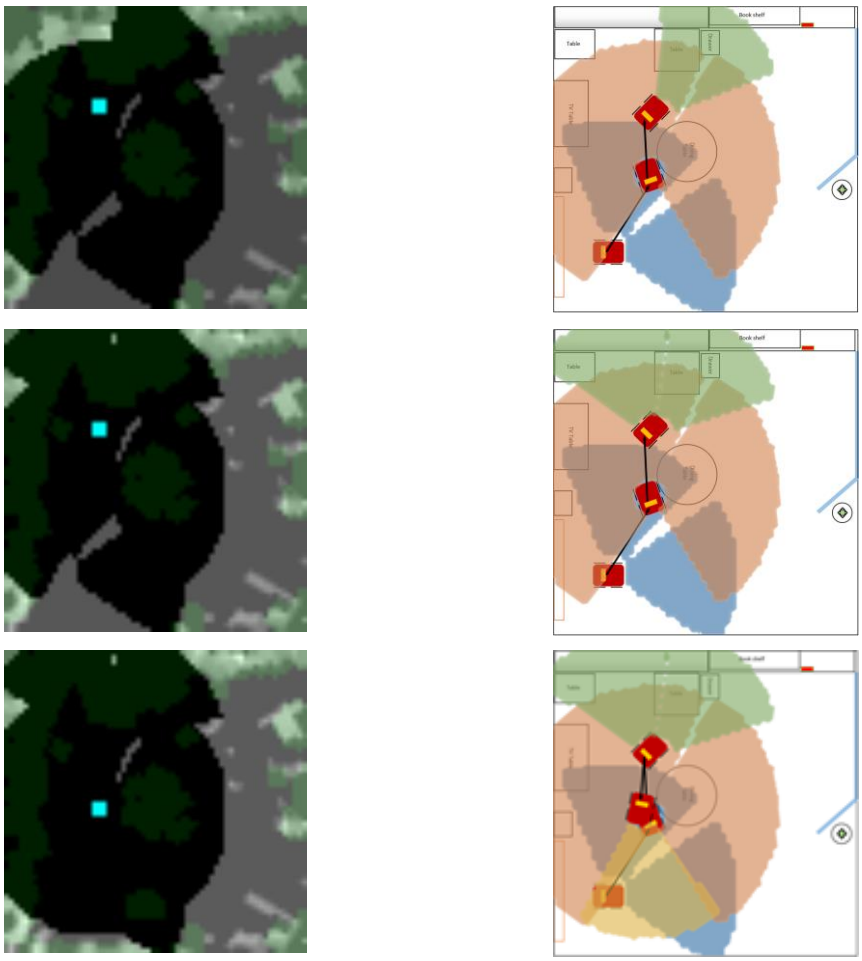


Figure 5.29: The search using the GSC algorithm with the time constraint. From the left to right, a 2D representation of the 3D probability distribution map and the top view of the environment. The search is terminated and the robot fails to find the target.

The EGS method (Figures 5.32-33) also fails to detect the target within the given time constraint. The robot very slowly expands the search domain by conservatively selecting the new locations. After conducting the search from three locations and performing 7 operations, the robot fails to move to the 4^{th} spot resulting in a new action sequence to be generated.

In compliance with the new operation sequence, the robot executes another action from its current location and then moves to the next one. From this position, three more directions are inspected by the robot after which the search is terminated unsuccessfully.

Applying the time constraint, the best performance was achieved by DLAS (see Figures 5.34-35). The robot looks toward two directions from its initial location and decides to relocate. Again the issue of destination and obstacle overlapping emerges, forcing DLAS to generate an improved version of the operation sequence.

The robot follows the new sequence and moves to the far up and searches two more directions. At the end, from the third location, the robot detects the object of interest by performing a total of 7 actions.

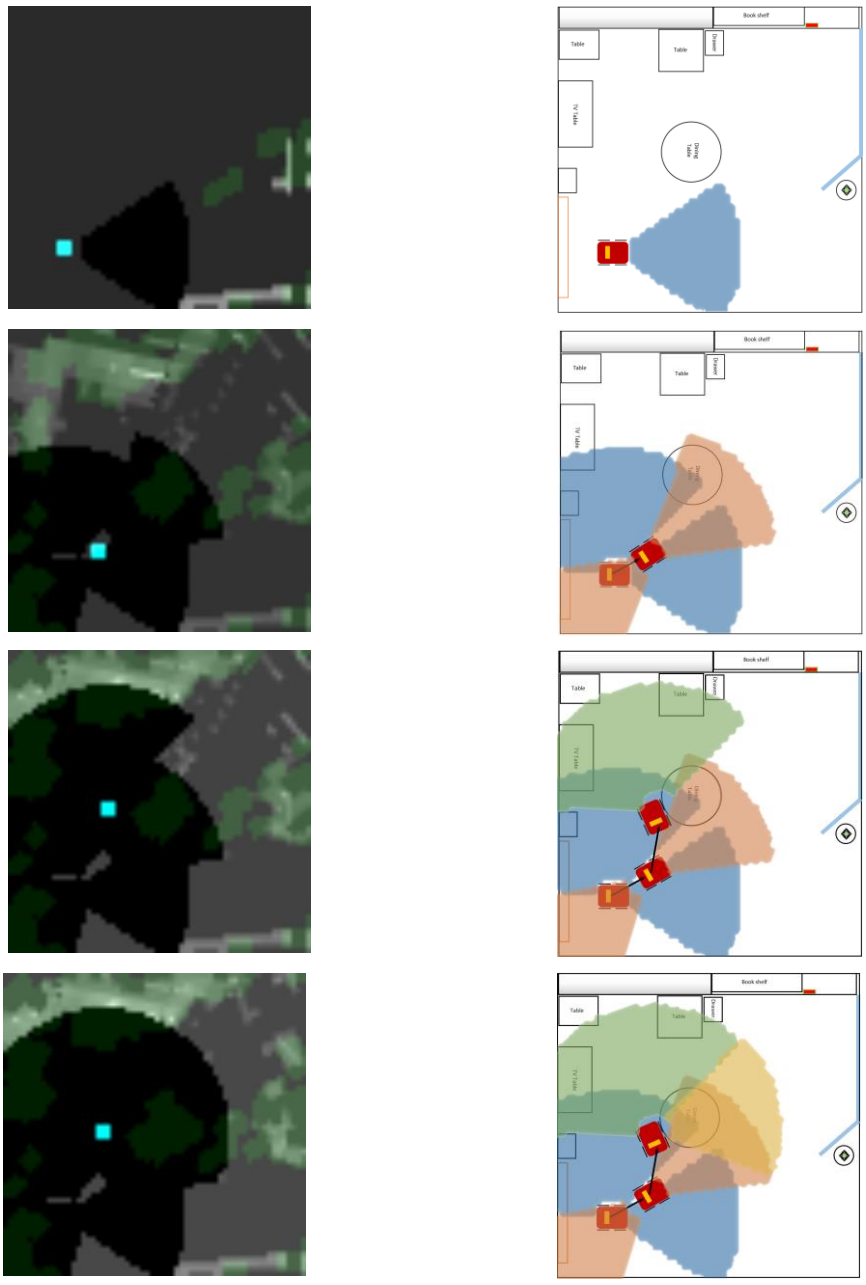


Figure 5.30: The search using the EGS algorithm with the time constraint. From the left to right, a 2D representation of the 3D probability distribution map and the top view of the environment. The robot searches the first 3 locations by performing 8 operations.

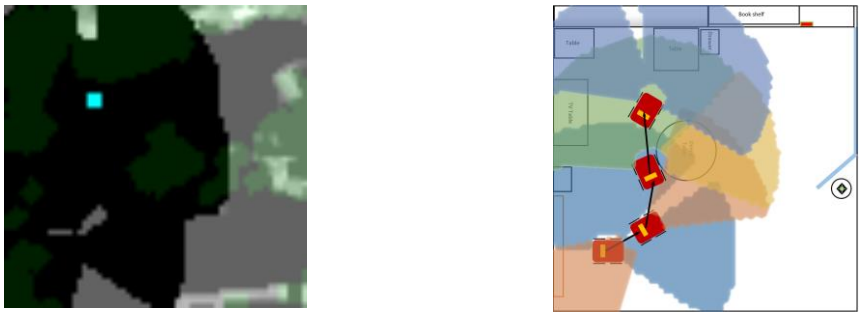


Figure 5.31: The search using the EGS algorithm with the time constraint. From the left to right, a 2D representation of the 3D probability distribution map and the top view of the environment. The robot finishes the search and fails to find the target.

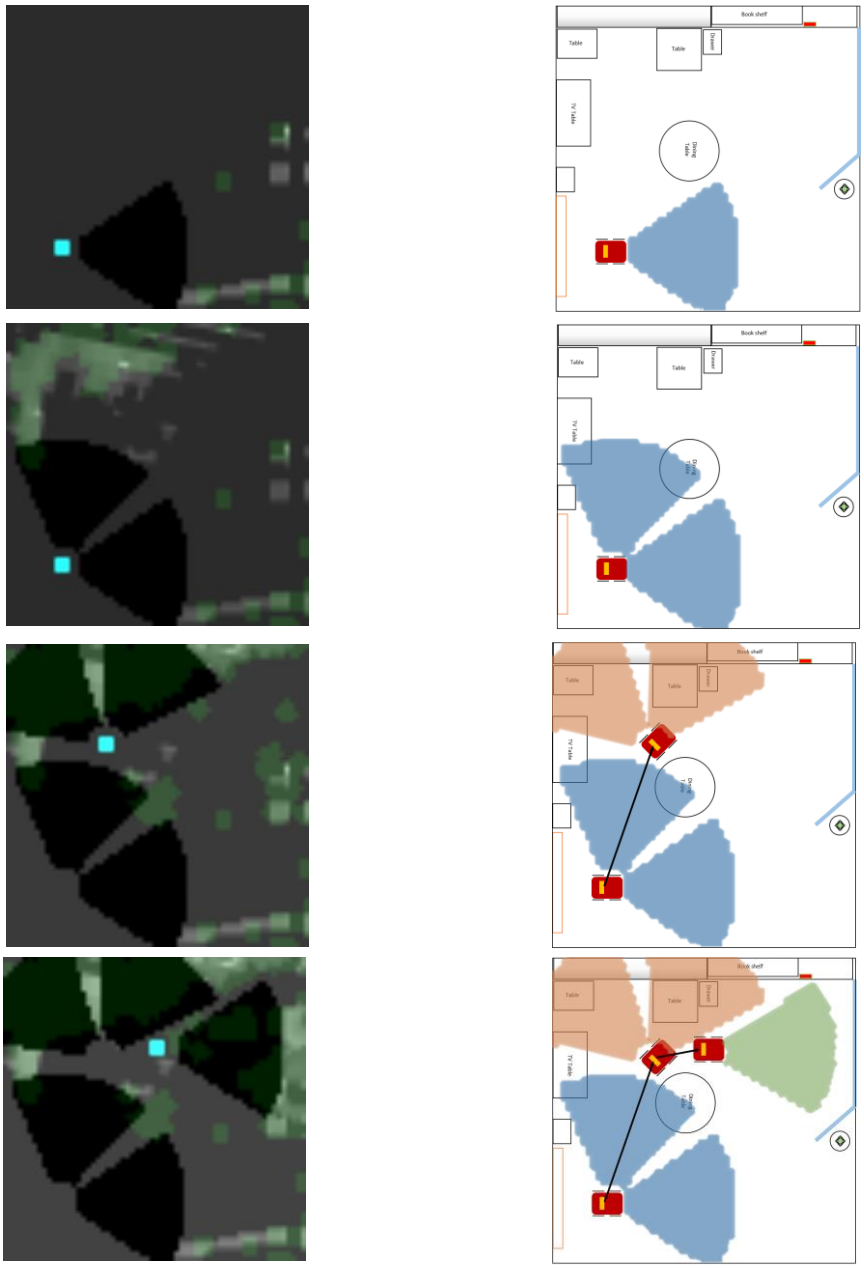


Figure 5.32: The search using the DLAS algorithm with the time constraint. From the left to right, a 2D representation of the 3D probability distribution map and the top view of the environment. The robot inspects its initial location.

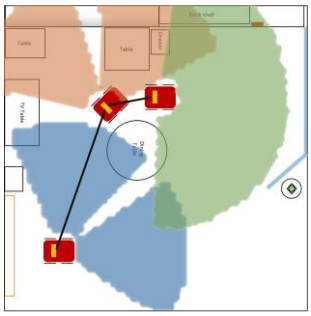


Figure 5.33: The search using the DLAS algorithm with the time constraint. The top view of the environment. The robot finds the target by looking toward the seventh direction.

In this instance of the search, the advantage of the DLAS global optimization is apparent. The search expands much faster and to more distant locations in comparison to EGS and GSC. This is particularly important in the cases where the search constraint is fairly low for the robot to explore an entire environment.

As of the previous subsections, a comparison of the approximate and complete solutions of DLAS is presented in Figure 5.36. The processing time of the solutions are 23.9 and 1824.9 seconds respectively.

5.2.3 Quantitative results

The results of the experiments are summarized for each individual environment in Tables 5.5-7 and the overall outcomes are reflected in table 5.8. The Tables are

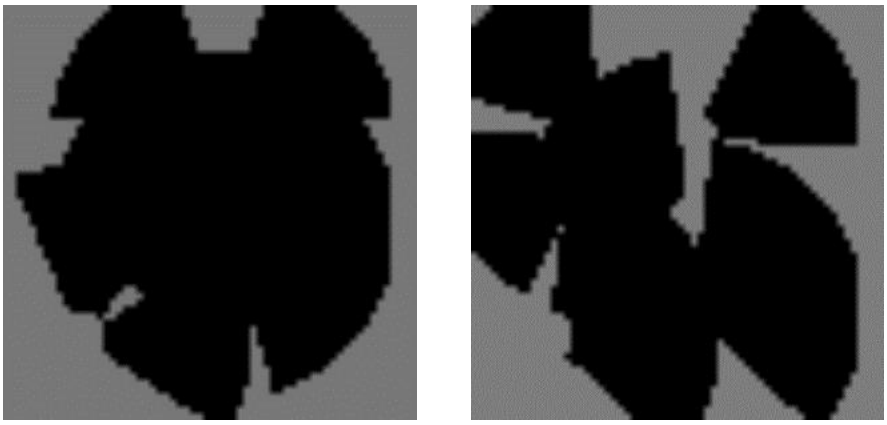


Figure 5.34: The 8 action approximate (on the left) and complete (on the right) solutions of DLAS.

categorized into 3 cost functions each divided into three sections reflecting the methods of search. The values are reflected in terms of the average time of the search and the energy spent by the system. Moreover, the average distance travelled by the robot and the number of actions performed by it are also included.

In addition, the percentage that each method found the target is indicated followed by the average number of times action sequences were recalculated (ANAR) (only applicable to EGS and DLAS). Finally, two additional measures are added to illustrate the efficiency of each method: the average percentage of the environment searched (APS) by each method and the units of cost incurred per APS (UCA). The UCA values show that how efficient methods are in terms of the cost they incur for searching each percentage of the environment. A lower rate of UCA means the method of the search is more efficient.

Table 5.4: The table of the results acquired from the experiments conducted in office a environment. The values that are colored green are the best in each category. The abbreviations are as follows: Greedy Search with Constraint(GSC), Extended Greedy Search (EGS), Dynamic Look Ahead Search (DLAS), Average Number of Actions Recalculated (ANAR), Average Percentage Searched (APS) and Unit of Cost incurred per APS.

Office a							
Constraint	***Method***	***Factor***	***Avg. Spent***	***% Found***	***ANAR***	***APS***	***UCA***
Distance	*GSC*	*Time (s)*	1006	89%	N/A	51.32%	59.43 (mm)
		Energy (kj)	70.390				
		Distance (m)	**3.050**				
		No. Actions	17.44				
	EGS	*Time (s)*	984	67%	0.11	48.79%	68.12 (mm)
		Energy (kj)	67.155				
		Distance (m)	**3.324**				
		No. Actions	15.44				
	DLAS	*Time (s)*	410	67%	0.44	37.97%	104.71 (mm)
		Energy (kj)	36.831				
		Distance (m)	**3.976**				
		No. Actions	5.44				
Energy	*GSC*	*Time (s)*	453	67%	N/A	42.98%	818.68 (j)
		Energy (kj)	**35.187**				
		Distance (m)	2.784				
		No. Actions	7.33				
	EGS	*Time (s)*	587	44%	0	46.04%	956.02 (j)
		Energy (kj)	**44.015**				
		Distance (m)	3.224				
		No. Actions	8.67				
	DLAS	*Time (s)*	503	56%	0.78	45.39%	976.71 (j)
		Energy (kj)	**44.333**				
		Distance (m)	4.299				
		No. Actions	6.89				
Time	*GSC*	***Time (s)***	**449**	44%	N/A	44.25%	10.14 (s)
		Energy (kj)	35.563				
		Distance (m)	3.012				
		No. Actions	7.22				
	EGS	***Time (s)***	**463**	44%	0.44	38.88%	11.90 (s)
		Energy (kj)	33.246				
		Distance (m)	2.357				
		No. Actions	7.44				
	DLAS	***Time (s)***	**459**	44%	1.11	43.65%	10.51 (s)
		Energy (kj)	37.946				
		Distance (m)	3.471				
		No. Actions	6.56				

Table 5.5: The table of the results acquired from the experiments conducted in office b environment. The values that are colored green are the best in each category. The abbreviations are as follows: Greedy Search with Constraint(GSC), Extended Greedy Search (EGS), Dynamic Look Ahead Search (DLAS), Average Number of Actions Recalculated (ANAR), Average Percentage Searched (APS) and Unit of Cost incurred per APS.

Office b							
Constraint	***Method***	***Factor***	***Avg. Spent***	***% Found***	***ANAR***	***APS***	***UCA***
Distance	*GSC*	*Time (s)*	720	89%	N/A	46.24%	55.13 (mm)
		Energy (kj)	51.150				
		Distance (m)	**2.549**				
		No. Actions	12.56				
	EGS	*Time (s)*	865	67%	0.78	50.34%	63.75 (mm)
		Energy (kj)	61.843				
		Distance (m)	**3.209**				
		No. Actions	14.67				
	DLAS	*Time (s)*	410	89%	0.56	41.79%	81.93 (mm)
		Energy (kj)	34.915				
		Distance (m)	**3.424**				
		No. Actions	5.67				
Energy	*GSC*	*Time (s)*	391	100%	N/A	42.34%	715.30 (j)
		Energy (kj)	**30.286**				
		Distance (m)	2.541				
		No. Actions	6.22				
	EGS	*Time (s)*	441	89%	0.44	45.40%	726.67 (j)
		Energy (kj)	**32.991**				
		Distance (m)	2.618				
		No. Actions	7.11				
	DLAS	*Time (s)*	445	78%	0.56	44.44%	831.72 (j)
		Energy (kj)	**36.962**				
		Distance (m)	3.285				
		No. Actions	5.89				
Time	*GSC*	***Time (s)***	**374**	89%	N/A	42.26%	8.84 (s)
		Energy (kj)	29.388				
		Distance (m)	2.555				
		No. Actions	6				
	EGS	***Time (s)***	**407**	89%	0.44	43.27%	9.40 (s)
		Energy (kj)	30.221				
		Distance (m)	2.393				
		No. Actions	6.56				
	DLAS	***Time (s)***	**443**	78%	1.11	41.86%	10.58 (s)
		Energy (kj)	36.202				
		Distance (m)	3.609				
		No. Actions	5.56				

Table 5.6: The table of the results acquired from the experiments conducted in office c environment. The values that are colored green are the best in each category. The abbreviations are as follows: Greedy Search with Constraint(GSC), Extended Greedy Search (EGS), Dynamic Look Ahead Search (DLAS), Average Number of Actions Recalculated (ANAR), Average Percentage Searched (APS) and Unit of Cost incurred per APS.

Office c							
Constraint	***Method***	***Factor***	***Avg. Spent***	***% Found***	***ANAR***	***APS***	***UCA***
Distance	*GSC*	*Time (s)*	676	67%	N/A	43%	62.07 (mm)
		Energy (kj)	41.585				
		Distance (m)	**2.669**				
		No. Actions	11.83				
	EGS	*Time (s)*	706	67%	0.33	45.13%	63.68 (mm)
		Energy (kj)	51.819				
		Distance (m)	**2.874**				
		No. Actions	12.17				
	DLAS	*Time (s)*	366	83%	0.33	40.48%	71.81 (mm)
		Energy (kj)	31.073				
		Distance (m)	**2.907**				
		No. Actions	5.50				
Energy	*GSC*	*Time (s)*	317	67%	N/A	37.28%	686.59 (j)
		Energy (kj)	**25.596**				
		Distance (m)	2.329				
		No. Actions	5.17				
	EGS	*Time (s)*	331	67%	0.33	37.89%	707.44 (j)
		Energy (kj)	**26.805**				
		Distance (m)	2.548				
		No. Actions	5.33				
	DLAS	*Time (s)*	305	83%	0	38.39%	680.44 (j)
		Energy (kj)	**26.122**				
		Distance (m)	2.747				
		No. Actions	4.33				
Time	*GSC*	***Time (s)***	**272**	67%	N/A	35.40%	7.68 (s)
		Energy (kj)	22.911				
		Distance (m)	2.836				
		No. Actions	4.33				
	EGS	***Time (s)***	**308**	67%	0	33.15%	9.29 (s)
		Energy (kj)	24.612				
		Distance (m)	2.337				
		No. Actions	4.33				
	DLAS	***Time (s)***	**326**	50%	1	35.69%	9.13 (s)
		Energy (kj)	26.302				
		Distance (m)	2.327				
		No. Actions	5				

In order to derive meaning from the data presented in the tables above, we start by reviewing the performance of the search models in office environment a. In scenarios where the distance and energy constraints were applied, GSC was able to achieve the best performance in terms of the percentage it found the target and the efficiency of the search.

GSC shared the same spot with the other algorithms by only detecting the target 44% of the times when the search was restricted by the time constraint. Despite the similar detection rates, GSC had the best efficiency by spending on average 10.40 s for searching each percentage of the environment. In this instance, the worst performance belongs to the EGS algorithm, which at its best did not outperform any of the other two methods. It, however, yielded a better efficiency in comparison to DLAS in the experiments with the distance and energy constraints.

The GSC algorithm attained the best search efficiency in office b along with the highest detection rate, sharing the first place with DLAS and EGS in the search instances with the distance and time constraints respectively.

The experiments in office c had very different outcomes. DLAS performed the best overall by having the highest percentage of detection in the cases of the distance and energy constraints. It was also the most efficient technique in terms of energy consumption. However, DLAS performed poorly with only 50% success rate in finding the target when the time constraint was applied.

Taking into account all the results, EGS exhibited the poorest performance thus far. The algorithm's rate of detection is the lowest overall despite having a better efficiency in comparison to DLAS in the majority of the cases. DLAS performed at its best in the environment c. It benefited from its global optimization, allowing the robot to spread the search domain further away from its initial location, resulting in the higher chance of detecting the target in far distances. Whereas relying on the greedy approaches, the robot stuck in a local neighbourhood of its initial location and the regions far away were ignored.

As it was anticipated, DLAS had its weakest performance when a time constraint was applied to the search, primarily due to the high time consumption of generating action sequences. This effect was less in the scenarios with the energy constraints because the processing energy was significantly lower in comparison to performing operations such as moving the pan-tilt unit or the robot. The operation generation cost is also irrelevant when distance is the constraint of the search, therefore DLAS had its best results in such cases.

The environments' settings had strong impacts on the performance of DLAS. The algorithm, as mentioned before, selects an initial operation sequence blindly without any prior information regarding the search environment. It is easy to see that if an environment is more cluttered, there is a higher chance of failure due to the unreachability of an action's location.

The aforementioned impact was apparent in the experiments. DLAS had the lowest performance rate comparing to GSC in office a, which was populated with the most number of furniture items in comparison to the other rooms. On average, DLAS recalculated sequences 0.77 of the times. This value was less in office b, 0.74, in which DLAS matched the performance of GSC in the search with the distance constraint.

As one would expect, DLAS regenerated operation sequences on average as low as 0.44 of the times in office c. This is why, it reached its highest performance comparing to GSC in this environment, where the furniture were concentrated on only two edges of the room leaving the central regions empty. This reduces the chance that operations' locations to be chosen in the areas occupied by obstacles. Furthermore, searching in a semi empty environment increases the accuracy of distance estimation when the search operations are calculated by DLAS. As a result, there is a better chance that the actions to be performed as predicted by the algorithm.

Figures 5.36 and 5.37 display the performance measures of the planning algorithms. Without any doubts, the best performance belongs to the GSC algorithm. This method achieved at least the highest percentage of finding the target in 7 out of 9 scenarios of the search and with only one exception, the best overall cost efficiency.

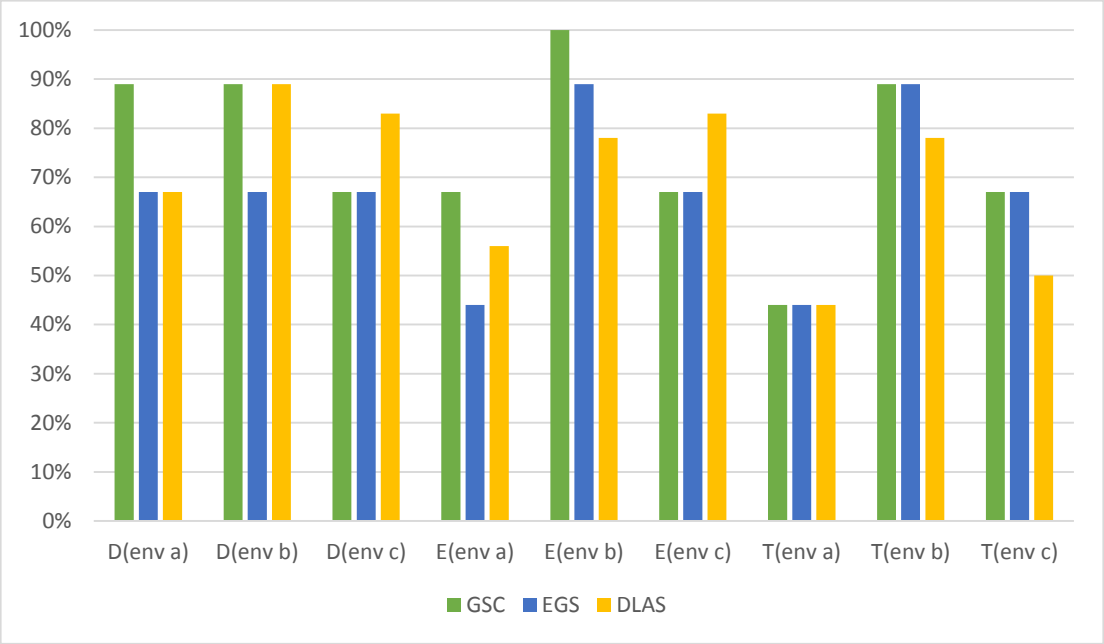


Figure 5.36. The comparison of the proposed search methods in terms of the percentage the target was detected. These results are presented for each cost constraint, Distance (D), Energy (E) and Time (T).

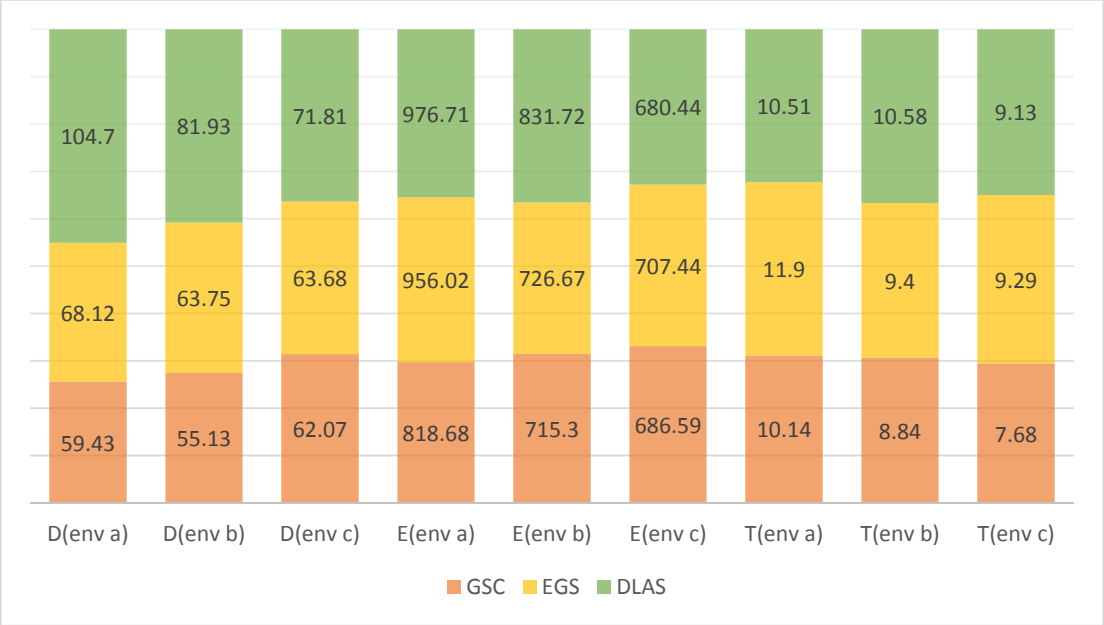


Figure 5.35. The comparison of the proposed search methods in terms of cost efficiency. The results are presented for each cost constraint, Distance (D), Energy (E) and Time (T) with units in mm, joules and seconds respectively.

Table 5.8 presents the average performance of the sensor planning algorithms across all the test environments. As anticipated, GSC stands out as the most cost efficient search strategy. It also achieved the best detection percentage with a small margin over DLAS in the experiments with distance and energy constraints, and shared the first place with EGS in the search instances with the time constraints.

It is important to note that in order to better highlight the differences between the detection ratios of the search models, it is necessary to conduct additional evaluations with various constraint values in a similar search environment. In our experiments, the constraints were set randomly and remained the same for experiments in each test environment. Merely relying on such fixed values increases the chance that they are set either too high or too low resulting in all the methods to whether find the target or not, thus, yielding similar detection ratios.

However, by taking into consideration the efficiency of the planning strategies in addition to their detection rates, we can gain a better insight into the performance of each model. Based on this approach, the GSC algorithm clearly stands out. On average, GSC incurred approximately 1s, 100 j and 27 mm less cost to explore each percentage of environment in comparison to DLAS, and similarly 1.30 s, 60 j, and 7 mm less than EGS. Having a method with a better efficiency implies that it more likely can find a target successfully in the situations with tighter constraints.

Moreover, the choice of the search strategy highly depends on the nature of the environment. The use of the DLAS algorithm is only justifiable in the large environments, where only a limited number of obstacles are available as was seen in the case of the experiments in office c. To be efficient, DLAS requires a prior knowledge of the environment assuming that there are no dynamic elements involved at any point of the search. As a result, the DLAS algorithm certainly is not a reliable option for conducting the search in unknown environments.

Nevertheless, GSC has the advantage to react to changes in the environment. It also relies on the visual clues, which allow this model to foresee into the future and estimate the outcome of its later actions with no significant cost of processing. To state this in a different manner, through the use of saliency, GCS can literally "look ahead" of itself in the environment and decide what to do next.

Moreover, the expanding radio of the search domain in GCS can be altered by varying the parameters of the algorithm such as the α threshold or the influence of saliency responses on increasing the probability of interest regions. The higher these values are, the larger the steps taken by the robot to explore new locations.

Table 5.7: The overall results of the experiments using the proposed search strategies. The values that are colored green are the best of each constraint. The abbreviations are as follows: Greedy Search with Constraint(GSC), Extended Greedy Search (EGS), Dynamic Look Ahead Search (DLAS), Average Number of Actions Recalculated (ANAR), Average Percentage Searched (APS) and Unit of Cost incurred per APS.

Total							
Constraint	***Method***	***Factor***	***Avg. Spent***	***% Found***	***ANAR***	***APS***	***UCA***
Distance	*GSC*	*Time (s)*	800	81.48%	N/A	46.85%	58.82 (mm)
		Energy (kj)	54.375				
		Distance (m)	**2.756**				
		No. Actions	13.94				
	EGS	*Time (s)*	840	66.67%	0.41	48.09%	65.19 (mm)
		Energy (kj)	60.273				
		Distance (m)	**3.135**				
		No. Actions	14.09				
	DLAS	*Time (s)*	396	79.63%	0.44	40.08%	85.72 (mm)
		Energy (kj)	34.273				
		Distance (m)	**3.436**				
		No. Actions	5.54				
Energy	*GSC*	*Time (s)*	387	77.78%	N/A	40.87%	742.76 (j)
		Energy (kj)	**30.356**				
		Distance (m)	2.551				
		No. Actions	6.24				
	EGS	*Time (s)*	453	66.67%	0.26	43.11%	802.69 (j)
		Energy (kj)	**34.604**				
		Distance (m)	2.797				
		No. Actions	7.04				
	DLAS	*Time (s)*	418	72.22%	0.44	42.74%	837.76 (j)
		Energy (kj)	**35.806**				
		Distance (m)	3.443				
		No. Actions	5.70				
Time	*GSC*	***Time (s)***	**365**	67%	N/A	40.64%	8.90 (s)
		Energy (kj)	29.287				
		Distance (m)	2.801				
		No. Actions	5.85				
	EGS	***Time (s)***	**393**	67%	0.30	38.43%	10.21 (s)
		Energy (kj)	29.359				
		Distance (m)	2.362				
		No. Actions	6.11				
	DLAS	***Time (s)***	**409**	57%	1.07	40.40%	10.08 (s)
		Energy (kj)	33.483				
		Distance (m)	3.135				
		No. Actions	5.70				

6 Conclusion

In this thesis, we presented a framework of generating saliency to be used in robotic visual search as a means of detecting regions with a higher chance of being spatially related to the target object. The proposed model exploits the use of two saliency techniques to produce clues regarding the structure of the environment that may contain the target as well as the target itself.

The model of search used in this work has no prior knowledge of the environment except those of its exterior boundaries. The knowledge describing the environment is dynamically produced using saliency, as the robot progresses through the search. The saliency responses are used to increase the importance of the regions yielding a higher chance of detecting the target, inducing the robot to search those locations first.

Through the extensive empirical evaluations, we showed that how such attentive processes benefit the search by reducing the overall time of the search, the distance travelled by the robot and the number of operations performed to detect the target.

The efficiency of the saliency mapping is somewhat dependent on the structure of the search environment. The presence and number of distractors in an environment can greatly alter the performance of the search using saliency. The more such components exist in an environment, the lower the effectiveness of saliency.

In the second part of this thesis, we tackled the problem of sensor planning for object search with predefined cost constraints. Three strategies were proposed, namely Greedy Search with Constraint (GSC), Extended Greedy Search (EGS) and Dynamic Look Ahead Search (DLAS). The first two methods greedily select actions and attempt to maximize the number of operations to be performed within the search constraint.

The DLAS algorithm, however, follows a more global approach to optimization at the expense of a higher processing time. It is a form of tree search, which generates a number of alternative solutions and selects the one that yields the highest probability of detecting the target.

Experiments were conducted in the environments of various sizes and configurations with the different values and types of constraints. The results indicate that the GSC algorithm has the highest rate of detecting the target, and at the same time, it is the most efficient method of the search. The GSC algorithm heavily relies on saliency information and makes decisions one at a time, whereas DLAS and EGS generate action sequences at the start of the search and only saliency comes into effect if the algorithms regenerate the sequence at some point during the search.

6.1 Future work

The target specific saliency proposed in our model only relies on the color distributions of an object. This can be easily distracted to other objects with similar colors within an environment. One way of addressing this issue is to include additional object features such as shape and orientation to filter out unwanted saliency responses.

Moreover, our assumption was that in a typical environment an object more likely is placed on surfaces such as tables or shelves which may stand out applying a saliency model. In this way, the saliency results can help in the form of indirect clues to guide the robot to locations with higher chance of detecting the target. As for indirect search applications, in search with saliency if such the spatial relation between objects does not hold, e.g. the object is placed individually on the ground, saliency would not be effective. In fact, in such scenarios saliency can rather be distracting. One way of addressing this issue is the use of a highly tuned salient algorithm for a particular object.

The test environments used in our experiments were limited in size in the sense that their dimensions did not significantly exceed the effective field of view of the recognition algorithm. It is anticipated that the performance gap between the search methods with and without saliency grows as a result of increasing the size of the environment, something to be studied in the future.

To evaluate the proposed sensor planning techniques, only one constraint was applied at a time to the search. In practice, however, more than one constraint might be of interest depending on the nature of the application it is being used in. For instance, in search and rescue missions, the timing of search is very vital but at the same time the traveling distance of the robot should be minimized to avoid hazards and damaging the hardware, which might jeopardize the mission. The proposed methods can be extended to include multiple cost constraints. One way of achieving this is unifying costs into a single unit with some bias which reflects the importance of each cost at a given time.

Furthermore, the GSC model can be tested with different strategies of operation selection after reaching the α threshold such as the actions with minimum cost, search locations closest to the robot or the next best greedy action.

Both series of the experiments presented in this thesis indicated that saliency plays an important role in the efficiency of visual search. Perhaps limiting the number of actions generated by DLAS and EGS and including saliency information in their decision making processes more frequently can enhance the performance of these techniques.

Finally, search strategies were evaluated with fixed constraint values within each environment. One may consider altering the constraints in the search spaces to gain a better insight into the behavior of each planning strategy.

Bibliography

[1] T. D. Garvey, *Perceptual strategies for purposive vision,* Technical Report, SRI International, 1976.

[2] A. Aydemir, K. Sjoo, J. Folkesson, A. Pronobis, "Search in the real world: Active visual object search based on spatial relations," in *ICRA*, Shanghai, 2011.

[3] M. Göbelbecker, A. Aydemir, A. Pronobis, K. Sjöö, and P. Jensfelt, "A planning approach to active visual search in large environments," in *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*, San Franseco, 2011.

[4] L. Kunze and N. Hawes, "Indirect Object Search based on Qualitative Spatial Relations," in *IROS, Workshop on AI-based Robotics*, Tokyo, 2013.

[5] R. Bajcsy, "Active Perception," *Proceedings of the IEEE,*, vol. 6, no. 8, p. 996–1005, 1988.

[6] D. Wilkes, and J. K. Tsotsos, "Active Object Recognition," in *CVPR*, Urbana, 1992.

[7] S. J. Dickinson, H. I. Christenen, J. K. Tsotsos, and G. Olofsson, "Active object recognition integrating attention and viewpoint control," *COMPUTER VISION AND IMAGE UNDERSTANDING*, vol. 67, no. 3, pp. 239-260, 1997.

[8] B. Browatzki, V. Tikhanoff, G. Metta, H. H. Bulthoff, and C. Wallraven, "Active Object Recognition on a Humanoid Robot," in *IEEE International Conference on Robotics and Automation*, Saint Paul, 2012.

[9] G. De Cubber, D. Serrano, K. Berns, K. Chintamani, R. Sabino, S. Ourevitch, D. Doroftei, C. Armbrust, T. Flamma, and Y. Baudoi, "Search And Rescue Robots Developed By The European Icarus Project," in *RISE workshop on Robotics for Risky Environments - Extreme Robotic*, Saint-Petersburg, 2013.

[10] C. Marques, J. Cristovao, P. Lima, J. Frazao, I. Ribiro, and R. Vecntura, "RAPOSA: Semi-Autonomous Robot for Rescue Operations," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, 2006.

[11] R. Wegner and J. Anderson, "An Agent-Based support to Balancing Teleoperation and Autonomy for Robotic Search and Rescue," *International Journal of Robotics and Automation*, vol. 21, no. 2, pp. 120-128, 2006.

[12] JPL_NASA, "NASA's Mars Curiosity debuts autonomous," 27 Aug 2013. [Online]. Available: http://www.jpl.nasa.gov/news/news.php?release=2013-259. [Accessed 15 Jun 2014].

[13] Y. Fukazwa, H. Yuasa, T. Arai, and H. Asama, "Controling a Mobile Robot That Searches for and Rearranges Obejcts with Unknown Locations and Shapes," in *IEEE/RSJ international Conference on INtelligent Robots and Systems*, Las Vegas, 2003.

[14] B. Tovar, S. M. LaValle, and R. Murrieta, "Optimal Navigation and Object Finding without Geometric Maps or Localization," in *ICRA*, Taipei, 2003.

[15] H. Lau, S. Huang and G. Dissanayake, "Optimal Search for Multiple Targets in a Built Environment," in *IROS*, Edmonton, 2005.

[16] S. A. Mehdi and K.Berns, "Probabilistic Search of Human by Autonomous Mobile Robot," in *PETRA*, New York, 2011.

[17] Y. Ye, *Sensor Planning For Object Search*, PhD. Thesis, University of Toronto, 1997.

[18] K. Shubina and J. K. Tsotsos, "Visual search for an object in a 3D environment using a mobile robot," *Computer Vision and Image Understanding*, vol. 114, pp. 535-547, 2010.

[19] F. Saidi, O. Stasse, and K. Yokoi, "Active Visual Search by a Humanoid Robot," *Recent Progress in Robotics: Viable Robotic Service for Human*, vol. 370, pp. 171-184, 2008.

[20] J. K. Tsotsos and K. Shubina, "Attention and Visual Search: Active Robotic Vision Systems that Search," in *The 5th International Conference on Computer Vision Systems*, Bielefeld, 2007.

[21] X. Hou and L. Zhang, "Saliency Detection: A Spectral Residual Approach," in *CVPR*, Minneapolis, 2007.

[22] A. Oliva, A. Torralba, M. S. Castelhano, J. M. Henderson, "TOP-DOWN CONTROL OF VISUAL ATTENTION IN OBJECT DETECTION," in *IEEE Proceedings of the International Conference on Image Processing*, Catalonia, 2003.

[23] N. J. Butko, L. Zhang, G. W. Cottrell, and J. R. Movellan, "Visual Saliency Model for Robot Cameras," in *ICVR*, Passadena, 2008.

[24] F. Orabona, G. Metta and G. Sandini, "Object-based Visual Attention: a Model for a Behaving Robot," in *CVPR Workshops*, San Diego, 2005.

[25] C. J. Chang, C. Siagian, and L. Itti, "Mobile Robot Vision Navigation & Localization Using Gist and Saliency," in *IROS*, Taipei, 2010.

[26] L. Itti, C. Koch, and E. Niebur, "A model of Saliency-based Visua Attentio for Rapid Scene Analysis," *Pattern Analysis and Machine Intelligence,* vol. 20, no. 11, pp. 1254-1259, 1998.

[27] R. Roberts, D. N. Ta, J. Straub, K. Ok, and F. Dellaert, "Saliency Detection and Model-based Tracking: a Two Part Vision System for Small Robot Navigation in Forested Environments," in *SPIE,* San Diego, 2012.

[28] Y. Ye and J. K. Tsotsos, "Sensor Planning in 3D Object Search: its Formulation and Complexity," in *Artificial Intelligence and Mathematics*, Florida, 1996.

[29] A. Sarmiento, R. Murrieta, and S. A. Huchinson, "An Efficient Strategy for Rapidly Finding an Object in a Polygonal World," in *IEEE/RJS international Conference on Intelligent Robots and Systems*, Las Vegas, 2003.

[30] K. Shubina, *Sensor Planning for 3D Object Search*, Masters Thesis, York University, 2007.

[31] N. Bruce and J. K. Tsotsos, "Attention based on information maximization," *Journal of Vision*, vol. 7, no. 9, p. 950, 2007.

[32] M. J. Swain and D. H. Ballard, "Color Indexing," *International Journal of Computer Vision*, vol. 7, no. 1, pp. 11-32, 1991.

[33] R. Achanta, *Finding Objects of Interest in Images using Saliency and superpixel*, PhD Thesis, Ecole Poythechnique Fedeale de Lausanne, 2011.

[34] L. Itti and C. Koch, "A saliency-based search mechanism for overt and covert shifts of visual attention," *Vision Research*, vol. 40, pp. 1489-1506, 2000.

[35] Y. Jia and M. Han, "Category-Independent Object-level Saliency Detection," in *ICCV2013*, Sydney, 2013.

[36] F. Moosmann, D. Larulus and F. Jurie, "Learning Saliency Maps for Object Categorization," in *ECCV Workshop on the Representation and Use of Prior Knowledge in Vision*, Graz, 2006.

[37] H. J. Seo and P. Milanfar, "Nonparametric Bottom-Up Saliency Detection by Self-Resemblance," in *CVPR*, Miami, 2009.

[38] V. Olesova and V. Benesova, "Modified Methods of Generating Saliency Maps Based on Superpixels," in *CESCG*, Smolenice, 2014.

[39] L. Zhang, T. K. Mark, M. H. Tong,H. Shan and G. W. Cottrell, "SUN: A Bayesian Framework for Saliency Using Natural Statistics," *Journal of Vision*, vol. 8, no. 7, pp. 1-20, 2008.

[40] S. Goferman, L. Zelnik-Manor, and A. Tal, "Context-Aware Saliency Detection," *Pattern Analysis and Machine Intelligence*, vol. 34, no. 10, pp. 1915-1926, 2012.

[41] K.Y. Chang, T.L Liu, H.T. Chen, and S.H. Lai, "Fusing Generic Objectness and Visual Saliency for Salient Object Detection," in *ICCV*, Barcelona, 2011.

[42] H. Jiang, J. Wang, Z. Yuan, T. Liu, N. Zheng, and S. Li, "Automatic Salient Object Segmentation Based on Context and Shape Prior," in *in Proc*, *The British Machine Vision Conference*, Dundee, 2011.

[43] A. Borji , D. N. Sihite, and L. Itti, "Quantitative Analysis of Human-Model Agreement in Visual Saliency Modeling: A Comparative Study," *IEEE TRANSACTIONS ON IMAGE PROCESSING*, vol. 22, no. 1, 2013.

[44] Zhang, X. Hou and L., "Dynamic Visual Attention: Searching for coding length increments," in *In Proc. NIPS*, Vancouver, 2008.

[45] X. H. a. L. Zhang, "Saliency Detection: A Spectral Residual Approach," in *In Proc. CVPR*, Minneapolis, 2007.

[46] J. Harel, C. Koch, and P. Perona, "Graph-Based Visual Saliency," in *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, Cambridge, MIT Press, 2007, pp. 545-552.

[47] A. Borji, D. N. Sihite and L. Itti, "Salient Object Detection: A Benchmark," in *ECCV*, Florence, 2012.

[48] M. Cerf, E. P. Frady and C. Koch, "Faces and text attract gaze independent of the task: Experimental data and computer model," *Journal of Vision*, vol. 9, no. 12, pp. 1-15, 2009.

[49] D. Swets and J. Green , Signal Detection Theory and Psychophysics, New York: Wuley, 1966.

[50] J. Zhu, Y. Qiu, R. Zhang, and J. Huang, "Top-Down Saliency Detection via Contextual Pooling," *Journal of Signal Processing Systems*, vol. 74, no. 1, pp. 33-46, 2014.

[51] D. Gao, S. Han, and N. Vasconcelos, "Discriminant Saliency, the Detection of Suspicious Coincidences, and Applications to Visual Recognition," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE,* vol. 31, no. 6, pp. 989-1005, 2009.

[52] j. Yang, and M.H. Yangm , "Top-Down Visual Saliency via Joint CRF and Dictionary Learning," in *CVPR*, Providence, 2012.

[53] D. Langli, S. Chartier, and D. Gosselin, "An Introduction to Independent Component Analysis: InfoMax and FastICA algorithms," *Tutorials in Quantitative Methods for Psychology,* vol. 6, no. 1, pp. 31-38, 2010.

[54] G. R. Naik and D. K. Kumar, "An Overview of Independent Component Analysis and Its Applications," *Infomatica,* vol. 35, pp. 63-81, 2011.

[55] C. Bugli and P. Lambert, "Comparison between Principal Component Analysis and Independent Component Analysis in Electroencephalograms Modelling," *Biometrical Journal,* vol. 48, no. 5, pp. 1-16, 2006.

[56] A. Hyvarinen and E. Oja, "Independent Component Analysis A Tutorial ," *Neural Networks,* vol. 13, no. 4-5, pp. 411-430, 2000.

[57] S. Amari, A. Cichocki, and H. H. Yang, "A new Learning Algorthm for Blind Signal Seperation," *Advances in Neural Information Processing Systems,* vol. 8, pp. 757-763, 1996.

[58] C. E. Shannon, "A Mathematical Theory of Commuiation," *The Bell System Technica Journal,* vol. 27, pp. 379-423, 1948.

[59] I. Robertson, "Calculating Percentile," Stanford University, 09 Jan. 2004. [Online]. Available: web.stanford.edu/class/archive/anthsci/.../calculating%20percentiles.pdf. [Accessed 15 Jun 2014].

[60] J. A. Bilme, "A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models," University of California, Berkeley, 1998.

[61] G. D. Finlayson, B. Schiele, J. L. Crowley, "Comprehesive Colour Image Normalization," in *ECCV*, Freiburg, 1998.

[62] Y. Ye and J. K. Tsotsos, "A Complexity-Level Analysis of The Sensor Planning Task for Object Search," *Computational Intelligence,* vol. 17, no. 4, 2001.

[63] Y. Ye and J. K. Tsotsos, "Sensor planning for 3D Object Search," *Computer Vision and Image Understanding,* vol. 73, no. 2, pp. 145-168, 1999.

[64] S. Martello and P. Toth, KANPSAK PROBLEMS, Chichester: John Wiley & Sons Ltd., 1990.

[65] C. Cotta and J. M. Troya, "A Hybrid Genetic Algorithm for the 0-1 Multiple Knapsack Problem," *Artificial Neural Nets and Genetic Algorithms,* vol. 3, pp. 250-254, 1998.

[66] D. Pisinger, "A Minimal Algorithm for the Bounded Knapsack Problem," *Journal on Computing,* vol. 12, no. 1, pp. 75-82, 2000.

[67] L. Moura and F. Dos Santos, *An Efficient Dynamic Programming Algorithm For The Unbounded Knapsack problem*`, Porto Alegre: Journal on Computing, 2012.

[68] J. Puchinger, G. R. Raidl, and U. Pferschy, "The Multidimensional Knapsack Problem: Structure and Algorithms," *INFORMS Journal on Computing,* vol. 22, no. 2, pp. 250-265, 2010.

[69] J. Noga and V. Sarbua, "An Online Partially Fractional Knapsack Problem," in *ISPAN*, Las Vegas, 2005.

[70] M. Bartlett, A. M. Frisch, Y. hamadi, I. Miguel, S. A. Tarim, and C. Unsworth, "The temporal knapsack problem and its soution," in *CPAIOR*, Berlin, 2005.

[71] I. Aho, "Interactive Knapsacks," *Fundamenta Infmaticae*, vol. 44, no. 1-2, pp. 1-23, 2000.

[72] A. J. Kleywegt and J. D. Papastavrou, "The Dynamic and Stochastic Knapsack Problem," *Operations Research*, vol. 46, pp. 17-25, 1998.

[73] S. G. Kolliopoulos and G. Steiner, "Partially-Ordered Knapsack and Applications to Scheduling," *Discrete Applied Mathematics*, vol. 155, no. 8, pp. 889-897, 2007.

[74] Y. Merzifonluoglu, J. Geunes, and H. E. Romeijn, "The static stochastic knapsack problem with normally distributed item sizes," *Mathematical Programming*, vol. 134, no. 2, pp. 459-489, 2012.

[75] D. Kozen and S. Zaks, *Automata, Languages and Programming*, vol. 700, pp. 150-161, 1993.

[76] B. Bhowmik, "DYNAMIC PROGRAMMING – ITS PRINCIPLES, APPLICATIONS, STRENGTHS, AND LIMITATIONS," *International Journal of Engineering Science and Technology,* vol. 2, no. 9, pp. 4822-4826, 2010.

[77] E. Horowitz and S. Sahni, "Computing Partitions with Applications to the Knapsack Poblem," *Journal of the ACM,* vol. 21, no. 2, pp. 277-292, 1974.

[78] J. Csirik, J. B. G. Frenk, M. Labbe, and S. Zhang, "Heuristics for the 0–1 min-knapsack problem," *Acta Cybernetica,* vol. 10, no. 1-2, pp. 15-20, 1991.

[79] H. Kellerer, U. Pferschy, and D. Pisinger, Knapsack Poblems, Berlin: Springer-Verlag, 2004.

[80] D. Stefankovic, S. Vempala, and E. Vigoda, "A Deterministic Polynomial-Time Approximation Scheme for Counting Knapsack Solutions," *SIAM Journal on Computing,* vol. 41, no. 2, pp. 356-366, 2012.

[81] P. Gopalan, A. Klivans, R. Meka, D. Stefankovic, S. Vempala, adn E. Vigoda, "An FPTAS for #Knapsack and Related Counting Problems," in *IEEE FOCS*, Palm Springs, 2011.

[82] J. MacLean and J. K. Tsotsos, "Fast Pattern Recognition Using Gradient-Descent Search in an Image Pyramid," *Pattern Recognition*, vol. 2, pp. 873-877, 2000.

[83] P. Hoyer, A. Hyvarinen, and N. Group, "Independent Component Analysis and its Extensions as Models of Natural Image Statistics of Natural Image Statistics," University of Helsink, 15 Feb 2000. [Online]. Available: http://research.ics.aalto.fi/ica/imageica/. [Accessed 15 June 2014].