A non-myopic approach to visual search

Julia Vogel and Kevin Murphy
Department of Computer Science
University of British Columbia
Vancouver, Canada

Abstract

We show how a greedy approach to visual search — i.e., directly moving to the most likely location of the target — can be suboptimal, if the target object is hard to detect. Instead it is more efficient and leads to higher detection accuracy to first look for other related objects, that are easier to detect. These provide contextual priors for the target that make it easier to find. We demonstrate this in simulation using POMDP models, focussing on two special cases: where the target object is contained within the related object, and where the target object is spatially adjacent to the related object.

1 Introduction

Consider the problem of getting a robot to find your keys which you lost somewhere in your office. Since keys are small objects, which can be hard to find, the robot will need to employ a high-resolution sensor in order to detect them. However, exhaustively scanning such a high-resolution sensor over the whole office would take a long time. It seems much more reasonable to first identify large regions of the office that are likely to contain your keys (e.g., desktops) and then to start a more detailed search. We provide experimental evidence that this hierarchical approach to object search is not only more efficient in time, but also results in lower false positive rates, since by focusing attention on the desk, other distractors can be ignored. Another example of indirect object search is to first identify large objects that tend to be nearby the keys (e.g., computer monitors), and then to start a more detailed search in the neighboring vicinity [26]. We provide experimental evidence that this indirect approach outperforms more greedy approaches that directly look for the target object.

Our methodology is to build a simulated world, containing objects at random locations, and then to build various controllers which interact with the world through a movable sensor. We consider two versions of the problem. In the multi-resolution version, the sensor can either operate at high resolution, but narrow field of view (FOV), or at low resolution and wide FOV. In the multi-object version, the sensor can detect several different kinds of objects. The controller is told which target object to look for, and then, at each step, it must decide where to look next, and what detector to use. When it is sufficiently confident it has found the target object, the controller stops, and it receives a reward, depending on whether it was right or wrong, and depending on how many steps it took. The goal is to design controllers that maximize the reward (i.e., find the right object as quickly as possible). We show that controllers that perform indirect object search (e.g., looking for the large object before the small target object, or looking for the general region before zooming in to the specific region) do better than controllers that greedily look for the target object.

More precisely, we compare controllers that use a fixed greedy policy (always look for the target object using the high resolution sensor) with decision-theoretic controllers that maintain a model of the world, represented in terms of a partially observed Markov decision process (POMDP) [11]. POMDPs are a useful formalism because they can represent the fact that the controller has uncertainty about the state of the world (e.g., the object's location), but can perform information-gathering actions to reduce its uncertainty.

Standard approaches to building POMDP-based controllers try to construct a policy offline that specifies what actions to take for many possible belief states. At run time, the action corresponding to the current belief state is just looked up, the action executed, and then the belief state updated based on the resulting observation from the simulator. In this paper, we consider an alternative approach based on receding horizon control. In this approach, we figure out at run time what action

to take by building a lookahead search tree based on our current belief state. We then perform the optimal action, update our belief state based on the resulting observation, and then replan by building another tree. (This process can be sped up over time by caching our computations, a technique known as real-time dynamic programming [1, 8].)

We study the performance of the controllers as a function of the lookahead planning horizon h. We show that myopic controllers ($h = 1$) do poorly, and do not exhibit the indirect object search behavior mentioned above. But by looking just 3 steps into the future, the controller can "figure out" that the indirect approach will result in higher expected reward. We show that these non-greedy controllers find the target object more often and more quickly.

The rest of the paper is structured as follows. In the next section, we summarize related work in the area of active reasoning for vision. Section 3 describes our the multi-resolution simulator, and Section 4 our multi-object simulator. The receding horizon controller is explained in detail in Section 5. In Section 6, we present our experimental results which show that hierarchical and indirect object search results in better performance. We discuss our findings and propose steps for future work in Section 7.

2 Related work

The idea of employing active reasoning for object recognition and scene understanding has been explored previously. However, our method differs from most previous work in that the exploitation of hierarchical or spatial scene context is combined with a non-myopic planning engine.

Early work on selective vision includes the TEA-1 system by Rimey and Brown [22]. The system exploits the spatial structure of a scene in order to support or reject a hypothesis. Based on handcrafted goodness functions, it sequentially collects visual evidence. Fu et al. [7] utilize the regularities and the hierarchical organization of grocery stores within their planning system SHOPPER to find items in a simulated grocery store. Similarly, Wixson [26] exploits spatial relationships of the target object with other objects to efficiently search for objects in indoor scenes. However, the decision to use an intermediate object in order to more efficiently locate the target object is hand-coded while in our system this decision is made based on the current belief of the system.

Callari and Ferrie [2], Paletta and Pinz [19], and Laporte and Arbel [13] explore decision making for active object recognition and pose estimation. The task is to select a viewpoint in 3D so as to decide on the object label (and pose [13]) with as few views as possible. To this end, several approaches based on probabilistic evidence fusion, minimizing the expected loss or maximizing entropy or mutual information are proposed and compared. Only the method of Paletta and Pinz [19] is non-myopic in the sense that an approximate utility table is learned offline and used for viewpoint selection.

Figure 1. An image from the Caltech Office DB [6], divided into 7 slices, each of which is divided into 4 quadrants.

Minut and Mahadevan [15] proposed a two-layered architecture to simulate selective attention for visual search tasks. The location of the next gaze is primed coarsely using reinforcement learning and subsequently more finely using bottom-up visual saliency. Oliver and Horvitz [18] employ selective perception in a multimodal system for recognizing office activity. Actions are selected myopically based on the expected value of information.

Recently, Elder et al. [3] proposed a camera system that combines a fixed, pre-attentive, low-resolution wide-field camera with a shiftable, attentive, high-resolution narrow-field camera. However, the current stage of the work only implements methods for locating low-resolution faces from low-resolution images and does not yet extend to making the decision to narrow the view to the high-resolution camera.

3 Hierarchical search

For the experiments with hierarchical (multi-scale) search, we simulate a robot with a pan-zoom camera, which is in the middle of an office at a fixed distance from the objects of interest (on the desk and walls). We use panoramic (360°) images of offices from the Caltech Office database [6]. We divide each image into 7 "slices" (representing view points), and divide each slice into 4 "quadrants" (see Figure 1). The camera can look at one slice at a time when zoomed out, or one quadrant at a time when zoomed in. When it is looking at a slice or quadrant, the controller can choose to run an

object detector, which responds with 0 or 1, depending on whether the detector "fires" or not. If the camera is in the same location as the object, the simulator returns 1 with probability p_{tp} (true positive); however, it may also return 0 with probability p_{fn} (false negative). If the camera is in a different location to the object, the simulator returns 0 with probability p_{tn} (true negative); however, it may also return 1 with probability p_{fp} (false positive). These parameters also depend on whether the camera is zoomed in or not. Specifically, we assume the error rate is higher when the camera is zoomed out, which reflects the fact that finding small objects is harder when looking from a distance.

Although we have performed some preliminary experiments using real object detectors[1], in this paper, we simulate the object detector. Specifically, we set $p_{tp}^{hi} = 0.95$ and $p_{fp}^{hi} = 0.05$ for the high resolution detector, and $p_{tp}^{lo} = 0.85$ and $p_{fp}^{lo} = 0.15$ for the low resolution detector. The reason we simulate the detector is that our real detector is not yet good enough for the system to be able to reliably find the object.

At every step, the controller must choose from 8 discrete actions: move to Q1, move to Q2, move to Q3, move to Q4, move left one slice, move right one slice, run the detector, and stop. If it chooses to move to a quadrant, it automatically zooms in; if it chooses to move to a slice, it automatically zooms out. We assume that movements are noise-free, i.e. that they have a deterministic effect on the camera location. The controller should autonomously choose to stop when it is confident enough that it has found the object. Once the stop action has been taken, the simulator returns the total reward for this run, depending on whether the camera stopped in the right location or not.

3.1 POMDP model

We assume that the POMDP model of the world used by the controller is an accurate model of the above simulator. (Such models can be learned offline.) The factored POMDP model we use for these experiments is shown in Figure 2. S_t is the state of the system at time t, Y_t is the observation, D_t is the decision (action), and U_t is the utility node that encodes the reward. The belief state is $b_t = p(S_t|Y_{1:t}, D_{1:t})$. We discuss these components in more detail below.

The state of the system at time t contains the camera's location, X_t, the object's location, O_t, and the

[1]We built a computer monitor detector using a binary classifier. At the low resolution level, we applied the classifier to global image texture features (called the "visual gist" in [17]). At the high resolution level, we applied the classifier to image patch features, as in [17].

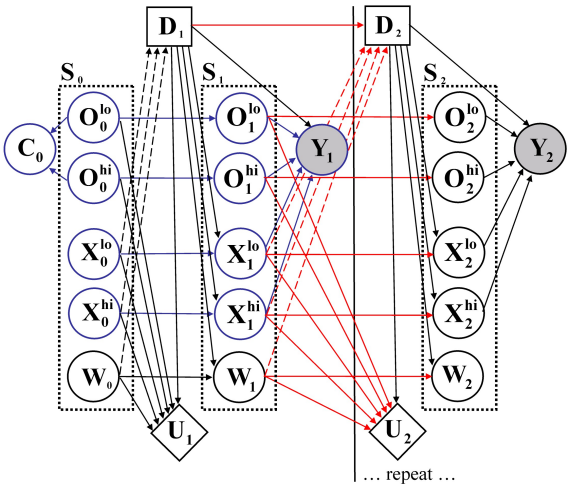

Figure 2. POMDP for the multi-resolution experiments, unrolled for two time slices. See text for details.

"world bit" W_t (this will be explained below). We assume X_t is fully observed, but O_t is hidden. Furthermore, we assume the target object does not move, so O_t is a constant. (We propagate it over time just for notational simplicity using $P(O_{t+1}^{hi}|O_t^{hi}) = 1$ and $P(O_{t+1}^{lo}|O_t^{lo}) = 1$.) We represent locations at two levels of resolution. Specifically, $O_t^{lo} \in \{1, \ldots, 7\}$ represents the slice, and $O_t^{hi} \in \{1, \ldots, 4\}$ represents the quadrant. Similarly for $X_t^{lo} \in \{1, \ldots, 7\}$ and $X_t^{hi} \in \{1, \ldots, 4, Z\}$ where $X_t^{hi} = Z$ means the camera is zoomed out, so the quadrant location is undefined. (A similar hierarchical representation of space was used in [25] for a POMDP approach to robot navigation.) In total, the state space for $O_t \times X_t$ has size $7 \times 4 \times 7 \times 5 = 980$. However, the belief state $p(X_t, O_t)$ will only be non-zero for those values that are consistent with the current X_t (which is observed).

The "stop bit" $W_t \in \{0, 1\}$ is turned on once the controller has executed the stop action. Thereafter, it will receive no further reward. This is how the controller knows that the simulator won't keep rewarding it for continually detecting the object. The stop bit doubles the state space so in fact $S_t \in \{1, \ldots, 1960\}$. The transition functions $P(X_{t+1}^{hi}|X_t^{hi})$, $P(X_{t+1}^{lo}|X_t^{lo})$, and $P(W_{t+1}|W_t)$ depend deterministically on the chosen action.

There is a single observable variable, $Y_t \in \{0, 1\}$, which is on if the simulator says that the detector fired, and off otherwise. We set $p(Y_t = 1|X_t, O_t)$ using the known p_{fp} etc values. In our current model, the detector is not running all the time, but must be explicitly invoked (as indicated by the $D_t \rightarrow Y_t$ arc). This is to

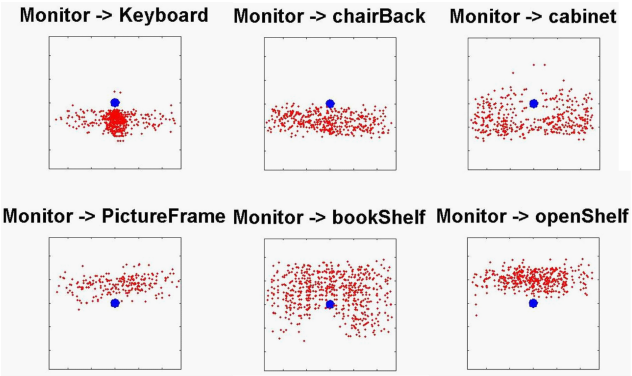

Figure 3. Relative spatial relationships between monitors (center of plot) and other objects in the manually labeled Caltech office database.

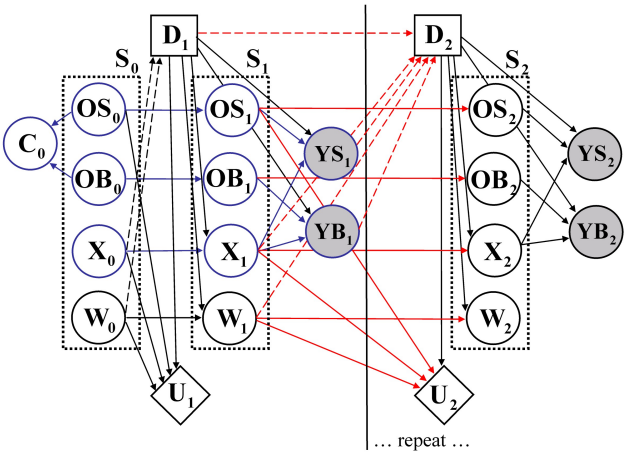

Figure 4. POMDP model for the multi-object system.

simulate a robot that has limited computational (or attentional) resources and decides consciously which of its detectors to employ, and only runs its object detection software some of the time. If the detector action is not taken, then the simulator returns no observation and Y_t is undefined (in this case, we set $p(Y_t = 1|X_t, O_t)$ to be independent of X_t, O_t, so no information is obtained).

Finally, the utility function U_t gives a reward of 200 if the system stops both in the right slice and the right quadrant (i.e., $X_t = O_t$) and it gives a reward of 100 if it stops in the right slice but wrong quadrant (i.e., $X_t^{lo} = O_t^{lo}$ but $X_t^{hi} \neq O_t^{hi}$). Note that if the system happens to move to the right location ($X_t = O_t$), but does not execute the stop action, it won't get rewarded. Otherwise it could continually rewarded for moving to the right location. It only gets the reward if it "believes" it is in the right location; it demonstrates this belief by choosing the stop action. We give a penalty of -300 for stopping in the wrong slice. We give a cost of -1 to all movements and detection actions, in order to encourage the system to find the object as quickly as possible.

4 Indirect object search

In this section, we investigate the ability to exploit spatial relationships between objects to help in visual search. Spatial relationships between *parts* of objects are widely used in passive object recognition systems (see e.g., [5, 4]). However, spatial relations between objects themselves are less commonly used, partly because they tend to be weaker (since objects can move) and thus the predictive density $p(x_2|x_1)$ for object 2's location given object 1's is more diffuse. Nevertheless, such relationships do exist (see Figure 3), and are worth exploiting. Not exploiting spatial relationships is equivalent to using a much less informed prior about the target object's location, and will result in inefficient search behavior, and higher false positive rates.

If the target object is small, it seems intuitively reasonable to first search for other bigger objects (that are easier to detect from afar) that are spatially nearby. This idea was formalized in [26], where it was proved that two-stage, indirect object search, whereby one first looks for the big (easy) object and then looks locally for the small (hard) object, is more efficient than directly looking for the small object. Here we wished to investigate if this behavior would "fall out" automatically from a generic decision theoretic model.

The world simulator we used was similar to the multi-resolution case, except now there are two objects, "big" and "small", which are always located next to each other, but can occur in any of the 7 slices. (There are no quadrants in this version of the model.) The controller can choose from 5 actions: move left, move right, run the big detector, run the small detector, and stop. We set the parameters so that the big object detector is more reliable than the small object detector, reflecting the fact that big objects are easier to find.

4.1 POMDP model

The POMDP model we used is shown in Figure 4. This is similar to the previous model except now we have two objects, OB_t (object big) and OS_t (object small), and we model location only at the slice level, so $X_t, OB_t, OS_t \in \{1, \ldots, 7\}$. Hence the state space is

now much smaller ($2 \times 7^3 = 686$ states). The known spatial relationship between the big and small object is encoded in the $p(C_t = 1|OS_t, OB_t)$ term, by enforcing that OS_t and OB_t can only differ by one slice.

The utility function only rewards the system for localizing the small object. Nevertheless, the hope is that the system can reason that it will "pay off" to use the big detector because it will help reduce uncertainty about the the big object's location, and hence indirectly reduce uncertainty about the small object's location. (For example, if the big detector were perfect, the belief on the small object's location would change (in general) from uniform over 7 slices to uniform over 2 slices, namely the ones on either side of the big object.)

5 Solving the POMDP model

A controller must decide what action to take at every time step. In return, the simulator returns an observation signal. The controller can maintain any internal state that it likes. If it had unbounded memory, it could memorize all previous actions and observations, which is clearly the optimal thing to do; this is called the no-forgetting assumption. This is indeed the standard approach used for solving finite-horizon decision problems, as modeled using influence diagrams. However, even in the finite horizon case, this can be expensive, so recently methods have been developed for solving limited information influence diagrams (LIMIDs), which relax the no-forgetting assumption [14]. We will use these techniques below.

For unbounded horizon decision problems, the optimal approach is to compress the past history of actions and observations into a finite-sized belief state, $b_t = p(S_t|y_{1:t}, d_{1:t})$. This requires a model of the world. One can then compute, offline, a mapping from b_t to actions for every possible belief state. This is known to be NP-hard [11]. Recently, an approximate approach called point-based value iteration has become popular [20, 24, 9]. This works by sampling a set of possible belief states (assuming a uniform or random controller), and then computing the value function (and hence optimal action) for these sampled states.

A disadvantage of the point-based approach is that they decide what actions to take for a large set of belief states that might never actually occur. Conversely, if something "unusual" should happen at run time, the belief state may not be close to the ones considered offline, and the resulting stored behavior could be far from optimal. For example, suppose the system learns that searching for the big object before the small object is the right strategy. While this may be true on average, it will not always be true. For example, if we already have a strong prior belief about the small object's location (e.g., because we have been tracking it), it makes more sense to directly look for the small object and ignore the big object.

This suggests a more adaptive/situation-aware approach to decision making. In particular, instead of computing what to do offline for many possible scenarios (belief states), why not just figure out what to do in the current scenario? To implement this, we use a method called receding horizon control [23, p. 630]. In this method, we consider all possible future actions we can take, and all possible observations that might result, out to a search depth h (the look-ahead horizon). This is a tree with $|D|^h|Y|^h$ leaves. (If the observation space is large, it is possible to approximate this by sampling [12].) At the root of the tree is the current belief state, b_t, and at each leaf is the belief state that would result if that sequence of actions and observations were to occur. We evaluate the expected utility of each possible sequence of actions and then take the first step of the optimal plan. Having taken action D_{t+1} and observed Y_{t+1}, we update our belief state to b_{t+1} and repeat the planning process. In the experiments below, we study the effect of the horizon length h. A value of $h = 1$ corresponds to a myopic (greedy) controller.

Rather than explicitly constructing this search tree, we use the machinery of LIMIDs (as implemented in the Bayes Net Toolbox [16]) to find the optimal h-step sequence of actions given the current belief state. Specifically, we unroll the POMDP model for h steps, and set the prior on the first slice to be b_t (see discussion below for how to do this). We then compute the best sequence of actions, $D_{t+1:t+h}$, and return D_{t+1} to the simulator. Within the lookahead tree, we make the assumption of no-forgetting, so the action sequence we compute would be optimal if we were to only "live" for h more steps. We could use the limited memory approximation to reduce the branching factor of $|D|^h|Y|^h$, but this would result in a locally optimal policy.

In order to be able to input the current belief state b_t into the POMDP model, we need to ensure there is a clique in the graph that can "store" the $p(O_t^{lo}, O_t^{hi})$ potential. We could add an undirected edge between O_0^{lo} and O_0^{hi} in the first slice, but we are restricted to work with directed models. However, we can simulate the effect of this undirected edge by adding a dummy binary child node, C_t, which is always on. We define $p(C_0 = 1|O_0^{lo}, O_0^{hi}) = p(O_t^{lo}, O_t^{hi})$. Similarly, in the indirect object search, we define $p(C_0 = 1|OB_0, OS_0) = p(OB_t, OS_t)$. This just copies the previous posterior into the current prior. Using a dummy variable is a standard trick for introducing undirected edges into directed graphical models [10].

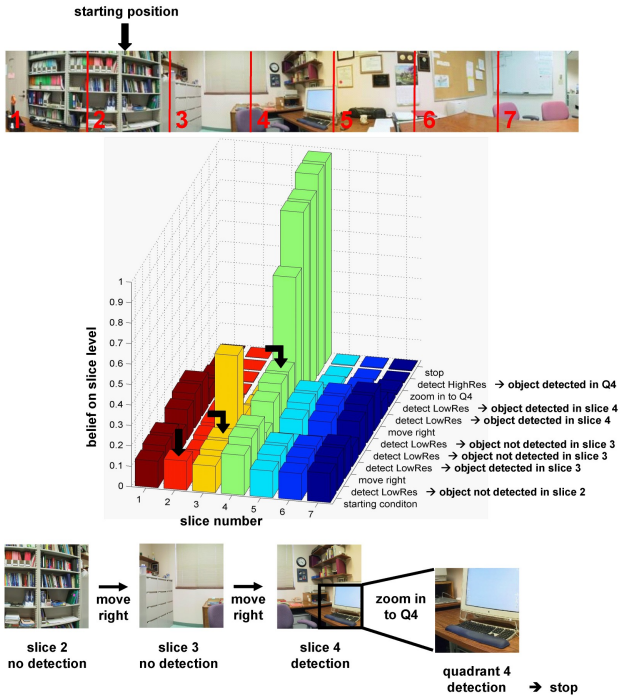

Figure 5. Typical action-observation sequence for the multi-resolution system when looking for a computer monitor.

Obviously it is expensive to continually build this lookahead tree. One improvement is to store the resulting optimal action and the associated belief state so that if it is encountered again, one can just look up the answer. This approach is called "real time dynamic programming" (RTDP) [1, 8]. By storing previously computed results (learning from experience), the system gets faster over time. However, we have not yet implemented this.

6 Experiments

In our experiments, we create 20 different random simulated worlds, and run the controllers in each one for a maximum of 100 steps while measuring the average performance.

6.1 Hierarchical search

Qualitative experimental results Figure 5 shows a typical action-observation sequence for the low-res/high-res search for a monitor. The central part of the figure displays the evolution of the belief state about the object location on the slice level for each action and time step. The simulation was randomly started in slice 2, as shown on top of the figure. The prior belief about the object location corresponds to the first row of colored bars. We chose a non-uniform initial prior in order to break symmetries. The current visual input of the system is depicted at the bottom part of the figure.

The system initially decides to use its low-res detector which does not detect the object. This leads to a decreased belief in slice 2 and a slight increase in all the other slices. The system then moves right and performs several detections in slice 3. This finally leads to the belief that the object is not present in slice 3 and makes the system decide to move on to slice 4. Again, the system uses its low-res detector several times, each of which return positive detections. These positive detections increase the belief in finding the object in slice 4 as the green bars in the bar plot show. The system decides to "take a closer look" and zooms into quadrant 4. When zooming into the high-resolution level, the reward is initially the same for all quadrants. The decision to check Q4 instead of another quadrant is thus chosen randomly. The next decision is to use the high-res detector at Q4. It returns a positive detection that increases the belief both on quadrant level (not displayed) and on slice level (displayed, see green bars). Thus, the next decision of the system is to stop and to return slice 4, quadrant 4 as the detected object location, which in this case is correct.

Other experiments are qualitatively similar in that the system first searches using the low-res detector until it is sufficiently confident the object is present in a given slice, and then it zooms in and uses the high-res detector. If it fails to find the object in any of the quadrants, it infers that its belief that the object is in this slice was mistaken (due to a false positive by the low-res detector). The belief in the other slices goes up in response, and the system moves to the next most promising candidate location.

One interesting aspect of the behavior is the repeated use of the same detector in the same location. This is because we incorrectly model the noise as i.i.d. In such a model, running the detector several times will decrease the chance of an error. This pathological behavior is common to all standard POMDP models, and has been seen before in real robots (Sebastian Thrun, personal communication). One way to solve the problem is to model the noise as correlated, by introducing extra latent factors. However, this approach increases the state space so much that it makes the problem intractable. We leave a more satisfactory solution to this issue to future work.

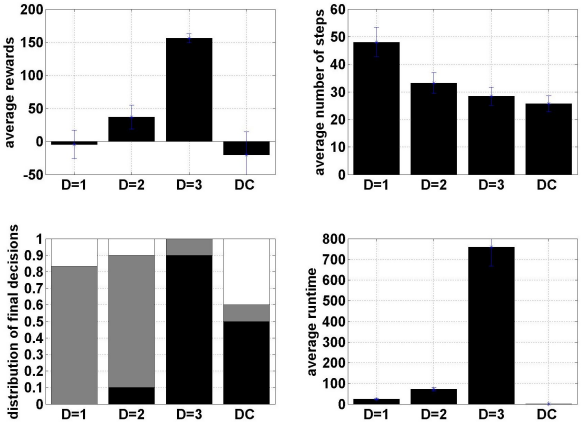

Figure 6. Performance of the hierarchical system vs planning horizon h. We also show results for the deterministic controller (DC).

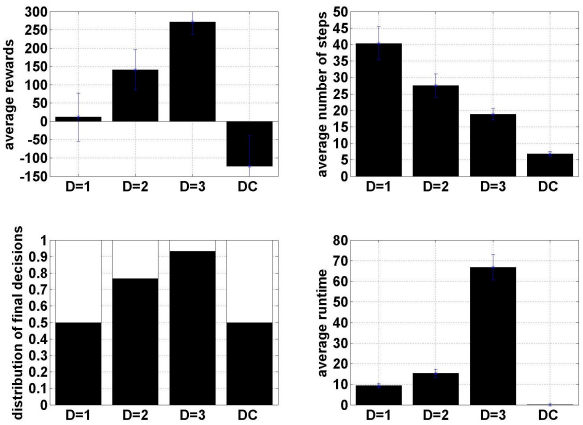

Figure 7. Performance of the multi-Object system vs planning horizon h. We also show results for the deterministic controller (DC).

Quantitative experimental results In addition to the above qualitative experiments, we measured the performance of the system quantitatively as a function of the planning horizon h. We also compared performance to a simple deterministic controller (DC) that searches for the object in a fixed raster-scan order, always using the high-resolution detector, and stopping as soon as the high-res detector fires. In each repetition of the experiment, we randomized the starting location of the robot and the true location of the object.

In Figure 6, we show four different performance measures. The top left is the average reward accumulated en route to finding the object. Not surprisingly, the systems with longer look ahead get higher reward, because (intuitively speaking) they think harder before they act. All the POMDP systems get higher average reward than the DC, since that is what they are trying to optimize.

However, internal reward is just a proxy for actual task performance. A more relevant metric is the fraction of times the object was successfully detected. When the system stops and says "I've found it", was it in the right location? And if not, did it at least identify the right slice? In the bottom left plot, the gray bar is the fraction of times the object was located correctly at the slice level; and the black bar is the fraction of times the object was correctly located at the slice *and* quadrant level. We see that for $h = 1$, the system identifies the right slice about 80% of the time (chance level would be $1/7 = 15\%$), but does not get the quadrant right. This is because it is too myopic to realize that running the high-res detector one or more times would help it localize the object more precisely, and thus get higher reward. However, as we increase h, the ability of the system to localize the object to the correct slice and quadrant dramatically increases. For $h = 3$, the system identifies the correct quadrant about 80% of the time (chance level would be 1/28=3%), and the correct slice 100% of the time. The DC is much better than chance (since it is using the high-res detector), but much worse than the $h = 3$ controller.

In addition to finding the object more often, increasing the planning horizon reduces the number of actions (steps) required. However, it also increases the computational cost. As shown in the bottom right, the average run time to locate the target object is exponential in h. In practice, we find it takes 0.5 sec *per decision* for the $h = 1$ controller to find the object, and 27 sec per decision for the $h = 3$ controller. (In Matlab on a 2.8 MHz Linux PC with 1GB memory.)

6.2 Indirect object search

Similarly to the multi-resolution case, we evaluated the performance of the system with different planning horizons on 20 different randomly generated worlds. We also compared to a deterministic controller that only used the small detector, and performed a fixed left-to-right sweep, stopping as soon as the detector fired.

The results are shown in Figure 7 and are qualitatively similar to before, namely that increasing the planning horizon increases all performance measures, but also increases planning time. Note that now we only count the number of correct localizations at the slice level, since there is no quadrant level. Furthermore, informal observation of the system in action con-

firms that indeed it does look for the big object before the small one.

7 Conclusions and future work

There is much evidence and theory from psychology (e.g., [21]) that natural vision systems build up interpretations of the scene in a sequential fashion, and that what we look for next depends on what we expect to see. In this paper, we have shown how such behavior could arise out of a simple decision theoretic model, provided one is not too myopic.

However, one of the major bottlenecks of our system is the essentially symbolic representation of the world that we use, with one discrete random variable per object, etc. In addition, by reducing the output of the detectors to a single bit, we throw away a lot of information, and the result is a lot of perceptual ambiguity. In the future, we hope to explore a more data-driven approach, in which the belief state is represented in terms of pieces of the image (or compressed versions thereof), rather than in terms of things that we can name. It might be that by bypassing the "linguistic" layer, and giving the decision making system "direct access" to the lower level internal representation of the scene (in terms of regions and surfaces etc.), that we can reduce ambiguity and increase efficiency.

Acknowledgments We would like to thank Pantelis Elinas, Per-Erik Forssen, Jim Little, and Rob Sim for helpful comments on earlier drafts of the manuscript.

References

[1] A. Barto, S. Bradtke, and S. Singh. Learning to act using real-time dynammic programming. *Artificial Intelligence Journal*, 72:81–138, 1995.

[2] F. Callari and F. Ferrie. Active object recognition: looking for differences. *Intl. J. Computer Vision*, 43(3):189–204, 2001.

[3] J. Elder, S. Prince, Y. Hour, and E. Olevskiy. Preattentive and attentive detection of humans in wide-field scenes. *Intl. J. Computer Vision*, 2006.

[4] P. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. *Intl. J. Computer Vision*, 61(1), 2005.

[5] R. Fergus, P. Perona, and A. Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *CVPR*, 2005.

[6] M. Fink and P. Perona. The full images for natural knowledge, caltech office db. Technical Report CSTR:2003.008, Caltech, 2003.

[7] D. Fu, K. Hammond, and M. Swain. Vision and navigation in man-made environments: Looking for syrup in all the right places. In *Proc. of CVPR Workshop on Visual Behaviors*, 1994.

[8] H. Geffner and B. Bonet. Solving large POMDPs using real time dynamic programming. In *Fall AAAI Symp. on POMDPs*, 1998.

[9] J. Hoey and P. Poupart. Solving POMDPs with Continuous or Large Discrete Observation Spaces. In *Intl. Joint Conf. on AI*, pages 1332–1338, 2005.

[10] F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer-Verlag, 2001.

[11] L. P. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101, 1998.

[12] M. Kearns, Y. Mansour, and A. Ng. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. In *Intl. Joint Conf. on AI*, 1999.

[13] C. Laporte and T. Arbel. Efficient discriminant viewpoint selection for active bayesian recognition. *Intl. J. Computer Vision*, 68(3):267–287, July 2006.

[14] S. Lauritzen and D. Nilsson. Representing and solving decision problems with limited information. *Management Science*, 47(9):1235–1251, 2001.

[15] S. Minut and S. Mahadevan. A reinforcement learning model of selective visual attention. In *Proc. 5th Intl. Conf. on Autonomous Agents*, pages 457–464, 2001.

[16] K. Murphy. The Bayes Net Toolbox for Matlab. In *Computing Science and Statistics: Proceedings of the Interface*, volume 33, 2001. www.ai.mit.edu/ murphyk/Software/BNT/bnt.html.

[17] K. Murphy, A. Torralba, and W. Freeman. Using the forest to see the trees: a graphical model relating features, objects and scenes. In *NIPS*, 2003.

[18] N. Oliver and E. Horvitz. Selective perception policies for guiding sensing and computation in multi-modal systems: a comparative analysis. In *Intl. Conf. on Machine Learning*, 2003.

[19] L. Paletta and A. Pinz. Active object recognition by view integration and reinforcement learning. *Robotics and Autonomous Systems*, 31(1-2):1–18, 2000.

[20] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for pomdps. In *Intl. Joint Conf. on AI*, 2003.

[21] R. Rensink. The dynamic representation of scenes. *Visual Cognition*, 7(1/2/3):17–42, 2000.

[22] R. Rimey and C. Brown. Control of selective perception using bayes nets and decision theory. *Intl. J. Computer Vision*, 12(2/3):172–207, 1994.

[23] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2002. 2nd edition.

[24] M. Spaan and N. Vlassis. Perseus: Randomized Point-based Value Iteration for POMDPs. *J. of AI Research*, 24:195–220, 2005.

[25] G. Theocharous, K. Murphy, and L. Kaelbling. Representing hierarchical POMDPs as DBNs for multi-scale robot localization. In *IEEE Intl. Conf. on Robotics and Automation*, 2004.

[26] L. Wixson and D. Ballard. Using intermediate objects to improve the efficiency of visual search. *Intl. J. Computer Vision*, 12(2-3):209–230, 1994.