Learning to Gather Information via Imitation

Sanjiban Choudhury[1], Ashish Kapoor[2], Gireeja Ranade[2], Debadeepta Dey[2]

***Abstract*— The budgeted information gathering problem - where a robot with a fixed fuel budget is required to maximize the amount of information gathered from the world - appears in practice across a wide range of applications in autonomous exploration and inspection with mobile robots. Although there is an extensive amount of prior work investigating effective approximations of the problem, these methods do not address the fact that their performance is heavily dependent on distribution of objects in the world. In this paper, we attempt to address this issue by proposing a novel data-driven imitation learning framework.**

We present an efficient algorithm, EXPLORE, that trains a policy on the target distribution to imitate a *clairvoyant oracle* - an oracle that has full information about the world and computes non-myopic solutions to maximize information gathered. We validate the approach on a spectrum of results on a number of 2D and 3D exploration problems that demonstrates the ability of EXPLORE to adapt to different object distributions. Additionally, our analysis provides theoretical insight into the behavior of EXPLORE. Our approach paves the way forward for efficiently applying data-driven methods to the domain of information gathering.

I. INTRODUCTION

This paper considers the budgeted information gathering problem. Our aim is to maximally explore a world with a robot that has a budget on the total amount of movement due to battery constraints. This problem fundamentally recurs in mobile robot applications such as autonomous mapping of environments using ground and aerial robots [1], [9], monitoring of water bodies [12] and inspecting models for 3D reconstruction [13], [11].

The nature of "interesting" objects in an environment and their spatial distribution influence the optimal trajectory a robot might take to explore the environment. As a result, it is important that a robot learns about the type of environment it is exploring as it acquires more information and adapts it's exploration trajectories accordingly. This adaptation must be done online, and we provide such an algorithm in this paper.

To illustrate our point, consider two extreme examples of environments for a particular mapping problem, shown in Fig. 1. Consider a robot equipped with a sensor (RGBD camera) that needs to generate a map of an unknown environment. It is given a prior distribution about the geometry of the world, but has no other information. This geometry could include very diverse settings. First it can include a world where there is only one ladder, but the form of the ladder must be explored (Fig. 1a), which is a very dense setting. Second, it could include a sparse setting with spatially distributed objects, such as a construction site (Fig. 1b).

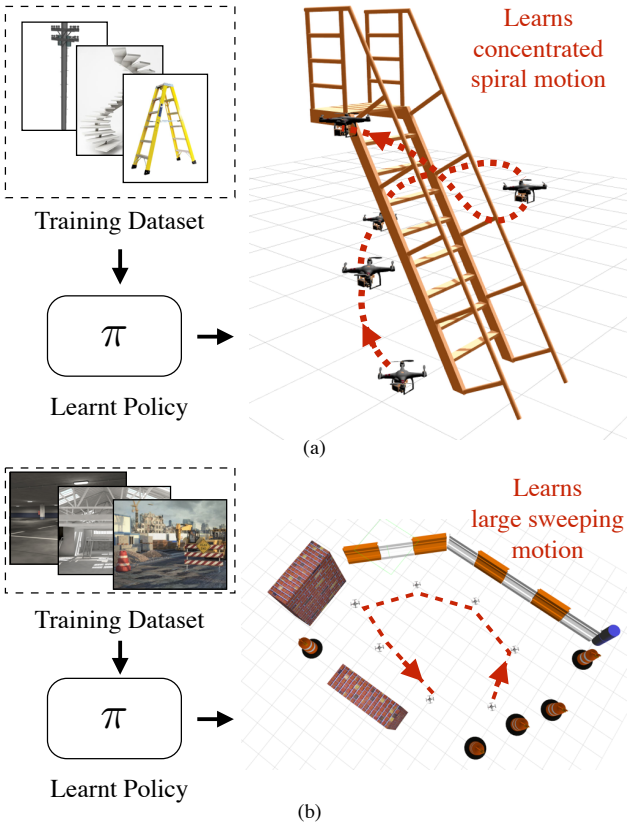


Fig. 1: EXPLORE trains a policy to gather information. The effectiveness of a policy to gather information depends on the distribution of worlds. (a) When the distribution corresponds to a scene containing ladders, the learnt policy executes a helical motion around parts of the ladder already observed as it's unlikely that there is information elsewhere. (b) When the distribution corresponds to a scene from a construction site, the learnt policy executes a large sweeping motion as information is likely to be dispersed.

The important task for the robot is to now try to infer which type of environment it is in based on the history of measurements, and thus plan an efficient trajectory. At every time step, the robot visits a sensing location and receives a sensor measurement (e.g. depth image) that has some amount of information utility (e.g. surface coverage of objects with point cloud). As opposed to naive lawnmower-coverage patterns, it will be more efficient if the robot could use a policy that maps the history of locations visited and measurements received to decide which location to visit next such that it maximizes the amount of information gathered in the finite amount of battery time available.

The ability of such a learnt policy to gather information efficiently depends on the prior distribution of worlds in

This project was formulated and conducted during an internship by Sanjiban Choudhury at Microsoft Research.

[1]Sanjiban Choudhury is with The Robotics Institute, Carnegie Mellon University, USA sanjiban@cmu.edu

[2]Ashish Kapoor, Gireeja Ranade and Debadeepta Dey are with Microsoft Research, Redmond, USA {akapoor, giranade, dedey}@microsoft.com

which the robot has been shown how to navigate optimally. Fig. 1a shows an efficient learnt policy for inspecting a ladder, which executes a helical motion around parts of the ladder already observed to efficiently uncover new parts without searching naively. This is efficient because given the prior distribution the robot learns that information is likely to be geometrically concentrated in a particular volume given it's initial observations of parts of the ladder. Similarly Fig. 1b shows an effective policy for exploring construction sites by executing large sweeping motions. Here again the robot learns from prior experience that wide, sweeping motions are efficient since it has learnt that information is likely to be dispersed in such scenarios.

Thus our requirements for an efficient information-gathering policy can be distilled to two points:

1) *Reasoning about posterior distribution over world maps:* The robot should use the history of movements and measurements to infer a posterior distribution of worlds. This can be used to infer locations that are likely to contain information and efficiently plan a trajectory. However the space of world maps is very large, and it is intractable to compute this posterior online.
2) *Reasoning about non-myopic value of information:* Even if the robot is able to compute the posterior and hence the value of information at a location, it has to be cognizant of the travel cost to get to that location. It needs to exhibit non-myopic behavior to achieve a trade-off that maximizes the overall information gathered. Performing this computationally expensive planning at every step is prohibitively expensive.

Even though it's natural to think of this problem setting as a POMDP, we frame this problem as a novel data-driven imitation learning problem [26]. We propose an algorithm EXPLORE (Exploration by Learning to Imitate an Oracle) that trains a policy on a dataset of worlds by imitating a *clairvoyant oracle*. During the training process, the oracle has full information about the world map (and is hence clairvoyant) and plans movements to maximize information. The policy is then trained to imitate these movements as best as it can using partial information from the current history of movements and measurements. As a result of our novel formulation, we are able to sidestep a number of challenging issues in POMDPs like explicitly computing posterior distribution over worlds and planning in belief space.

Our contributions are as follows

1) We map the budgeted information gathering problem to a POMDP and present an approach to solve it using imitation learning.
2) We present an approach to train a policy on the non-stationary distribution of event traces induced by the policy itself. We show that this implicitly results in the policy operating on the posterior distribution of world maps.
3) We show that by imitating an oracle that has access to the world map and thus can plan optimal routes, the policy is able to learn non-myopic behavior. Since the oracle is executed only during train time, the computational burden does not affect online performance.

The remainder of this paper is organized as follows. Section II presents the formal problem, while Section III contains relevant work. The algorithm is presented in Section IV and Section VI presents experimental results. Finally we conclude in Section VII with discussions and thoughts on future work.

II. PROBLEM STATEMENT

A. Notation

Let $\mathcal{V}$ be a set of nodes corresponding to all sensing locations. The robot starts at node v_s. Let $\xi = (v_1, v_2, \ldots, v_p)$ be a sequence of nodes (a path) such that $v_1 = v_s$. Let Ξ be the set of all such paths. Let $\phi \in \mathcal{M}$ be the world map. Let $y \in \mathcal{Y}$ be a measurement received by the robot. Let $\mathcal{H} : \mathcal{V} \times \mathcal{M} \to \mathcal{Y}$ be a measurement function. When the robot is at node v in a world map ϕ, the measurement y received by the robot is $y = \mathcal{H}(v, \phi)$. Let $\mathcal{F} : 2^{\mathcal{V}} \times \mathcal{M} \to \mathbb{R}_{\geq 0}$ be a utility function mapping a subet of nodes and a world map to a utility. For a path ξ and a world map ϕ, $\mathcal{F}(\xi, \phi)$ assigns a utility to executing the path on the world. Note that $\mathcal{F}$ is a set function. Given a node $v \in \mathcal{V}$, a set of nodes $V \subseteq \mathcal{V}$ and world ϕ, the discrete derivative of the utility function $\mathcal{F}$ is $\Delta_{\mathcal{F}}(v \mid V, \phi) = \mathcal{F}(V \cup \{v\}, \phi) - \mathcal{F}(V, \phi)$ Let $\mathcal{T} : \Xi \times \mathcal{M} \to \mathbb{R}_{\geq 0}$ be a travel cost function. For a path ξ and a world map ϕ, $\mathcal{T}(\xi, \phi)$ assigns a travel cost to executing the path on the world.

B. Problem Formulation

We first define the problem setting when the world map is fully known.

Problem 1 (Fully Observable World Map; Constrained Travel Cost)**.** *Given a world map ϕ, a travel cost budget B and a time horizon T, find a path ξ that maximizes utility subject to travel cost and cardinality constraints.*

$$\begin{aligned} \underset{\xi \in \Xi}{\arg\max} \quad & \mathcal{F}(\xi, \phi) \\ \textit{s.t.} \quad & \mathcal{T}(\xi, \phi) \leq B \\ & |\xi| \leq T + 1 \end{aligned} \tag{1}$$

Now, consider the setting where the world map ϕ is unknown. Given a prior distribution $P(\phi)$, it can be inferred only via the measurements y_i received as the robot visits nodes v_i. Hence, instead of solving for a fixed path, we compute a policy that maps history of measurements received and nodes visited to decide which node to visit.

Problem 2 (Partially Observable World Map; Constrained Travel Cost)**.** *Given a distribution of world maps, $P(\phi)$, a travel cost budget B and a time horizon T, find a policy that at time t, maps the history of nodes visited $\{v_i\}_{i=1}^{t-1}$ and measurements received $\{y_i\}_{i=1}^{t-1}$ to compute node v_t to visit at time t, such that the expected utility is maximized subject to travel cost and cardinality constraints.*

C. Mapping to MDP and POMDP

1) Mapping fully observable problems to MDP: The Markov Decision Process (MDP) is a tuple $(\mathcal{S}, \mathcal{M}, \mathcal{A}, \Omega, R, T)$ defined up to a fixed finite horizon T. It is defined over an augmented state space comprising of the ego-motion state space $\mathcal{S}$ (which we will refer to as simply the state space) and the space of world maps $\mathcal{M}$.

Let $s_t \in \mathcal{S}$ be the state of the robot at time t. It is defined as the set of nodes visited by the robot till time t, $s_t = (v_1, v_2, \ldots, v_t)$. This implies the dimension of the state space is exponentially large in the space of nodes, $\mathcal{S} = 2^{|\mathcal{V}|}$. The initial state $s_1 = v_s$ is the start node. Let $a_t \in \mathcal{A}$ be the action executed by the robot at time t. It is defined as the node visited at time $t+1$, $a_t = v_{t+1}$. The set of all actions is defined as $\mathcal{A} = \mathcal{V}$. Given a world map ϕ, when the robot is at state s the utility of executing an action a is $\mathcal{F}(s \cup a, \phi)$. Let $\mathcal{A}_{\text{feas}}(s, \phi) \subset \mathcal{A}$ be the set of feasible actions that the robot can execute when in state s in a world map ϕ. This is defined as follows

$$\mathcal{A}_{\text{feas}}(s, \phi) = \{a \mid a \in \mathcal{A}, \mathcal{T}(s \cup a, \phi) \leq B\} \tag{2}$$

Let $\Omega(s, a, s') = P(s'|s, a)$ be the state transition function. In our setting, this is the deterministic function $s' = s \cup a$. Let $R(s, \phi, a) \in [0, 1]$ be the one step reward function. It is defined as the normalized marginal gain of the utility function, $R(s, \phi, a) = \frac{\Delta_{\mathcal{F}}(a|s,\phi)}{\mathcal{F}(\mathcal{A},\phi)}$. Let $\pi(s, \phi) \in \Pi$ be a policy that maps state s and world map ϕ to a feasible action $a \in \mathcal{A}_{\text{feas}}(s, \phi)$. The value of executing a policy π for t time steps on a world ϕ starting at state s.

$$V_t^{\pi}(s, \phi) = \sum_{i=1}^{t} \mathbb{E}_{s_i \sim P(s'|s,\pi,i)} \left[R(s_i, \phi, \pi(s_i, \phi))\right] \tag{3}$$

where $P(s'|s, \pi, i)$ is the distribution of states at time i starting from s and following policy π. The state action value $Q_t^{\pi}(s, \phi, a)$ is the value of executing action a in state s in world ϕ and then following the policy π for $t-1$ timesteps

$$Q_t^{\pi}(s, \phi, a) = R(s, \phi, a) + \mathbb{E}_{s' \sim P(s'|s,a)} \left[V_{t-1}^{\pi}(s', \phi)\right] \tag{4}$$

The value of a policy π for T steps on a distribution of worlds $P(\phi)$ and starting states $P(s)$

$$J(\pi) = \mathbb{E}_{s \sim P(s), \phi \sim P(\phi)} \left[V_T^{\pi}(s, \phi)\right] \tag{5}$$

The optimal MDP policy is $\pi_{\text{MDP}} = \arg\max_{\pi \in \Pi} J(\pi)$.

2) Mapping partially observable problems to POMDP: The Partially Observable Markov Decision Process (POMDP) is a tuple $(\mathcal{S}, \mathcal{M}, \mathcal{A}, \Omega, R, \mathcal{O}, Z, T)$. The first component of the augmented state space, the ego motion state space $\mathcal{S}$, is fully observable. The second component, the space of world maps $\mathcal{M}$, is partially observable through observations received.

Let $o_t \in \mathcal{O}$ be the observation at time step t. This is defined as the measurement received by the robot $o_t = y_t$. Let $Z(s, a, \phi, o) = P(o|s, a, \phi)$ be the probability of receiving an observation o given the robot is at state s and executes action a. In our setting, this is the deterministic function $o = \mathcal{H}(a, \phi)$.

Let the belief at time t, ψ_t, be the history of state, action, observation tuple received so far, i.e. $\{(s_i, a_i, o_i)\}_{i=1}^{t}$. Note that this differs from the conventional use of the word belief which would usually imply a distribution. However, we use belief here to refer to the history of state, action, observations conditioned on which one can infer the posterior distribution of world maps $P(\phi)$. Let the belief transition function be $P(\psi'|\psi, a)$. Let $\tilde{\pi}(s, \psi) \in \tilde{\Pi}$ be a policy that maps state s and belief ψ to a feasible action $a \in \mathcal{A}_{\text{feas}}(s, \phi)$. The value of executing a policy $\tilde{\pi}$ for t time steps starting at state s and belief ψ is

$$\tilde{V}_t^{\tilde{\pi}}(s, \psi) = \sum_{i=1}^{t} \mathbb{E}_{\substack{\psi_i \sim P(\psi'|\psi,\tilde{\pi},i),\\ \phi \sim P(\phi|\psi_i)\\ s_i \sim P(s'|s,\tilde{\pi},i)}} \left[R(s_i, \phi, \tilde{\pi}(s_i, \psi_i))\right] \tag{6}$$

where $P(\psi'|\psi, \tilde{\pi}, i)$ is the distribution of beliefs at time i starting from ψ and following policy $\tilde{\pi}$. $P(\phi|\psi_i)$ is the posterior distribution on worlds given the belief ψ_i. Similarly the action value function $\tilde{Q}_t^{\tilde{\pi}}$ is defined as

$$\begin{aligned}\tilde{Q}_t^{\tilde{\pi}}(s, \psi, a) =& \mathbb{E}_{\phi \sim P(\phi|\psi)} \left[R(s, \phi, a)\right] + \\ & \mathbb{E}_{\psi' \sim P(\psi'|\psi,a), s' \sim P(s'|s,a)} \left[\tilde{V}_{t-1}^{\tilde{\pi}}(s', \psi')\right]\end{aligned} \tag{7}$$

The optimal POMDP policy can be expressed as

$$\pi^* = \arg\max_{\tilde{\pi} \in \tilde{\Pi}} \mathbb{E}_{\substack{t \sim U(1:T),\\ s \sim P(s|\tilde{\pi},t),\\ \psi \sim P(\psi|\tilde{\pi},t)}} \left[\tilde{Q}_{T-t+1}^{\tilde{\pi}}(s, \psi, \tilde{\pi}(s, \psi))\right] \tag{8}$$

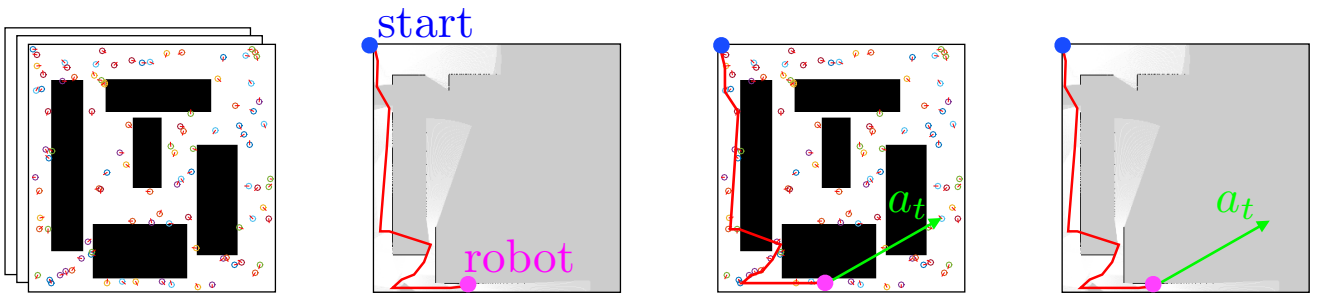
where $U(1:T)$ is a uniform distribution over the discrete interval $\{1, 2, \ldots, T\}$, $P(s \mid \tilde{\pi}, t)$ is the distribution of states following policy $\tilde{\pi}$ for t steps, $P(\psi \mid \tilde{\pi}, t)$ is the distribution of belief following policy $\tilde{\pi}$ for t steps. The value of a policy $\tilde{\pi} \in \tilde{\Pi}$ for T steps on a distribution of worlds $P(\phi)$, starting states $P(s)$ and starting belief $P(\psi)$

$$J(\tilde{\pi}) = \mathbb{E}_{s \sim P(s), \psi \sim P(\psi)} \left[V_T^{\tilde{\pi}}(s, \psi)\right] \tag{9}$$

where the posterior world distribution $P(\phi \mid \psi)$ uses $P(\phi)$ as prior.

III. Related Work

Problem 1 is a submodular function optimization (due to nature of $\mathcal{F}$) subject to routing constraints (due to $\mathcal{T}$). In absence of this constraint, there is a large body of work on near optimality of greedy strategies by Krause et al.[19], [21], [20] - however naive greedy approaches can perform arbitrarily poorly. Chekuri and Pal [2] propose a quasi-polynomial time recursive greedy approach to solving this problem. Singh et al.[30] show how to scale up the approach to multiple robots. However, these methods are slow in practice. Iyer and Bilmes [14] solve a related problem of submodular maximization subject to submodular knapsack constraints using an iterative greedy approach. This inspires Zhang and Vorobeychik [35] to propose an elegant generalization of the cost benefit algorithm (GCB) which we use as an oracle in this paper. Yu et al.[34] frame the problem as a correlated orienteering problem and propose a mixed integer based approach - however only correlations between neighboring nodes are considered. Hollinger and Sukhatme [12] use sampling based approaches which require a lot of evaluations of the utility function in practice.



Sample world ϕ from database $P(\phi)$ | Rollin with policy π_{mix} to get state s_t, belief ψ_t | Query oracle π_{OR} with a_t to get $Q^{\pi_{\text{OR}}}(s_t, \phi, a_t)$ | Update learner $\hat{\pi}$ to map (s_t, ψ_t, a_t) to $Q^{\pi_{\text{OR}}}$

Fig. 2: Overview of EXPLORE which iteratively trains a learner $\hat{\pi}$ to imitate a clairvoyant oracle π_{OR}.

Problem 2 in the absence of the travel cost constraint can be efficiently solved using the framework of adaptive submodularity developed by Golovin et al.[6], [7] as shown by Javdani et al.[16], [15] and Chen et al.[4], [3]. Hollinger et al.[10], [11] propose a heuristic based approach to select a subset of informative nodes and perform minimum cost tours. Singh et al.[31] replan every step using a non-adaptive information path planning algorithm. Such methods suffer when the adaptivity gap is large [10]. Inspired by adaptive TSP approaches by Gupta et al.[8], Lim et al.[25], [24] propose recursive coverage algorithms to learn policy trees. However such methods cannot scale well to large state and observation spaces. Heng et al.[9] make a modular approximation of the objective function. Isler et al.[13] survey a broad number of myopic information gain based heuristics that work well in practice but have no formal guarantees.

Online POMDP planning also has a large body of work ([17], [28], [22]. Although there exists fast solvers such as POMCP (Silver and Veness [29]) and DESPOT (Somani et al.[32]), the space of world maps is too large for online planning.

IV. APPROACH

A. Overview

Fig. 2 shows an overview of our approach. The central idea is as follows - we train a policy to imitate an algorithm that has access to the world map at train time. The policy $\hat{\pi}(s, \psi)$ maps features extracted from state s and belief ψ to an action a. The algorithm that is being imitated has access to the corresponding world map ϕ.

B. Imitation Learning

We now formally define imitation learning as applied to our setting. Let $\tilde{\pi} \in \tilde{\Pi}$ be a policy defined on a pair of state and belief (s, ψ). Let the distribution of states and beliefs induced by roll-in [1] with policy $\tilde{\pi}$ be $P(s|\tilde{\pi})$ and $P(\psi|\tilde{\pi})$. Let $\mathcal{L}(s, \psi, \tilde{\pi})$ be the loss of a policy $\tilde{\pi}$ when executed on state s and belief ψ. This loss function implicitly captures how well policy $\tilde{\pi}$ imitates a reference policy (such as an oracle algorithm). Our goal is to find a policy $\hat{\pi}$ which minimizes the observed loss under its own induced distribution of state and beliefs.

$$\hat{\pi} = \arg\min_{\tilde{\pi} \in \tilde{\Pi}} \mathbb{E}_{s \sim P(s|\tilde{\pi}), \psi \sim P(\psi|\tilde{\pi})} \left[\mathcal{L}(s, \psi, \tilde{\pi}) \right] \tag{10}$$

This is a non-i.i.d supervised learning problem. Ross and Bagnell [26] show how such problems can be reduced to no-regret online learning using dataset aggregation for one step loss (DAGGER) and later [27] extend the approach to where the loss is the cost-to-go of an oracle reference policy by aggregating values to imitate (AGGREVATE).0

C. Solving POMDP via Imitation of a Clairvoyant Oracle

When (8) is compared to the imitation learning framework in (10), we see that in addition to the induced state belief distributions, the loss function analogue $\mathcal{L}(s, \psi, \tilde{\pi})$ is $\tilde{Q}^{\tilde{\pi}}_{T-t+1}(s, \psi, \tilde{\pi}(s, \psi))$. This implies rolling out with policy $\tilde{\pi}$. For poor policies $\tilde{\pi}$, the action value estimate $\tilde{Q}^{\tilde{\pi}}_{T-t+1}$ would be very different from optimal values $\tilde{Q}^{\pi^*}_{T-t+1}$.

In our approach, we alleviate this problem by defining a surrogate value functions to imitate - the cumulative reward gathered by a clairvoyant oracle.

Definition 1 (Clairvoyant Oracle)**.** *Given a distribution of world map $P(\phi)$, a clairvoyant oracle $\pi_{\text{OR}}(s, \phi)$ is a policy that maps state s and world map ϕ to a feasible action $a \in \mathcal{A}_{\text{feas}}(s, \phi)$ such that it approximates the optimal MDP policy, $\pi_{\text{OR}} \approx \pi_{\text{MDP}} = \arg\max_{\pi \in \Pi} J(\pi)$.*

The term *clairvoyant* is used because the oracle has full access to the world map ϕ at train time. The oracle can be used to compute state action value as follows

$$Q^{\pi_{\text{OR}}}_t(s, \phi, a) = R(s, \phi, a) + \mathbb{E}_{s' \sim P(s'|s,a)} \left[V^{\pi_{\text{OR}}}_{t-1}(s', \phi) \right] \tag{11}$$

Our approach is to imitate the oracle during training. This implies that we train a policy $\hat{\pi}$ by solving the following optimization problem

$$\hat{\pi} = \arg\max_{\tilde{\pi} \in \tilde{\Pi}} \mathbb{E}_{\substack{t \sim U(1:T), \\ s \sim P(s|\tilde{\pi}, t), \\ \phi \sim P(\phi), \\ \psi \sim P(\psi|\phi, \tilde{\pi}, t)}} \left[Q^{\pi_{\text{OR}}}_{T-t+1}(s, \phi, \tilde{\pi}(s, \psi)) \right] \tag{12}$$

D. Algorithm

Alg. 1 describes the EXPLORE algorithm. It takes as input the training distribution $P(\phi)$, the action set $\mathcal{A}$, time

[1] *Roll-in* be the process of executing a policy $\tilde{\pi}$ from the start to a certain time horizon. Similarly *roll-out* is the process of executing a policy from the current state and belief till the end.

Algorithm 1 EXPLORE: Imitation Learning of Oracle

1: Initialize $\mathcal{D} \leftarrow \emptyset$, $\hat{\pi}_1$ to any policy in $\tilde{\Pi}$
2: **for** $i = 1$ **to** N **do**
3: Initialize sub dataset $\mathcal{D}_i \leftarrow \emptyset$
4: Let roll in policy be $\pi_{\text{mix}} = \beta_i \pi_{\text{OR}} + (1 - \beta_i)\hat{\pi}_i$
5: Collect m data points as follows:
6: **for** $j = 1$ **to** m **do**
7: Sample world map ϕ from dataset $P(\phi)$
8: Sample uniformly $t \in \{1, 2, \ldots, T\}$
9: Assign initial state $s_1 = v_s$
10: Execute π_{mix} up to time $t-1$ to reach (s_t, ψ_t)
11: Execute any action $a_t \in \mathcal{A}_{\text{feas}}(s_t, \phi)$
12: Execute oracle π_{OR} from $t+1$ to T on ϕ
13: Collect value to go $Q_i^{\pi_{\text{OR}}} = Q_{T-t+1}^{\pi_{\text{OR}}}(s_t, \phi, a_t)$
14: $\mathcal{D}_i \leftarrow \mathcal{D}_i \cup \{s_t, \psi_t, a_t, t, Q_i^{\pi_{\text{OR}}}\}$
15: Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{D}_i$
16: Train cost-sensitive classifier $\hat{\pi}_{i+1}$ on $\mathcal{D}$
17: *(Alternately: use any online learner $\hat{\pi}_{i+1}$ on $\mathcal{D}_i$)*
18: **Return** best $\hat{\pi}_i$ on validation

horizon T, the oracle policy π_{OR}, number of EXPLORE iterations N and number of episodes per iteration m. The algorithm iteratively trains a sequence of learnt policies $(\hat{\pi}_1, \hat{\pi}_2, \ldots, \hat{\pi}_N)$ by aggregating data for an online cost-sensitive classification problem and returns the best policy $\hat{\pi}_i$ on validation.

$\hat{\pi}_1$ is initialized as a random policy (Line 1). At iteration i, the policy that is used to roll-in is a mixture policy of learnt policy $\hat{\pi}_i$ and the oracle policy π_{OR} (Line 4) using mixture parameter β_i. A set of m cost-sensitive classification datapoints are captured as follows: a world ϕ is sampled (Line 7). The π_{mix} is used to roll-in to a random time from an initial state to reach (s_t, ψ_t) (Lines 8–10). An exploratory action is selected (Line 11). The clairvoyant oracle is given full access to ϕ and asked to roll-out and provide an action value $Q^{\pi_{\text{OR}}}$ (Lines 12–13). $\{s_t, \psi_t, a_t, t, Q_i^{\pi_{\text{OR}}}\}$ is added to a dataset $\mathcal{D}_i$ of cost-sensitive classification problem and the process is repeated (Line 14). $\mathcal{D}_i$ is appended to the original dataset $\mathcal{D}$ and used to train an updated learner $\hat{\pi}_{i+1}$ (Lines 15–17).

V. ANALYSIS

Following the analysis style of AGGREVATE [27], we first introduce a *hallucinating oracle*.

Definition 2 (Hallucinating Oracle)**.** *Given a prior distribution of world map $P(\phi)$ and roll-outs by a policy $\tilde{\pi}$, a hallucinating oracle $\tilde{\pi}_{\text{OR}}$ computes the instantaneous posterior distribution over world maps and takes the action with the highest expected state-action value as computed by the clairvoyant oracle.*

$$\tilde{\pi}_{\text{OR}}(s, \psi) = \arg\max_{a \in \mathcal{A}} \mathbb{E}_{\phi' \sim P(\phi'|\psi, \tilde{\pi}, t)} \left[Q_{T-t+1}^{\pi_{\text{OR}}}(s, \phi', a) \right] \tag{13}$$

The hallucinating oracle is an effective policy for information gathering as alluded to by Koval et al.[18] and we now show that we implicitly imitate it.

Lemma 1. *The policy optimization rule (12) is equivalent to*

$$\hat{\pi} = \arg\max_{\tilde{\pi} \in \tilde{\Pi}} \mathbb{E}_{\substack{t \sim U(1:T), \\ s \sim P(s|\tilde{\pi}, t), \\ \phi \sim P(\phi), \\ \psi \sim P(\psi|\phi, \tilde{\pi}, t)}} \left[Q_{T-t+1}^{\tilde{\pi}_{\text{OR}}}(s, \phi, \tilde{\pi}(s, \psi)) \right]$$

Consequently our learnt policy has the following guarantee

Theorem 1. *N iterations of* EXPLORE*, collecting m regression examples per iteration guarantees that with probability at least $1 - \delta$*

$$J(\hat{\pi}) \geq J(\tilde{\pi}_{\text{OR}}) - 2\sqrt{|\mathcal{A}|} T \sqrt{\varepsilon_{\text{class}} + \varepsilon_{\text{reg}} + O\left(\sqrt{\log\left((1/\delta)/Nm\right)}\right)} - O\left(\frac{T^2 \log T}{\alpha N}\right)$$

where ε_{reg} is the empirical average online learning regret on the training regression examples collected over the iterations and $\varepsilon_{\text{class}}$ is the empirical regression regret of the best regressor in the policy class.

For both proofs, refer to [5]. We now analyze the computational complexity during training and testing phase. From Alg. 1, the training complexity can be derived to be

$$N\left[m\left(0.5T\text{O}(|\mathcal{V}|) + \text{O}(\text{oracle})\right) + \text{O}(\text{learn})\right] \tag{14}$$

We use an efficient greedy oracle (GCB [35]) such that $\text{O}(\text{oracle}) = 0.5T\text{O}(|\mathcal{V}|)$ but in general can be exponential in T. The test-time complexity is $\text{O}(|\mathcal{V}|)$.

VI. EXPERIMENTAL RESULTS

A. Implementation Details

Our implementation is open sourced for both MATLAB and C++ (https://bitbucket.org/sanjiban/matlab_learning_info_gain) .

1) Problem Details: The utility function $\mathcal{F}$ is selected to be a fractional coverage function (similar to [13]) which is defined as follows. The world map ϕ is represented as a voxel grid representing the surface of a 3D model. The sensor measurement $\mathcal{H}(v, \phi)$ at node v is obtained by raycasting on this 3D model. A voxel of the model is said to be 'covered' by a measurement received at a node if a point lies in that voxel. The coverage of a path ξ is the fraction of covered voxels by the union of measurements received when visiting each node of the path. The set of nodes $\mathcal{V}$ is created by uniformly randomly sampling poses. The travel cost function $\mathcal{T}$ is chosen to be the euclidean distance. The values of total time step T and travel budget B varies with problem instances and are specified along with the results.

2) Learning Details: We now describe how the tuple (s, a, ψ) is mapped to a vector of features $f = \begin{bmatrix} f_{\text{IG}}^T & f_{\text{mot}}^T \end{bmatrix}^T$. The belief ψ is represented by an occupancy map. Given a candidate action a, a sensor model can be used to raycast from corresponding node v on the occupancy map and compute various information gain metrics as described in [13] denoted by f_{IG}. Given a state s (corresponds to a path ξ) and action a (corresponds to a node v), the vector f_{mot} encodes the relative rotation and translation required to visit v from ξ. We observed that this feature representation

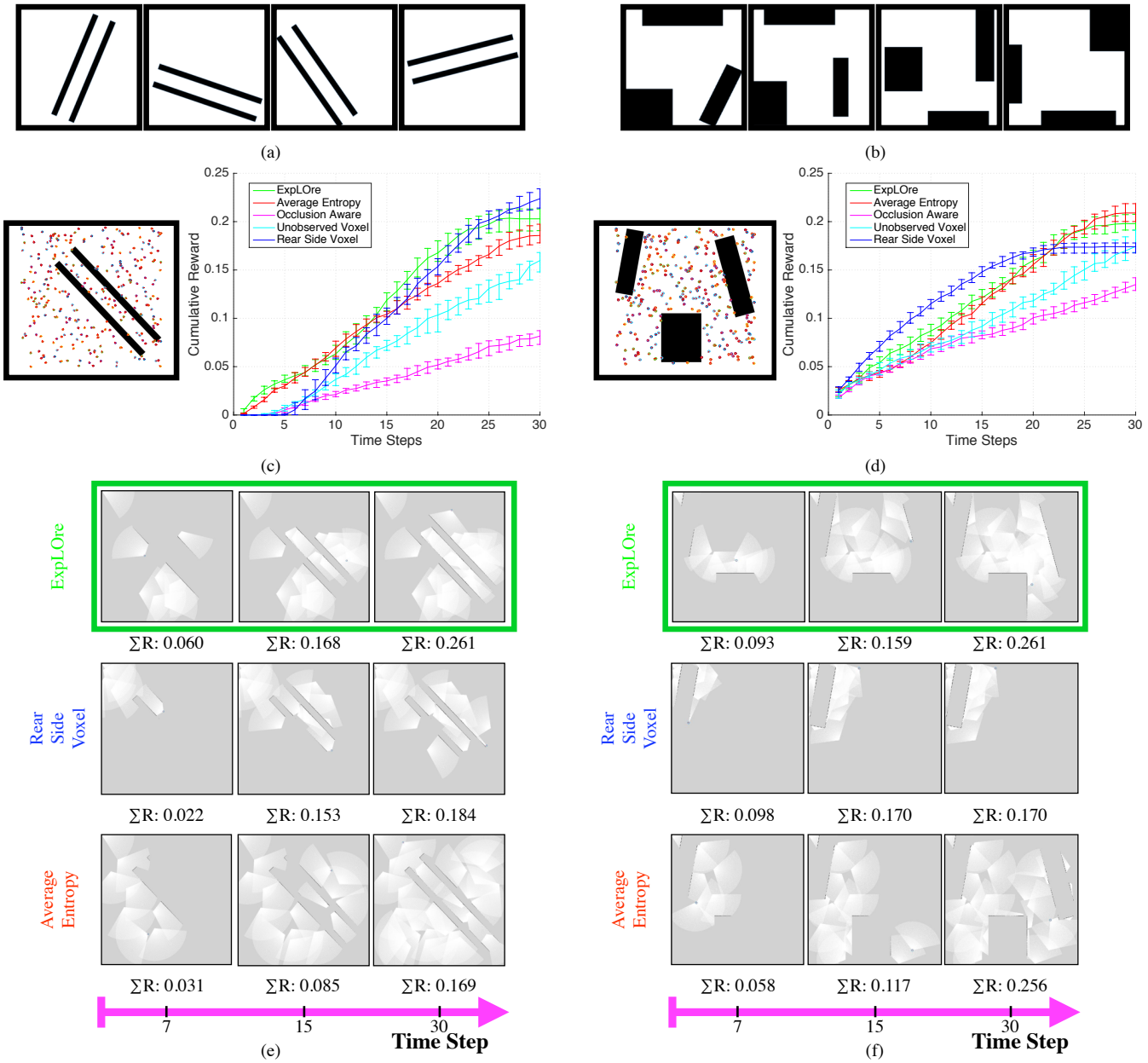


Fig. 3: Comparison of ExpLOre with baseline heuristics on a 2D exploration problem on 2 different datasets - dataset 1 (concentrated information) and dataset 2 (distributed information). The problem details are: $T = 30, B = 2500, |\mathcal{A}| = 300$. Sample world maps from (a) dataset 1 and (b) dataset 2. Training dataset is created with 10 world maps, each with 10 random node sets to create a dataset size of 100. Test results on 1 representative world map with 100 random node sets are shown for (c) dataset 1 and (d) dataset 2. A sample test instance is shown along with a plot of cumulative reward with time steps for ExpLOre and other baseline heuristics. The error bars show 95% confidence intervals. Snapshots of execution of ExpLOre, Rear Side Voxel and Average Entropy are shown for (e) dataset 1 and (f) dataset 2. The snapshots show the evidence grid at time steps $7, 15$ and 30.

has powerful generalization capabilities given the 'local' nature of information gain metrics. Hence, if a completely different world is encountered at test time, as long as there is sufficient overlap between artifacts such as walls, corners or object sizes, the feature distributions will match. We use random forest [23] as a function approximator. We use the generalized cost benefit algorithm (GCB) [35] as the oracle owing to its small run times. Details are specified in Table. I.

3) Baseline: For baseline policies, we compare to the class of information gain heuristics discussed in [13]. The heuristics are remarkably effective, however, their performance depends on the distribution of objections in a world map. As ExpLOre uses these heuristic values as part of its feature vector, it will implicitly learn a data driven trade-off between them.

TABLE I: Learning Details

Problem	Train Dataset m	Test Dataset	ExpLOre Iterations N	Feature Dimension $\lvert f \rvert$
2D	100	100	100	16
3D	100	10	10	16

B. 2D Exploration

We create a set of 2D exploration problems to gain a better understanding of the behavior of the ExpLOre and

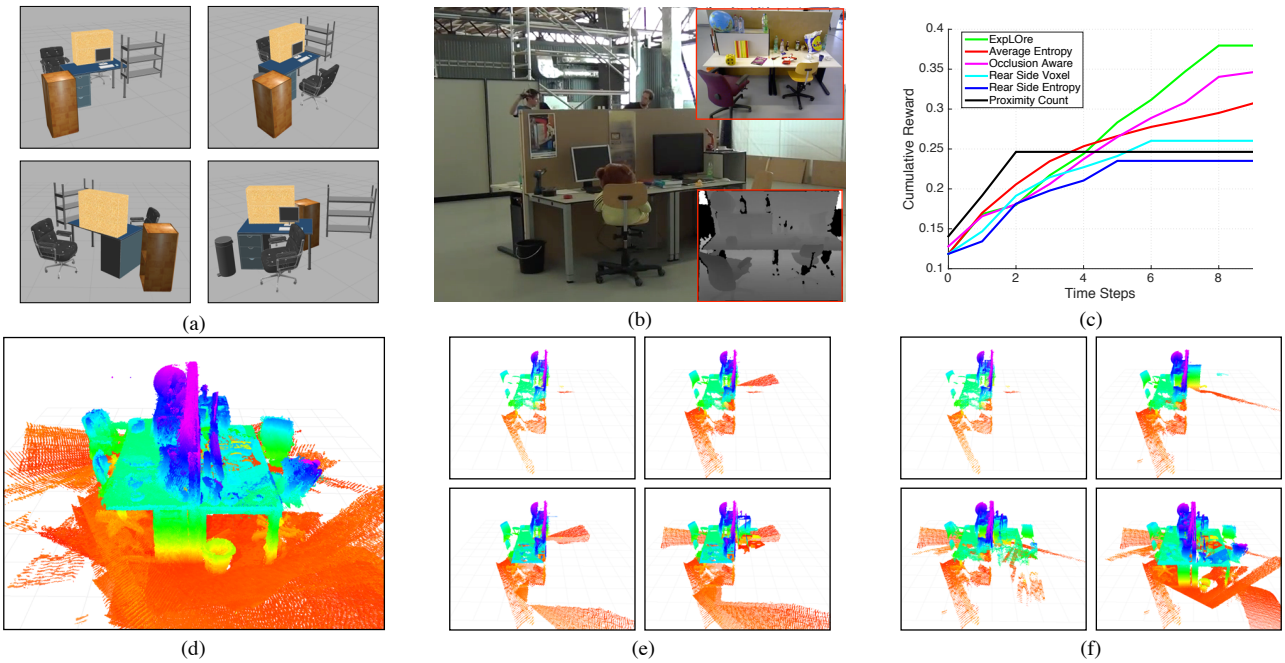


Fig. 4: Comparison of ExpLOre with baseline heuristics on a 3D exploration problem where training is done on simulated world maps and testing is done on a real dataset of an office workspace. The problem details are: $T = 10$, $B = 12$, $|\mathcal{A}| = 50$. (a) Samples from 100 simulated worlds resembling an office workspace created in Gazebo. (b) Real dataset collected by [33] using a RGBD camera. (c) Plot of cumulative reward with time steps for ExpLOre and baseline heuristics on the real dataset. (d) The 3D model of the real office workspace formed by cumulating measurements from all poses. (e) Snapshots of execution of Occlusion Aware heuristic at time steps $1, 3, 5, 9$. (f) Snapshots of execution of ExpLOre heuristic at time steps $1, 3, 5, 9$.

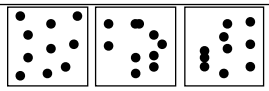
Dataset	Sample World Maps	Details	ExpLOre	Average Entropy	Occlusion Aware	Unobserved Voxels	Rear Side Voxels	Rear Side Entropy
Poisson Forest of Circular Discs (2D)		Train Size (m): 100 Test Size:10 ExpLOre Iter:10 T:30, B:2500, A:300	**(0.54, 0.59)**	(0.54, 0.59)	(0.42, 0.46)	(0.34, 0.41)	(0/37, 0.43)	(0.39, 0.44)
Tabular World of Rectilinear Blocks (2D)		Train Size (m): 100 Test Size:10 ExpLOre Iter:10 T:30, B:2500, A:300	**(0.27, 0.33)**	(0.26, 0.29)	(0.18, 0.23)	(0.21, 0.28)	(0.18, 0.24)	(0.21, 0.27)
Bookshelves and Tables (3D)		Train Size (m): 100 Test Size:10 ExpLOre Iter:10 T:10, B:55, A:100	**(0.05, 0.24)**	(0.01, 0.04)	(0.01, 0.04)	(0.01, 0.04)	(0.01, 0.22)	(0.01, 0.19)
Cluttered Construction Site (3D)		Train Size (m): 100 Test Size:10 ExpLOre Iter:10 T:10, B:55, A:100	**(0.08, 0.12)**	(0.01, 0.12)	(0.01, 0.09)	(0.01, 0.09)	(0.01, 0.11)	(0.01, 0.10)
Office Desk and Chairs (3D)		Train Size (m): 100 Test Size:10 ExpLOre Iter:10 T:10, B:12, A:50	**(0.55, 0.72)**	(0.46, 0.59)	(0.48, 0.63)	(0.48, 0.63)	(0.43, 0.52)	(0.41, 0.53)

Fig. 5: Comparison of ExpLOre with baseline heuristics on a number of experiments both 2D and 3D. Each row corresponds to different datasets. The columns contain information about the dataset, representative pictures and performance results for all algorithms. The numbers are the lower and upper confidence (for 95% CI) of cumulative reward at the final time step. The algorithm with the highest median performance is emphasized in bold.

baseline heuristics. A dataset comprises of 2D binary world maps, uniformly distributed nodes and a simulated laser. The training size is 100, $T = 30$, $B = 2500$.

1) Dataset 1: Concentrated Information: Fig. 3a shows a dataset created by applying random affine transformations to a pair of parallel lines. This dataset is representative of information being concentrated in a particular fashion. Fig. 3c shows a comparison of ExpLOre with baseline heuristics. The heuristic Rear Side Voxel performs the best, while ExpLOre is able to match the heuristic. Fig. 3e shows progress of ExpLOre along with two relevant heuristics - Rear Side Voxel and Average Entropy. Rear Side Voxel takes small steps focusing on exploiting viewpoints along the already observed area. Average Entropy aggressively visits unexplored area which is mainly free space. ExpLOre initially explores the world but on seeing parts of the lines reverts to exploiting the area around it.

2) Dataset 2: Distributed Information: Fig. 3b shows a dataset created by randomly distributing rectangular blocks around the periphery of the map. This dataset is representative of information being distributed around. Fig. 3c shows that the heuristic Average Entropy performs the best, while ExpLOre is able to match the heuristic. Rear Side Voxel saturates early on and performs worse. Fig. 3e shows that Rear Side Voxel gets stuck exploiting an island of information. Average Entropy takes broader sweeps of the area thus gaining more information about the world. ExpLOre also shows a similar behavior of exploring the world map.

Thus we see that on changing the datasets the performance of the heuristics reverse while our data driven approach is able to adapt seamlessly. Other datasets are shown in Fig. 5.

C. 3D Exploration

We create a set of 3D exploration problems to test the algorithm on more realistic scenarios. The datasets comprises of 3D worlds created in Gazebo and simulated Kinect. To show the practical usage of our pipeline, we show a scenario where a policy is *trained on synthetic data and tested on a real dataset*.

Fig. 4a shows some sample worlds created in Gazebo to represent an office desk environment on which ExpLOre is trained. Fig. 4b shows a dataset of an office desk collected by TUM Computer Vision Group [33]. The dataset is parsed to create a pair of pose and registered point cloud which can then be used to evaluate different algorithms. Fig. 4c shows that ExpLOre outperforms all heuristics. Fig. 4f shows ExpLOre getting good coverage of the desk while the best heuristic Occlusion Aware misses out on the rear side of the desk. Fig. 5 shows more datasets where training and testing is done on synthetic worlds.

VII. Conclusion

We presented a novel data-driven imitation learning framework to solve budgeted information gathering problems. Our approach, ExpLOre, trains a policy to imitate a clairvoyant oracle that has full information about the world and can compute non-myopic plans to maximize information. The effectiveness of ExpLOre can be attributed to two main reasons: Firstly, as the distribution of worlds varies, the clairvoyant oracle is able to adapt and consequently ExpLOre adapts as well. Secondly, as the oracle computes non-myopic solutions, imitating it allows ExpLOre to also learn non-myopic behaviors.

VIII. Acknowledgement

The authors thank Sankalp Arora for insightful discussions and open source code for exploration in MATLAB.

References

[1] Benjamin Charrow, Gregory Kahn, Sachin Patil, Sikang Liu, Ken Goldberg, Pieter Abbeel, Nathan Michael, and Vijay Kumar. Information-theoretic planning with trajectory optimization for dense 3d mapping. In *RSS*, 2015.

[2] Chandra Chekuri and Martin Pal. A recursive greedy algorithm for walks in directed graphs. In *FOCS*, 2005.

[3] Yuxin Chen, S. Hamed Hassani, and Andreas Krause. Near-optimal bayesian active learning with correlated and noisy tests. *CoRR*, abs/1605.07334, 2016.

[4] Yuxin Chen, Shervin Javdani, Amin Karbasi, J. Bagnell, Siddhartha Srinivasa, and Andreas Krause. Submodular surrogates for value of information. In *AAAI*, 2015.

[5] Sanjiban Choudhury, Ashish Kapoor, Gireeja Ranade, and Debadeepta Dey. Learning to gather information via imitation. *preprint arXiv:1611.04180*, 2016.

[6] Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *JAIR*, 2011.

[7] Daniel Golovin, Andreas Krause, and Debajyoti Ray. Near-optimal bayesian active learning with noisy observations. In *NIPS*, 2010.

[8] Anupam Gupta, Viswanath Nagarajan, and R Ravi. Approximation algorithms for optimal decision trees and adaptive tsp problems. In *International Colloquium on Automata, Languages, and Programming*, 2010.

[9] Lionel Heng, Alkis Gotovos, Andreas Krause, and Marc Pollefeys. Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments. In *ICRA*, 2015.

[10] Geoffrey A Hollinger, Brendan Englot, Franz S Hover, Urbashi Mitra, and Gaurav S Sukhatme. Active planning for underwater inspection and the benefit of adaptivity. *IJRR*, 2012.

[11] Geoffrey A Hollinger, Urbashi Mitra, and Gaurav S Sukhatme. Active classification: Theory and application to underwater inspection. *preprint arXiv:1106.5829*, 2011.

[12] Geoffrey A Hollinger and Gaurav S Sukhatme. Sampling-based motion planning for robotic information gathering. In *RSS*, 2013.

[13] Stefan Isler, Reza Sabzevari, Jeffrey Delmerico, and Davide Scaramuzza. An information gain formulation for active volumetric 3d reconstruction. In *ICRA*, 2016.

[14] Rishabh K Iyer and Jeff A Bilmes. Submodular optimization with submodular cover and submodular knapsack constraints. In *NIPS*, 2013.

[15] Shervin Javdani, Yuxin Chen, Amin Karbasi, Andreas Krause, J. Andrew Bagnell, and Siddhartha Srinivasa. Near optimal bayesian active learning for decision making. In *AISTATS*, 2014.

[16] Shervin Javdani, Matthew Klingensmith, J. Andrew Bagnell, Nancy Pollard, and Siddhartha Srinivasa. Efficient touch based localization through submodularity. In *ICRA*, 2013.

[17] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 1998.

[18] Michael Koval, Nancy Pollard, and Siddhartha Srinivasa. Pre- and post-contact policy decomposition for planar contact manipulation under uncertainty. In *RSS*, 2014.

[19] Andreas Krause and Daniel Golovin. Submodular function maximization. *Tractability: Practical Approaches to Hard Problems*, 2012.

[20] Andreas Krause and Carlos Guestrin. Near-optimal observation selection using submodular functions. In *AAAI*, 2007.

[21] Andreas Krause, Jure Leskovec, Carlos Guestrin, Jeanne VanBriesen, and Christos Faloutsos. Efficient sensor placement optimization for securing large water distribution networks. *Journal of Water Resources Planning and Management*, 2008.

[22] Hanna Kurniawati, David Hsu, and Wee Sun Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *RSS*, 2008.

[23] Andy Liaw and Matthew Wiener. Classification and regression by randomforest.

[24] Zhan Wei Lim, David Hsu, and Wee Sun Lee. Adaptive stochastic optimization: From sets to paths. In *NIPS*. 2015.

[25] Zhan Wei Lim, David Hsu, and Wee Sun Lee. Adaptive informative path planning in metric spaces. *IJRR*, 2016.

[26] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *AISTATS*, 2010.

[27] Stephane Ross and J Andrew Bagnell. Reinforcement and imitation learning via interactive no-regret learning. *preprint arXiv:1406.5979*, 2014.

[28] Guy Shani, Joelle Pineau, and Robert Kaplow. A survey of point-based pomdp solvers. *Autonomous Agents and Multi-Agent Systems*, 2013.

[29] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *NIPS*, 2010.

[30] Amarjeet Singh, Andreas Krause, Carlos Guestrin, William Kaiser, and Maxim Batalin. Efficient planning of informative paths for multiple robots. In *IJCAI*, 2007.

[31] Amarjeet Singh, Andreas Krause, and William J. Kaiser. Nonmyopic adaptive informative path planning for multiple robots. In *IJCAI*, 2009.

[32] Adhiraj Somani, Nan Ye, David Hsu, and Wee Sun Lee. Despot: Online pomdp planning with regularization. In *NIPS*, 2013.

[33] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *IROS*, 2012.

[34] Jingjin Yu, Mac Schwager, and Daniela Rus. Correlated orienteering problem and its application to informative path planning for persistent monitoring tasks. In *IROS*, 2014.

[35] Haifeng Zhang and Yevgeniy Vorobeychik. Submodular optimization with routing constraints. In *AAAI*, 2016.