IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 19, 2019, accepted September 10, 2019, date of publication September 16, 2019, date of current version October 1, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2941706

Task-Oriented Active Sensing via Action Entropy Minimization

TIPAKORN GREIGARN[1], (Student Member, IEEE), MICHAEL S. BRANICKY[2], (Fellow, IEEE), AND M. CENK ÇAVUŞOĞLU[1], (Senior Member, IEEE)

[1]Department of Electrical Engineering and Computer Science, Case Western Reserve University, Cleveland, OH 10900, USA
[2]Department of Electrical Engineering and Computer Science, The University of Kansas, Lawrence, KS 66045, USA

Corresponding author: Tipakorn Greigarn (txg92@case.edu)

This work was supported in part by the National Science Foundation under Grant CISE IIS-1524363 and Grant IIS-1563805, and in part the National Institutes of Health under Grant R01 EB018108.

ABSTRACT In active sensing, sensing actions are typically chosen to minimize the uncertainty of the state according to some information-theoretic measure such as entropy, conditional entropy, mutual information, etc. This is reasonable for applications where the goal is to obtain information. However, when the information about the state is used to perform a task, minimizing state uncertainty may not lead to sensing actions that provide the information that is most useful to the task. This is because the uncertainty in some subspace of the state space could have more impact on the performance of the task than others, and this dependence can vary at different stages of the task. One way to combine task, uncertainty, and sensing, is to model the problem as a sequential decision making problem under uncertainty. Unfortunately, the solutions to these problems are computationally expensive. This paper presents a new task-oriented active sensing scheme, where the task is taken into account in sensing action selection by choosing sensing actions that minimize the uncertainty in future task-related actions instead of state uncertainty. The proposed method is validated via simulations.

INDEX TERMS Active sensing, decision making, uncertainty, entropy.

I. INTRODUCTION

Uncertainty is unavoidable in any real-world problem, whether it is from modeling errors, changes in the environment, numerical approximations, etc. An effective way of dealing with uncertainty is to take sensor data into consideration, and make appropriate adjustments to the system [40]. This is a common practice across many fields, ranging from systems and control to sequential decision making under uncertainty. When the sensor can be controlled, the sensor can be utilized to obtain more informative observations. Choosing sensing actions to gain more useful data is known as *active sensing*.

The goal of active sensing is typically to reduce uncertainty in the state of the system by minimizing some information-theoretic measure such as entropy, conditional entropy, mutual information, etc. While this is sufficient for many applications where the goal is to obtain information, when the information on the state is used to complete a task, minimizing state uncertainty may not lead to sensing actions that provide information that is most useful for the task. This is because the reduced uncertainty may not be the most relevant to the current stage of the task. For example, consider a driving analogy. Let the state space be the positions of nearby cars and the sensing actions are looking left and right. Suppose the driver wants to merge right to change lanes; the positions of the cars on the right determine whether or not they can merge. Therefore looking to the right is the logical sensing action to take. However, from a state uncertainty perspective, the uncertainty of the positions of the cars on the left and right are equivalent. Since only the positions of the cars to the right affects our action of changing lanes, only the uncertainty of the position of the car to the right matters. If we try to reduce the uncertainty of the merging action, we will arrive at the same conclusion that the driver should look to the right.

In some decision making problems, the action space naturally decomposes into task and sensing action spaces, where the task-action space is the set of actions that affect the state of the system, while the sensing-action space is the

The associate editor coordinating the review of this manuscript and approving it for publication was Fatih Emre Boran.

set of actions that affect possible observations. For example, in image-guided robotic surgical intervention, the task-action space is all possible movements of the surgical robot, while the sensing-action space is the range of possible image slices from the imaging system. In this case, it is possible to decompose the sequential decision making problem into task-related planning and active sensing. So, the surgical robot planner only solves for the best way of performing the surgery, while the active sensing algorithm selects the image slices that are most useful for the surgery. Another example of a decoupled system is a Mars rover that is guided by an orbital satellite. In this scenario, the rover decides where to explore next based on the information provided by the satellite. This problem can be decoupled by considering where the rover goes next as the task action, while the path of the satellite can be considered as the sensing action. The key feature shared by both examples is the fact that the state, i.e., the configuration of the surgical robot in the first example, and the location of the rover in the latter, does not influence the sensing capability. In other words, the robot (or the rover) does not have to be in a particular state to obtain relevant information. For such systems, the sensing and the acting aspects of the are decoupled, which allows us to solve the problem separately. The proposed decoupling of sensing and acting shares the same basic ''divide and conquer'' concept with the principle of separation of estimation and control, where the state estimator is designed to reduce the error between the actual and the estimated state, while the controller is designed to perform the task assuming perfect information.

This paper presents a new task-oriented active sensing method that integrates the task into active sensing by choosing sensing actions that minimize the uncertainty of future task actions. The proposed algorithm reduces the ambiguity of the next task-related action by choosing the sensing action that minimizes the conditional entropy of the next task-related action. First, a discrete version of the algorithm, which works with systems with discrete state and action spaces, is presented as a proof of concept. For systems with continuous state and action spaces, the particle filter is used to approximate belief propagation over continuous state space. Differential entropy is estimated from particles using k-nearest neighbor estimator.

The rest of the paper is organized as follows. Related work is reviewed in Section II. Problem formulation is described in Section III. A quick review of information theory is given in Section IV. The proposed active sensing algorithm for discrete systems is described in Section V. The proposed active sensing algorithm for continuous systems is described in Section VI. Simulations and results of discrete system examples are presented in Section VII. Simulations and results of continuous system examples are presented in Section VIII. Conclusions are presented in Section IX.

II. RELATED WORK

Information theory provides a basis on which various active sensing algorithms for many different applications are built. This section describes some of the related work and highlights the differences between this work and existing ones.

Typically an active sensing algorithm chooses a sensing action to optimize some information measure. Schmaedeke proposes one of the early active sensing algorithms that utilize information theory [35]. In the paper, sensor-target assignment is performed by maximizing the difference in Kullback-Leibler divergence formulated as a linear program. Manyika and Durrant-Whyte present a data fusion method that maximizes the information content of the state [25]. Information of the state is estimated recursively using an information filter. Porta et al. solve visual localization by choosing camera control actions that minimize the conditional entropy of the belief [29]. A Gaussian mixture observation model is learned a priori from training data. The particle filter is used in propagating the belief in continuous spaces and the conditional entropy of the belief is estimated from the particles. Kreucher et al. perform sensor scheduling for multi-target tracking by minimizing Rényi divergence of the belief, which is represented by a set of particles [20]. Hoffman and Tomlin propose an information gathering method for a mobile sensor network that performs decentralized mutual information maximization [17].

Partially-Observable Markov Decision Processes (POMDPs) is a general framework for describing sequential decision making under uncertainty. However, this generality comes at a cost [24]. Only very small problems, with a few states, can be solved exactly due to the curse of dimensionality and the curse of history [24], [28]. Researchers have put a lot of effort into approximate methods that solve POMDPs more efficiently. A survey of earlier value function approximation techniques can be found in [16]. One of the earlier work that aims at approximating POMDP is the QMDP proposed by Littman *et al.* [24]. The Q in QMDP represents the Q function in Q-learning, while MDP stands for Markov Decision Process. The QMDP algorithm first learn the Q function of the MDP part of a POMDP, ignoring the sensing aspect of the task. During execution, an action that maximizes the expected Q function is chosen. A milestone in approximate POMDP method is the Point-Based Value Iteration (PBVI) method proposed by Pineau *et al.* [28]. The PBVI method approximates a POMDP value function using a small set of belief points and the hyperplanes at the belief points. Online POMDP is another class of approximate method that solves POMDPs forward in time, instead of performing backup as in value iteration. Determinized Sparse Partially Observable Tree (DESPOT), presented by Somani et al. in [37], is an example of such algorithms. Porta et al. extend the PBVI method to a subset of continuous POMDPs, where the model can be described by Gaussian mixtures [30].

Instead of finding a sensing action, some work focuses on finding a control trajectory or a policy that produces state trajectory which leads to more effective information gathering. Ryan and Hedrick calculate an information-seeking control trajectory by minimizing the finite horizon sum of

the entropy of the belief [33]. The finite horizon entropy is calculated by Monte-Carlo simulation. Araya et al. propose ρPOMDP, which extends the POMDP formulation to the case where the reward function is defined in the belief space [3]. By defining the reward function over the belief space, ρPOMDP is capable of formulating decision making problems where information acquiring is the end goal. Chakravorty and Erwin formulate sensor scheduling as a receding horizon control problem with information gain as the objective function [6]. The policy gradient is obtained from Monte-Carlo simulations. A probabilistic policy allows soft maximization using the policy gradient. Charrow et al. present a multi-robot information gathering algorithm [7]. Control trajectories for multi-robot information gathering are obtained by maximizing the mutual information over the state trajectory. Wang et al. formulate information gathering as a POMDP where the action space is decoupled into state-changing actions and observation-making actions, where the state-changing action is assumed to be deterministic [42]. Javdani et al. propose a hyperedge cutting algorithm that drives the uncertainty over hypotheses down to one decision region by greedily maximizing the expected marginal gain of a test [18]. Spaan et al. formulate cooperative perception of networked robot systems as a POMDP [38]. The positions of the targets and the robots, as well as the features of the targets, are considered as the state of the system. Instead of defining a reward function on the belief to encourage information gathering, classification actions and book-keeping variables are incorporated into the formulation to maintain the piecewise linear and convex property of the POMDP. Satsangi et al. solve sensor selection problem by modifying the backup step in the point-based value iteration (PBVI) algorithm [34]. Instead of considering a combination of sensors as an action in POMDP, each sensor is added via greedy maximization of the value function during the PBVI backup, which reduces the computation complexity of each iteration of PBVI significantly.

While it is rather straightforward to describe a sequential decision making under uncertainty as a POMDP, it is generally difficult to solve in many real-world applications. As a results, many researchers have come up with different planning algorithms that exploit different characteristics of the problem in order to deal with the uncertainty in the system. Chhatpar and Branicky solve robotic assembly problems where clearance is much smaller than the visual feedback by constructing a contact map in the configuration space [8]. Prentice and Roy present the belief roadmap, where the covariance matrices are factored and stored in the edges, which simplifies posterior distribution and expected cost calculations [31]. Van den Berg et al. formulate belief space planning problems with Gaussian motion and measurement models as iterative LQG (iLQG) problems, where the initial trajectory is generated from a sampling-based method in the state, then the trajectory is updated iteratively using iLQG algorithm [41]. Agha-Mohammadi et al. solve belief space planning using PRM where the edges are stabilizing controllers [1].

There is some prior work on active sensing methods that are designed to provide information related to a given task. Kwok and Fox use reinforcement learning for sensing action selection for soccer robots [22]. The long-term value of a sensing action is learned in the training phase. The robot then performs the best action according to the learned action-value function. Guerrero et al. present two task-oriented active sensing methods for soccer robots [15]. The difference between the two methods is the objective function. In the first method, the variance of the value function of the next iteration is minimized, while in the second method, the expected value of the value function of the next iteration is maximized. Experimental results show that the expected value maximization method is more effective than the variance minimization method.

The proposed active sensing method is different from the existing work for several reasons. First, it is different from POMDP-based methods because it does not solve the planning problem in the belief space. The action-entropy active sensing algorithm aims at providing a tractable solution to a subset of sequential decision making problems where the sensing and the acting aspects of the task are naturally decoupled. The action-entropy active sensing algorithm chooses a sensing action that minimizes the uncertainty in the next task action according to the policy provided by the state-space planner. While the proposed solution to the decoupled problem may be suboptimal in terms of accumulated reward, the decoupled approach could make up for the smaller reward in terms of computational time. Moreover, the proposed method is different from the other information gathering methods such as [3] and [38] because it is designed to be used in the context of task completion. So, the ultimate goal is not to obtain information but to perform the task well. Finally, the proposed method is different from [22] and [15] because no training or value function is required. Instead, the proposed method only assumes the availability of a policy that maps a state into a task-related action. A preliminary study of the proposed method, specifically the continuous formulation of the problem with a Python implementation, is presented in [14]. In this paper, both discrete and continuous formulations are presented, with the discrete algorithm implemented in Python, and the continuous algorithm implemented in C++. The proposed method is also explored and validated in more depth with two examples for the discrete case and two new examples for the continuous case.

III. PROBLEM FORMULATION

The objective of the active sensing algorithm in this work is to provide useful information for task completion under uncertainty, where both motion and sensing uncertainties are considered. The proposed method is based on Bayesian estimation, which is reviewed in this section.

In decision-making problems where sensing actions do not affect the states, such as looking around while driving or taking image slices in image-guided robotic surgery, the action space can be divided into two action spaces. First, the space of actions that change the state of the system is denoted by $\mathcal{U}$. This set of action is used in completing the task so it will be referred to as the *task-action space*. The other action space, denoted by $\mathcal{V}$, is the space of actions that affect the measurement but not the state. Since this set of actions is only used in sensing, it shall be called the *sensing-action space*. Let $\mathcal{X}$ and $\mathcal{Z}$ denote the state space and the measurement space respectively. The state-transition model, $p(x_t|u_t, x_{t-1})$, is a probability distribution function of the next state $x_t \in \mathcal{X}$ given that a task action $u_t \in \mathcal{U}$ is performed at a state $x_{t-1} \in \mathcal{X}$. The measurement model, $p(z_t|v_t, x_t)$, is a distribution of the measurement $z_t \in \mathcal{Z}$ given that a sensing action $v_t \in \mathcal{V}$ is performed at a state $x_t \in \mathcal{X}$.

The information the robot has about the state of the environment is represented by its belief, $b(x_t)$, which is a distribution defined over the state space given all past actions and measurements, i.e.,

$$b(x_t) = p(x_t|u_{1:t}, v_{1:t}, z_{1:t}), \forall x_t \in \mathcal{X}, \tag{1}$$

where $b_0 = p(x_0)$ is the probability distribution of the initial state. For brevity, the time subscript t will be dropped where it would not cause confusion. When the robot interacts with the environment, its belief propagates according to the state-transition model as follows,

$$\bar{b}(x_t) = \int_{\mathcal{X}} p(x_t|u_t, x_{t-1}) b(x_{t-1}) dx_{t-1}. \tag{2}$$

For systems with discrete state space, the integral in belief propagation becomes a summation as follows,

$$\bar{b}(x_t) = \sum_{x_{t-1} \in \mathcal{X}} p(x_t|u_t, x_{t-1}) b(x_{t-1}). \tag{3}$$

When the robot performs a sensing action and receives a measurement, its belief is updated according to its measurement model as follows,

$$b(x_t) = \eta p(z_t|v_t, x_t) \bar{b}(x_t). \tag{4}$$

where η is the prior probability of observation that normalizes the expression on the right-hand side.

In this work, it is assumed that a state-space planner is available for task-related planning. This planner will be referred to as the *task planner*. Various state-space planners can be used as the task planner [9], [23]. Fundamentally, the only requirement of the task planner is to generate a policy $\pi : \mathcal{X} \to \mathcal{U}$ that tells the robot what to do at each state. The desirable behavior of the robot is encoded in the policy. For example, mobile robot navigation would have a policy that brings the robot from any state to the goal state.

The goal of this work is to create an active sensing algorithm that works with a state-space planner in completing a task. The proposed active sensing method integrates sensing and acting aspects of the task by choosing the sensing action v_t that minimizes the conditional entropy of the next task action $u_{t+1} = \pi(x_t)$. Since the objective function of the active sensing method is minimizing task-action entropy, the proposed method shall be referred to as the *action-entropy* active sensing method.

IV. BACKGROUND ON INFORMATION THEORY

A short review on conditional entropy of discrete and continuous random variables is presented in this section. More detail on information theory can be found in [10].

The entropy of a discrete random variable X with probability mass function $p(x)$ and range $\mathcal{X}$ is defined as

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x). \tag{5}$$

Suppose there is another discrete random variable Y with range $\mathcal{Y}$. If X and Y have a joint mass function $p(x, y)$, then the conditional entropy $H(X|Y)$ is defined as follows,

$$H(X|Y) = -\sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} p(x, y) \log p(x|y). \tag{6}$$

Since $p(x, y) = p(x|y)p(y)$, we can write the conditional entropy as the expected value of the entropy $H(X|Y = y)$ as follows,

$$\begin{aligned} H(X|Y) &= \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} p(x|y)p(y) \log p(x|y), \\ &= -\sum_{y \in \mathcal{Y}} p(y) \sum_{x \in \mathcal{X}} p(x|y)p(y) \log p(x|y), \\ &= \sum_{y \in \mathcal{Y}} p(y) H(X|Y = y). \end{aligned} \tag{7}$$

The concept of entropy has been extended to continuous variables. The entropy of a continuous variable is called the differential entropy. Let X be a continuous random variable with probability density function $p(x)$ and range $\mathcal{X}$. The differential entropy of X is defined as follows,

$$h(X) = -\int_{\mathcal{X}} p(x) \log p(x) dx. \tag{8}$$

Suppose there is another continuous random variable Y with range $\mathcal{Y}$. If X and Y have a joint density function $p(x, y)$, then similar to the discrete case, the conditional differential entropy $h(X|Y)$ is defined as the entropy of the conditional distribution as follows,

$$h(X|Y) = -\int_{\mathcal{Y}} \int_{\mathcal{X}} p(x, y) \log p(x|y) dx dy. \tag{9}$$

Since $p(x, y) = p(x|y)p(y)$, the conditional differential entropy can be written as the expected value of the differential entropy $h(X|Y = y)$ as follows,

$$\begin{aligned} h(X|Y) &= -\int_{\mathcal{Y}} \int_{\mathcal{X}} p(x|y)p(y) \log p(x|y) dx dy, \\ &= -\int_{\mathcal{Y}} p(y) \int_{\mathcal{X}} p(x|y) \log p(x|y) dx dy, \\ &= \int_{\mathcal{Y}} p(y) h(X|Y = y) dy. \end{aligned} \tag{10}$$

The conditional entropy and conditional differential entropy described in (7) and (10) are used in measuring the degree of uncertainty for sensing action selection.

V. ACTION-ENTROPY ACTIVE-SENSING ALGORITHM FOR DISCRETE SYSTEMS

The action-entropy active sensing algorithm for discrete systems is presented in this section. The discrete version of the active sensing method is meant to be a proof of concept of the algorithm that illustrates how well the method works in its most basic form. The active sensing algorithm is a part of the planner, as depicted in Fig. 1. The other two components of the planner include the task planner, which provides a state-space policy, and the coordinator, which maintains the internal belief of the planner and interacts with the environment.

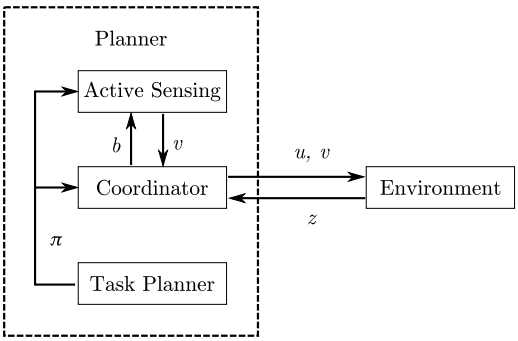

FIGURE 1. The planner is divided into three submodules. The sensing module handles active sensing, while the task planner solves the planning problem in the state space. The coordinator maintains the internal belief and interacts with the environment. When the planner is initialized, the task planner solves the planning problem in the state space for a policy π, which is passed on to the active sensing module. The coordinator passes its internal belief b to the active sensing module. Then the active sensing module maps the belief through the policy to obtain a probability distribution of task-action, calculates the task-action entropy, and returns the sensing action v that minimizes the entropy of the task action. The coordinator obtains the sensing action, performs it, receives a measurement z, and updates its belief according to the measurement update in (4). Next, the coordinator selects the task action u according to the policy based on the most likely state from the belief. Then the task action is performed and the coordinator updates its internal belief according to the belief prediction in (3).

A mathematical derivation of the algorithm is presented first in Section V-A. An analysis of the algorithm is given in Section V-B.

A. MATHEMATICAL DERIVATION

Since a policy is often non-injective, it could map many states to the same action, so the same action would be taken for all of those states. In such cases, the uncertainties among the states that map to the same task action do not affect action selection, and therefore, sensing action selection based on state uncertainty could be misleading or suboptimal in terms of task performance. In this work, the active sensing module assists the task planner by selecting the sensing actions that makes future task actions the least ambiguous by minimizing the conditional entropy of the task-action distribution.

Specifically, the sensing actions are chosen to minimize the conditional entropy of the task actions given a prior belief on the state,

$$v_t = \underset{v_t \in \mathcal{V}}{\operatorname{argmin}} H(\hat{u}_{t+1} | \hat{z}_t, v_{1:t}, u_{1:t}, z_{1:t-1}), \tag{11}$$

where $\hat{u}_{t+1}$ and $\hat{z}_t$ denote the random variables of the next task action and the observation, respectively.

The conditional entropy, $H(\hat{u}_{t+1}|\hat{z}_t, v_{1:t}, u_{1:t}, z_{1:t-1})$, on the right hand side of (11), is the expected entropy of the task-action distribution with $\hat{z}_t$ as the conditioning variable. From (7), the conditional entropy is calculated as follows,

$$\begin{aligned} &H(\hat{u}_{t+1}|\hat{z}_t, v_{1:t}, u_{1:t}, z_{1:t-1}) \\ &\quad = \sum_{z_t \in \mathcal{Z}} p(z_t|v_{1:t}, u_{1:t}, z_{1:t-1}) H(\hat{u}_{t+1}|v_{1:t}, u_{1:t}, z_{1:t}) \end{aligned} \tag{12}$$

There are two components in (12). The first term in the summation, $p(z_t|v_{1:t}, u_{1:t}, z_{1:t-1})$, is the probability of observing z_t when sensing action v_t is taken, given a priori observation and actions. This term is calculated by expanding the probability using the law of total probability to obtain

$$\begin{aligned} &p(z_t|v_{1:t}, u_{1:t}, z_{1:t-1}) \\ &\quad = \sum_{x_t \in \mathcal{X}} p(z_t|x_t, v_{1:t}, u_{1:t}, z_{1:t-1}) p(x_t|v_{1:t}, u_{1:t}, z_{1:t-1}) \\ &\quad = \sum_{x_t \in \mathcal{X}} p(z_t|x_t, v_t) \bar{b}_t. \end{aligned} \tag{13}$$

The first term in (13), $p(z_t|x_t, v_{1:t}, u_{1:t}, z_{1:t-1})$, reduces to the measurement model, $p(z_t|x_t, v_t)$, when the state is given. The second term in (13), $p(x_t|v_{1:t}, u_{1:t}, z_{1:t-1})$, is the prior belief $\bar{b}_t$, since, without the observation z_t the state x_t is independent of the sensing action v_t, and the distribution over the state is simply the a priori belief, $\bar{b}$.

The second term in (12), $H(\hat{u}_{t+1}|v_{1:t}, u_{1:t}, z_{1:t})$, is the entropy of the next task action given the current belief. Since the action u_{t+1} is selected from the state x_t according to the policy, i.e., $u_{t+1} = \pi(x_t)$, the distribution of $\hat{u}_{t+1}$ can be obtained by mapping the belief distribution over the states through π. The entropy of the task action is then calculated from (5).

The proposed active sensing method is summarized in Algorithm 1. The inputs to the algorithms are the policy π and the a priori belief $\bar{b}_x$ (the subscript x is used to distinguish the belief over the state from the belief over the task action in the algorithm). The conditional entropy associated with each sensing action is initialized in Line 2. The conditional entropy for sensing action v is calculated in Lines 3 to 13. The probability of observing z from (13) is calculated in Lines 5 to 8, where ω in Line 7 is a dummy variable that stores the probability of observing z. The belief is updated according to observation z in Line 9. The belief over the next task action is calculated in Line 10 by mapping the belief over the state through the policy. Then the summation over $z \in \mathcal{Z}$ is carried out in Line 11.

Algorithm 1 Active-Sensing Algorithm for Discrete Systems

1: **procedure** DiscreteActiveSensing($\pi, \bar{b}_x$)
2: $H_v = 0, \forall v \in \mathcal{V}$
3: **for all** $v \in \mathcal{V}$ **do**
4: **for all** $z \in \mathcal{Z}$ **do**
5: $\omega = 0$
6: **for all** $x \in \mathcal{X}$ **do**
7: $\omega \mathrel{+}= p(z|v, x)\bar{b}_x$
8: **end for**
9: $b_x = \eta p(z|v, x)\bar{b}_x$
10: Calculate b_u from π and b_x
11: $H_v \mathrel{+}= \omega H(b_u)$
12: **end for**
13: **end for**
14: **return** $\text{argmin}_v H_v$
15: **end procedure**

B. ANALYSIS

This active sensing scheme is attractive for a couple of reasons. First, the uncertainty of the task actions is defined over the action space, which is often "smaller" than the state space. From this perspective, the policy compresses the uncertainty into a smaller space. Additionally, when many states are mapped into the same action, the uncertainty among those states does not contribute to the uncertainty in the task actions. In this case, the states are clustered together according to the policy, and the active sensing module only has to make its decision based on the clusters.

The time complexity of this algorithm is simply $O(|\mathcal{V}||\mathcal{Z}||\mathcal{X}|)$, where $|\cdot|$ is the counting measure. Time complexity of a state-entropy active sensing algorithm has the same order as the action-entropy algorithm. They only differ by a constant factor because the time it takes to map the state through the policy (line 10) is also $O(|\mathcal{X}|)$.

VI. ACTION-ENTROPY ACTIVE-SENSING ALGORITHM FOR CONTINUOUS SYSTEMS

Most problems in robotics are continuous in nature, so an active sensing algorithm for continuous systems will be compatible with a wider variety of applications. In this section, the action-entropy active sensing algorithm for systems with continuous state and action spaces is presented. There are two classes of methods that are widely used in robotics for dealing with beliefs over continuous spaces. Parametric methods such as Gaussian filters often make more assumptions on the underlying system than nonparametric methods such as the particle filter. In order to make our algorithm the least restrictive, the particle filter is chosen here.

For a belief defined over a continuous space, the entropy is replaced by the differential entropy. Since belief propagation is represented by the particle filter, the differential entropy has to be estimated from the particles. Estimation of the differential entropy is presented in Section VI-A.

The structure of the system is the same as in the discrete case, shown in Fig. 1. The mathematical derivation of the active sensing algorithm for continuous system is presented in Section VI-B. The analysis of the algorithm is given in Section VI-C.

A. DIFFERENTIAL ENTROPY ESTIMATION

This section presents estimation of differential entropy from a set of particles. Suppose there is a set of particles $\mathcal{P}$ containing N particles. The weight and the position of the i-th particle shall be denoted by w^i and x^i, respectively. A belief is approximated by the set of particles $\mathcal{P}$ as follows,

$$b_{\mathcal{P}}(x) = \sum_{i=1}^{N} w^i \delta(x - x^i), \tag{14}$$

where $b_{\mathcal{P}}$ denotes a belief represented by a set of particles $\mathcal{P}$, and δ is the Dirac delta function.

An overview of nonparametric estimation of differential entropy can be found in [4]. While some estimators are defined for one or two dimensional datasets, nearest neighbor methods do not share the same restriction. Kozachenko and Leonenko propose the nearest neighbor estimator which estimates the probability distribution function around each sample using the distance to its nearest neighbor [19]. Singh et al. and Goria et al. generalize the nearest neighbor estimator to k-nearest neighbor estimators, where the distance to the k-th nearest neighbor is used to approximate the local density function [13], [36]. Mnatsakanov et al. have shown that the root-mean-square error of the estimation is lower with $k > 1$, and an empirical formula for k is proposed [27]. Ajgl and Šimandl extend the k nearest neighbor to the case when the underlying distribution is represented by particles instead of samples [2]. The estimator uses the sum of the weights of k nearest neighbors particles as well as the distance to the k-th nearest particle to estimate differential entropy.

The estimator presented in [2] is chosen in this work because it is capable of estimating the entropy associating with a set of particles. The estimator, denoted by $\hat{h}_{\mathcal{P}}$, is defined as follows,

$$\hat{h}_{\mathcal{P}}(X) = -\sum_{i=1}^{N} \frac{\sum_{j\in\mathcal{N}_k^i} w^j}{k} \log \frac{\sum_{j\in\mathcal{N}_k^i} w^j}{|B_n(d_k^i)|} + \log k - \Psi(k), \tag{15}$$

where $\mathcal{N}_k^i$ is the set of indices of the i-th particle's k nearest neighbors. $|B_n(d_k^i)|$ is the volume of a ball in $\mathbb{R}^n$ with radius d_k^i, where the radius is the distance from the i-th particle to its k-th nearest neighbor. Finally, $\log k - \Psi(k)$ is the bias term, where Ψ is the digamma function.

B. MATHEMATICAL DERIVATION

The active sensing algorithm minimizes the uncertainty of the future task action by choosing the sensing action that minimizes the entropy of the future task action. Since the observation is not known when the sensing action is being

chosen, the entropy is conditioned on the observation. The sensing action is the minimizer of the conditional entropy, i.e.,

$$v_t = \underset{v_t \in \mathcal{V}}{\operatorname{argmin}}\, h(\hat{u}_{t+1}|\hat{z}_t; v_{1:t}, u_{1:t}, z_{1:t-1}), \tag{16}$$

where $\hat{u}_{t+1}$ denotes the random variable of the next task action and $\hat{z}_t$ denotes the random variable of the observation. The conditional entropy is calculated for each sensing action v_t, and the sensing action that minimizes it is chosen. The sensing-action space is assumed to be discrete, and if it is continuous, sampling or discretization can be employed.

The entropy of the future task action conditioned on the observation, according to (10), is as follows,

$$\begin{aligned} &h(\hat{u}_{t+1}|\hat{z}_t; v_{1:t}, u_{1:t}, z_{1:t-1}) \\ &\quad = \int_{\mathcal{Z}} p(z_t|v_{1:t}, u_{1:t}, z_{1:t-1}) h(\hat{u}_{t+1}|v_{1:t}, u_{1:t}, z_{1:t}) dz_t. \end{aligned} \tag{17}$$

The first term in the integration, $p(z_t|v_{1:t}, u_{1:t}, z_{1:t-1})$, is the probability density function of the observation z_t given prior actions and observations. Using the law of total probability, the density function can be expanded as follows,

$$\begin{aligned} &p(z_t|v_{1:t}, u_{1:t}, z_{1:t-1}) \\ &\quad = \int_{\mathcal{X}} p(z_t|x_t, v_{1:t}, u_{1:t}, z_{1:t-1}) p(x_t|v_{1:t}, u_{1:t}, z_{1:t-1}) dx_t, \\ &\quad = \int_{\mathcal{X}} p(z_t|v_t, x_t) \bar{b}(x_t) dx_t. \end{aligned} \tag{18}$$

The first term in the integral, $p(z_t|x_t, v_{1:t}, u_{1:t}, z_{1:t-1})$, reduces to the measurement model, $p(z_t|v_t, x_t)$, when the state x_t is given. The second term in the integral, $p(x_t|v_{1:t}, u_{1:t}, z_{1:t-1})$, is simply the predicted belief, $\bar{b}(x_t)$.

The remaining term in (17), $h(\hat{u}_{t+1}|v_{1:t}, u_{1:t}, z_{1:t})$, is the differential entropy of the next task action given all prior actions and observations. Similar to the discrete case, the distribution of the task action is obtained by mapping the belief particles through the policy. The entropy is then estimated from the task-action particles using (15).

The active sensing algorithm is shown in Algorithm 2. The inputs include the policy π and the particles $\mathcal{P}_x$, which represent the current belief of the coordinator, $\bar{b}$. The algorithm estimates the task-action entropy of each sensing action and return the sensing action with minimum entropy. The integration in (17) is approximated using Monte Carlo simulation in the inner loop, where the integral in (18) is replaced by sampling in Lines 5 and 6. The particles are updated according to the sampled measurement in Line 7. Then the updated particles are mapped into task-action particles in Line 8, and the entropy of the task-action particles are estimated using (15) in Line 9. Finally, the sensing action that results in the lowest conditional entropy is returned.

Algorithm 2 Minimum Task-Action Entropy Active Sensing

```
1:  procedure GetSensingAction(π, P_x)
2:      for all v ∈ V do
3:          h_v = 0
4:          for i = 1, ..., M do
5:              sample x from P_x
6:              sample z from p(z|v, x)
7:              P'_x = MeasurementUpdate(P_x, v, z)
8:              P_u = π(P'_x)
9:              h_v += ĥ_{P_u}(u)
10:         end for
11:     end for
12:     return argmin_v h_v
13: end procedure
```

C. ANALYSIS

Let the number of available sensing actions be denoted by $|\mathcal{V}|$. Monte Carlo simulation is performed M times for each sensing action, so Lines 5 to 9 in Algorithm 1 are executed $|\mathcal{V}|M$ times. Note that it is desirable to perform as many Monte Carlo simulations as possible, so M should be maximized such at a sensing action can be calculated within a given time constraint of the problem. Two bottlenecks in the algorithm are policy evaluation in Line 8 and entropy estimation in Line 9. Entropy estimation requires k-th nearest-neighbor search for each particle. A brute-force nearest neighbor search is $O(N^2)$, where N is the number of particles. It is possible to speed up nearest neighbor search using an algorithm that preprocesses the data so that each nearest neighbor query can be made more efficiently. A widely used nearest neighbor algorithm is the k-d tree algorithm, which has construction time complexity $O(dN \log N)$, where d is the dimension of the particles, and query time complexity $O(\log N)$ under some assumptions [12]. If policy evaluation for each particle does not take more than $O(d \log N)$, then entropy estimation dominates the algorithm's time complexity. In this case the action-entropy active sensing algorithm has time complexity $O(d|\mathcal{V}|MN \log N)$.

VII. SIMULATIONS WITH DISCRETE SYSTEMS

Three examples are presented in this section. They are designed to represent common problems in decision making under uncertainty. In the first example, the robot has to make a decision that would result in a large reward or a large penalty in the future. The second example represents a localization problem where the robot is placed randomly in a symmetric corridor. The last example tests the algorithm in a more general setting using randomly generated obstacles.

The proposed method will be compared with state-entropy active sensing [20], [29], the expected-value maximization method [15], and an algorithm that selects a sensing action uniformly at random. The state-entropy active sensing method selects the sensing action that minimizes the state entropy, i.e.,

$$v_t = \underset{v_t \in \mathcal{V}}{\operatorname{argmin}}\, H(\hat{x}_t|\hat{z}_t, v_{1:t}, u_{1:t}, z_{1:t-1}).$$

The expected-value active sensing method selects the sensing action that maximizes expected future value, i.e.,

$$v_t = \underset{v_t \in \mathcal{V}}{\operatorname{argmax}} \sum_{x_{t+1} \in \mathcal{X}} V(x_{t+1}) p(x_{t+1} | v_{1:t}, u_{1:t}, z_{1:t-1}),$$

where V is the value function [40]. For simplicity, the proposed method, the standard state-entropy method, the expected-value method, and the randomized method shall be abbreviated as AE (Action-Entropy), SE (State-Entropy), EV (Expected-Value), and RN (Randomized) methods, respectively. The results from DESPOT [37], an online POMDP algorithm, are also presented alongside the active sensing results.

The simulations were run on an Ubuntu 14.04 machine with Intel Core 2 Quad 2.40 GHz processor and 4 GB memory. The active sensing algorithms are written in Python.

A. EXAMPLE: FORK IN THE ROAD

The first example is a 5 × 2 grid-world problem, shown in Fig. 2. The robot starts at the bottom. The goal is on the top-right corner and a trap is on the top-left corner. There is a wall between the left and the right side of the map, so the robot has to be able to determine the side of the wall it is on to be able to avoid the trap and reach the goal. This example highlights the ability to make a decision based on future rewards, a central theme in planning. It is critical for the active sensing algorithm to first identify the side of the wall the agent is on. If it is on the right side of the wall, the agent will just move upwards, but if it is on the left side of the wall, then it must move down to the opening then to the right side. With the uniform initial belief, sensing along the north-south and the east-west directions are equally good for the state-entropy method. However, sensing in the north-south direction does not distinguish between being on the left and the right side. So, this will lead to a lower reward. The proposed method on the other hand, will make a sensing action along the east-west direction first because it reduces the uncertainties in the actions the most.

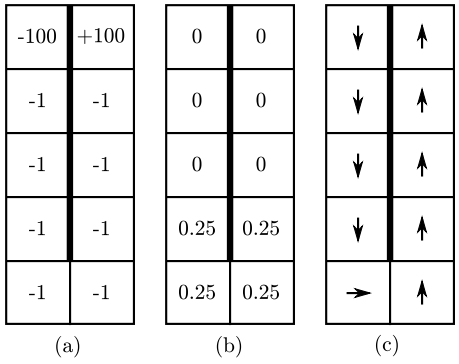

FIGURE 2. A 5 × 2 grid-world with a wall between the left and the right sides and a small opening at the bottom. (a) The reward at each state. (b) The initial belief. (c) The policy.

The intervals between two sensing actions are varied to see how well the active sensing algorithms cope with less information. Between two sensing actions, the belief is propagated according to the robot's motion model without using observations to correct the belief. The robot follows the policy according to the belief until the next sensing opportunity arrives. For AE and SE, the sensing action is chosen to minimize the sum of the entropy up until the next sensing action is possible.

The parameters used in the simulations are as follows. There is a wall between the left and the right sides with one opening at the bottom. The actions are to move in the principle directions (`mov_n`, `mov_s`, `mov_e`, `mov_w`). All the move actions have a probability of 0.8 to end up in the correct state, and a probability of 0.2 to end up in the same state as before the action was taken. The sensing actions are to sense the wall of the perimeter along the north-south and east-west directions (`sen_ns` and `sen_ew`). The observations are seeing a wall in a particular direction, or that no wall is detected (`wall_n`, `wall_s`, `wall_e`, `wall_w`, `none`). All the sensing actions have a probability of 0.9 to sense the wall in the correct direction, and a probability of 0.1 to sense the wall in the opposite direction. There is a reward of +100 at the top-right corner, a penalty of −100 at the top-left corner, a penalty of −1 at other states, and the discount factor is 0.99. The initial belief is uniformly distributed in the bottom four states. The average rewards and computational time for each simulation are taken from 10,000 repeated trials.

The average rewards and the 95% confidence intervals obtained from the four methods are listed in Table 1. First, note that AE performs better than the other three methods for all sensing intervals. The average rewards drop monotonically for all methods when the sensing interval is increased. It takes AE, SE, and EV, 0.282 ms, 0.247 ms, and 0.226 ms, respectively, to calculate a sensing action. The average simulation time of a complete trial, including the offline time required for solving for the policy, of AE, SE, EV, and RN, are 12.132 ms, 11.735 ms, 24.429 ms, and 9.641 ms, respectively. Prior to the simulations, the policy for AE and SE is calculated via policy iteration, which converges in 6.034 ms. The value function for EV is calculated via value iteration, which converges in 18.438 ms. So, while EV is the fastest in sensing action calculation, it takes longer to initialize. DESPOT has an average reward of 78.97, and it takes 5.75 s on average to complete a trial.

B. EXAMPLE: SYMMETRIC CORRIDOR

The second example, shown in Fig. 3, is also a 5 × 2 grid-world problem. This example is inspired by the active localization in a symmetric corridor problem in [40], where being in the right state has an impact on how well the robot can localize itself. In this example, the opening is now in the middle instead of at the bottom. The initial belief is uniformly distributed on the left side. The reward is the same except the penalty at the top-left corner is now -1 instead of -100. Everything else is the same as the previous example. The interval between two sensing actions is varied to see how well the active sensing algorithms handle diminishing information in

TABLE 1. Fork-in-the-road simulation results.

	Sensing Interval				
Results	0	1	2	3	4
AE Reward	64.60 ± 0.41	62.41 ± 0.86	62.16 ± 0.98	57.33 ± 1.06	52.16 ± 1.13
SE Reward	60.71 ± 0.42	51.24 ± 0.96	20.03 ± 1.64	0.89 ± 1.71	-9.08 ± 1.81
EV Reward	63.50 ± 0.47	56.76 ± 0.95	55.37 ± 0.96	50.43 ± 1.07	39.75 ± 1.19
RN Reward	47.97 ± 0.95	38.68 ± 1.29	28.30 ± 1.49	23.75 ± 1.52	19.35 ± 1.58

TABLE 2. Symmetric corridor simulation results.

	Sensing Interval				
Results	0	1	2	3	4
AE Reward	58.70 ± 0.43	54.54 ± 0.74	50.96 ± 0.81	44.10 ± 0.92	48.17 ± 0.91
SE Reward	57.59 ± 0.48	52.28 ± 0.78	48.20 ± 0.85	42.95 ± 0.94	42.63 ± 0.96
EV Reward	58.92 ± 0.40	49.18 ± 0.83	51.46 ± 0.80	43.58 ± 0.92	46.21 ± 0.92
RN Reward	49.03 ± 0.58	40.97 ± 0.85	38.65 ± 0.89	31.30 ± 0.94	30.68 ± 0.94

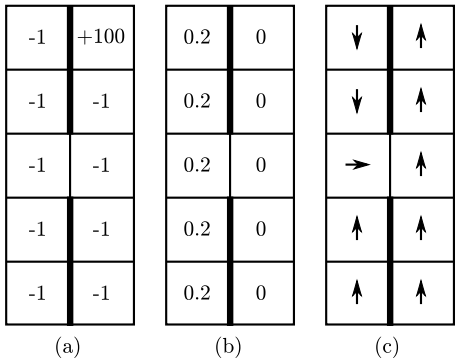

FIGURE 3. A 5 × 2 grid-world with a wall between the left and the right sides and a small opening in the middle. (a) The reward at each state. (b) The initial belief. (c) The policy.

the same way as in the first example. The average rewards and run times for each simulation are taken from 10,000 repeated trials.

The average rewards and the 95% confidence intervals for the four methods are listed in Table 2. In this example, AE, SE, and EV perform similarly in terms of rewards across all sensing intervals, with AE performs slightly better than SE and EV most of the time, and RN always performs worst. Computational time is very similar to the previous example. It takes AE, SE, and EV, 0.281 ms, 0.249 ms, and 0.237 ms, respectively, to calculate a sensing action. The average simulation time of a complete trial, including the offline time required for solving for the policy, of AE, SE, EV, and RN, are 11.529 ms, 11.211 ms, 23.114 ms, and 9.004 ms, respectively. Policy iteration for AE and SE takes 5.33 ms, while value iteration for EV takes 17.70 ms, prior to the simulations. DESPOT has an average reward of 76.579, and it takes 16.97 s on average to complete a trial.

C. DISCUSSION

AE yields higher reward than the other methods in the first example, where the early decisions can lead to a large reward or penalty. The key difference that leads to AE performing better in the fork-in-the-road example is that AE selects the sensing action that minimizes the uncertainty in the next task action. Thus, when the belief is distributed between the two columns of the map, AE will pick the sensing action that resolves the uncertainty regarding the side of the map the robot is in. EV seems to share the same characteristic with AE in this regard. However, as the information become more scarce, i.e., the interval between two sensing opportunities increases, EV's accumulated reward drops faster than AE's. SE performs poorly in this example because it simply tries to minimize state uncertainty without taking the task into consideration.

In the second example, there is no clear winner among the decoupled methods. This is reasonable, because in this problem the robot has to be at the corner to localize itself. Since the active sensing algorithms can only pick what to measure in a given state, they lack the exploration characteristic presented in POMDP. An active-localization algorithm [11], which moves the robot to the corner first, is known to also work well in this scenario. The results from the second example reveal that when it comes to exploration tasks, AE does not do much better than the other methods. In Table 2, there are some non-monotonicity in the rewards as the sensing interval is increased. This could be due to the fact that the problem is rather small, so the number of steps before an action is taken could affect the results.

In the two examples, AE takes slightly longer than the SE because it maps the belief through the policy first. However, entropy calculation is faster because there are less task actions than there are states, so the AE is not significantly slower than the SE. Since policy iteration is faster than value iteration in general [32], AE and SE do not require as much preprocessing time as the EV.

While the decoupled methods have lower rewards compared to POMDP results in the two examples, they are much faster at solving the problems. This trade-off between suboptimality and speed is essential in choosing the tool to

solve a problem. In smaller problems such as in the two examples presented in this section, a POMDP solver may be a better tool since the problems can be solved in a reasonable amount of time. However, if the state space is large, or even continuous, the decoupled methods may offer a better trade-off.

VIII. SIMULATIONS WITH CONTINUOUS SYSTEMS

Two examples are presented in this section. The first example is a different take on the classic *peg-in-hole* problem, which represents a very common insertion task in assemblies [26]. The second example generalizes this to a sequential peg-in-hole, also known as the *peg-in-maze* problem [5]. The purpose of the examples is to compare the action-entropy active sensing algorithm with the state-entropy active sensing algorithms in decision making for continuous problems. State-entropy active sensing [20], [29] is chosen for comparison because it is widely used and it is closely related to the proposed method.

The simulations were performed on a machine running Ubuntu 14.04 with Intel Core-i5 3.10 GHz processor and 8 GB memory. The entropy estimation algorithm and the simulations are written in C++. For brevity, the action-entropy and the state-entropy active sensing algorithms shall be denoted by AE and SE in this section. The two methods are compared with a baseline method that selects a sensing action uniformly at random, which is denoted by RN.

A. EXAMPLE: PEG-IN-HOLE PROBLEM

The peg-in-hole problem is an example of a common process in assemblies, where a peg is inserted into a slightly larger hole [23]. Traditionally, the peg-in-hole problem is solved by sequentially reducing the uncertainty of the position of the peg relative to the hole through manipulation and force feedback [26]. The peg-in-hole problem is chosen as an example because it has the desired characteristic where the state space can be decomposed into multiple subspaces depending on the stage of the task. First, when the peg is not in front of the hole, the peg is moved toward the hole. The subspace that is important here is the position of the peg relative to the hole's axis. Once the peg is in front of the hole, it is reoriented and inserted into the hole. In this case, the orientation and the distance along the hole's axis are more important. In this paper, a planar version of the peg-in-hole problem is considered.

The state of the peg at time t, denoted by $x_t \in \mathbb{R}^3$, is the x- and y-axis positions and the orientation of the peg with respect to the hole. Without loss of generality, the hole's center is placed at the origin, with the axis pointing along the positive y-axis. The initial belief is $b_0 = \mathcal{N}(\mu_x, \sigma_x)$. The motion model is $x_t = x_{t-1} + u_t \min(1, \alpha/\|u_t\|) + n_u$, where $x_t \in \mathbb{R}^3$ is the position and orientation of the peg at time t, and $u_t \in \mathbb{R}^3$ is the control action that translates and rotates the peg at time t, $n_u \sim \mathcal{N}(0, \sigma_u)$ is the motion noise, and α is the simulation step size. The sensing action, $v_t \in \{1, 2, 3\}$ determines which dimension of the state space is to be observed. In other words, let's $x_{i,t}$ denotes the i-th element of x_t, then the measurement model is $z_t = x_{v_t,t} + n_v$, where $n_v \sim \mathcal{N}(0, \sigma_v)$ is the measurement noise. The state-space policy for the peg-in-hole example is illustrated in Fig.4.

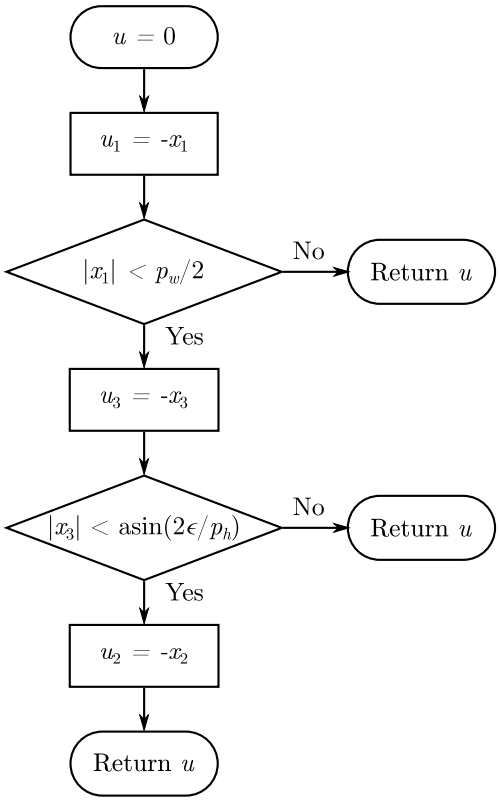

FIGURE 4. This flowchart illustrates the policy of the peg-in-hole problem, where p_w and p_h are the peg's width and height, and ϵ is the hole's tolerance. The policy first initializes u to zero. Then it moves the peg along the x-axis toward to hole. The first conditional statement checks whether or not the peg is within half of peg's width to the hole along the x-axis. If x-axis position is within that range, the policy reorients the peg. The second conditional statement is a heuristic that determines if the peg is too tilted or not. If the orientation is within that range, the policy lowers the peg into the hole.

The parameters used in the simulation is as follows. The peg's width and height are 1 and 2, respectively. The hole has a tolerance of 0.1. $\mu_x = [4, 2, 0]^T$, $\sigma_x^2 = \text{diag}([2, 1, 1.57])$, $\sigma_u^2 = \text{diag}([0.01, 0.01, 0.01])$, and $\sigma_v^2 = 0.001$, and $\alpha = 0.1$. The translation and rotation step size in the simulation is 0.1 and the simulation is repeated for 1,000 trials. Each trial terminates once the peg reaches the goal, or the number of steps exceeds 500 iterations. If the peg is not inserted within the iteration limit, the task ends in failure.

The results are shown in Table 3. Two metrics are used to evaluate different active sensing methods. The first metric is the ratio between the number of successful trials and the number of trials. AE has higher success ratios than the other two method across all sensing intervals. AE's success ratio does not drop as fast as the other two methods as the sensing interval is increased, which indicates that when sensing is more limited, AE is more effective in picking the sensing action that is important to the task. Also note that the number successful trials goes down when the sensing interval

TABLE 3. Peg-in-Hole simulation results.

Results	Sensing Interval			
	0	2	4	6
AE Success Ratio	0.957	0.914	0.876	0.847
SE Success Ratio	0.929	0.868	0.845	0.774
RN Success Ratio	0.862	0.851	0.791	0.735
AE Travel Distance	16.64 ± 0.61	20.95 ± 0.71	23.46 ± 0.79	25.03 ± 0.86
SE Travel Distance	16.31 ± 0.62	22.37 ± 0.78	25.06 ± 0.91	26.62 ± 0.93
RN Travel Distance	17.11 ± 0.62	23.57 ± 0.80	26.27 ± 0.91	29.10 ± 1.04

TABLE 4. Peg-in-Maze simulation results.

Results	Sensing Interval			
	8	10	12	14
AE Success Ratio	0.656	0.572	0.475	0.430
SE Success Ratio	0.573	0.532	0.448	0.393
RN Success Ratio	0.238	0.210	0.192	0.170
AE Travel Distance	105.12 ± 2.30	106.05 ± 2.48	106.33 ± 2.58	106.78 ± 2.79
SE Travel Distance	112.90 ± 2.66	113.68 ± 2.64	112.65 ± 2.85	111.79 ± 3.17
RN Travel Distance	96.93 ± 3.18	99.90 ± 3.42	98.99 ± 3.77	105.21 ± 4.22

increase, which is reasonable, since we have less information to complete the task.

The other metric considered here is the total distance the peg travels before reaching the goal. The average distances and the 95% confidence intervals are listed in Table 3. The travel distance can be thought of as the negative cost associated with moving the peg to the goal. While the distance generally goes up with the sensing interval, AE has the lowest distances in all cases. For the computational time, AE and SE use 0.139 s and 0.131 s, respectively, to calculate each sensing action, while RN's computational time is negligible.

B. EXAMPLE: PEG-IN-MAZE PROBLEM

This example is a generalization of the peg-in-hole problem, in which the peg has to be inserted through a sequence of holes. Real world examples of the peg-in-maze problem includes clutch mating and gear meshing [5].

The system the same as in the peg-in-hole problem. The main difference is that instead of one going one hole, the peg has to go though a series of holes to complete the task. The policy in Fig. 4 is used for each hole.

The parameters used in the simulation is as follows. The peg's width and height are 1 and 2, respectively. The holes are centered at (4, 6), (-4, 0), (0, -6), and they have tolerances of 0.1. $\mu_x = [0, 9, 0]^T$, $\sigma_x^2 = \text{diag}([2, 1, 1.57])$, $\sigma_u^2 = \text{diag}([0.01, 0.01, 0.01])$, and $\sigma_v^2 = 0.001$, and $\alpha = 0.1$. The configuration of the map is shown in Fig. 5. The translation and rotation step size in the simulation is 0.1 and the simulation is repeated for 1,000 trials. Each trial terminates once the peg reaches the goal, or the number of steps exceeds 1,500 iterations. If the peg is not inserted within the iteration limit, the task ends in failure. In this example we continue to increase the sensing interval from the previous example.

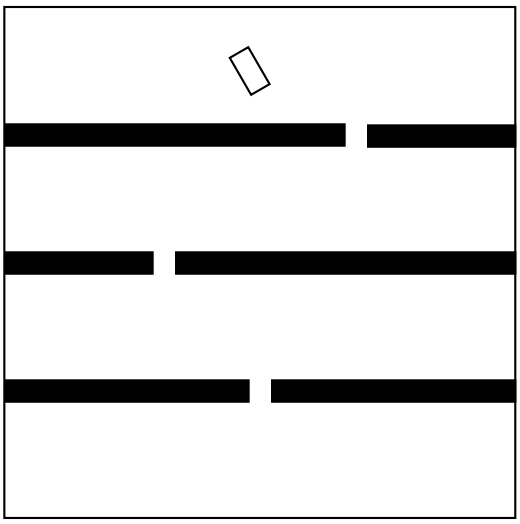

FIGURE 5. TThis figure shows the configuration of the peg-in-maze problem.

From the results in Table 4, the AE's success ratios are higher than the other two methods in all cases. AE's travel distances are also lower than SE's, but higher than RN's. Success ratios of the three methods decrease monotonically when the sensing interval goes up.

C. DISCUSSION

In both the peg-in-hole and peg-in-maze examples, AE performs better than SE and RN, while the computation time is comparable to the state-entropy method. The reason the computational time is comparable to the state-entropy method is that the active sensing calculation of AE is dominated by entropy estimation.

The reason why AE performs better in the two examples is due to the fact AE takes advantage of how the tasks can be decomposed into searching and insertion stages. When the peg is far away from the hole, AE only considers the uncertainty on the distance of the peg with respect to the hole's axis. Once the peg is close enough, AE considers the orientation. Similarly, when the orientation is sufficiently small, AE focuses on the insertion distance. In other words, AE minimizes the uncertainty of the action as required by the policy.

While it is possible to model some continuous decision making under uncertainty problems as continuous POMDPs [30], there are restrictions on the model, e.g., Gaussian mixture reward function, and the computation time is likely to be high as suggested by discrete POMDPs. The expected-value method is not included in the continuous examples because the expected-value method requires a value function. Continuous MDP is an active research topic that is beyond the scope of this paper.

The sensing actions in the two examples are observing the x-coordinate position, the y-coordinate position, and the orientation. The sensing actions in the two continuous examples are meant to illustrate the point that different subspaces are important during different stages of the tasks. In a more realistic setting, the measurement model can be replaced by a mobile stereo camera system, where depth perception along the axis perpendicular to the image plane is limited. So, we can obtain more information by moving the camera around and looking at different angles.

IX. DISCUSSION AND CONCLUSION

This paper presents action-entropy active sensing algorithms that are designed to be used in conjunction with a task planner to perform tasks in discrete and continuous environments. The proposed method incorporates the objective of the task into active sensing by selecting the sensing action that minimizes the ambiguity of the next task action. For discrete systems, the active sensing algorithm selects the sensing action that minimizes the conditional entropy of the next task action. The basic characteristics of the proposed algorithm is explored in two simulation examples. For continuous systems, belief propagation is approximated by the particle filter, and the active sensing algorithm minimizes an estimate of the conditional differential entropy based on the particles. Two examples demonstrate the performance of the proposed method in comparison with the state-entropy method. The proposed method not only has higher success rates, but it also completes the tasks with lowest cost (travel distance).

POMDP is a general framework that allows many sequential decision making under uncertainty problems to be defined in a consistent manner. Problems that cannot be decoupled into sensing and acting aspects can be modeled and solved more effectively as POMDPs. This is demonstrated in the symmetric corridor example in Section VII-B, where the states at the end of the hallway on the left is essential for localization. Consequently, the difference in the reward between the proposed method and DESPOT is larger in the symmetric corridor example than in the fork-in-the-road example.

The proposed solution to the decision making uncertainty problem is to combine a state-space policy with an active sensing algorithm. Motion uncertainty can be considered in calculating the state-space policy. In the discrete examples, we solve the state-space planning problems as MDPs, which inherently consider motion uncertainty. For the continuous examples, the policies are "funnel-like", in the sense that they funnel the states into smaller and smaller regions [26]. For the discrete examples, we have included POMDP results, which show that while the proposed method is suboptimal, they are much faster than POMDPs.

The proposed decoupled method borrows from the principle of separation in classical control, where we separate state estimation from control when the system can be described well enough locally using a linear approximation with Gaussian noise. This leads to the LQG controller, where we have the LQR controller designed in the state space, and EKF as the state estimator.

One possible limitation of the action entropy method is when the policy does not decompose the state space into subspaces. For example, when the policy is a simple linear feedback control law $\pi(x) = -x$, the action-entropy method becomes the state-entropy method. Or in general, if the policy is "too smooth", then the belief in the action space will be similar to belief in the state space, and there will be little benefit in using action entropy. Conversely, if the policy maps all of the states to the same action, which can happen in the case that the policy or the actuator saturates, then action entropy becomes negative infinity for all sensing actions. In this case, it might be useful to use a hybrid method, where the active sensing algorithm switches to state-entropy when the belief in the state space is mapped to one point in the action space.

Algorithm 2 assumes the availability of a policy that maps a state into an action. A very common family of planning methods in robotics is sampling-based methods. In practical implementations, the sampling-based methods are used in conjunction with a feedback controller. In this case, policy evaluation takes $O(d \log N)$ because we have to find the closest node for all the particles, and AE's computation complexity will be different from SE's by a constant factor, as discussed in Section VI-C. This planner-controller scheme provides a finite-parametric planner that could work well with the active sensing algorithm. A feedback planner that has a built-in feedback controller is also a good candidate for the state-space planner. Examples of feedback planners include LQR-tree [39], Sampling-Based Neighborhood Graph [43], etc. While it is possible to use an online planner as the policy, the online planner could make the action-entropy method much slower than the state-entropy method if online planning has to be done inside of the active sensing algorithm, i.e., in Line 8 of Algorithm 2. However, if the online planner generates a plan concurrently with the active sensing algorithm, i.e., a receding horizon plan is generated independently with

the active sensing algorithm, then the active sensing algorithm can potentially use the policy as if it is precomputed.

There are several possible directions for future work. In this work, we have studied the effect of the scarcity of information on the performance by increasing the interval between two sensing opportunities. It will be interesting to solve the inverse problem where we use the action entropy to measure the degree of uncertainty, and determine when a sensing action has to be performed. Physical robot implementation or more sophisticated simulations are two possible future directions. Kulick et al. propose a new objective function for information gathering based on cross entropy [21], which will be explored in future work. Another interesting future direction is to try different ways of selecting the task action from the belief in the state space. In this paper, we focus on the sensing aspect of decision make and select task actions based on the most likely state from the belief. Other possibilities include, the expected value of action value function as in QMDP [24], minimizing regret by select the action that maximized the minimum value, etc.

ACKNOWLEDGMENT

This article was presented in part at the IEEE/RSJ International Conference on Intelligence Robots and Systems, Daejeon, South Korea, 2016.

REFERENCES

[1] A.-A. Agha-Mohammadi, S. Chakravorty, and N. M. Amato, "FIRM: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements," *Int. J. Robot. Res.*, vol. 33, no. 2, pp. 268–304, 2014.

[2] J. Ajgl and M. Šimandl, "Differential entropy estimation by particles," *IFAC Proc. Volumes*, vol. 44, no. 1, pp. 11991–11996, 2011.

[3] M. Araya, O. Buffet, V. Thomas, and F. Charpillet, "A POMDP extension with belief-dependent rewards," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 23. Red Hook, NY, USA: Curran Associates, 2010, pp. 64–72.

[4] J. Beirlant, E. J. Dudewicz, L. Györfi, and E. C. Van der Meulen, "Nonparametric entropy estimation: An overview," *Int. J. Math. Stat. Sci.*, vol. 6, no. 1, pp. 17–39, 1997.

[5] M. S. Branicky and S. R. Chhatpar, "A computational framework for the simulation, verification, and synthesis of force-guided robotic assembly strategies," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 3, Oct./Nov. 2001, pp. 1471–1476.

[6] S. Chakravorty and R. S. Erwin, "Information space receding horizon control," in *Proc. IEEE Symp. Adapt. Dyn. Program. Reinforcement Learn. (ADPRL)*, Apr. 2011, pp. 302–309.

[7] B. Charrow, V. Kumar, and N. Michael, "Approximate representations for multi-robot control policies that maximize mutual information," in *Robotics: Science and Systems*. Berlin, Germany, Jun. 2013.

[8] S. R. Chhatpar and M. S. Branicky, "Particle filtering for localization in robotic assemblies with position uncertainty," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Aug. 2005, pp. 3610–3617.

[9] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementation*. Cambridge, MA, USA: A Bradford Book, May 2005.

[10] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Hoboken, NJ, USA: Wiley, 1991.

[11] D. Fox, W. Burgard, and S. Thrun, "Active Markov localization for mobile robots," *Robot. Auto. Syst.*, vol. 25, nos. 3–4, pp. 195–207, Nov. 1998.

[12] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Trans. Math. Softw.*, vol. 3, no. 3, pp. 209–226, 1997.

[13] M. N. Goria, N. N. Leonenko, V. V. Mergel, and P. L. N. Inverardi, "A new class of random vector entropy estimators and its applications in testing statistical hypotheses," *J. Nonparametric Statist.*, vol. 17, no. 3, pp. 277–297, 2005.

[14] T. Greigarn and M. C. Çavuşoğlu, "Active sensing for continuous state and action spaces via task-action entropy minimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2016, pp. 4678–4684.

[15] P. Guerrero, J. Ruiz-Del-Solar, M. Romero, and S. Angulo, "Task-oriented probabilistic active vision," *Int. J. Humanoid Robot.*, vol. 7, no. 3, pp. 451–476, 2010.

[16] M. Hauskrecht, "Value-function approximations for partially observable Markov decision processes," *J. Artif. Intell. Res.*, vol. 13, pp. 33–94, Aug. 2000.

[17] G. M. Hoffman and C. J. Tomlin, "Mobile sensor network control using mutual information methods and particle filters," *IEEE Trans. Autom. Control*, vol. 55, no. 1, pp. 32–47, Jan. 2010.

[18] S. Javdani, Y. Chen, A. Karbasi, A. Krause, J. A. Bagnell, and S. Srinivasa, "Near optimal Bayesian active learning for decision making," in *Proc. 17th Int. Conf. Artif. Intell. Statist.*, 2014, pp. 430–438.

[19] L. F. Kozachenko and N. N. Leonenko, "Sample estimate of the entropy of a random vector," *Problemy Peredachi Inform.*, vol. 23, no. 2, pp. 9–16, 1987.

[20] C. Kreucher, K. Kastella, and A. O. Hero, III, "Sensor management using an active sensing approach," *Signal Process.*, vol. 85, no. 3, pp. 607–624, 2005.

[21] J. Kulick, R. Lieck, and M. Toussaint, "The advantage of cross entropy over entropy in iterative information gathering," Sep. 2014, *arXiv:1409.7552*. [Online]. Available: https://arxiv.org/abs/1409.7552

[22] C. Kwok and D. Fox, "Reinforcement learning for sensing strategies," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, vol. 4, Sep./Oct. 2004, pp. 3158–3163.

[23] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2006.

[24] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling, "Learning policies for partially observable environments: Scaling up," in *Proc. 12th Int. Conf. Int. Conf. Mach. Learn.*, 1995, pp. 362–370.

[25] J. M. Manyika and H. F. Durrant-Whyte, "Information-theoretic approach to management in decentralized data fusion," *Proc. SPIE*, vol. 1828, Nov. 1992, pp. 202–213.

[26] M. Mason, "The mechanics of manipulation," in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 2, Mar. 1985, pp. 544–548.

[27] R. M. Mnatsakanov, N. Misra, S. Li, and E. J. Harner, "K_n-nearest neighbor estimators of entropy," *Math. Methods Statist.*, vol. 17, no. 3, pp. 261–277, 2008.

[28] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: An anytime algorithm for POMDPs," in *Proc. 18th Int. Joint Conf. Artif. Intell.* San Francisco, CA, USA: Morgan Kaufmann Publishers, 2003, pp. 1025–1030.

[29] J. M. Porta, B. Terwijn, and B. Krose, "Efficient entropy-based action selection for appearance-based robot localization," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, Sep. 2003, pp. 2842–2847.

[30] J. M. Porta, N. Vlassis, M. T. J. Spaan, and P. Poupart, "Point-based value iteration for continuous POMDPs," *J. Mach. Learn. Res.*, vol. 7, pp. 2329–2367, Nov. 2006.

[31] S. Prentice and N. Roy, "The belief roadmap: Efficient planning in belief space by factoring the covariance," *Int. J. Robot. Res.*, vol. 28, nos. 11–12, pp. 1448–1465, Jul. 2009.

[32] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ, USA: Prentice-Hall, 2009.

[33] A. Ryan and J. K. Hedrick, "Particle filter based information-theoretic active sensing," *Robot. Auto. Syst.*, vol. 58, pp. 574–584, May 2010.

[34] Y. Satsangi, S. Whiteson, and F. A. Oliehoek, "Exploiting submodular value functions for faster dynamic sensor selection," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 3356–3363.

[35] W. W. Schmaedeke, "Information-based sensor management," *Proc. SPIE*, vol. 1955, Sep. 1993, pp. 156–164.

[36] H. Singh, N. Misra, V. Hnizdo, A. Fedorowicz, and E. Demchuk, "Nearest neighbor estimates of entropy," *Amer. J. Math. Manage. Sci.*, vol. 23, nos. 3–4, pp. 301–321, 2003.

[37] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "DESPOT: Online POMDP planning with regularization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 26, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, 2013, pp. 1772–1780.

[38] M. T. J. Spaan, T. S. Veiga, and P. U. Lima, "Decision-theoretic planning under uncertainty with information rewards for active cooperative perception," *Auton. Agents Multi-Agent Syst.*, vol. 29, no. 6, pp. 1157–1185, Nov. 2015.
[39] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "LQR-Trees: Feedback motion planning via sums-of-squares verification," *Int. J. Robot. Res.*, vol. 29, no. 8, pp. 1038–1052, 2010.
[40] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2005.
[41] J. van den Berg, S. Patil, and R. Alterovitz, "Motion planning under uncertainty using iterative local optimization in belief space," *Int. J. Robot. Res.*, vol. 31, no. 11, pp. 1263–1278, 2012.
[42] M. Wang, S. Canu, and R. Dearden, "Improving robot plans for information gathering tasks through execution monitoring," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 5285–5291.
[43] L. Yang and S. M. LaValle, "The sampling-based neighborhood graph: An approach to computing and executing feedback motion strategies," *IEEE Trans. Robot. Autom.*, vol. 20, no. 3, pp. 419–432, Jun. 2004.

TIPAKORN GREIGARN (S'14) received the B.S. degree in electrical engineering from Chulalongkorn University, Bangkok, Thailand, in 2008, and the M.S. and Ph.D. degrees in systems and control engineering from Case Western University, in 2011 and 2018, respectively. He is currently a Senior Robotics Engineer with Transenterix, Inc. His research interests include robotics, systems and control, and machine learning.

MICHAEL S. BRANICKY (M'87–SM'01–F'16) received the Sc.D. degree from the Massachusetts Institute of Technology. He is currently the Dean of engineering and a Professor of electrical engineering and computer science with The University of Kansas. His research interests include control systems, robotics, hybrid systems, intelligent control, and learning.

M. CENK ÇAVUŞOĞLU (S'93–M'01–SM'06) received the B.S. degree in electrical and electronic engineering from Middle East Technical University, Ankara, Turkey, in 1995, and the M.S. and Ph.D. degrees in electrical engineering and computer sciences from the University of California, at Berkeley, in 1997 and 2000, respectively.

He was a Visiting Researcher with the INRIA Rhones-Alpes Research Center, Grenoble, France, in 1998, a Postdoctoral Researcher and a Lecturer with the University of California at Berkeley, from 2000 to 2002, and a Visiting Associate Professor with Bilkent University, Ankara, from 2009 to 2010. He is currently a Professor of electrical engineering and computer science, biomedical engineering, and mechanical and aerospace engineering with Case Western Reserve University, Cleveland, OH, USA. His research interests include robotics, systems and control theory, and human–machine interfaces, with an emphasis on medical robotics, haptics, virtual environments, surgical simulation, and bio-system modeling and simulation. His current research involves the applications of robotics and control engineering to biomedical and biologically inspired engineered systems. He has served as an Associate Editor for the IEEE TRANSACTIONS ON ROBOTICS and a Technical Editor for the IEEE/ASME TRANSACTIONS ON MECHATRONICS.