In *POMDP Practitioners Workshop*,
Toronto, Canada, May 12 2010.

Vision-based Scene Analysis on Mobile Robots using Layered POMDPs

Shiqi Zhang and **Mohan Sridharan**
Department of Computer Science
Texas Tech University, USA
{s.zhang,mohan.sridharan}@ttu.edu

Abstract

Mobile robots are increasingly being deployed in different applications as a result of progress in sensor technology and the development of state of the art algorithms for processing sensory inputs. A key challenge to the widespread deployment of robots is the ability to autonomously tailor sensing and information processing to the task at hand. In this paper, we describe on-going work towards such general-purpose processing of visual input. We pose *visual processing management* as an instance of probabilistic sequential decision making, and specifically as a three-layered hierarchical partially observable Markov decision process (POMDP). We use a two-layered POMDP hierarchy to enable a robot to plan a sequence of visual operators in order to reliably and efficiently analyze the state of the world represented by salient regions of interest in images (Sridharan, Wyatt, and Dearden 2008; 2009). We then incorporate scene analysis as the highest layer, where the goal is to maximize the known information about the environment. The POMDP models at the different layers are tailored to the task at hand by enabling the robot to learn the model parameters and trade-off computational efficiency and reliability. The proposed approach is tested on robots and simulated agents in human robot interaction scenarios. **Keywords: Scene analysis, Processing management, Hierarchical POMDPs, Human robot interaction.**

Motivation

In recent times, high-fidelity sensors have become available at moderate costs (Hokuyo 2008; Videre Design 2010), and sophisticated algorithms have been developed to process sensory inputs. Mobile robots equipped with multiple sensors are hence being used in applications such as disaster rescue. navigation and medicine (Casper and Murphy 2003; Pineau and Thrun 2002; Thrun 2006). However, the ability to accurately sense the environment and interact reliably with other robots and humans remains an open challenge. The following features and requirements characterize mobile robots operating in dynamic environments:

- Features:
  - *Non-deterministic actions*: robot sensing and actuation are unreliable.
  - *Partial observability*: the robot cannot directly observe the state of the world. It can only update its *belief* of the state of the world by applying operators on sensory inputs and observing the outcomes.
  - *Computationally intensive*: algorithms processing sensory inputs are often computationally expensive.
- Requirements:
  - *Performance*: robots interacting with a dynamic environment have to respond in real-time.
  - *Reliability*: though individual actions are unreliable, high overall reliability is required.

These features and requirements make it all the more challenging to process images from a camera, which constitute a rich and high-bandwidth source of information in comparison to other popular sensors such as range finders. However, visual input is more sensitive to environmental changes such as illumination, and operators (i.e. algorithms) that process visual input are typically computationally expensive. In addition, a robot can process the images using a variety of algorithms, each with a different level of uncertainty and computational complexity. One appealing solution is to retain operators to support many tasks and then tailor the sensing and processing to the task at hand i.e. use a subset of operators that can accomplish the task reliably and efficiently.

An approach based on probabilistic sequential decision making, or more specifically partially observable Markov decision processes (POMDPs) (Kaelbling, Littman, and Cassandra 1998), elegantly captures the non-determinism and the partial observability of robot domains. However, POMDP formulations of many practical problems have a state space that increases exponentially, and the worst case time complexity of POMDPs is exponential in the state space dimensions. Even state of the art approximate POMDP solvers (Smith and Simmons 2005; Ross et al. 2008) are computationally expensive for robots deployed in dynamic domains. Promising results have been obtained by introducing a hierarchy in the state or action spaces of the POMDP formulation (Pineau and Thrun 2002; Foka and Trahanias 2005). However, many such methods require manual supervision for creating the hierarchy or the corresponding POMDP models.

This paper builds on our prior work on hierarchical image processing (Sridharan, Wyatt, and Dearden 2008) to describe a three-layered hierarchy for scene analysis. The top layer determines *where to look* i.e. which scene of the 3D space to focus on; the intermediate layer determines *what to*

ICAPS 2010 POMDP Practitioners Workshop, May 12, 2010, Toronto, Canada.

process i.e. which regions of images of the chosen scene to analyze; and the bottom layer focuses on *how to process* i.e. what visual operators to apply on the chosen image regions. This paper therefore makes the following contributions:

- It proposes a *functional* hierarchical decomposition for visual processing, which (partially) ameliorates the computational complexity challenge and enables the robot to better exploit the visual input.
- It enables the robot to automatically create the POMDP models required at the different levels, which are then solved to generate the overall policy.

The remainder of this paper is organized as follows. We begin with a brief review of some related work, followed by a description of the overall goal and the specific task addressed in this paper. We then present the proposed layered POMDP hierarchy. Finally, we present some proof-of-concept evaluation results on physical robots and simulated agents in human-robot interaction scenarios.

Related Work

Computer vision research has produced several methods for using a user-specified high-level goal to plan a pipeline of visual operators. However, many of these methods use deterministic action models, with the action pre-conditions and effects being propositions that are required to be true a priori, or are made true by the application of the operator. Unsatisfactory results are handled by re-planning the operator sequence or modifying operator parameters (Clouard et al. 1999). However, the state of the system is not directly observable in robot domains and actions are unreliable.

Recent research in AI planning has focused on relaxing the limitations of classical planners to make them suitable for practical domains. The PKS planner (Petrick and Bacchus 2004) uses a first-order language to describe actions in terms of their effect on the agent's knowledge, rather than their effect on the world. The model is hence non-deterministic because the true state of the world is determined uniquely by the actions performed, but the agent's knowledge of that state is not. PKS captures the initial state uncertainty and constructs conditional plans based on the agent's knowledge. The Continual Planner (CP) (Brenner and Nebel 2008), on the other hand, interleaves planning, plan execution and monitoring, in order to postpone reasoning about uncertain states until more information is available. CP asserts that action preconditions will be met when that point is reached during plan execution. If the preconditions are not met during execution, or are met earlier, replanning is triggered. In robot domains, it may be necessary to accumulate evidence by applying operators more than once, which cannot be done using PKS or CP.

Unlike the approaches based on classical planners, Li et al. (2003) modeled image interpretation as a Markov Decision Process (MDP). Human-annotated images are used to determine the reward structure, explore the state space and compute value functions that are extrapolated to the entire state space. Online operation involves action choices that maximize the value of the learned functions. Darrell (1997) used manual feedback, memory-based reinforcement learning and POMDPs to learn what foveation actions to execute, and when to execute the terminal recognition action, in order to foveate salient body parts in an active gesture recognition system. Manual feedback and extensive training trials are however infeasible in many mobile robot domains.

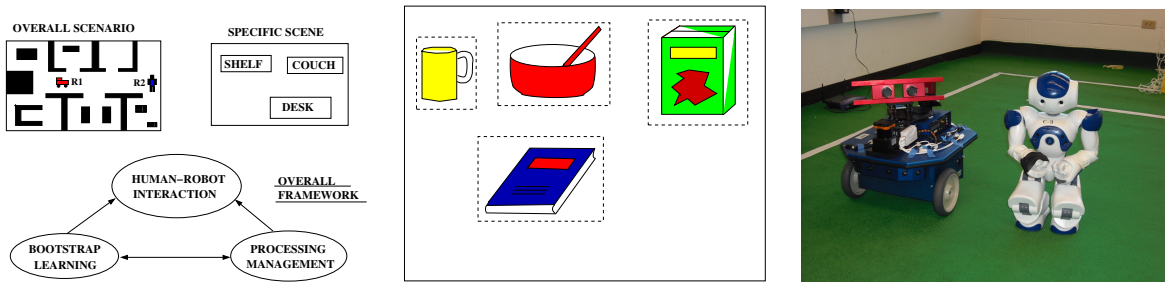
The POMDP formulation is appropriate for domains with partially observable state and unreliable actions. However, a POMDP formulation is intractable for many practical domains. Pineau and Thrun (2002) proposed an elegant hierarchical POMDP approach for behavior control of a robot assistant. In their hierarchy, the top level action is a collection of simpler actions that are represented by smaller POMDPs and solved completely; planning is done in a bottom-up manner, combining individual policies to provide the total policy. All model parameters are defined over the same state space, but the relevant space is abstracted for each POMDP using a dynamic belief network. Similar approaches have been used for robot navigation (Foka and Trahanias 2005) but a significant amount of data for the hierarchy and model creation is hand-coded. POMDPs have also be used for visual search based on models of the human visual system (Butko and Movellan 2008). In parallel, faster solution methods (Ross et al. 2008) and schemes for hierarchy discovery in POMDPs have been developed (Hansen and Zhou 2003; Toussaint, Charlin, and Poupart 2008). POMDPs have also been used on robots that sense and interact in specific real-world tasks such as grasping (Hsiao, Kaelbling, and Lozano-Perez 2007; Saxena, Driemeyer, and Ng 2008). However, most of these methods are computationally expensive for applications with large state spaces. Instead, we propose a three-layered hierarchy for visual scene analysis, with the model parameters being learned by the robot.

Domain Description

Figure 1(a) describes the long-term goal of the on-going project: reliable and autonomous human-robot interaction in indoor scenarios. Achieving this goal requires, among other things, autonomous learning and processing management. Here, we focus on autonomous visual processing management. Consider, for instance, the task of finding a *red bowl* i.e. addressing the query: "where is the red bowl?" or command: "fetch the red bowl". Ideally, the robot should focus on a suitable scene of the environment as shown in the top right of Figure 1(a), resulting in images such as Figure 1(b).

The images can be processed to obtain salient *regions of interest* (ROIs). The robot can process such image ROIs using one or more of a large set of visual operators, for tasks such as segmentation, object recognition and scene reconstruction. However, only a small subset of these operators are appropriate for any specific task—in the current example, we need operators to identify a ROI that is *red* and contains a *bowl*. In this paper, we model sensing and information processing operators as actions and use the terms "operators", "routines" and "actions" interchangeably. We focus on tasks that require a sequence of visual operators to be applied on specific images of specific scenes, so that a human and a robot can converse about (text-based and not language-based) objects of interest.

The experimental platforms include a humanoid robot: Aldebaran Nao (Nao 2008) and a wheeled robot: Videre



(a) Overall scenario. (b) Example image of a desk. (c) Robot platforms.

Figure 1: Visual processing management: (a) Overall scenario; (b) Example image of a portion of the scene with regions-of-interest (ROIs) bounded by rectangles; (c) Robot platforms.

Design ERA-Mobi (Videre Design 2010), as shown in Figure 1(c). The Nao is 58cm tall and equipped with color cameras, ultrasound sensors and touch sensors. The ERA-Mobi is quipped with stereo and monocular cameras, and laser range finders. These platforms have on-board processing (500MHz, 2.2GHz) and wi-fi capabilities, in addition to algorithms for extracting different types of information from the sensory inputs. Next, we describe the hierarchical POMDP formulation for scene analysis.

Hierarchical POMDP Formulation

The proposed hierarchical POMDP formulation has three layers to match the *functional* and *cognitive* requirements of visual scene analysis. The goal here is not to create a POMDP solver but to enable reliable and efficient visual processing using operators that are individually unreliable. As shown in Figure 2(a), the top layer of the hierarchy is the high-level POMDP ("HL-POMDP") that addresses the question: *where to look?* i.e. it determines the specific scene of the 3D space under consideration that is to be analyzed next. The intermediate layer ("IL-POMDP") analyzes images of the chosen scene by determining the image regions to be processed next (*what to process?*). Finally, each salient image region (i.e. ROI) is analyzed using a lower-level POMDP ("LL-POMDP") that determines *how to process?* i.e. the sequence of visual operators to be applied in order to address the specific task (e.g. finding the *red bowl*). The layers of the hierarchy are described below.

Each LL-POMDP operates on a specific image ROI (i.e. salient region) and computes a sequence of visual operators that would identify the desired object reliably and efficiently. Without loss of generality, consider the specific task of locating all *red bowls* i.e. addressing the query: *where are the red bowls?* In addition, assume that the robot can use the following operators: a $color$ operator (based on color histograms) that classifies the dominant color of the ROI, a $shape$ operator (based on shape moments) that classifies the dominant shape within the ROI, and a $sift$ operator (based on the SIFT features (Lowe 2004)) that detects the presence of one of the previously trained object models.

Since the true state of the system cannot be observed, the robot maintains a probability distribution over the underlying state (*belief state*). Each action considers the true underlying state to be composed of the class labels (e.g. *red(R), green(G), blue(B)* for color; *circle(C), triangle(T), square(S)* for shape; *bowl, box, book* for sift), a label to denote the absence of any valid object class—*empty* (ϕ), and a label to denote the presence of *multiple* classes (M). The model for each action's outcomes provides a probability distribution over the set composed of the corresponding class labels, the label *empty* (ϕ) which implies that the match probability corresponding to the class labels is very low, and *unknown* (U) which means that multiple class labels are equally likely. U is an observation whereas M is part of the state: though they are correlated they are not the same.

Since visual operators only update belief states, we include task-specific "special actions" that indicate the presence or absence of the target object, or identify which underlying state is most likely to be the true state. Such actions cause a transition to a terminal state where no further actions are applied. Below, without loss of generality and for ease of explanation , consider two operators: $color$ and $shape$ denoted by subscripts c, s respectively; other operators are included in the experiments. States and observations are denoted by superscripts a, o respectively. The POMDP tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{Z}, \mathcal{O}, \mathcal{R} \rangle$ for a single image ROI is:

- $\mathcal{S} : \mathcal{S}_c \times \mathcal{S}_s \cup term$, the set of states, is a Cartesian product of the variables describing different aspects of the underlying state (e.g. color, shape). It also includes a *terminal* state ($term$). $\mathcal{S}_c : \{\phi_c^a, R_c^a, G_c^a, B_c^a, M_c\}$, $\mathcal{S}_s : \{\phi_s^a, C_s^a, T_s^a, S_s^a, M_s\}$.
- $\mathcal{A} : \{color, shape, A_S\}$ is the set of actions. The first two entries are the visual operators. The rest are special actions. For a query such as: "is there a circle in the scene?", $A_S = \{sFound, sNotFound\}$ describes the presence/absence of the target object, while the query: "what is the color of the ROI?" has $A_S = \{sRed, sGreen, sBlue\}$. Special actions lead to $term$.
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S}' \rightarrow [0, 1]$ represents the state transition function. For e.g. it is an identity matrix for visual operators that do not change the state, such as $color$ and $shape$. For special actions it represents a transition to *term*.
- $\mathcal{Z} : \{\phi_c^o, R_c^o, G_c^o, B_c^o, U_c, \phi_s^o, C_s^o, T_s^o, S_s^o, U_s\}$ is the set of observations of all operators.
- $\mathcal{O} : \mathcal{S} \times \mathcal{A} \times \mathcal{Z} \rightarrow [0, 1]$ is the observation function. It is learned by the robot for the visual actions and it is a uniform distribution for the special actions.
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \Re$, specifies the reward, a mapping from

the state-action space to real numbers.

$$\forall s \in \mathcal{S},\ \mathcal{R}(s, \text{operators}) = -\beta \cdot f(\text{ROI\_size}) \quad (1)$$
$$\mathcal{R}(s, \text{special actions}) = \pm 100 \cdot \alpha$$

For visual operators, the cost depends on the ROI-size and the relative computational complexity (β for $color$ is twice that for $shape$). Special actions are assigned a large positive (negative) reward for giving the correct (incorrect) answer. Parameter α trades-off computation and reliability. Values for α, β are tuned experimentally.

Given the belief state (i.e. a probability distribution over the underlying state) at time t, B_t, the belief update proceeds as:

$$B_{t+1}(s') = \frac{\mathcal{O}(s', a_t, o_{t+1}) \sum_{s \in \mathcal{S}} \mathcal{T}(s, a_t, s') \cdot B_t(s)}{P(o_{t+1}|a_t, b_t)} \quad (2)$$

The visual planning for a single ROI involves solving this POMDP to find a policy ($\pi : B_t \mapsto a_{t+1}$) that maximizes reward over a range of belief states. Plan execution corresponds to using the policy to repeatedly choose the action with the highest value at the current belief state, and updating the belief after executing that action and getting an observation. However, for a single ROI with m features (color, shape etc.) each with n values (e.g. R, G, B for color), the POMDP has an underlying space of $n^m + 1$; for k ROIs it is: $n^{mk} + 1$. For instance, with three ROIs and two actions we get ≈ 15000 states i.e. the planning task soon becomes intractable, even with sophisticated approximate solvers.

The proposed approach therefore models each ROI with a lower-level (LL) POMDP as described above, and introduces a IL-POMDP that chooses, at each step, the ROI whose policy (generated by solving the corresponding LL-POMDP) is to be executed. The overall problem is then decomposed into one POMDP with state space $2^k + 1$, and k POMDPs with state space $n^m + 1$. Without loss of generality, consider an input image with two ROIs, whose IL-POMDP is given by the tuple $\langle \mathcal{S}^I, \mathcal{A}^I, \mathcal{T}^I, \mathcal{Z}^I, \mathcal{O}^I, \mathcal{R}^I \rangle$:

- $\mathcal{S}^I = \{R_1 \wedge \neg R_2, \neg R_1 \wedge R_2, \neg R_1 \wedge \neg R_2, R_1 \wedge R_2\} \cup term^I$ is the set of states. It represents the presence or absence of the object in one or more of the ROIs, and includes a terminal state ($term^I$).
- $\mathcal{A}^I = \{u_1, u_2, A^I_{\mathcal{S}}\}$ are the actions, where (u_i) denotes the choice of executing LL ROI R_i's policy. For queries such as "is there a circle in the scene?", the special action set $A^I_{\mathcal{S}} = \{sFound^I, sNotFound^I\}$. However, for queries such as "where is the circle?", $A^I_{\mathcal{S}}$ represents the fact that one of the entries of $\mathcal{S}^I$ is the answer. All special actions lead to $term^I$.
- $\mathcal{T}^I$ is the state transition function that leads to $term^I$ for special actions and is an identity matrix otherwise.
- $\mathcal{Z}^I = \{FR_1, \neg FR_1, FR_2, \neg FR_2\}$ is the set of observations, which represents finding or not-finding the desired object when each ROI's policy is executed.
- $\mathcal{O}^I : \mathcal{S}^I \times \mathcal{A}^I \times Z^I \rightarrow [0, 1]$, the observation function, is a uniform matrix for special actions. For other actions, it is learned from the LL-POMDP policy trees.
- $\mathcal{R}^I$ is the reward specification. For a special action, it is a large positive (negative) value if it predicts the state correctly (incorrectly). For other actions, it is a "cost" computed from the LL policy trees.

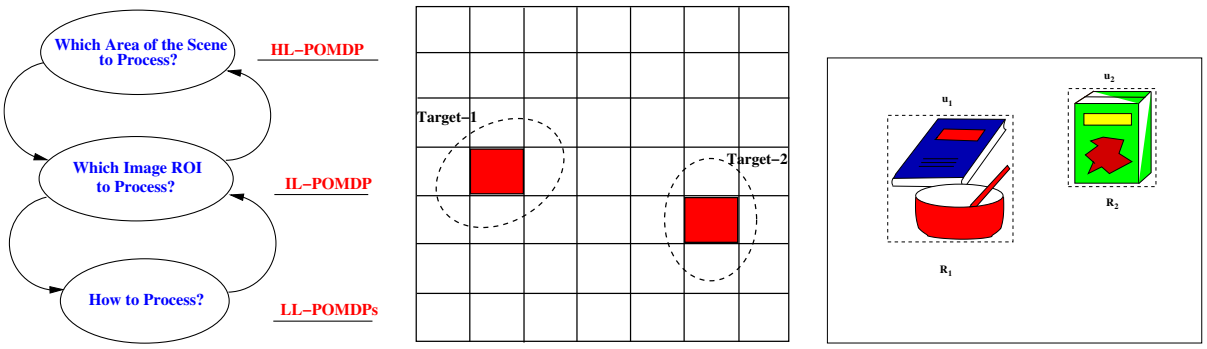
The required LL observation functions and rewards are learned in an initial training phase (see the next section). Given a task (i.e. query) and images of the scene, the robot creates a LL-POMDP for each ROI based on the available visual operators. These POMDPs are created in the format used by the ZMDP package (ZMDP Code 2008) and solved using a point-based solver in the package (Smith and Simmons 2005). The LL policy trees and the query are used to compute the appropriate observation functions and rewards for the IL-POMDP model at run-time. The IL-POMDP is then solved to generate the policy that operates on the entire image, and analyzes relevant ROIs using the corresponding LL policies, in order to identify the target object. More details on the automatic belief propagation between layers can be found in (Sridharan, Wyatt, and Dearden 2008).

Images of real-world scenes contain overlapping objects that would be considered as a single ROI. Such situations can be handled using operators that split an existing ROI based on one of more underlying features such as color or shape. However, planning with such operators is difficult because they change the perceptual state of the system by creating new ROIs. Our approach plans the effects of such ROI-splitting operators by replacing the solution of several new POMDPs with one in which we compute the likelihood that one of the resultant ROIs has the desired feature value. During execution, we replan with a new POMDP after a split. Details are not provided here due to space limitations but are available in (Sridharan, Wyatt, and Dearden 2009)—an execution example is provided below.

In practical settings, the target object can exist in different locations in the room or even in different rooms. The robot may need to move and analyze different scenes to locate the desired target. We therefore introduce a HL-POMDP as the topmost layer of the hierarchy. The HL-POMDP is based on recent work by Butko and Movellan (2008), where visual search was posed as the task of maximizing information gain i.e. reducing the entropy in the belief state. We represent the 3D area under consideration using a discrete 2D *occupancy grid*—Figure 2(b). Each grid cell is associated with a probability that represents the likelihood of occurrence of the target object(s). For a grid with N cells, the definition of the HL-POMDP tuple $\langle \mathcal{S}^I, \mathcal{A}^H, \mathcal{T}^H, \mathcal{Z}^H, \mathcal{O}^H, \mathcal{R}^H \rangle$ is:

- $\mathcal{S}^H : s_i, i \in [1, N]$ is the state vector, where entry s_i corresponds to the event that a target is in grid cell i.
- $\mathcal{A}^H : a_i, i \in [1, N]$ is the set of actions, where entry a_i corresponds to the event that the robot analyzes the scene corresponding to grid cell i.
- $\mathcal{T}^H : \mathcal{S}^H \times \mathcal{A}^H \times \mathcal{S}'^H \rightarrow [0, 1]$ is the state transition function. For the actions that only observe the state, it is an identity matrix. It can be extended to actions that change the state.
- $\mathcal{Z}^H$: {present, absent} is the set of observations that indicates the presence or absence of the target object in the grid cell being analyzed.
- $\mathcal{O}^H : \mathcal{S}^H \times \mathcal{A}^H \times \mathcal{Z}^H \rightarrow [0, 1]$ is the observation function. It is determined automatically as given below.

Such a formulation can represent uniform initial belief and also situations when some prior knowledge of object location is available. The key difference is that the action utili-



(a) Three-layered hierarchy. (b) HL scenario and observation models. (c) Example image with two ROIs.

Figure 2: Visual processing management: (a) Proposed hierarchical decomposition; (b) HL grid cells and observation Gaussians; and (c) Example image with three objects enveloped in two ROIs.

ties (i.e. rewards) are based on the information likely to be gained by executing that action. For the belief state B_t at time t, we use an approach similar to (Butko and Movellan 2008) to define reward $\mathcal{R}^H$ as the InfoMax objective function i.e. the negative entropy of the belief:

$$\mathcal{R}^H(B_t) := -\sum_i B_t^i log(B_t^i) \tag{3}$$

The goal then is to learn a policy: $\pi : B_t \mapsto a_{t+1}$ that decreases the entropy in B_t over a planning horizon of T steps. The other major feature is the observation function specification. We set it to be a function of the expected performance of the lower levels of the hierarchy:

$$\mathcal{O}^H(z_i, s_j, a_k) = p(z_i|s_j, a_k) \sim \mathcal{N}(\mu, \sigma^2) \tag{4}$$
$$\mu = f_\mu(s_j, a_k), \quad \sigma^2 = f_{\sigma^2}(\mathcal{O}, \mathcal{O}^I|s_j, a_k)$$

where the probability of obtaining a specific observation z_i given the state s_j (i.e. desired target is in grid cell j) and action a_k (i.e. focus on grid cell k) is given by a Gaussian distribution. The mean of the Gaussian depends on the target location, the grid cell being examined and the camera's field of view. The variance of the Gaussian denotes the likelihood that the IL-POMDP and the LL-POMDPs will identify the target: it is hence a function of the corresponding observation functions. Figure 2(b) is a pictorial representation of targets and observation Gaussians in a 7×7 grid.

In such a HL-POMDP representation, the number of grid cells can be large for a practical application domain, making it challenging to find policies. We currently use an existing implementation of policy gradient algorithms (Buffet and Aberdeen 2009) to solve the HL-POMDP. The overall operation then involves using the HL policy to choose an appropriate portion of the environment for analysis. Images of the specific scene are analyzed using the IL-POMDP and LL-POMDPs that are created at run-time. The result of this analysis updates the beliefs in the HL, resulting in the choice of a grid cell for subsequent analysis. The key contributions are: (a) the extension of the existing hierarchical approach for image analysis to complex scenes; (b) the incorporation of high-level scene analysis to maximize information gain; and (c) the automatic belief propagation between layers that operate over different state spaces. The proposed hierarchical approach is hence called: I-HiPPo.

Experimental Setup and Results

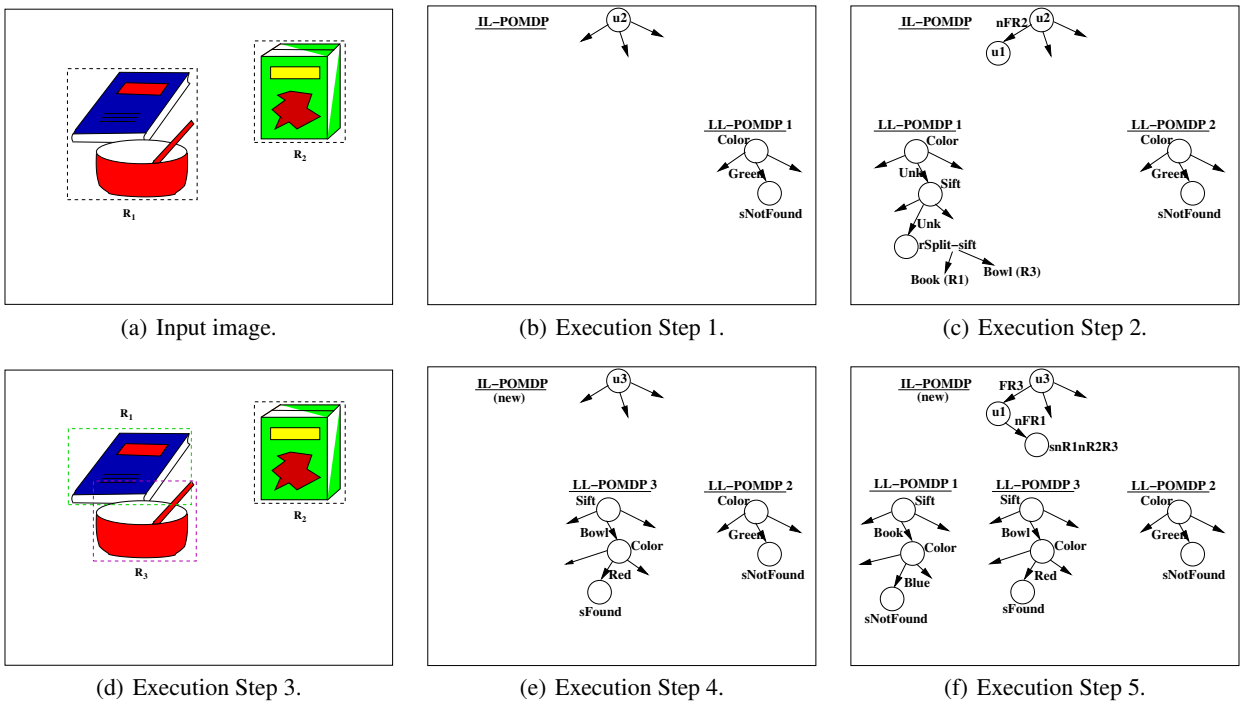
In this section, we describe the learning of the LL-POMDP components, followed by the experimental results.

The model creation in the LL-POMDPs requires observation and reward functions. Unlike other POMDP-based applications, we enable the robot to explicitly model the uncertainty in the operators. Objects with known labels (e.g. "red book", "blue square box") are placed at positions known to the robot. The robot executes different visual operators on images of the scenes to collect statistics of various action outcomes. These statistics are used to generate the observation functions, assuming the observations are mutually independent and produced by different actions. The LL action costs are a function of the relative run-time complexity of the operators and the size of the ROI (Equation 1). The run-time complexity is experimentally determined based on the statistics collected during observation function learning. The effect of ROI-size is modeled as a polynomial function:

$$f(r) = a_0 + \sum_{k=1}^{N} a_k \cdot r^k \tag{5}$$

where r is the ROI-size in pixels. The polynomial degree and coefficients are estimated such that they *best fit* the performance statistics collected during observation function learning ($N = 1, 3, 4$ for $shape, color, sift$ respectively). An existing saliency operator (Itti, Koch, and Niebur 1998) is used to determine the ROIs. For simple scenes, the system can use background subtraction to extract the ROIs.

We conducted several experiments on robot platforms to evaluate our algorithm. First, we evaluated the ability of the lower and intermediate layers of the proposed hierarchy (i.e. IL-POMDP and LL-POMDPs) on images of objects on a desk. Consider the execution example that answers the query: "where are the red bowls?" on the image shown in Figure 3(a)—this is a pictorial representation of an actual image captured by the robot. We include the actions: $color, shape, sift$ in addition to the region-splitting actions that split input ROIs based on these features i.e. $rSplit_{color}, rSplit_{shape}, rSplit_{sift}$. As stated earlier, such splitting actions would result in a modified state space for the IL-POMDP. In Figure 3(a) two of the three objects overlap, resulting in two ROIs. Since both ROIs are equally



(a) Input image. (b) Execution Step 1. (c) Execution Step 2.

(d) Execution Step 3. (e) Execution Step 4. (f) Execution Step 5.

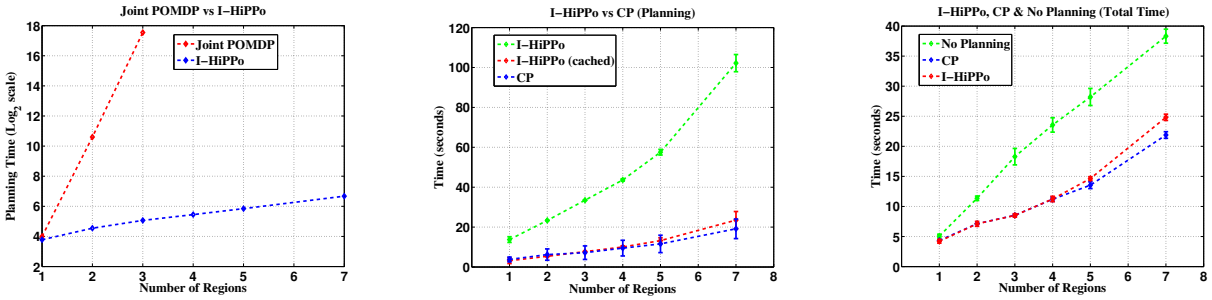
Figure 3: Example query: "Where are the red bowls?" Pictorial representation of actual execution on a robot. Region-split operators enable separation of overlapping objects.

likely target locations, the IL-POMDP chooses to examine the smaller ROI R_2 first, because of its smaller action costs—action u_2 in Figure 3(b). The corresponding LL-POMDP runs the color operator on the ROI, leading to the outcome of *green*. This outcome significantly reduces the likelihood of finding a red bowl, and a terminal action is chosen as the *best* action in the next step: *sNotFound*. The IL-POMDP receives the input that the target object was not found in R_2, leading to a belief update and subsequent action selection—action u_1 in Figure 3(c). The LL policy of R_1 is invoked, causing $color$ and $sift$ to be applied in turn on the ROI. Both operators come up with outcomes of *unknown* because the ROI has two different colors and objects. At this point, $rSplit_{sift}$ is chosen as the *best* action and R_1 is split into R_1 and R_3 on the basis of the sift (i.e. local gradient) features within the ROI—Figure 3(d). Our system includes other algorithms that can be invoked to split a ROI on the basis of color histograms or shape contours. The execution of $rSplit_{sift}$ is followed by the application of $sift$ on each sub-region, leading to the observations: *book* and *bowl* in R_1 and R_3 respectively—Figure 3(c). The current beliefs are used to create and solve a new IL-POMDP for three ROIs. The subsequent action selection in the IL (u_3) results in the execution of the LL-policy of R_3, whose initial belief reflects the previous application of $sift$. Hence, $color$ and $sift$ are applied just once before a terminal action (*sFound*) is chosen—Figure 3(e). The subsequent belief update and action choice in the IL leads to the processing of R_1 since the goal is locate *all* red bowls. The terminal action *sNotFound* for R_1 results in the action $s\neg R_1 \wedge \neg R_2 \wedge R_3$ in the IL—Figure 3(f).

As seen in the example above, some actions are applied more than once on a ROI to accumulate evidence. In such cases, conditional independence of the observations is ensured by taking a new image of the scene before repeating the action. Though the images are not strictly independent, the independence assumption is required for the belief update (Equation 2) and works in practice to account for small changes in factors such as illumination.

Similar to our prior work, we compared our approach against a non-deterministic planner that has been used successfully in human-robot interaction domains: Continual Planning (CP) (Brenner and Nebel 2008). We also considered the case where all operators were applied on the image ROIs ("no planning") until the target was found or all ROIs were analyzed. We considered a range of queries on scene properties, object occurrence, object location and object properties. For each query category, experiments were run on 20 different scenes, with 20 trials for each query. The queries represent combinations of object properties (colors, shapes, sift features). A subset of these experiments (on another robot platform) were reported in (Sridharan, Wyatt, and Dearden 2008; 2009).

Figure 4(a) shows that the "joint POMDP" that operates in the space of all ROIs and actions soon becomes intractable. The plot includes several trials on simulated scenes (with different number of ROIs of varying sizes) in addition to the robot experiments. The efficiency may be improved using faster POMDP solvers but the intractability would still be observed. The hierarchical approach significantly reduces



(a) HiPPo vs. joint POMDP. Joint POMDP soon becomes intractable.

(b) Planning times of HiPPo vs. CP. Policy-caching makes results comparable.

(c) Planning+execution times of HiPPo, CP vs. No planning.

Figure 4: Experimental Results: comparing planning and execution times of I-HiPPo and CP against no planning, on specific scenes. I-HiPPo and CP reduce processing time.

the planning time—though the computed policies are not optimal the performance is reliable. Next, Figure 4(b) compares the planning times of the hierarchical approach and CP as a function of the number of ROIs. The default hierarchical approach needs to compute policies for all ROIs (LL-POMDPs) and is hence more expensive than CP. However, if no prior information is available regarding the contents of the different ROIs, the LL-POMDPs only differ in the action costs. The ROI-sizes are hence discretized, and the policies of ROIs within specific size ranges are cached and reused. Though this policy caching results in planning times comparable to CP, it introduces value estimation errors. We estimate these errors to trade-off reliability and efficiency (Sridharan, Wyatt, and Dearden 2009).

Figure 4(c) compares the planning approaches against the no-planning approach in terms of the combined planning and execution times. Planning provides benefits even on scenes with just two ROIs, and the benefits are much more significant as the number of ROIs and visual operators increase. In these experiments the robot had modules operating in parallel to analyze the sensory inputs. Therefore, though the individual operators are optimized for performance, the parallelism results in the times reported above.

As mentioned earlier, the goal is not to create a POMDP solver but to provide a hierarchy that sequences available operators to achieve reliable and efficient visual processing. Table 1 summarizes the reliability performance over the range of tasks, with the ground truth provided manually. With no planning, the average reliability (i.e. classification accuracy) is 76.67% i.e. the visual operators provide incorrect results approximately once every four trials. Though CP is very efficient, it does not model the action outcomes and hence cannot achieve higher reliability than the naive approach of applying all available operators. Our approach explicitly models the uncertainty and accumulates evidence to provide higher reliability.

Approach	% Reliability
No planning	76.67
CP	76.67
I-HiPPo	90.75

Table 1: Reliability of visual processing

Finally, we ran experiments to evaluate the HL-POMDP's capabilities. The robot's environment was discretized into grid cells, and the robot had to locate specific objects (e.g. soccer ball, bowl). Since running extensive trials on a physical robot is infeasible, we also ran experiments in a simulated domain that mimicked the robot's sensing capabilities (based on the computed observation functions). The simulated grid currently includes the locations of certain known obstacles (e.g. walls) and objects (e.g. tables) though this can also be learned by the robot, using existing methods (Thrun, Burgard, and Fox 2005). The different grid sizes correspond to the different regions over which the evaluation was conducted. The size of each cell in the grid is set based on the field of view of the robot's camera (e.g. 60cm, 100cm).

Grid size	% Accuracy
5×5	88
7×7	90
10×10	92

Table 2: Accuracy of visual search for different grid resolutions.

Table 2 summarizes the results, including ≈ 1000 trials (for each grid size) in the simulated domain. In a training phase, the robot learns the LL observation functions and a HL policy for visual search. Then, the policy is evaluated by placing target objects in different grid cells. A trial is deemed to be successful if the policy results in identifying the location of the target in the correct cell. In Table 2, most failures correspond to cases where the target object is identified but its estimated location is no more than one cell-width away from the actual location. Such errors are due to the grid cell resolution or the errors in robot sensing and motion. In addition, the policy performs significantly better than a random selection of target grid cells, converging quickly after the cell with the target is analyzed once. The planning time does vary from a few minutes to a few hours based on the size of the grid being analyzed. We are currently investigating the use of other POMDP solvers and convolutional policies to speed up the planning process.

Conclusions

Mobile robot environments are characterized by partial observability, non-determinism and computational complex-

ity. Though POMDP formulations elegantly encapsulate all these factors, they are intractable for all but the most simple problems. In this paper, we have summarized our recent work on using a three-layered hierarchical POMDPs for visual scene analysis. We have presented evaluation results of our approach on physical robot platforms and simulated domains. We are currently in the process of fully integrating the different components of the proposed approach on multiple robot platforms, in order to extensively evaluate our approach in challenging indoor scenarios. We are also working on including several additional visual operators, which may require that we factor the state and action spaces. Furthermore, we aim to incorporate actions that change the physical state of the system (e.g. move an object), in order to reason about action affordances.

Our goal here is to enable a robot to fully utilize the rich information encoded in camera images. Our approach is not a POMDP solver but a scheme to sequence a subset of existing (unreliable) operators to achieve reliable and efficient visual processing for the task at hand. Our ultimate aim is to enable robots to collaborate autonomously, reliably and efficiently with humans in dynamic scenarios.

Acknowledgments

This work was sponsored in part by the ONR award N00014-09-1-0658.

References

Brenner, M., and Nebel, B. 2008. Continual Planning and Acting in Dynamic Multiagent Environments. *Journal of Autonomous Agents and Multiagent Systems*.

Buffet, O., and Aberdeen, D. 2009. The Factored Policy-Gradient Planner. *Artificial Intelligence* 173(5-6):722–747.

Butko, N. J., and Movellan, J. R. 2008. I-POMDP: An Infomax Model of Eye Movement. In *The IEEE International Conference on Development and Learning (ICDL)*.

Casper, J., and Murphy, R. R. 2003. Human-robot Interactions during Urban Search and Rescue at the WTC. *Transactions on SMC*.

Clouard, R.; Elmoataz, A.; Porquet, C.; and Revenu, M. 1999. Borg: A Knowledge-Based System for Automatic Generation of Image Processing Programs. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 21(2):128–144.

Darrell, T. 1997. Reinforcement Learning of Active Recognition Behaviors. Technical report, Interval Research Technical Report 1997-045.

Foka, A. F., and Trahanias, P. E. 2005. Real-time Hierarchical POMDPs for Autonomous Robot Navigation. In *IJCAI Workshop on Reasoning with Uncertainty in Robotics*.

Hansen, E. A., and Zhou, R. 2003. Synthesis of Hierarchical Finite-State Controllers for POMDPs. In *ICAPS*, 113–122.

Hokuyo. 2008. Hokuyo Laser. `http://www.hokuyo-aut.jp/products/`.

Hsiao, K.; Kaelbling, L. P.; and Lozano-Perez, T. 2007. Grasping POMDPs. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Itti, L.; Koch, C.; and Niebur, E. 1998. A Model of Saliency-Based Visual Attention for Rapid Scene Analysis. *PAMI* 20(11):1254–1259.

Kaelbling, L.; Littman, M.; and Cassandra, A. 1998. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence* 101:99–134.

Li, L.; Bulitko, V.; Greiner, R.; and Levner, I. 2003. Improving an Adaptive Image Interpretation System by Leveraging. In *Australian and New Zealand Conference on Intelligent Information Systems*.

Lowe, D. 2004. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision (IJCV)* 60(2):91–110.

Nao. 2008. The Aldebaran Nao Robots. `http://www.aldebaran-robotics.com/`.

Petrick, R., and Bacchus, F. 2004. Extending the Knowledge-Based approach to Planning with Incomplete Information and Sensing. In *ICAPS*, 2–11.

Pineau, J., and Thrun, S. 2002. High-level Robot Behavior Control using POMDPs. In *AAAI*.

Ross, S.; Pineau, J.; Paquet, S.; and Chaib-draa, B. 2008. Online Planning Algorithms for POMDPs. *Journal of Artificial Intelligence Research* 32:663–704.

Saxena, A.; Driemeyer, J.; and Ng, A. Y. 2008. Robotic Grasping of Novel Objects using Vision. *International Journal of Robotics Research* 27(2):157–173.

Smith, T., and Simmons, R. 2005. Point-based POMDP Algorithms: Improved Analysis and Implementation. In *UAI*.

Sridharan, M.; Wyatt, J.; and Dearden, R. 2008. HiPPo: Hierarchical POMDPs for Planning Information Processing and Sensing Actions on a Robot. In *ICAPS*.

Sridharan, M.; Wyatt, J.; and Dearden, R. 2009. POMDP-based Planning for Visual Processing Management on a Mobile Robot. In *Cognitive Vision Workshop at IROS*.

Thrun, S.; Burgard, W.; and Fox, D. 2005. *Probabilistic Robotics*. Cambridge, USA: MIT Press.

Thrun, S. 2006. Stanley: The Robot that Won the DARPA Grand Challenge. *Field Robotics* 23(9):661–692.

Toussaint, M.; Charlin, L.; and Poupart, P. 2008. Hierarchical POMDP Controller Optimization by Likelihood Maximization. In *Uncertainty in AI (UAI)*.

Videre Design. 2010. Videre Design Robots. `http://www.videredesign.com/robots/era_mobi.htm`.

2008. ZMDP Planning Code. `http://www.cs.cmu.edu/~trey/zmdp/`.