Active Visual Sensing and Collaboration on Mobile Robots using Hierarchical POMDPs

Shiqi Zhang
Department of Computer Science
Texas Tech University
s.zhang@ttu.edu

Mohan Sridharan
Department of Computer Science
Texas Tech University
mohan.sridharan@ttu.edu

ABSTRACT

A key challenge to widespread deployment of mobile robots in the real-world is the ability to robustly and autonomously sense the environment and collaborate with teammates. Real-world domains are characterized by partial observability, non-deterministic action outcomes and unforeseen changes, making autonomous sensing and collaboration a formidable challenge. This paper poses vision-based sensing, information processing and collaboration as an instance of probabilistic planning using partially observable Markov decision processes. Reliable, efficient and autonomous operation is achieved using a hierarchical decomposition that includes: (a) convolutional policies to exploit the local symmetry of high-level visual search; (b) adaptive observation functions, policy re-weighting, automatic belief propagation and online updates of the domain map for autonomous adaptation to domain changes; and (c) a probabilistic strategy for a team of robots to robustly share beliefs. All algorithms are evaluated in simulation and on physical robots localizing target objects in dynamic indoor domains.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics

General Terms

Algorithms, Experimentation

Keywords

Integrated perception, cognition, and action; Robot planning (including action and motion planning); Robot teams, multi-robot systems, robot coordination.

1. INTRODUCTION

Autonomous and robust sensing and collaboration is a key challenge to widespread deployment of mobile robots in the real-world. Real-world application domains are characterized by partial observability, non-deterministic action outcomes and unforeseen dynamic changes. A robot equipped with multiple sensors (e.g., cameras and range finders) can use different algorithms to process sensory inputs with varying levels of reliability and computational complexity. It is not feasible for the robot to observe the entire domain or process all sensory inputs with all available algorithms and still respond to dynamic changes. At the same time, robust operation requires that the robot make best use of relevant information. Furthermore, each robot in a team can possess different capabilities and communication between robots can be unreliable. Autonomous and robust sensing and collaboration on robots deployed in the real-world is hence a formidable challenge.

This paper poses vision-based sensing and collaboration as a planning task and uses partially observable Markov decision processes (POMDPs) [8] to enable each robot in a team to tailor sensing and processing to the task at hand. Although POMDPs elegantly model the non-determinism and partial observability of real-world domains, the state space of these domains typically increases exponentially and even state of the art (approximate) POMDP solvers have high computational complexity [14, 19]. Our prior work introduced a hierarchical decomposition in POMDPs for reliable and efficient visual sensing and processing in simulation and simplistic tabletop scenarios [22, 23]. This paper builds on our prior work to enable a robot to autonomously direct sensing to relevant locations, and consider the reliability and complexity of available algorithms to determine the sequence of sensing and processing actions best suited to a given task. Each robot then shares beliefs (acquired by processing sensory cues) with teammates to collaborate robustly in real-world domains. The following novel contributions are made:

- Local symmetries in visual sensing are exploited to learn convolutional policies for efficient operation over large state spaces.
- Adaptive observation functions, policy re-weighting and automatic belief propagation in the hierarchy are used in conjunction with online revisions to domain map (based on range data) to enable the robot to adapt to dynamic changes.
- A probabilistic belief sharing strategy is used to enable a team of robots to merge individual and communicated beliefs to collaborate robustly despite unreliable communication.

These contributions enable the use of POMDPs for reliable, efficient and autonomous visual sensing and collaboration on mobile robots. All algorithms are evaluated in simulation and on physical robots deployed to localize target objects in dynamic indoor domains. The remainder of the paper is organized as follows. Section 2 summarizes related work, while Section 3 describes the hierarchical planning approach. Experimental results are described in Section 4, followed by conclusions in Section 5.

2. RELATED WORK

Research in vision, planning and robotics has produced sophisticated algorithms for planning a pipeline of visual operators for a high-level goal. Many such algorithms use deterministic action models whose preconditions and effects are propositions that need to be true apriori, or are made true by executing the operator. However, such formulations are insufficient for application domains with partially observable state and non-deterministic action outcomes.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

In vision research, image interpretation has been modeled using MDPs and POMDPs. Li et al. [12] used human-annotated images to determine the reward structure, explore the state space and compute value functions—actions that maximize the learned functions are chosen during online operation. Similarly, active sensing has been used to decide sensor placement and information processing, using particle filters and relative entropy maximization for estimating a joint multitarget probability density [10]. Sensor placements in spatial phenomena have also been modeled as Gaussian processes using submodular functions [9]. However, many visual planning tasks are not submodular, and it is difficult to model probability densities using manual feedback over many trials on robots.

Since a POMDP formulation can become intractable due to the exponential state explosion of real-world domains, researchers have focused on imposing structure on application domains. Pineau and Thrun [16] proposed a hierarchical approach for behavior control of a robot assistant. The top level action is a collection of simpler actions modeled as smaller POMDPs and solved completely to enable bottom-up planning and top-down plan execution. Similar approaches have been used for robot navigation [7] but a significant amount of data for the hierarchy and model creation is hand-coded. Recent work has focused on learning POMDP observation models [1]; using information maximization for POMDP-based visual search [4, 23], and developing factored representations and faster POMDP solvers [14, 19]. Researchers have also focused on integrating human input in POMDPs for human-robot interaction [18]. However, these methods are still not suitable for dynamic domains with large state spaces, and do not enable automatic model creation and belief propagation that is essential for robot domains.

Many algorithms continue to be developed for multiagent and multirobot collaboration in a variety of domains [15]. Sophisticated algorithms have also been developed recently for using decentralized POMDPs (Dec-POMDPs) for multiagent and multirobot collaboration [11]. However, the computational complexity of these formulations is more than that of POMDP formulations [2]. Research has also shown that using complex communication strategies does not necessarily improve task completion times [21]. This paper addresses these challenges using hierarchical POMDPs that enable autonomous active visual sensing on each robot and robust collaboration between a team of robots.

3. PROBLEM FORMULATION

Figure 1 summarizes the POMDP hierarchy for visual sensing, processing and collaboration. Each robot uses the hierarchy to locate one or more target objects. The top-level visual sensing (**VS**)-**POMDP** determines the sequence of 3D scenes to process to locate a specific target, as described in Sections 3.1–3.3. For each chosen scene, the scene processing (**SP**)-**POMDP** determines the sequence of regions to process in a sequence of images using the appropriate set of algorithms. The SP-POMDP has one or two layers depending on the characterization of the learned object models, as described in Section 3.4. The hierarchy is then augmented with a *communication layer* that enables each robot in a team to share beliefs with teammates to collaborate robustly despite unreliable communication, as described in Section 3.5.

3.1 POMDP Planning

In real-world domains, the robot has to move and analyze different scenes to locate target objects that can exist in different locations. Consider the situation where a robot has learned a domain map [6] and has to locate a specific target. The 3D area is represented as a discrete 2D *occupancy grid* and each grid cell stores the probability of occurrence of the target object. The VS-POMDP

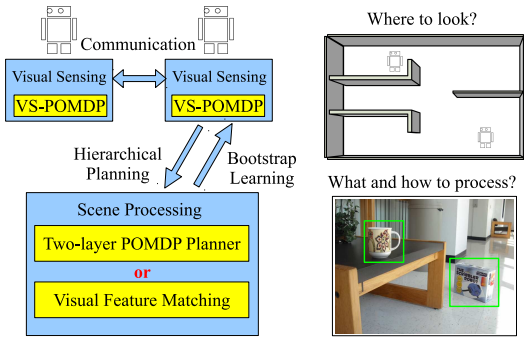


Figure 1: Overview of POMDP hierarchy for target localization.

poses sensing as the task of maximizing information gain, i.e., reducing the belief state entropy in a grid with N cells. The POMDP tuple $\langle S, A, T, Z, O, R\rangle$ is defined as:

- $S : s_i,\ i \in [1, N]$ is the state vector; s_i corresponds to the event that the target is in grid cell i.
- $A : a_i, i \in [1, N]$ is the set of actions. Executing a_i causes the robot to move to and analyze grid cell i.
- $T : S \times A \times S \rightarrow [0, 1]$ is the state transition function. It is an identity matrix here because actions do not change state.
- Z : {present, absent} is the observation set that indicates if the target is detected.
- $O : S \times A \times Z \rightarrow [0, 1]$ is the observation function (see below).
- $R : S \times A \rightarrow \mathbb{R}$ is the reward specification that is based on belief entropy (see below).

The robot maintains a *belief state*, a probability distribution over the state. The *entropy* of belief distribution B_t is given by:

$$\mathscr{H}(B_t) = -\sum_{i=1}^{N} b_t^i log(b_t^i) \tag{1}$$

where b^i is the i^{th} entry of the belief distributed over the N grid cells. With no prior knowledge of target location, the belief is uniformly distributed and entropy is maximum. The VS-POMDP aims to choose actions that significantly reduce the entropy by causing the belief distribution to converge to likely target locations. The reward of action a_t at time t is hence defined as the entropy reduction between belief state B_{t-1} and the resultant belief state B_t:

$$\begin{aligned} R(a_t) :=& \mathscr{H}(B_{t-1}) - \mathscr{H}(B_t) \\ =& \sum_k b_t^k log(b_t^k) - \sum_j b_{t-1}^j log(b_{t-1}^j) \end{aligned} \tag{2}$$

The observation function models the probability of target detection as a function of the robot position and target position:

$$\begin{aligned} &\textbf{if } isBlocked(s_j, a_k) \\ &\quad O(z_i = present, s_j, a_k) = Pr(z_i = present | s_j, a_k) = \beta \\ &\textbf{else} \\ &\quad O(z_i = present, s_j, a_k) = \eta \cdot exp\{-\lambda \mu^2 / 2\sigma^2\} \\ &O(z_i = absent, s_j, a_k) = 1 - O(z_i = present, s_j, a_k) \end{aligned} \tag{3}$$

where the probability of observation "present" in cell i given that the target is in cell j and the focus is on cell k, i.e., $p(z_i|s_j, a_k)$, is a Gaussian distribution whose mean depends on the target location, the grid cell being examined and the field of view: $\mu = f_\mu(s_j, a_k)$. The variance of the Gaussian represents the sensitivity of sensory cues to the object's distance from the sensor—there is more uncertainty associated with the observation of a target at a greater distance. The factor η is a normalizer. If there is any obstacle be-

tween the robot and the target, i.e., $isBlocked(s_j, a_k)$, β is a small probability that the target can still be observed. This observation function is used to perform belief updates after sensing actions provide observations, and to generate observations in the simulated experiments. Given these model parameters, belief update in the VS-POMDP proceeds as follows:

$$B_{t+1}(s') = \frac{O(s', a_{t+1}, o_{t+1}) \sum_s T(s, a_{t+1}, s') \cdot B_t(s)}{p(o_{t+1}|a_{t+1}, b_t)} \quad (4)$$

POMDP solvers take such a model and compute a *policy* that maps belief states to actions: $\pi : B_t \mapsto a_{t+1}$. In the VS-POMDP, the computed policy has to minimize entropy in B_t over a planning horizon. Policy gradient algorithms are used to compute the policy in the form of stochastic action choices, i.e., the policy is learned as a matrix of "weights" that are used (during plan execution) to probabilistically choose an action for specific belief states [3]. Actions in the VS-POMDP require the robot to physically move between grid cells, expending time and effort. Instead of the formulation described above, motion costs are addressed in a post-processing step, as described in Section 3.3.

3.2 Convolutional Policy

In real-world domains, the state space of the VS-POMDP can increase exponentially, making it intractable to compute the policy in real-time even with sophisticated solvers. This challenge is addressed by exploiting the local shift and rotation symmetries of visual processing. Specifically, if the robot is analyzing a specific grid cell, only the beliefs immediately around that grid cell change substantially, i.e., the performance is a function of (and can affect) only a small number of surrounding cells. The robot captures this local influence by learning a *policy kernel* based on a *baseline* policy for a map with a small number of grid cells. The policy for a larger map with a larger number of grid cells is generated automatically by an inexpensive convolution operation. This section describes the creation of policy kernels and the use of convolutional policies for efficient sensing and processing.

3.2.1 Kernel Extraction

Consider the stochastic baseline policy generated for a 5×5 map, which has 25 states and 25 actions. In the 2D matrix of action weights, each column corresponds to an action and each row corresponds to a state. The matrix is re-organized into layers, where each layer corresponds to action weights for a particular state and is a represented as a 2D matrix of the same size as the map. This re-organization enables the robot to use the local symmetries (i.e., shift and rotation invariance) to extract a kernel without significant loss of information:

$$\bar{K}(s) = (\pi^V \otimes C_m^K)(s) = \int \pi^V(\tilde{s}) C_m^K(s - \tilde{s}) d\tilde{s}, \quad (5)$$
$$K = (\sum_{states} \bar{K}) \cdot / W$$

where $\bar{K}$ is the un-normalized kernel, π^V is baseline policy generated for the VS-POMDP over the 5×5 map and C_m^K is the convolution mask of the same size as the target kernel. Since the mask only considers action weights within a local region, the layers of the resultant kernel are summed up and normalized using W, a matrix that stores the count of the number of accumulated weights across all layers. For instance, a 3×3 policy kernel is computed by convolving a 3×3 mask with the 5×5 policy layers and normalizing the weights in the region covered by the mask.

The computed kernel does not assign action weights to grid cells further away from the center of the convolution mask. Since these action weights are usually much lower than values in the kernel, they can all be set to a small default value:

$$w^d = \frac{\sum_{actions} \sum_{states} \pi^V - \sum_{states} \bar{K}}{N_{actions} \times N_{states} - \sum W} \quad (6)$$

where the default action weight w^d is a function of the number of states (N_{states}) and actions ($N_{actions}$). To prevent the summation of "small weights" from overwhelming the kernel's weights when generating policies for large maps, w^d is revised to make the ratio of importance assigned to the area covered and left uncovered by the kernel to be similar over maps of different sizes:

$$\hat{w}^d = w^d - ln(\frac{N_{states}^E - sz(W)}{N_{states}^K - sz(W)}) \quad (7)$$

where N_{states}^E and N_{states}^K are the number of states in the large map and kernel respectively, and $sz(W)$ is the number of entries in W.

3.2.2 Policy Extension

Once a policy kernel has been learned, it can be used to efficiently compute the convolutional policy for a larger map:

$$\pi_C^V(s) = (K \otimes C_m^E)(s) = \int K(\tilde{s}) C_m^E(s - \tilde{s}) d\tilde{s} \quad (8)$$

where π_C^V is the convolutional policy, K is the policy kernel and C_m^E is the convolution mask of the same size as the target map. For instance, for a 10×10 map, C_m^E is a 10×10 mask over which the 3×3 policy kernel is convolved. The desired policy is generated one layer at a time by centering the kernel on the state represented by the layer. Since the kernel covers only grid-cells in a small area, other cells are assigned the weight computed in Equation 7 and the resultant policy is normalized. Although it may take some time for the robot to learn a baseline policy for a small map, it is a one-time computation. The kernel extracted from a baseline policy needs to be revised only when the robot's sensors change substantially.

3.3 Motion Costs and Path Planning

Unlike visual search over an image, a mobile robot has to physically move between grid cells. The movement takes time and is associated with unreliability that has a cumulative effect as the distance traveled increases. Each action is hence assigned a cost proportional to the distance to be traveled by revising the action's policy weights during policy execution:

$$\hat{w}(i) = w(i) \frac{1}{1 + \frac{d_{A^*}(a_i, a_j)}{\text{speed}}} \quad (9)$$

where $d_{A^*}(a_i, a_j)$ is the distance between the current grid cell and the candidate grid cell, which is computed using the A^* search algorithm [20]. The A^* search includes a heuristic cost to the target grid cell and a path cost to account for obstacles (e.g., walls) in the domain map. The revised policy trades off the expected likelihood of locating the target in a specific grid cell against the cost of traveling to that location. When the domain map changes due to changes in object configurations (e.g., objects are moved and/or new obstacles are created), the robot automatically revises the map using laser-based simultaneous localization and mapping (SLAM) algorithms. The modified map is used to recompute distances between grid cells and revise action weights for subsequent computations.

In addition to revising action weights to model motion-based costs, hill-climbing is used to make the search more efficient in large maps. Consider Figure 2, which shows a domain map (similar to Figure 8) discretized into grid cells. The green grid is the current position of the robot after executing the most recent action. At this point, there are three grid cells in the map with significantly

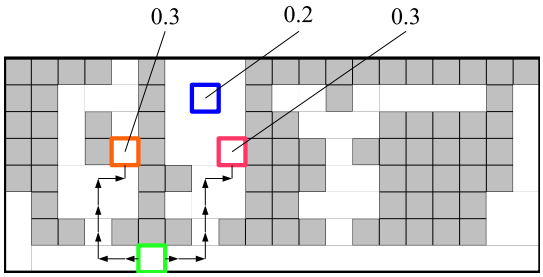


Figure 2: Illustration of hot-spot detection.

higher weights than the other cells: the orange and pink grids have $w = 0.3$ and the blue grid has $w = 0.2$. Since the robot's current position is equidistant from the pink and orange grids, these grids have an equal chance of being the next grid cell visited by the robot. However, given that the robot has three valid candidates of similar relevance, it makes sense to visit the pink cell first because it is also close to the blue grid cell. Instead of looking for a grid cell with the largest $\hat{w}$ (Equation 9), the robot therefore selects the path through the candidate grid cells that has the largest summation of $\hat{w}$ values. Since it is computationally expensive to estimate an optimal path by evaluating all paths through all grid cells in a large map, the robot detects "hot-spots", i.e., grid cells with sufficiently large beliefs, and plans a path through them.

To compute hot-spots, N seeds are randomly selected and then refined based on hill-climbing to arrive at local maxima, i.e., cells similar to the orange, blue and pink grids in Figure 2. These hot-spots are considered to be the interesting areas for further analysis. The robot then computes the values of paths w^p through combinations of these hot-spots:

$$w^p([h_0, h_1, \ldots, h_N]) = \sum_{i=1}^{N} f(w_i, \sum_{j=1}^{i} d_{A^*}(h_{j-1}, h_j)) \qquad (10)$$

where, h_n is the nth hot-spot, h_0 is the current position of the robot and other entries are chosen by hill-climbing. The function f is defined in Equation 9. In Figure 2, the values of the *pink-blue-orange* and *orange-pink-blue* paths are 0.0672 and 0.0591 respectively, making the pink grid cell the most likely choice for being analyzed next. This path planning does *not* imply that the robot will move through all the hot-spots—once a robot arrives at a grid cell, the corresponding observation revises the belief distribution and hence the planned path. The path planning ensures that the robot's attention is directed towards the most interesting grid cells.

3.4 Scene Processing

Invoking the VS-POMDP policy computed for a specific target causes a 3D scene to be chosen for analysis. The robot moves and captures images of this scene. As stated earlier, there are two options for scene processing depending on the scene complexity and learned object models—specific examples are provided in Section 4. In uncluttered scenes with unique objects, the SP-POMDP is a two layered POMDP as described in [22]. Each input image is analyzed to extract salient regions of interest (ROI). Each ROI is modeled as a lower-level (**LL**)-**POMDP**, where actions are information processing operators (e.g., to detect color or shape). The LL policy provides the best sequence of operators to apply on a specific ROI to detect the target. The LL policies of all image ROIs are used to automatically create a high-level (**HL**)-**POMDP**. Executing an action in the corresponding HL policy directs robot's attention to a specific ROI. The result of executing the corresponding LL policy causes an HL belief update and action choice until presence or absence of the target in the image is determined. In cluttered scenes with sophisticated learned object models, the robot may need to process the entire image. Scene processing is then reduced to a single POMDP over the image. With either version of SP-POMDP, the result of scene processing causes a belief update in the VS-POMDP and subsequent analysis of grid cells until the target is found or a time limit is exceeded. The entire hierarchy operates automatically and efficiently for dynamic domains.

3.5 Multirobot Collaboration

Consider (next) a team of X robots trying to locate Y targets. Each robot maintains a belief vector for each target, and uses the hierarchical POMDPs to detect each target. This section describes an algorithm for a team of robots to share beliefs and collaborate to locate all targets reliably and efficiently.

We assume that the targets are visually distinguishable and that the observations of different targets are independent of each other. Each robot now stores a data structure:

$$\{B_i, f_i\}, \forall i \in [1, |TL|] \qquad (11)$$

where B_i is the belief vector for a specific target i among the list of target objects (TL) and f_i is a binary flag that indicates discovery of a target. The robot also stores an action map $\mathscr{M}$, a vector of the same size as the belief vector. Each entry in this vector stores the number of times the robot has visited the corresponding grid cell:

$$\mathscr{M} = \langle m_1, \cdots, m_N \rangle \qquad (12)$$

where m_i is the count of the number of times grid-cell i has been visited. For moving targets, values in the action map decay over time if they are not reinforced by more recent visits. Each robot uses the POMDP hierarchy to update the appropriate belief vectors based on observations. After the belief update, each robot shares the belief information with its teammates by broadcasting a package that includes its current belief vectors ($\forall i\, B_i$), discovery flags ($\forall i\, f_i$) and the action map ($\mathscr{M}$).

There is uncertainty associated with sensing on each robot and communication between robots—the information from a teammate (when received successfully) may reinforce or contradict the information acquired by the robot by processing sensory inputs. At the same time, the communicated estimates provide useful information about map locations that the robot has not visited. Each robot hence merges own and communicated beliefs by assigning a trust factor to beliefs based on whether the robot that generated this belief vector has recently observed the corresponding map region:

$$b_i^{j,own} = \frac{m_i^{j,own} \cdot b_i^{j,own} + m_i^{j,comm} \cdot b_i^{j,comm}}{m_i^{j,own} + m_i^{j,comm}} \qquad (13)$$
$$\forall j \in [1, N], \quad \forall i \in [1, |TL|]$$

where b_i^j is j^{th} entry of the belief vector of the i^{th} target, while $m_i^{j,own}$ and $m_i^{j,comm}$ are action map entries of the robot and the teammate whose communicated belief is being merged. Although this merging process can be sensitive to processing order, it works well in practice. Next, the target discovery flags are updated:

$$\mathscr{F} = \{ f_i^{own} || f_i^{comm}; \forall i \in [1, |TL|] \} \qquad (14)$$

where each target is considered to be found when at least one robot has localized it. Once a target is discovered, a robot that requires a new target chooses an undiscovered object from the list ($|TL|$):

$$targetID = argmax_i \{ \max_j B_i(j) \} \qquad (15)$$

where the robot chooses the target object whose location it is most certain about, i.e., the target that is likely to require the least amount

of work to localize. The robot makes this choice based on current beliefs that include the beliefs communicated by teammates. This target selection approach (intentionally) includes some overlap of targets among robots to account for unreliable communication, but robots in the team distribute tasks and rarely go to the same target. Furthermore, an additional cost is included to trade-off distance of travel against expected likelihood of locating the target (or priority of target, if known), similar to Equation 9.

4. EXPERIMENTAL RESULTS

This section describes the results of experiments performed to evaluate the robot's ability to: (a) use *convolutional policies* and the POMDP hierarchy for reliable, autonomous and efficient visual sensing and processing in complex domains; and (b) probabilistically merge own beliefs with communicated beliefs of teammates to achieve robust collaboration. Experiments were hence conducted in simulation and on robots to evaluate the following hypotheses: (I) the constrained convolution (CC) policy is more efficient than the non-convolutional (i.e., baseline) policy while providing similar accuracy; (II) the POMDP hierarchy results in better target localization in comparison to heuristic search strategies; and (III) the belief merging strategy enables a team of robots to share beliefs and collaborate robustly despite unreliable communication.

4.1 Experimental Setup

Before describing the experimental results, this section describes the initial setup and the modifications necessary for experimental trials on robots. The initial setup consisted of a semi-supervised learning phase, where some objects with known labels were placed in front of the robot. The robot applied different processing operators on images of these objects to learn object models and some model parameters of the VS-POMDP and SP-POMDP (e.g., observation functions, reward specifications). Examples of learned object models are described in Sections 4.3.1 and 4.3.2. The robot also used data from a laser range finder to learn a domain map that was revised continuously during experimental trials.

For any detected object, the robot computes the relative distance and bearing using geometric transforms. However, including orientation as a parameter in the observation set will destroy the local symmetry in visual sensing. The belief update in Equation 4 was therefore modified as:

if $\neg target$ (16)

$$B(s') = \frac{O(s',a,o)\sum_{s\in S} T(s,a,s')b(s)}{Pr(o|a,b)} = \frac{O(s',a,o)b(s)}{Pr(o|a,b)}$$

else

$$B(s') = \frac{O(s',\hat{a},o)\sum_{s\in S} T(s,\hat{a},s')b(s)}{Pr(o|\hat{a},b)} = \frac{O(s',\hat{a},o)b(s)}{Pr(o|\hat{a},b)}$$

where $B(s')$ is the updated belief for state s' after action a. Since the transition functions are identity matrices, the update equation can be simplified as shown. The robot's estimate of its own position and the relative distance and bearing of a detected target are used to find the target's global location in the domain map. The belief is then updated as if the action corresponding to this global location had been executed: $\hat{a}$. This belief update scheme also models the fact that false positives are rare while false negatives are common when sensing (or processing) actions are executed on mobile robots. Furthermore, a robot moving between grid cells may receive sensory inputs relevant to the current task, e.g., it may unexpectedly have the target in its field of view. The robot therefore periodically processes input images at low-resolution to update the current belief.

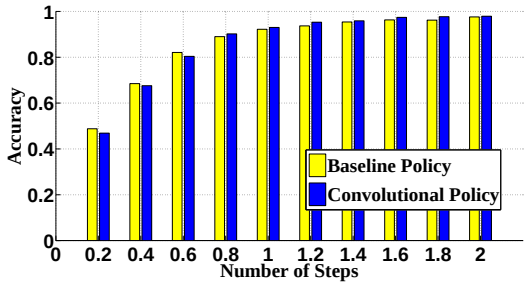


Figure 3: CC policy performs as good as the baseline policy.

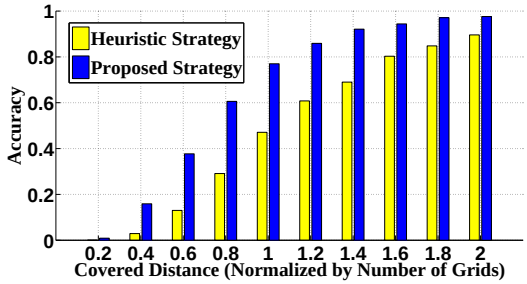


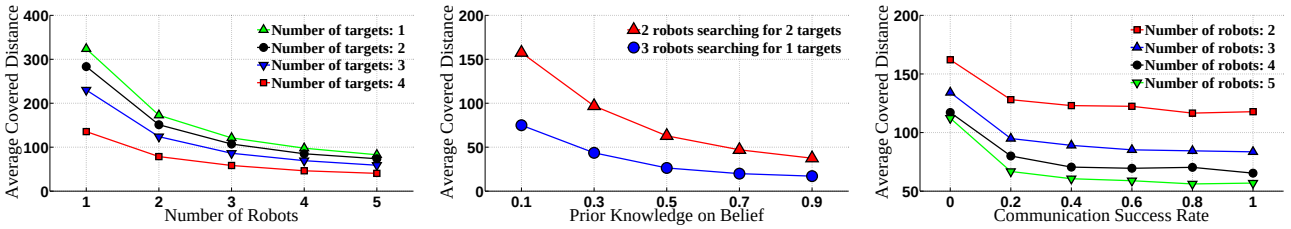
Figure 4: CC policy performs better than a heuristic strategy.

4.2 Simulation Experiments

All three hypotheses were evaluated extensively in simulation using domain maps that represented different sections of the map shown in Figure 8. Each data point in the figures in this section is the average of 1000 simulated trials. To evaluate hypothesis I, a baseline policy computed for a 5×5 map was used to extract a policy kernel that was used to compute policies for larger maps. Figure 3 compares the CC policy against the baseline policy for a 7×7 map—the x-axis shows the number of times the policy was invoked, as a fraction of the number of states. In each trial, the initial positions of the target and the robot were set randomly and the trial was deemed successful if the target was localized correctly. There is no statistically significant difference in the target localization accuracies of the CC and baseline policies. However, it takes a few hours to compute the baseline policy for the 7×7 map.

Hypothesis II was evaluated by comparing the CC policy's performance against a heuristic policy that makes greedy action choices or selects random actions based on the presence/absence of prior knowledge. The results shown in Figure 4 correspond to a 15×15 convolutional policy generated from a 5×5 kernel. The locations of the robot and the target were randomly selected for each trial. Existence of prior knowledge was simulated by adding bias to the initial belief—70% of the belief was uniformly distributed over all grid cells, while the remaining 30% was Gaussian-distributed around the target. To generate the data points in Figure 4, trials were terminated after a certain distance had been traveled and the grid cell with the largest belief value was taken to be the target's location. The robot's performance is scored as the weighted distance between the actual and detected locations of the target. Figure 4 shows that the CC policy significantly reduces the number of action steps required to locate the target with high accuracy.

Experiments were conducted next to evaluate the multirobot collaboration capability, i.e., hypothesis III. Assuming that all robots in a team move at the same speed, the average distance moved by the robots in a team (in an episode/trial) was used as a measure of the team's performance. In each trial, robots and targets were placed randomly in a grid map, with no more than one robot or target in each grid-cell. A Gaussian bias (20%) was added to the initial



(a) Multirobot search. (b) Effect of Bias. (c) Effect of CSR.

Figure 5: (a) Belief merging and hierarchical POMDPs result in robust multirobot collaboration; (b) Performance improves if prior information is incorporated; and (c)Performance is robust to dropped communication packages.

Table 1: Proposed algorithms enable a robot team to localize targets more accurately than random and heuristic search strategies.

Algorithm	Normalized covered distance			
	0.5	1.0	1.5	2.0
Random	0.033	0.171	0.382	0.537
Heuristic	0.079	0.334	0.549	0.817
Proposed	0.153	0.544	0.825	0.957

belief in a 3×3 area around every target—the belief vector was then normalized. When the belief in a grid cell exceeded 0.9, the grid cell was assumed to contain a target. To simulate unreliable communication, a *communication success rate* (CSR) parameter was introduced and set to 0.5, i.e., every other broadcasted package was not received. Figure 5(a) shows results for different combinations of robots and targets in a 15×15 grid map—the robots collaborate effectively to find the targets. Similar results were obtained for grid maps of different sizes (4×4 to 25×25) that represent different sections of the real-world office domain shown in Figure 8.

Next, the ability of the proposed collaboration algorithm to incorporate prior knowledge of target locations was evaluated. Figure 5(b) shows examples of the team's performance for a specific number of robots and targets as a function of the bias in the initial belief. As expected, the performance improves, i.e., the robots are able to localize targets faster, as more information about the locations of targets is made available.

Next, the effect of communication uncertainty on multirobot collaboration was measured. Figure 5(c) shows results of experiments as a function of varying CSR, where robot teams were asked to locate two targets. Though a low likelihood of successful communication hurts the team's performance, the target localization capability soon stabilizes and is then no longer sensitive to the CSR.

Table 1 shows results of an experiment where two robots localized two targets in a 15×15 map. The initial positions of robots and targets were randomly assigned in each trial. The POMDP-based approach is compared to a policy that randomly selects actions and assigns targets to robots, and a heuristic policy which selects targets and actions based on the grid cell with the largest belief. To simulate more realistic scenarios, prior belief was assigned to multiple areas in the map (including the target location). The proposed approach results in significantly better performance, with the robots traveling a much smaller distance to localize targets with high accuracy. Over extensive simulation experiments (and robot trials, see below) in different maps (3×3 to 25×25), using the hierarchical POMDP and collaboration strategy enables a team of robots to collaborate and localize target objects reliably and efficiently.

Figure 6 is a pictorial representation of the proposed approach for multirobot collaboration, with two robots repeatedly localizing two targets in a 20×20 map with obstacles. The robots had no prior knowledge of target locations. Intuitively, each robot should first look around its starting position and then explore other areas. Once

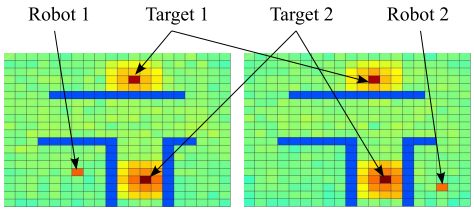
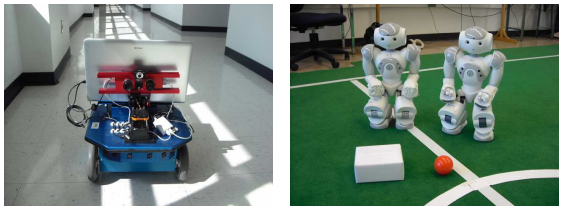


Figure 6: Simulated trials with 2 robots and 2 targets. Obstacles are shown in *blue*, targets in *dark red* and robot starting positions in *red*. Other cells show the number of times they were visited using colors ranging from *blue* to *red* along the visible spectrum.

a target is sighted, the robot should localize the target accurately. The actions taken by the robots are recorded over 100 simulated trials—each trial ends when the targets are located. In Figure 6, each grid cell's color changes from *blue* to *red* along the visible spectrum based on the relative number of visits by a robot—results are shown separately for each robot. Figure 6 shows that obstacles are avoided and grid cells near the targets are visited more often than other grid cells. In the absence of prior knowledge, there is no clear path from initial robot positions to the targets. The radius of the yellow region reflects the largest distance of effective observation. The two robots start searching for different targets in different trials, but there are hardly any trials when they both go for the same target. Similar performance is observed for different grid maps with different numbers of targets and robots.

4.3 Robot Experiments



(a) Erratic robot (b) Nao robots

Figure 7: Robot platforms used in experiments.

Experiments were conducted on a wheeled robot and a team of humanoid robots to test the proposed algorithms for reliable, efficient and autonomous sensing and collaboration.

4.3.1 *Experiments on Wheeled Robot*

The algorithms for POMDP-based visual sensing and processing were evaluated on the *Erratic* robot platform shown in Figure 7(a). This robot is equipped with stereo and monocular cameras, in addition to a laser range finder that can provide range information over an angular range of $\pm 135^o$ for a distance of $30m$.

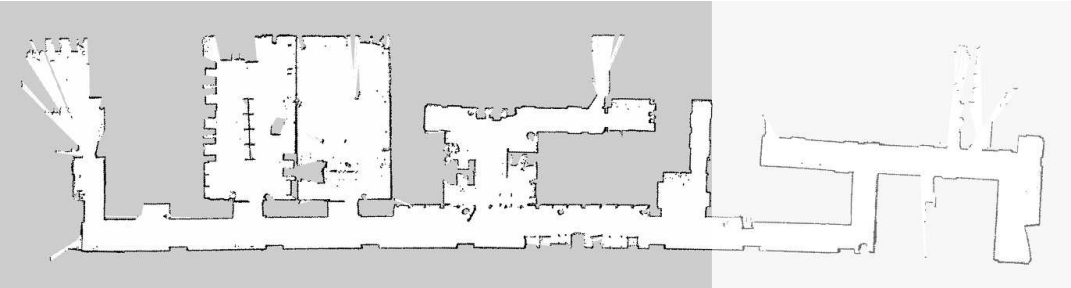


Figure 8: Occupancy-grid map of the third floor of the Computer Science department at Texas Tech University.

All processing is performed using an on-board dual-core 2.6*GHz* processor. The robot was used to conduct experiments in an indoor office domain—the corresponding occupancy-grid map was generated using a SLAM algorithm, as shown in Figure 8. This map corresponds to an entire floor of the CS department at Texas Tech University—it has three research labs, 13 faculty offices and a conference room. The experiments reported below were mostly conducted over the shaded portion of this map, which includes all research labs and nine rooms—this region was discretized into cells to form the grid map.

Given the complexity of the domain, objects were characterized using color distributions and the Binary Robust Independent Elementary Features (BRIEF) [5], i.e., local image gradients. Although BRIEF features are not inherently rotation and scale invariant, images of an object (captured during the learning phase) are automatically rotated and scaled to generate a set of images that encapsulate a range of rotations and scale changes—features extracted from these images are used to populate the object model. Figure 9 shows a screenshot of local feature detection and matching on a test object. Target objects consist of *boxes*, *cups*, *books* and other *robots* in complex (i.e., cluttered) backgrounds.

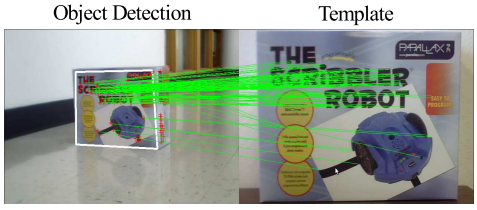


Figure 9: BRIEF descriptor.

To enable modular software architecture, the popular Robotics Operating System (ROS) [17] was installed on the robot and the algorithms described above were implemented on top of ROS. Figure 10 presents an overview of the implementation—it is a subset of the graph generated by the ROS command `<rxgraph>`. The planning algorithms are placed within the *vs_planner* node that is the control center of the system. It repeatedly accepts messages from the *vs_vision* node, which processes input images to provide the ID of any detected object, in addition to relative distance, bearing and detection probability, in the `<v_pack>` package. Belief updates occur under two situations: (1) robot arrives at a desired grid cell and processes some images of the scene—updates consider presence or absence of the target object; or (2) robot detects the target by processing images during navigation to a desired grid cell. The planner node sends the coordinates of any desired grid cell to the motion control node *move_base* and then waits for a response from the node, which can be one of: `arrived`, `canceled` or `not-arrived`. The `not-arrived` response is usually caused by a dynamic change in the environment, e.g., a door being closed, which makes an office unavailable to the robot. The node of the platform driver *erratic_base_driver* moves the robot platform based on the velocity command `cmd_vel`. The *hokuyo_node* provides the laser (range) readings to the motion control node and the localization node *amcl*. The *amcl* node computes the robot's `pos` (position and orientation) and the *map_server* revises the domain `map` continuously.

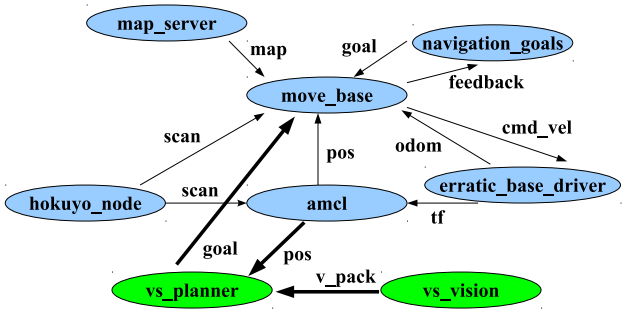


Figure 10: Node connections in ROS.

Over a sequence of 40 trials, the robot successfully identified the desired target objects. The robot only fails when a valid path to the target does not exist. The performance was significantly better than the heuristic search strategy used in Table 1. Videos of the robot's performance can be viewed online: `www.cs.ttu.edu/~smohan/Movies/Planning/visplan_aamas12.mp4`

4.3.2 *Experiments on Humanoid Robots*

The humanoid Nao robots [13] were used for multirobot collaboration experiments because multiple wheeled robots were not available. Since stable navigation is a challenge on humanoids, experiments were conducted in the robot soccer domain, where a team of robots play a competitive game of soccer on a $4m \times 6m$ indoor soccer field. This moderately constrained domain still captures all the collaboration challenges we seek to address. Each robot has a domain map and localizes based on domain landmarks such as goals and field corners (whose positions in the map are known) detected in input images. All computation is performed on-board the robots using a 500*MHz* processor.

Target objects include boxes and balls of different colors and shapes, as shown in Figure 7(b). Since objects are composed of homogeneous colors, gradient features cannot be used to learn object models. The robot has to process 30 frames/sec and computational resources are limited. Algorithms that detect object color and shape were hence used. Scene processing was modeled as a two-layered POMDP, with a POMDP that selects operators to apply on each salient region of interest in an image, and a POMDP that controls the selection of image ROIs for processing. The transfer of con-

trol between SP-POMDP and VS-POMDP occurred as described in Section 3.4. Obstacles were artificially introduced to force the robot to walk around to see the desired targets.

Experiments consisted of 25 trials, where a team of robots had to detect and localize one or more targets. The robots successfully localized all targets in all trials, and the performance was significantly better than the heuristic (i.e. greedy) policy for target and action selection, similar to the results reported in Table 1. The collaboration strategy was also robust to sudden changes in the team composition. For instance, when a robot was suddenly introduced in an existing team of robots, the new robot automatically (and quickly) chose to search for a relevant target using the communicated beliefs of teammates. Similarly, when a robot was removed from the team, the remaining robots automatically distributed the targets among themselves. These experiments show that the robots are able to use visual cues to reliably, efficiently and autonomously sense the environment and collaborate with teammates.

5. CONCLUSION

This paper described an approach for reliable, efficient and autonomous visual sensing and multirobot collaboration. A hierarchical POMDP with convolutional policies, adaptive observation functions, policy re-weighting and automatic belief propagation enables each robot to adapt sensing and information processing to that task at hand in dynamically changing environments. Each robot shares its beliefs with teammates and the multirobot collaboration algorithm enables the robot to merge its beliefs with the communicated beliefs of teammates. As a result, a team of mobile robots is able to collaborate robustly in simulation and in the real-world. The experiments reported in this paper assumed that robots have similar actuation capabilities. One direction of further investigation is to model and incorporate the sensing and actuation capabilities of heterogeneous robot platforms in the collaboration algorithm. Experiments will also be conducted using a larger number of physical robots and targets. Furthermore, the proposed hierarchy will be adapted to inputs from other sensors on mobile robot platforms. The ultimate goal is to enable reliable, efficient and autonomous multirobot (and human-robot) interaction in complex and dynamic real-world application domains.

Acknowledgment

This work was supported in part by the ONR Science of Autonomy award N00014-09-1-0658.

6. REFERENCES

[1] A. Atrash and J. Pineau. A Bayesian Method for Learning POMDP Observation Parameters for Robot Interaction Management Systems. In *The POMDP Practitioners Workshop*, 2010.

[2] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The Complexity of Decentralized Control of Markov Decision Processes. *Mathematics of Operations Research*, 27(4), November 2002.

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

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

[5] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary Robust Independent Elementary Features. In *European Conference on Computer Vision*, September 2010.

[6] G. Dissanayake, P. Newman, and S. Clark. A Solution to the Simultaneous Localization and Map Building (SLAM) Problem. *Transactions on Robotics and Automation*, 17(3):229–241, 2001.

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

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

[9] A. Krause, A. Singh, and C. Guestrin. Near-optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies. *JMLR*, 9:235–284, 2008.

[10] C. Kreucher, K. Kastella, and A. Hero. Sensor Management using An Active Sensing Approach. *IEEE Transactions on Signal Processing*, 85(3):607–624, 2005.

[11] J. Kwak, R. Yang, Z. Yin, M. Taylor, and M. Tambe. Teamwork and Coordination under Model Uncertainty in DEC-POMDPs. In *The AAAI Workshop on Interactive Decision Theory and Game Theory*, 2010.

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

[13] Nao. The Aldebaran Nao Robots, 2008. `http://www.aldebaran-robotics.com/`.

[14] S. C. Ong, S. W. Png, D. Hsu, and W. S. Lee. Planning Under Uncertainty for Robotic Tasks with Mixed Observability. *International Journal of Robotics Research*, 29(8):1053–1068, July 2010.

[15] L. Panait and S. Luke. Cooperative Multi-Agent Learning: The State of the Art. *JAAMAS*, 11(3):387–434, 2005.

[16] J. Pineau, M. Montemerlo, M. Pollack, N. Roy, and S. Thrun. Towards Robotic Assistants in Nursing Homes: Challenges and Results. In *RAS Special Issue on Socially Interactive Robots*, 2003.

[17] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[18] S. Rosenthal, M. Veloso, and A. Dey. Learning Accuracy and Availability of Humans who Help Mobile Robots. In *Twenty-Fifth Conference on Artificial Intelligence (AAAI)*, San Francisco, USA, August 2011.

[19] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa. Online Planning Algorithms for POMDPs. *JAIR*, 32:663–704, 2008.

[20] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, New Jersey, USA, 2003.

[21] P. E. Rybski, A. Larson, H. Veeraraghavan, M. LaPoint, and M. Gini. Communication strategies in Multi-Robot Search and Retrieval: Experiences with MinDART. In *Symposium on Distributed Autonomous Robotic Systems*, 2004.

[22] M. Sridharan, J. Wyatt, and R. Dearden. Planning to See: A Hierarchical Aprroach to Planning Visual Actions on a Robot using POMDPs. *Artificial Intelligence*, 174:704–725, 2010.

[23] S. Zhang, M. Sridharan, and X. Li. To Look or Not to Look: A Hierarchical Representation for Visual Planning on Mobile Robots. In *International Conference on Robotics and Automation*, 2011.