Learning Search Heuristics for Finding Objects in Structured Environments

Dominik Joho*, Martin Senk, Wolfram Burgard

University of Freiburg, Department of Computer Science, Georges-Köhler-Allee 79, D-79110 Freiburg, Germany

Abstract

We consider the problem of efficiently finding an object with a mobile robot in an initially unknown, structured environment. The overall goal is to allow the robot to improve upon a standard exploration technique by utilizing background knowledge from previously seen, similar environments. We present two conceptually different approaches. Whereas the first method, which is the focus of this article, is a reactive search technique that decides where to search next only based on local information about the objects in the robot's vicinity, the second algorithm is a more global and inference-based approach that explicitly reasons about the location of the target object given all observations made so far. While the model underlying the first approach can be learned from data of optimal search paths, we learn the model of the second method from object arrangements of example environments. Our application scenario is the search for a product in a supermarket. We present simulation and real-world experiments in which we compare our strategies to alternative methods and also to the performance of humans.

Keywords: Object Search, Search Heuristics, Sequential Decision Making, Object Maps

1. Introduction

Consider the situation where you want to find a product in a supermarket that you have never been to before. Certainly, you will not just wander around randomly through the market nor will you systematically visit each so far unvisited aisle in the market until you find the product. Your search will rather be guided by the current observations and the expectations you have about how objects in supermarkets are usually arranged. Over time, you might even have developed some heuristics that proved to be useful for quickly finding a certain product, like "if you want to find yogurt, follow the aisle with the cooling shelves."

The search for a product in an unknown supermarket is a problem that everyone is familiar with and it therefore is an illustrative instance of the kind of search problems we want to tackle with the techniques presented in this article. Even for humans [1] this task is not an easy one and we will therefore also compare our search techniques to the performance of human participants that took part in a field study conducted in a real supermarket [2]. However, the supermarket is just an example scenario. Searching for objects or places in offices or domestic environments is conceptually similar. All we assume is that the environment is structured and the object arrangements exhibit some spatial dependencies such that a generalization to an unknown yet similarly structured environment is possible. We regard this as a rather weak assumption that holds for a huge variety of man-made environments. We thus strive for a general way to model, learn, and utilize background knowledge such that a mobile robot is able to find an object more efficiently than it would have been possible without such domain-specific knowledge. For this, we present and evaluate two approaches and provide alternative views on the search problem.

The first approach is a reactive search technique that only depends on local information about the objects in the robot's vicinity when deciding where to search next. This approach emphasizes the sequential nature of the search process, which is a sequential decision making process. Being in a certain state we must choose among a set of available actions. In this setting, background knowledge can be encoded as a state-to-action mapping, a policy, that tells us what to do in a certain situation. In the supermarket scenario, a state includes the currently observed objects in direction of the different aisles and the available actions correspond to the aisle we may choose to visit next. To learn this state-to-action mapping, we draw on ideas from imitation learning [3]. In particular, we want to imitate a simulated robot that exhibits an optimal search behavior by approaching the target object on the shortest path. In each visited state of a demonstrated example path, the robot takes a certain action and discards the other available actions in this state. Thereby, it provides positive and negative examples of state-action pairs to be taken or not, respectively. These examples can be used to learn a classifier for state-action pairs, which yields a classifier-based policy representation [4]. This might either be a multi-class classifier that directly outputs the action to be taken in a given state, or it might be a binary classifier that labels each available action in a state as promising or non-promising (if there is a tie, we may choose randomly among the promising actions). The latter has been empirically shown [4] to yield policies that perform better than the ones that

*Corresponding author.

Email addresses: joho@informatik.uni-freiburg.de (Dominik Joho), senk@informatik.uni-freiburg.de (Martin Senk), burgard@informatik.uni-freiburg.de (Wolfram Burgard)

are represented by multi-class classifiers. We use decision trees as binary classifiers which result in compact policy representations that resemble search heuristics like the above-mentioned heuristic for finding yogurt.

The second approach treats the search problem as an inference problem. This is motivated by the observation that we are constantly reasoning about the location of the object while searching for it. This reasoning process will be influenced by the thus far observed objects and structure of the environment as well as our expectations about usual object arrangements in such environments. In the supermarket scenario this means, for example, that if we are searching for beer and in one aisle we observe milk we may conclude that the beer is probably not in the same aisle. In this setting, background knowledge is encoded as expectations about how objects co-occur. However, co-occurrence of objects can only be defined with regard to a spatial context – like objects being "in the same aisle", or one object being "in the neighboring aisle" of the other. Each particular spatial context induces a different local co-occurrence model. In general, there is no single best spatial context and object arrangements in real-world environments are too complex to be faithfully represented by any of these rather basic models alone. Nevertheless, each local model captures useful statistical properties of such object arrangements. Based on these considerations and motivated by the idea of combining an ensemble of base classifiers to form a more robust classifier, we proceed as follows: we use a diverse set of local co-occurrence models, each considering a different spatial relation, and fuse their outcomes as features in a maximum entropy model (MaxEnt) [5, 6] which in our case models the discrete distribution over all possible locations of the target object. The robot then essentially moves to the location which most likely contains the target object. Each time new information becomes available, e.g., newly detected objects or newly discovered parts of the environment, the robot recomputes the distribution.

These two approaches have quite different properties. The first approach uses only local information, as it depends only on the objects in the vicinity of the robot, while the second takes into account all observations made so far. Furthermore, the underlying model of the first approach is learned by observing optimal search behavior, while the model of the second is learned from object arrangements of similarly structured example environments.

This article is organized as follows. After discussing related work, Section 3 introduces the representation of the supermarket environments utilized by the first approach, the decision-tree strategy. Section 4 describes how the search heuristics of this strategy have been learned from data of optimal search paths. Section 5 then describes several alternative search strategies, including variants of the decision-tree strategy, the inference-based approach, and an exploration strategy that serves as a baseline approach. Finally, in Section 6 we present the results of an experimental evaluation including simulation and real-world experiments. The results demonstrate that our proposed techniques yield significantly shorter search paths than a search strategy that does not take domain-specific information into account.

2. Related Work

There exists considerable theoretical work on general search problems in the fields of robotics and artificial intelligence [7] as well as operations research [8]. Finding an optimal search path in a graph that either minimizes the expected time to detection [9] or the expected search costs [10] is known to be NP-hard. Besides complexity considerations in theoretical work, some prior work evaluated proposed search strategies in simulation. The approach presented in [11], for example, used a computationally involved dynamic programming technique for planning an optimal search path to find multiple stationary targets. In [12], a framework was proposed that additionally allows to reason about the possible absence of the target in the search area. In contrast to these works, we additionally assume that the environment is initially unknown. Most of these approaches allow to incorporate background knowledge as a prior distribution over the target location. But this distribution is assumed to be given in advance and then updated during the search based on simple presence or absence detections. Our work additionally aims at modeling such a distribution based on object co-occurrences, or by implicitly incorporating background knowledge as search heuristics.

Thus, a further related problem is how to model expectations about object arrangements in indoor environments. In prior work this has been realized by modeling different types of places in terms of object counts and inter-object distances [13], by utilizing object co-occurrence statistics [14], by utilizing a full 3D constellation model [15], or by using a manually designed ontology about indoor environments [16, 17]. Of these works, only [14], and to some extent [17], also considered the search problem and used their model for improving search efficiency.

To the best of our knowledge, modeling background knowledge about indoor environments for improving search efficiency has received far less attention in the literature so far. In [14], known object locations in a given global map were used for efficiently finding another object at an unknown location. A Markov random field based on statistics of object co-occurrences was used to infer a likelihood map and to plan a search path that minimizes the expected search path length. Again, this is a work that assumed that the structure of the environment, and some of the objects therein, are known. In [18], the application scenario was to efficiently find the entrance hall in a hotel. A relational navigation policy was learned which utilized information about the type of rooms and corridors that are directly connected to the robot's current location. The work presented in [17] aimed at improving the task planning of a mobile robot by relying on semantic information about its domain. In particular, they defined an ontology about typical home-like environments and generated plans to find unseen objects or type of rooms, e.g. a bedroom.

3. Modeling the Environment

A supermarket $m \in \mathcal{M}$ contains a set of shelves $\mathcal{S}_m \subset \mathcal{S}$ and a graph $\mathcal{G}_m = (V, E)$, as illustrated in Fig. 1. Each shelf $s \in \mathcal{S}$

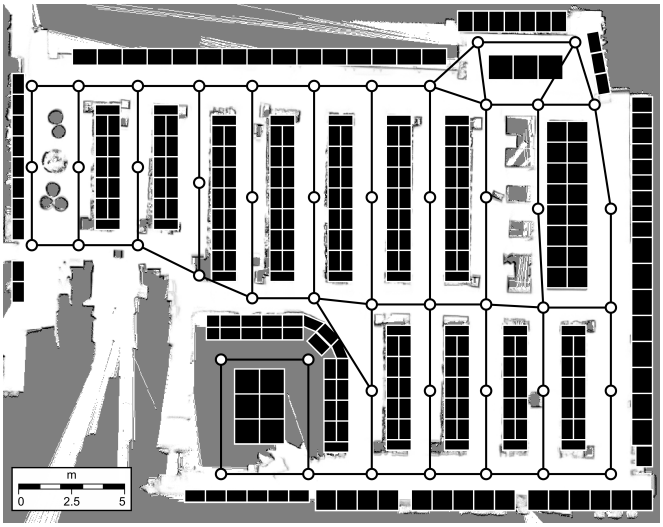


Figure 1: Example map of a real supermarket environment. In our approach, maps contain shelf locations (black boxes) and the products within the shelves. The underlying structure is a graph. The task of the robot is to efficiently find a certain product.

is associated with a location $\boldsymbol{\ell}_s = (x_s, y_s)$ and an orientation θ_s. The relation $\texttt{inMarket} \subset \mathcal{S} \times \mathcal{M}$ associates each shelf with its corresponding market. Furthermore, we define a set of shelf types $\mathcal{T} = \{\texttt{Normal}, \texttt{Cooling}, \texttt{Freezer}, \texttt{Counter}, \texttt{Grocery}\}$ and each shelf is associated with exactly one type as defined by the relation $\texttt{type} \subset \mathcal{S} \times \mathcal{T}$. Each shelf contains at least one product and the same product might be placed in several shelves, as defined by the relation $\texttt{inShelf} \subset \mathcal{P} \times \mathcal{S}$. The relation $\texttt{categOf} \subset \mathcal{P} \times C$, associates each product with a product category.

For the experiments carried out in this paper, we used a set of 196 products at the granularity of small categories like "sugar", "pizza", "apples", "tea", etc. We furthermore used 20 product categories with a coarser granularity like "breakfast", "dairy products", "vegetables & fruits", etc.

The nodes V of a graph $\mathcal{G}_m = (V, E)$ model the decision points in the supermarket and the directed edges define the reachability between decision points. While the reachability could have been modeled with undirected edges, the visibility of the shelves also depends on the current node (the robot's current location) and therefore is defined over directed edges. We use two variants of a visibility relation that defines the shelves that are visible when looking into the direction of a certain edge. The first one is a long range variant $\texttt{shelfVisL} \subset E \times \mathcal{S}$ and the second is a short range variant $\texttt{shelfVisS} \subseteq \texttt{shelfVisL}$. This is motivated by the fact that, although certain information, like the type of a shelf, can be determined reliably over long distances, some information can only be determined when one is in close vicinity to a shelf, like for example the products contained within a shelf. Three example situations illustrating the short range visibility are depicted in Fig. 2. On the basis of the two visibility relations we define several other visibility relations, like the visible products

$$\texttt{prodVis} = \{(e, p) \mid \texttt{shelfVisS}(e, s), \texttt{inShelf}(p, s)\}, \quad (1)$$

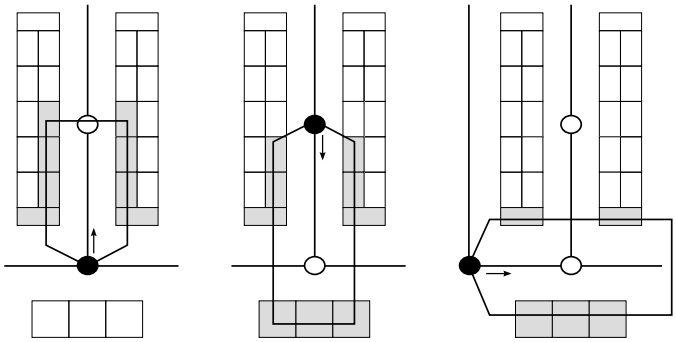


Figure 2: Three example situations for illustrating the short range visibility. Whereas gray shelves are visible, white shelves are not visible. The location of the robot is indicated by the black node and its orientation is indicated by the arrow.

and the visible product categories

$$\texttt{categVis} = \{(e, c) \mid \texttt{prodVis}(e, p), \texttt{categOf}(p, c)\}. \quad (2)$$

The visibility of shelf types is modeled in such a way that we can distinguish whether the shelf type is seen in the direct vicinity, or being observed at a further distance:

$$\texttt{typeVisS} = \{(e, t) \mid \texttt{shelfVisS}(e, s), \texttt{type}(s, t)\} \quad (3)$$

$$\texttt{typeVisL} = \{(e, t) \mid \texttt{shelfVisL}(e, s), \texttt{type}(s, t)\}. \quad (4)$$

The proposed search strategy utilizes the information associated with each edge to decide which edge to follow. In the next section we describe how we learn such a strategy from data by observing optimal search paths.

4. Learning Search Heuristics

We are interested in learning a reactive search strategy that depends only on local information in order to find a certain target product. We therefore classify the outgoing edges of the current node by a decision tree into promising and non-promising directions based on the information associated with each edge. For learning such a decision tree, we first need to define appropriate edge attributes and then generate training data by observing optimal search paths in a training set of supermarkets. To evaluate the strategy, we apply it to a previously unseen market.

4.1. Defining Edge Attributes

One obvious piece of information, by which the search should be guided, is which products and product categories are visible at a certain edge. If we are searching for coffee and an aisle contains tea, or in general breakfast products, then this edge is certainly a promising candidate. But the decision should also be influenced by additional factors. If we know that an edge has been visited already, we can reject it in order to avoid loops. Also the type of an edge might be of interest, such as if an edge belongs to an aisle that follows a wall (wall aisle), because some products, like milk, are only located in such aisles. Likewise, we define main aisles as aisles that follow a main direction in a market and from which many narrow side aisles

branch off. Next, it is informative if the robot is approaching certain landmarks in the supermarket, like the entrance, the exit, or the back of the market. Vegetables, for example, are always located near the entrance in our markets. Thus, each optimal search path for finding apples would mostly contain edges that are approaching the entrance. Likewise, frozen food is usually in the back of the market and wine and non-food are near the exit of the market.

We also use statistics about the expected relative product position between the entrance and the exit based on the layout of all training markets. The relative position of a shelf s with respect to the location $\boldsymbol{\ell}_{en}$ of the entrance node and the location $\boldsymbol{\ell}_{ex}$ of the exit node of the corresponding market is defined as

$$\text{relPos}(s) = \frac{\|\boldsymbol{\ell}_s - \boldsymbol{\ell}_{en}\|}{\|\boldsymbol{\ell}_s - \boldsymbol{\ell}_{en}\| + \|\boldsymbol{\ell}_s - \boldsymbol{\ell}_{ex}\|}. \quad (5)$$

The expected relative position of a *product* is then defined as the average of these values for all shelves that contain this product in the training markets M_t

$$S_p = \{s \mid \texttt{inShelf}(p, s), \texttt{inMarket}(s, m), m \in M_t\} \quad (6)$$

$$\text{expRelPos}(p) = \left|S_p\right|^{-1} \sum_{s \in S_p} \text{relPos}(s). \quad (7)$$

We define a binary edge attribute (No. 222 in Table 1) that indicates if the robot would be approaching the expected relative position of the target product by following that edge.

Furthermore, we calculate the average Euclidean distance $\text{prodDist}(p_i, p_j)$ for each pair (p_i, p_j) of products based on their locations in the training markets. If we denote by $P_{i,j}$ the visible products that are associated with an outgoing edge $e_{i,j}$ from the current node v_i to a possible successor node v_j, then the average product distance of this edge to the target product p_t is defined as

$$\text{avgProdDist}(e_{i,j}, p_t) = \left|P_{i,j}\right|^{-1} \sum_{p \in P_{i,j}} \text{prodDist}(p_t, p). \quad (8)$$

We define an indicator attribute (attribute No. 223) that is set to true if an edge has the lowest average product distance of all outgoing edges of the current node, and thus can be considered to be the most promising edge with respect to the expected product distances. Likewise, we define an attribute that uses the path distance on the graph between products instead of the Euclidean distance (attribute No. 224). As it is not easy to decide beforehand whether the path distance or the Euclidean distance is a more reliable indicator for product distances we use both attributes and let the learning algorithm decide which one to use during the induction of the tree. A complete list of all attributes can be seen in Table 1.

4.2. Generating Training Data

We use a fixed set of 15 target products. These are the same products that human participants had to find in a field study conducted in the very same supermarket in which we will evaluate our strategy. We learn a separate decision tree for each of these 15 target products.

Table 1: The attributes that are used to characterize an edge. All attributes are binary. In the experimental evaluation we test different combinations of subsets (a–d) of these attributes.

Subset	Att. No.	Description
a	1	Edge already visited
a	2–197	Product $p_i \in \mathcal{P}$ visible
a	198–217	Product of category $c_i \in C$ visible
a	218	Shelf of type `Normal` visible (short range)
a	219	Shelf of type `Cooling` visible (short range)
a	220	Shelf of type `Freezer` visible (short range)
a	221	Shelf of type `Counter` visible (short range)
b	222	Leads to expected relative position
b	223	Has smallest avg. Euclidean distance to product
b	224	Has smallest avg. path distance to product
c	225	Current node belongs to a main aisle
c	226	Next node belongs to a main aisle
c	227	Next node belongs to a wall aisle
c	228–230	Leads to the entrance, exit, or back of the market
d	231	Shelf of type `Cooling` visible (long range)
d	232	Shelf of type `Freezer` visible (long range)
d	233	Shelf of type `Grocery` visible (long range)

We determine for each node in a training supermarket the shortest path to a given target product. Each node of an optimal path corresponds to a local decision for taking a certain outgoing edge (the one that leads to the next node of the optimal path) and for rejecting all other outgoing edges of that node. In this way, each optimal search path contributes a set of positive and negative examples of edges to be taken or not, respectively. Fig. 3 illustrates the basic idea. The positive and negative examples of all paths for all starting positions in all training supermarkets then constitute the training data for learning the decision tree for a given target product.

As there might exist more than one optimal path from a starting location to the target location, we search for more than just a single shortest path to generate training data. Additionally, as the decision points are placed manually, there might be small differences between nearly optimal paths. For this reason, we also generate training data from paths, which are not longer than a given small threshold when compared to the actual shortest path.

4.3. Decision Tree Learning and Pruning

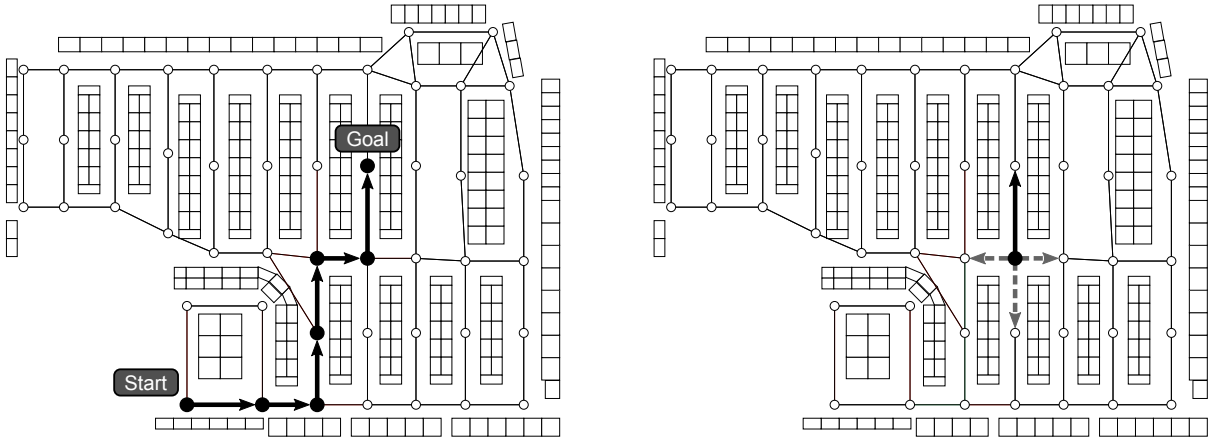
We use the well known ID3 algorithm [19] to learn a decision tree. For convenience, we restate the basic idea of the algorithm. The tree is constructed top-down and each node is associated with a set of positive e_p and negative e_n examples and a set A of yet untested attributes. At each node an attribute $a \in A$ is chosen that maximizes the information gain

$$G(a) = I\left(\left|e_p\right|, \left|e_n\right|\right) - \sum_{v \in a} \frac{\left|e_{p(v)}\right| + \left|e_{n(v)}\right|}{\left|e_p\right| + \left|e_n\right|} I\left(\left|e_{p(v)}\right|, \left|e_{n(v)}\right|\right) \quad (9)$$

where

$$I(p, n) = -\frac{p}{p+n} \log_2 \frac{p}{p+n} - \frac{n}{p+n} \log_2 \frac{n}{p+n} \quad (10)$$

denotes the information entropy and $e_{p(v)}$ and $e_{n(v)}$ are the set of positive and negative examples, respectively, where attribute a



(a) Shortest path between starting location and goal location.

(b) Local decision at one of the nodes along the path.

Figure 3: (a) For generating training data, we compute the shortest path from every possible starting location in the market for a given goal location of a target product. One such path is depicted above. (b) Each node of an optimal path corresponds to a local decision for taking a certain outgoing edge (black solid arrow) and for rejecting all other outgoing edges of that node (gray dashed arrows). In this way, each optimal search path contributes a set of positive and negative examples of edges to be taken or not, respectively.

has the value v. If the examples of a node belong to one class only, the node becomes a leaf node with the respective class. If no other attributes are left, the class of a leaf node is defined by the majority vote of the associated examples.

A technique to avoid overfitting in decision tree learning is to prune the learned tree. In the experimental section we will therefore also investigate the influence of two pruning techniques, namely a simple restriction on the maximum depth of the tree (max-depth-pruning, MDP) and reduced error pruning (REP) [20]. In MDP, every subtree that has its root node at a given depth of the original tree will be collapsed into a leaf node. For REP, we need to divide the training data set into an induction set, which is used during induction of the decision tree, and a pruning set, which is used to evaluate which part of the tree should be pruned. REP then replaces any subtree with a leaf node if this does not lead to a higher classification error on the pruning data set.

The learned decision tree is then used to guide the search for the target product by classifying each outgoing edge of the robot's current location into promising and non-promising directions. It may happen that more than one edge will be classified as promising. In this case, we choose randomly among the different candidates. If there are no promising edges, we randomly choose among the unvisited edges. If all outgoing edges have been visited already, the robot moves on the shortest path to the nearest known node, which has at least one unvisited outgoing edge.

5. Alternative Search Strategies

Besides discussing some variants of our proposed strategy based on decision trees, we describe two other search strategies – including the inference-based approach – that we evaluate in comparison to the decision tree strategy. The results will be presented in the next section.

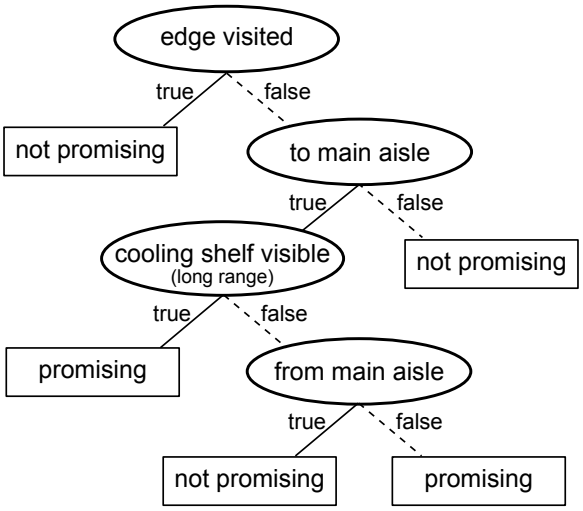
5.1. Exploration Strategy

As a baseline approach, we use an exploration strategy that selects randomly among the unvisited edges at the current node. If all outgoing edges of a node have been visited already, an edge will be chosen that leads to the nearest node with at least one unvisited edge. This strategy rapidly explores unvisited areas and avoids searching the known parts of the environment. This is akin to the frontier-based exploration strategy [21] known in mobile robot exploration. If a search technique does not perform better than the exploration technique, it obviously is not able to utilize domain-specific information, which is the ambition of our strategy.

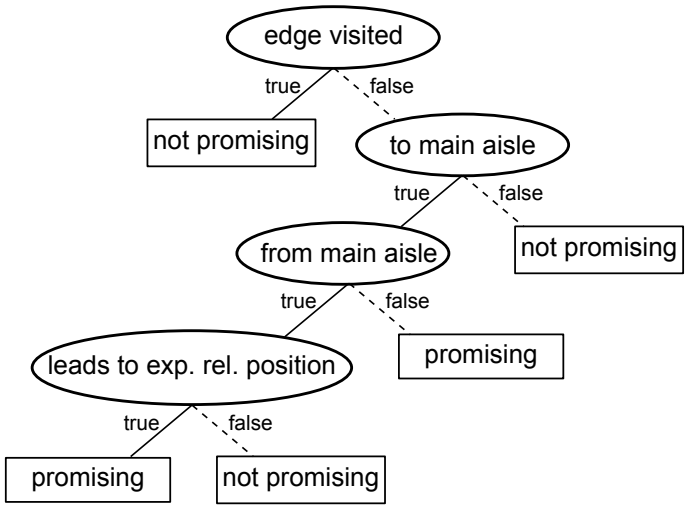
The exploration strategy as a baseline approach allows us to relate the performance of the search strategies to an expected upper bound on the search path length, defined by the average search path length of the exploration strategy, and to a strict lower bound defined by the shortest path.

5.2. Variants of the Decision Tree Strategy

In total, we evaluate five variants of the decision tree strategy. The first four variants differ by the set of attributes they are allowed to use. We start from a simple variant, which uses only subset "a" of the attributes (see Table 1), while the three subsequent variants can use increasingly more attributes (including subsets "b", "c", and "d"). The resulting decision trees are not pruned in any way and therefore might be prone to overfitting. We therefore also investigate the influence of two pruning techniques. We tried several alternatives by restricting the maximum depth of the trees to different levels (MDP) or by applying reduced error pruning (REP), or a combination of both to any of the four attribute subset variants. We found the best variant to be a combination of both pruning techniques applied to a tree that uses the full set of attributes. We first applied MDP using a maximum depth of four and then additionally applied REP. To do so, the training data set was split into an induction set (75%



(a) Decision tree for finding yogurt.



(b) Decision tree for finding UHT milk.

Figure 4: Two examples of pruned decision trees that have been learned from optimal search paths in the training supermarkets. The trees use the attribute variant (a–d, pruned) mentioned in Table 2.

of the data) and a pruning set (25% of the data). Two examples of learned and pruned decision trees can be seen in Fig. 4.

5.3. MaxEnt Search Strategy

As already mentioned, we also consider an additional search strategy which is conceptually rather contrary to the decision tree search strategy. Due to lack of space, we can only briefly explain this method here and refer the interested reader to [22] for a more detailed description.

Instead of making decisions reactively based only upon locally available information, this second strategy will take all observations made throughout the search into account when deciding where to search next. This will make it necessary to maintain and update a global map of the structure of the environment and the locations of the seen products throughout the search process. The strategy decomposes the action selection problem during the search into two parts. First, the strategy computes a discrete belief $P(x \mid \mathbf{z})$ over possible locations $x \in X$ of the target object o_q that the robot is searching for, given the observations $\mathbf{z}$ made so far. Subsequently, the belief distribution is used to select the next location the robot should visit.

Possible product locations $x \in X$ correspond to the supermarket shelves in our scenario. The observations $\mathbf{z}$ describe how the locations of observed products are related to the observed shelves in the market, e.g., if they are in "the same aisle" or in "the neighboring aisle" of a shelf $x \in X$. To be more specific, an observation z in our model corresponds to a tuple $z = (x_l, r_j, A_i, a_k)$ that states that an object with the attribute key A_i and attribute value a_k has been observed at a location that is related to the shelf location x_l by the spatial relation r_j. Thus, a single newly detected object introduces several basic observations z. By $\mathbf{z}$ we denote all observations made so far and $\mathbf{z}_{|x_l,r_j,A_i}$ denotes the subset of observations that are constraint to have values x_l, r_j, and A_i. The object attributes should not be mistaken for the edge attributes mentioned above. They rather describe general properties of the products, like if they are edible, if they need to be cooled, etc. (see Fig. 5 for an illustration of the underlying idea).

At its heart, the model is based on background knowledge about the co-occurrence of objects and object attributes in different spatial contexts (defined by spatial relations r_j). This background knowledge is expressed by discrete conditional probability distributions $P(A_i = a_k \mid o_q, r_j)$ that specify the probability of the following event: given that object o_q exists at some location, then there will exist another object with attribute $A_i = a_k$ at any location that is related to the location of o_q by the spatial relation r_j. For example, we might ask that under the assumption that the "coffee" that we are searching for is in shelf x, what is the probability that we observe a "breakfast product" in the "same aisle" as shelf x. Additionally, we need to model $P(A_i = a_k \mid \neg o_q, r_j)$ that we will see the attribute in a related location, given the object is *not* present. These probabilities can be estimated directly from the layouts of the training supermarkets.

To use this background knowledge for computing the desired final distribution $P(x \mid \mathbf{z})$, we follow a two step process. The basic idea is to use an ensemble of local co-occurrence models, each considering only a certain aspect of the observations, and then to fuse the local models in a combined model over all possible locations. This is motivated by the assumption that the distribution of objects in real-world environments is too complex to be faithfully captured by just a single model and it therefore would be beneficial to combine a diverse set of more simple models. These local models compute the binary probability $L_{A_i,r_j}(x \mid \mathbf{z})$ that the object exists at a certain location x versus that it does not exist at this location $L_{A_i,r_j}(\neg x \mid \mathbf{z})$. Each local model considers only a certain attribute A_i of those observations $\mathbf{z}_{|x,r_j,A_i}$ that are related to x by the relation r_j. Now let $\mathbf{a}(\mathbf{z})$ denote the set of attribute values that occur in the observations $\mathbf{z}$. Then we model the local models as binary naive Bayes

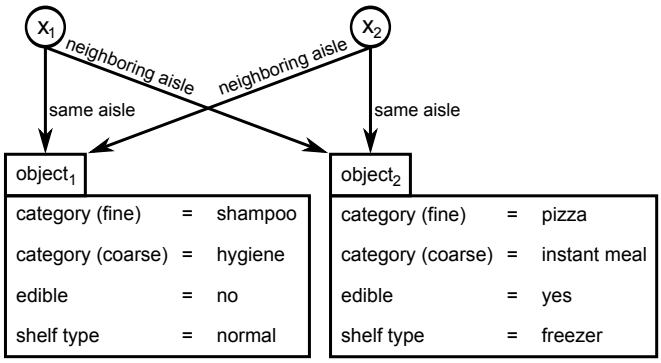


Figure 5: Illustration of the basic idea for inferring the product location in the MaxEnt search strategy: Possible locations (x_1, x_2) of the product which the robot is searching for are linked to locations of already seen products $(\text{object}_1, \text{object}_2)$ by different spatial relations (r_1 = "same aisle", r_2 = "neighboring aisle", etc.). Products are described by four different attribute keys (A_1 = "Category (fine)", etc.).

classifiers as follows:

$$L_{A_i,r_j}(x \mid \mathbf{z}) = \frac{P(x) \prod_{z \in \mathbf{z}_{|x,r_j,A_i}} P(z \mid x)}{\sum_{x' \in \{x,\neg x\}} P(x') \prod_{z \in \mathbf{z}_{|x,r_j,A_i}} P(z \mid x')} \tag{11}$$

$$= \frac{\prod_{a \in \mathbf{a}(\mathbf{z}_{|x,r_j,A_i})} P(A_i = a \mid o_q, r_j)}{\sum_{o' \in \{o_q,\neg o_q\}} \prod_{a \in \mathbf{a}(\mathbf{z}_{|x,r_j,A_i})} P(A_i = a \mid o', r_j)}. \tag{12}$$

In (12) we dropped the prior $P(x)$, which we assume to be uniform. The log-likelihood of these local models will be used as features f_{A_i,r_j} in a maximum entropy model (MaxEnt) [5, 6]

$$P(x \mid \mathbf{z}) = \frac{\exp\left(\sum_{A_i,r_j} \lambda_{A_i,r_j} f_{A_i,r_j}(x, \mathbf{z})\right)}{\sum_{x'} \exp\left(\sum_{A_i,r_j} \lambda_{A_i,r_j} f_{A_i,r_j}(x', \mathbf{z})\right)}, \tag{13}$$

with feature weights λ_{A_i,r_j} and defined over all possible locations $x \in X$ of the target object. In [22] we additionally evaluated different mappings between the output L_{A_i,r_j} of the local models and the features f_{A_i,r_j}.

The feature weights λ_{A_i,r_j} can be learned in a supervised way by maximizing the likelihood of a training data set using gradient ascent. This is a convex optimization problem and the feature weights will therefore approach a global optimum. A single training example consists of a supermarket layout, the target object o_q, and the correct "label" $x \in X$, which corresponds to the shelf that contains the target product. We have four supermarkets and 141 products that are available in all four markets. This yields 4×141 training examples. Because we train a single set of weights, the learned weights reflect the general importance of each local model – independent of a specific target product or market.

Based on this distribution $P(x \mid \mathbf{z})$ over possible product locations and by considering the distance necessary to reach a location the robot computes a utility for each node of the graph and moves to the node with the highest utility. As soon as new information is available, such as newly detected shelves or products, the belief and the utilities will be re-evaluated.

6. Experimental Evaluation

In the following, we present several experimental evaluations. The first experiment is aimed at comparing the performance of the different search strategies in comparison to the performance of humans searching in a real supermarket environment. The second experiment is aimed at a more thorough evaluation of the search strategies, though we do not have data from human participants for this setting. And finally, we present the results obtained by real-world experiments in which a robot autonomously searched for a product using the presented decision tree strategy.

The supermarket data, including the layouts and the product placement, was collected in real supermarkets. Three of the supermarkets were used as a training set for learning the underlying models (decision trees, local models) of the search strategies and the fourth supermarket was used for evaluation of the strategies.

6.1. Evaluation in Comparison to Humans

A field study involving 38 human participants was conducted in a real supermarket [2]. The participants had to find 15 products in a given order and we used the same 15 products as target products in our simulated search. As the supermarket in which the study took place was the same market that we used as a model for our evaluation market, we can compare the path distances of the human participants to the path distances traveled by the robot in the simulated environment. In order to assure that we have a metrically comparable model of the real market, we first built an occupancy grid map of the supermarket using a laser-based FastSLAM implementation [23] and then placed the shelves according to the grid-map, as can be seen in Fig. 1. The product placement in our virtual markets also resembles the product locations in the real markets. The participants were tracked using a RFID-based localization technique [24] and the resulting trajectories were then mapped upon the graph for a fair comparison with the path distances of the simulated robot.

The human participants had to find the 15 products in a given order (see Fig. 6), and so the location of a found target product was the starting location for the search for the next target product. Therefore, each target product was associated with a certain starting location and we evaluated the *simulated* search strategies for the same 15 pairs of starting location and target product.

As a performance measure we consider the length of a complete search path, that is the path length of a search for all 15 products. We simulated a thousand search trials for the exploration strategy and the decision tree strategies. The MaxEnt search strategy is deterministic. We only have a sample size of 26 complete search trials of the human participants, because some search sub-trials (for a single product) have been canceled if the search took too long or the participants gave up. This introduces a slight bias to the comparison for the benefit of the human participants, because the *simulated* search trials were not canceled if they took "too long". Nevertheless, we think that the available data of the human search paths still constitutes a usable basis for a comparison.

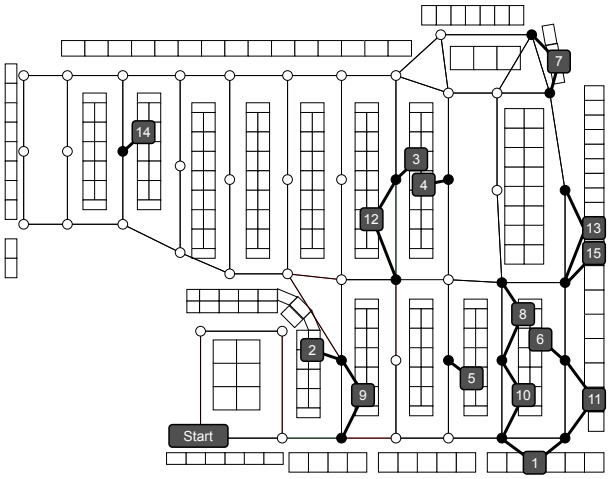


Figure 6: In the first experiment the simulated robot and the human participants had to find 15 products in a given order. They start at the entrance of the market in the lower left corner. The product locations are connected to their respective target nodes on the graph. If the robot enters a target node of the current product, the product is found and the robot will search for the next one.

In Table 2 we present the mean and standard deviation of the search path lengths. We performed a one-tailed paired t-test[1] for the sampled strategies and found all improvements indicated by the means to be significant at the 0.01 level, except for the difference between the decision trees with attribute combinations (a) and (a–c).

The exploration strategy yielded search paths that are on average 7.9 times longer than the optimal path. This can be improved to a ratio of 4.8 when the search was guided by our proposed strategy based on the pruned decision trees. If we used unpruned decision trees then the best ratio we achieved was 5.8. This seems to suggest that the unpruned decision trees overfit the data of the three training supermarkets. The MaxEnt strategy achieved the best result with a ratio of 1.4. However, by using the MaxEnt strategy the robot built up a global map during the search that also included the locations of all products seen so far. Thus, if a target product has been seen previously while searching for another product, the product could later be approached directly on the shortest path. As the other strategies lack the ability to memorize the global locations of previously seen products, we also considered a modified version of the MaxEnt strategy (the "restart" version in Table 2) in which the map was cleared when a product has been found. Thus, when searching for the next product on the list, the robot could not utilize information about the market stemming from the search sub-trial for the previous product. This resulted in a ratio of 3.7, which is still better than the decision tree strategy, though it now performs worse than the human participants who achieved a ratio of 2.3. However, one might argue that humans certainly do build up some kind of global map of the market during the search and thus are able to utilize information from previous search sub-trials when searching for the next product. Clearly, this information is less accurate and more sketchy than the map utilized in the "single run" version of the MaxEnt strategy that memorizes the exact location of all seen products.

[1] If the sample sizes differed, we used the sample size of the smaller sample. We also applied Welch's t-test, which is applicable for unequal sample sizes and unequal variances, and got the same results regarding the statistical significance at the 0.01 level.

Table 2: Mean and standard deviation (SD) of the overall search path lengths for different search strategies. For further comparison we list the length of the optimal path and the path length ratio defined as the average path length of a strategy divided by the length of the optimal path.

Strategy	Search Path Length		Ratio	Samples
	Mean (km)	SD (km)		
Exploration	1.959	0.297	7.9	1000
Dec. Tree (a–d, pruned)	1.176	0.211	4.8	1000
Dec. Tree (a)	1.609	0.263	6.5	1000
Dec. Tree (a–b)	1.425	0.193	5.8	1000
Dec. Tree (a–c)	1.620	0.257	6.6	1000
Dec. Tree (a–d)	1.717	0.238	7.0	1000
MaxEnt (single run)	0.342	–	1.4	–
MaxEnt (restarts)	0.911	–	3.7	–
Human Participants	0.565	0.110	2.3	26
Optimal Path	0.247	–	1.0	–

Though the proposed strategies did not achieve the same performance as humans, the results clearly indicate that the utilization of background knowledge by our proposed strategies leads to significantly shorter search paths when compared to an uninformed search strategy. The exploration strategy performed significantly worse than our approaches, because it is not able to take domain-specific background knowledge into account, which is the advantage of our proposed techniques.

6.2. Evaluation with Varying Starting Locations

The setting presented in the previous section was restricted to a single starting location for each target product. This was motivated by the desired comparison to the performance of the human participants, for which we had to replicate the conditions of the field study.

For a more thorough evaluation of the search strategies, we started the search for the target products from several different locations. We randomly chose 20 starting locations in the market (see Fig. 7). Each of the 15 products from the previous experiment had to be searched for from each starting location, yielding 300 independent search trials per strategy. For the MaxEnt strategy, the map was cleared between the individual search trials. As in the previous experiment, we considered the total search path length of all individual search trials as a performance measure for the search strategies. We repeated each experiment a thousand times and list the average total search path length and its standard deviation in Table 3.

Beside the exploration strategy we evaluated the variant of our decision tree strategy that performed the best in the previous experiment. Though the improvement over the exploration strategy was now less pronounced than in the previous setting, it still yields significantly[2] shorter search paths, as can be seen in

[2] We performed one-tailed paired t-tests with $p < 0.01$ for the sampled

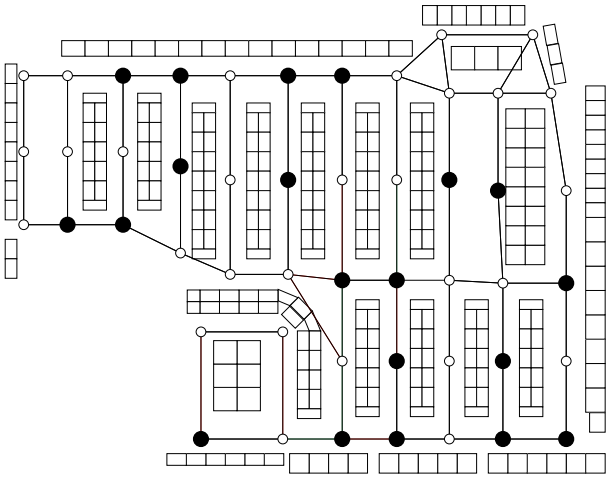


Figure 7: For the second experiment we randomly chose 20 starting locations (filled nodes) for the search trials. The target locations of the products remain the same as in the previous experiment (see Fig. 6).

Table 3. The MaxEnt strategy yields a significant improvement over the decision tree strategy, which is now more pronounced than in the previous experiment. To summarize, this second experiment confirms our finding of the first experiment, that it is beneficial to integrate domain-specific background knowledge when searching for an object.

6.3. Searching with a Real Robot

As a proof of concept we let a mobile robot autonomously search for a product in a real supermarket. We used a Pioneer 3DX equipped with a SICK LMS 291 laser range scanner and a SICK RFI 641 RFID reader (Fig. 8). The target product was marked with a passive UHF RFID tag and the robot stopped searching as soon as it had detected the corresponding RFID tag of the product. For navigation purposes the robot mapped its environment in a local occupancy grid-map with a side length of 16 meters. The local map was successively re-centered at the robots location if the robot moved more than one meter. We used a virtual sensor for detecting the relevant edge attributes and the location of the decision points in the reference frame of the local map. Extracting this information directly from sensor data is a problem in its own that we consider to be beyond the scope of this article which focuses on high-level decision making.

In the first experiment the robot started at the entrance of the market and had to find yogurt by utilizing the decision tree depicted in Fig. 4 (a). In Fig. 9 we depict a sequence of snapshots[3] of this search run. The small image in the upper right corner shows the path taken by the robot as well as the current location of the robot and the local map with respect to a map of the whole market. The rest of the image shows a detail of the map with the current decision point and the possible successor nodes with their respective edges. The successor nodes of which the robot may choose randomly are marked by a black dot. These nodes either belong to edges that were classified as promising or to unvisited edges if no edge was classified as promising. As can be seen in Fig. 9 (first and second picture) the robot first proceeded straight down the main aisle, because at each decision point there was only one promising edge in front of the robot: an unvisited edge in a main aisle with a cooling shelf visible in its direction. At the end of the main aisle the robot then selected the only unvisited edge, which led to the node to the robot's left side. After a few more meters it finally detected the product's RFID tag and successfully ended the search (Fig. 9, third picture).

In the second experiment the robot started in a side aisle located nearly in the center of the market. At the beginning, it had two promising choices for leaving the side aisle and entering a main aisle and randomly chose to enter the lower main aisle. Arriving there (Fig. 10, first picture) it encountered only one promising direction: a node to its left side, which lies in a main aisle with a cooling shelf visible at its end. It then proceeded down the main aisle – encountering two similar situations – until it was left with a choice of two non-promising but unvisited edges at the end of the main aisle (Fig. 10, second picture). It randomly chose to turn right and successfully ended the search

Table 3: Mean and standard deviation (SD) of the overall search path lengths for different search strategies. For further comparison we list the length of the optimal path and the path length ratio defined as the average path length of a strategy divided by the length of the optimal path.

Strategy	Search Path Length		Ratio	Samples
	Mean (km)	SD (km)		
Exploration	32.2	1.30	6.2	1000
Dec. Tree (a–d, pruned)	27.0	0.96	5.2	1000
MaxEnt	17.4	–	3.3	–
Optimal Path	5.2	–	1.0	–

Figure 8: We equipped a Pioneer 3DX with a SICK laser range scanner and a SICK RFID reader with two antennas. The robot autonomously searched for a product in a supermarket using our proposed search strategy.

strategies. The result of the MaxEnt strategy also differs significantly ($p < 0.01$) from the results of the sampled strategies.

[3] Note that some parts of the real market have been rearranged while we have been working on this article. This accounts for the differences in the market layout of Figs. 9–11 when compared to the previously shown figures in this article.

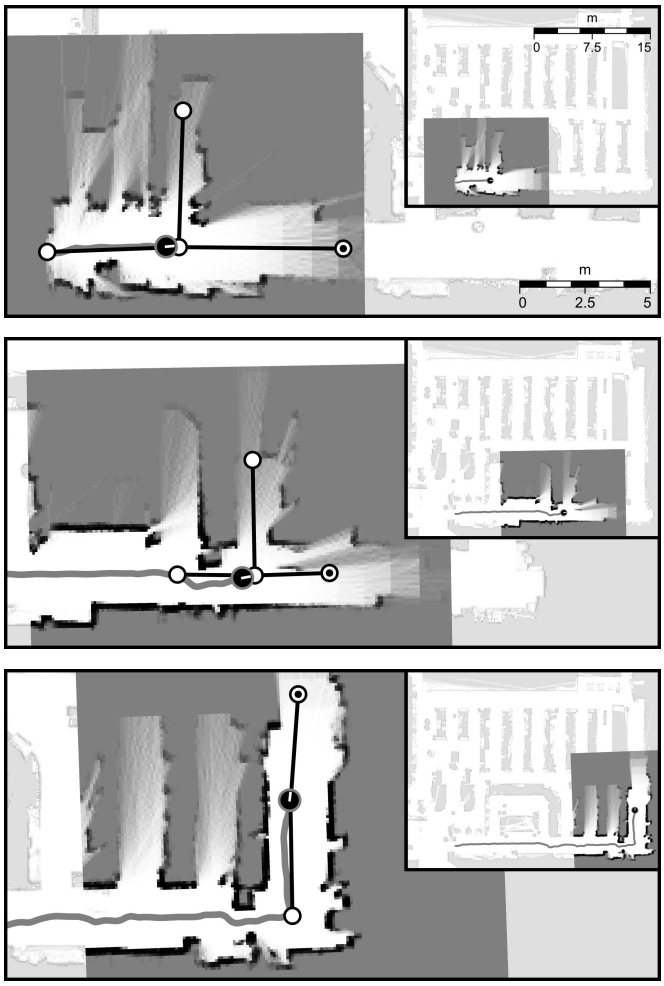


Figure 9: In the first search run the robot started at the entrance of the market.

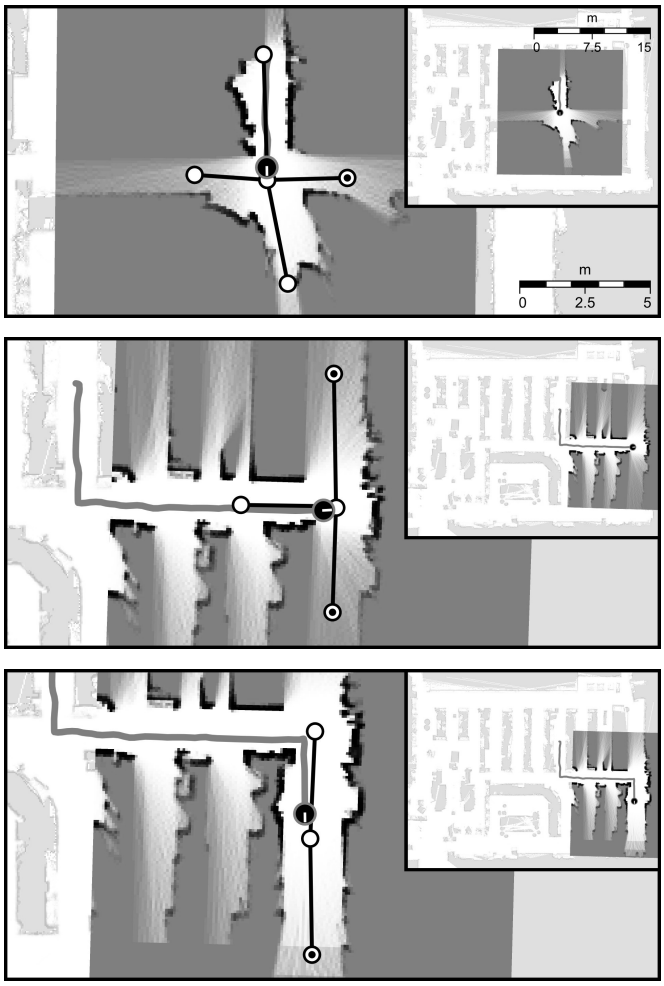


Figure 10: The next search run began in a side aisle in the center of the market.

after detecting the RFID tag of the product after a few meters (Fig. 10, third picture).

As in the previous experiment, the robot made an optimal decision at each decision point. Of course, this will not always be the case. For example, in a replication of the second experiment the robot chose to proceed to the upper node when it once again was confronted with the situation depicted in the second picture of Fig. 10 in which it had to choose randomly among two nodes. This resulted in the longer search path shown in Fig. 11. In general, it is inevitable that the robot makes a suboptimal decision at some point during the search. The purpose of our proposed technique is to learn heuristics that support the robot in making the right decision such that on average the product is found faster than with an uninformed search strategy. We believe that the results obtained both in simulation and real-world experiments highlight the potential of this idea.

7. Conclusions

We presented two approaches for efficiently finding an object in an unknown environment. The first approach, which was the focus of this article, is a reactive search technique that only depends on local information about the objects in the robot's vicinity when deciding where to search next. This strategy is based on search heuristics that can be learned from data of optimal search paths. As a proof of concept, we presented real-world experiments in which a mobile robot searched autonomously for a product using the proposed decision tree strategy. We furthermore presented a second, inference-based search strategy, that explicitly reasons about the location of the target object given all observations made so far. It thereby takes more global information into account. The underlying model of this MaxEnt search strategy can be learned from object arrangements of similarly structured example environments.

The MaxEnt strategy achieved shorter search paths than the decision tree strategy. But this advantage comes at additional cost as it needs to maintain and update a global map and perform inference by taking into account all previously seen products. This has important practical ramifications. The decision tree strategy can be implemented using a shifting local grid-map as we have demonstrated in the real-world experiments. Thus, the mapping overhead is constant in the size of the search area – the only exception being the decision point graph, which is needed for backtracking to a node with an unvisited edge if all outgoing edges of the current node have been visited already.

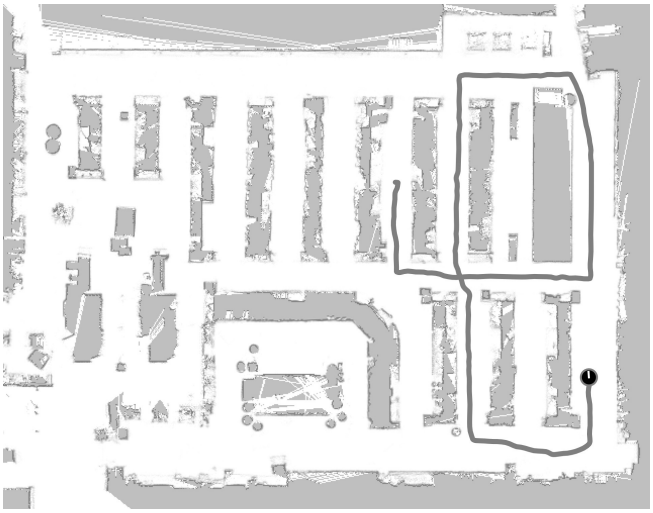


Figure 11: In a replication of the second search run the robot chose the upper node in the situation depicted in the second picture of Fig. 10, which resulted in a longer search path.

However, this graph only needs to be topologically correct and the computational overhead during the search can be considered negligible compared to the maintenance of the local grid-map. The MaxEnt strategy, on the other hand, needs to resort to a computationally much more demanding online SLAM approach for maintaining a consistent global map that preserves the spatial relations between all detected objects. On the downside the mapping overhead now grows linearly with the size of the search area. The upside is that more informed decisions can be made which eventually leads to shorter search paths.

To summarize, in this article we presented two general approaches for modeling, learning, and utilizing background knowledge about indoor environments such that a mobile robot is able to find an object in an initially unknown environment more efficiently than would have been possible without such domain-specific knowledge. The choice between these two approaches constitutes a trade-off between the complexity of the underlying model and the resulting search efficiency. Extensive experiments showed that both strategies significantly outperform an exploration strategy. This demonstrates the benefits of utilizing background knowledge when searching for objects in unknown environments.

Acknowledgment

This work has been supported by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) under contract number SFB/TR 8 Spatial Cognition (R6-[SpaceGuide]). The field study involving human participants in the real supermarket was carried out by Christopher Kalff and colleagues of the Center for Cognitive Science at the University of Freiburg, Germany. The support provided by Jörg Müller (Department of Computer Science, University of Freiburg) during the preparation of the real-world experiments is gratefully acknowledged.

[1] P. A. Titus, P. B. Everett, Consumer wayfinding tasks, strategies, and errors: An exploratory field study, Psychology and Marketing 13 (3) (1996) 265–290.

[2] C. Kalff, G. Strube, Where is the fresh yeast? The use of background knowledge in human navigation, in: Spatial Cognition 2008: Poster Presentations, Freiburg, Germany, 2008, pp. 17–20.

[3] B. D. Argall, S. Chernova, M. Veloso, B. Browning, A survey of robot learning from demonstration, Robotics and Autonomous Systems 57 (5) (2009) 469–483. doi:10.1016/j.robot.2008.10.024.

[4] I. Rexakis, M. G. Lagoudakis, Classifier-based policy representation, in: Proc. of the Int. Conf. on Machine Learning and Applications (ICMLA), San Diego, CA, USA, 2008, pp. 91–98. doi:10.1109/ICMLA.2008.31.

[5] E. T. Jaynes, Information theory and statistical mechanics, The Physical Review 106 (4) (1957) 620–630.

[6] A. L. Berger, V. J. Della Pietra, S. A. Della Pietra, A maximum entropy approach to natural language processing, Computational Linguistics 22 (1) (1996) 39–71.

[7] S. Koenig, Agent-centered search, AI Magazine 22 (4) (2001) 109–131.

[8] S. J. Benkoski, M. G. Monticino, J. R. Weisinger, A survey of the search theory literature, Naval Research Logistics (NRL) 38 (4) (1991) 469–494.

[9] K. E. Trummel, J. R. Weisinger, The complexity of the optimal searcher path problem, Operations Research 34 (2) (1986) 324–327.

[10] I. Wegener, Optimal search with positive switch cost is NP-hard, Information Processing Letters 21 (1) (1985) 49–52. doi:10.1016/0020-0190(85)90108-5.

[11] H. Lau, S. Huang, G. Dissanayake, Optimal search for multiple targets in a built environment, in: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Edmonton, AB, Canada, 2005, pp. 3740–3745. doi:10.1109/IROS.2005.1544986.

[12] T. H. Chung, J. W. Burdick, A decision-making framework for control strategies in probabilistic search, in: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), Roma, Italy, 2007, pp. 4386–4393. doi:10.1109/ROBOT.2007.364155.

[13] S. Vasudevan, R. Siegwart, Bayesian space conceptualization and place classification for semantic maps in mobile robotics, Robotics and Autonomous Systems 56 (6) (2008) 522–537. doi:10.1016/j.robot.2008.03.005.

[14] T. Kollar, N. Roy, Utilizing object-object and object-scene context when planning to find things, in: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), Kobe, Japan, 2009, pp. 2168–2173. doi:10.1109/ROBOT.2009.5152831.

[15] A. Ranganathan, F. Dellaert, Semantic modeling of places using objects, in: Proc. of Robotics: Science and Systems (RSS), Atlanta, GA, USA, 2007.

[16] H. Zender, O. Martínez Mozos, P. Jensfelt, G. M. Kruijff, W. Burgard, Conceptual spatial representations for indoor mobile robots, Robotics and Autonomous Systems 56 (6) (2008) 493–502. doi:10.1016/j.robot.2008.03.007.

[17] C. Galindo, J.-A. Fernández-Madrigal, J. González, A. Saffiotti, Robot task planning using semantic maps, Robotics and Autonomous Systems 56 (11) (2008) 955–966. doi:10.1016/j.robot.2008.08.007.

[18] A. Cocora, K. Kersting, C. Plagemann, W. Burgard, L. De Raedt, Learning relational navigation policies, in: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Beijing, China, 2006, pp. 2792–2797. doi:10.1109/IROS.2006.282061.

[19] J. R. Quinlan, Induction of decision trees, Machine Learning 1 (1) (1986) 81–106. doi:10.1007/BF00116251.

[20] L. A. Breslow, D. W. Aha, Simplifying decision trees: A survey, The Knowledge Engineering Review 12 (1) (1997) 1–40.

[21] B. Yamauchi, A frontier-based approach for autonomous exploration, in: Proc. of the IEEE Int. Symp. on Computational Intelligence in Robotics and Automation (CIRA), 1997, pp. 146–151. doi:10.1109/CIRA.1997.613851.

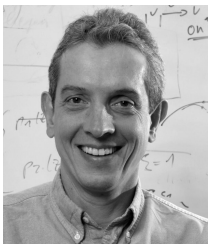
[22] D. Joho, W. Burgard, Searching for objects: Combining multiple cues to object locations using a maximum entropy model, in: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), Anchorage, AK, USA, 2010, pp. 723–728.

[23] G. Grisetti, C. Stachniss, W. Burgard, Improved techniques for grid mapping with Rao-Blackwellized particle filters, IEEE Trans. on Robotics 23 (1) (2007) 34–46.

[24] D. Joho, C. Plagemann, W. Burgard, Modeling RFID signal strength and tag detection for localization and mapping, in: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), Kobe, Japan, 2009, pp. 3160–3165. doi:10.1109/ROBOT.2009.5152372.

Dominik Joho is a research assistant at the University of Freiburg, Germany working in the Laboratory for Autonomous Intelligent Systems headed by Wolfram Burgard. He studied computer science at the University of Freiburg and received his diploma degree in 2007. He is currently working towards his Ph.D. degree. His general research interests lie in artificial intelligence, machine learning, and mobile robotics.

Martin Senk studies computer science at the University of Freiburg, Germany. He received his bachelor's degree from the University of Freiburg in 2009 and is currently working towards his master's degree.



Wolfram Burgard is a professor for computer science at the University of Freiburg, Germany where he heads the Laboratory for Autonomous Intelligent Systems. He received his Ph.D. degree in computer science from the University of Bonn in 1991. His areas of interest lie in artificial intelligence and mobile robots. In the past, Wolfram Burgard and his group developed several innovative probabilistic techniques for robot navigation and control. They cover different aspects such as localization, map-building, path-planning, and exploration. For his work, Wolfram Burgard received several best paper awards from outstanding national and international conferences. In 2009, Wolfram Burgard received the Gottfried Wilhelm Leibniz Prize, the most prestigious German research award.