Author Manuscript

Author Manuscript

Author Manuscript

Author Manuscript

HHS Public Access

Author manuscript

Rep U S. Author manuscript; available in PMC 2017 December 07.

Published in final edited form as:

Rep U S. 2016 October ; 2016: 4678–4684. doi:10.1109/IROS.2016.7759688.

Active Sensing for Continuous State and Action Spaces via Task-Action Entropy Minimization

Tipakorn Greigarn and **M. Cenk Çavuşoğlu**

Department of Electrical Engineering and Computer Science, Case Western Reserve University, Cleveland, OH

Abstract

In this paper, a new task-oriented active-sensing method is presented. Most active sensing methods choose sensing actions that minimize the uncertainty of the state according to some information-theoretic measure. While this is reasonable for most applications, minimizing state uncertainty may not be most relevant when the state information is used to perform a task. This is because the uncertainty in some subspace of the state space could have more impact on the performance of the task than the others at a given time. The active-sensing method presented in this paper takes the task into account when selecting sensing actions by minimizing the uncertainty in future task action.

I. Introduction

Sequential decision making under uncertainty is difficult even for discrete problems due to the dual curses of dimensionality and history [1, 2]. However, most robotics problems are continuous in nature, and the complexity of sequential decision making increases tremendously when the state space is continuous. On the other hand, motion planning for systems with continuous state spaces has matured in recent years with the invention of various planners [3]. In real-world applications, these state-space planners and their variants are often used in conjunction with feedback control which employs sensing to obtain measurements necessary for uncertainty reduction.

In some applications, there are actions that directly affect sensing but do not change the state of the system. These actions are called sensing actions. For example, while driving, looking in different directions provides different information about the surrounding without affecting the state of the car and the environment. In image-guided robotic surgical intervention, taking different image slices of a volume give you different information without affecting the state of the robot and the environment. In such cases, the sensing actions can be utilized to obtain the most useful information. The act of choosing the best sensing action is known as *active sensing*.

Typically, sensing actions are chosen to reduce state uncertainty by optimizing some information theoretic measure of the belief over the state. When the information about the state is used to perform a task, there could be some subspace of the state space that has more impact to the task at a given moment. 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.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

Suppose the driver wants to merge right, the positions of the cars on the right is determined whether or not they can merge. Therefore looking to the right is the logical sensing action to take. However, from state uncertainty perspective, the uncertainty of the positions of the cars on the left and right are equivalent. Since only the position of the car to the right affects our action of changing lane, only the uncertainty of the position of the car to the right matters. If we try to reduce the uncertainty of whether the driver can merge right or not, we will arrive at the same conclusion that the driver should look to the right.

This paper presents a new active-sensing method that is designed to be used in conjunction with a state-space planner to solve sequential decision making under uncertainty problems with continuous state and action spaces. The proposed active-sensing method takes the task at hand into account by choosing sensing actions that minimize the entropy of the future task actions. This new method shall be referred to as *action-entropy active sensing*.

In this paper, particle filters are used to model belief propagation in continuous state spaces. A belief over a task-action space is obtained by mapping the belief over the state space through a policy provided by the task planner. Entropy of the future task action is estimated from the particles using a nearest-neighbor method described in [4]. Then the sensing action that minimizes the estimated task-action entropy is chosen.

The rest of the paper is organized as follows. Related work is reviewed in Section II. Problem formulation is presented in Section III. The proposed active-sensing method is described in Section IV. Simulation results are presented in Section V. The results are discussed in Section VI. Conclusions are given in Section VII.

II. Related Work

Motion planning for systems with continuous state spaces has been one of the most important subjects in robotics research. Earlier planning methods include potential fields, roadmaps, and cell decompositions [3]. Sampling-based planners are also widely used in robotics. Kavraki et al. [5] presents the Probabilistic Roadmap (PRM) algorithm. LaValle and Kuffner [6] presents the Rapidly-exploring Random Tree algorithm. Many PRM and RRT variants are conceived thereafter [7]. In this work, we assume that a planner that solves the planning in the state space is available. The active-sensing algorithm’s objective is to reduce the uncertainty such that the planner can perform the task effectively.

Information-theoretic measures have been used in sensor selection and data fusion problems. Iterative data fusion using Kalman filter is presented by Manyika and Durrant-Whyte [8]. Target assignment via information gain maximization is presented by Schmaedeke [9]. Appearance-based localization of a robot equipped with a pan-and-tilt camera using entropy minimization is presented by Porta et al. [10]. Sensor data selection using mutual information maximization is presented by Denzler and Brown [11].

There are some prior work on active-sensing methods that are designed to provide information related to a given task. Kwok and Fox [12] uses reinforcement learning for sensing action selection for soccer robots. Long-term value of a sensing action is learned in the training phrase. The robot then performs the best action according the learned action-

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

value function. Guerrero [13] presents an active-sensing method that selects sensing actions that maximize the expected value, where, a value function over the states is calculated via value iteration.

Decision making under uncertainty can be also modeled as Partially-Observable Markov Decision Processes (POMDPs) [14]. While most POMDP algorithms are designed for discrete problems, there also exists algorithms for continuous POMDPs. For example, Porta et al. [15] presents point-based value iteration for continuous POMDPs, where Gaussian Mixture is used to model belief and reward function. Bai et al. [16] presents Monte Carlo value iteration for continuous-state POMDPs with discrete action and observation spaces.

The proposed method is different to 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. In a sense, the proposed active-sensing method functions as a state estimator while the state-space planner functions as the controller. The active-sensing algorithm chooses a sensing action that minimize the uncertainty in the next task actions according to the policy provided by the state-space planner. Moreover, the proposed method is different from the other sensor selection and data fusion methods because it is designed to be used in the context of planning under uncertainty. So, the ultimate goal is not to obtain information but to perform the task well. Finally, the proposed method is different from the work in [12] and [13] 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.

III. Problem Formulation

Let $\mathcal{X}$ and $\mathcal{Z}$ denote the state space and the measurement space respectively. 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*. The state-transition model, $p(x'|u, x)$, is a probability density function (PDF) of the next state $x' \in \mathcal{X}$ given that a task action $u \in \mathcal{U}$ is performed at a state $x \in \mathcal{X}$. The measurement model, $p(z|v, x)$, is a PDF of the measurement $z \in \mathcal{Z}$ given that a sensing action $v \in \mathcal{V}$ is performed at a state $x \in \mathcal{X}$.

The information the robot has about the state of the environment is represented by its belief. A belief is a PDF defined over the state space given all past actions and measurements, i.e.,

$$b(x_t)=p(x_t|b_0, u_{1:t}, v_{1:t}, z_{1:t}), \forall x_t \in \mathcal{X}, \quad (1)$$

where $b_0 = p(x_0)$ is the initial belief. For brevity, the time subscript t will be dropped where it would not cause confusion. When the robot interacts with the environment, a predicted

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

belief, denoted by $\bar{b}$, is calculated by propagating the belief according to the state-transition model as follows,

$$\bar{b}(x') = \int_{\mathscr{X}} p(x'|u, x) b(x) dx. \quad (2)$$

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

$$b(x) = \eta p(z|v, x) \bar{b}(x). \quad (3)$$

where $\eta = 1/\int p(z|v, x)\bar{b}(x)dx$ is the normalizer of the expression on the right-hand side.

In this work we assume 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. Fundamentally, the only requirement of the task planner is it generates a policy $\pi : \mathscr{X} \rightarrow \mathscr{U}$.

The proposed active-sensing method minimizes the entropy of the future task action. The distribution of the future task action is obtained by mapping the belief over the state space through the policy π. This is the focus of the next section.

IV. Active Sensing

The action-entropy active-sensing method is presented in this section. Some background in differential entropy and entropy estimation is reviewed in Section IV-A. The proposed active-sensing algorithm is presented in Section IV-B.

A. Differential Entropy Estimation

Let X be a continuous random variable with PDF $p(x)$ and the range of X is $\mathscr{X}$. Differential Entropy is defined as follows [17],

$$h(X) = -\int_{\mathscr{X}} p(x) \log p(x) dx. \quad (4)$$

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

$$h(X|Y) = -\int_{\mathscr{Y}} \int_{\mathscr{X}} p(x, y) \log p(x|y) dx dy. \quad (5)$$

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

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,

$$\begin{aligned} h(X|Y) &= -\int_{\mathcal{Y}} \int_{\mathcal{X}} p(x|y)p(y)\log p(x|y)\,dxdy, \\ &= -\int_{\mathcal{Y}} p(y) \int_{\mathcal{X}} p(x|y)\log p(x|y)\,dxdy, \\ &= \int_{\mathcal{Y}} p(y)h(X|Y=y)dy. \end{aligned} \tag{6}$$

In this work, belief propagation over continuous state spaces is approximated using particle filters. 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,

$$b_{\mathcal{P}}(x) = \sum_{i=1}^{N} w^i \delta(x - x^i), \tag{7}$$

where δ is the Dirac delta function.

Differential entropy of a belief approximated by a set of particles is estimated using the nearest neighbor method described in [4]. The estimator uses the distance to the k-th nearest neighbor and the weights of the particles within the distance to approximate the PDF locally at each particle. The estimated entropy is given by,

$$\hat{h}_{\mathcal{P}}(X) = -\sum_{i=1}^{N} \frac{\sum_{j \in \mathcal{N}_k^i} w^j}{k} \log \frac{\sum_{j \in \mathcal{N}_k^i} w^j}{|B_n(d_k^i)|} + \log k - \Psi(k), \tag{8}$$

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. Sensing Action Selection

The action-entropy active-sensing method is presented in this section. The algorithm is designed to collaborate with a task planner to provide a solution to decision making under uncertainty. Fig. 1 illustrates a planner for decision making under uncertainty that contains the action-entropy active-sensing algorithm and a task planner. In this setting, the task planner solves a given planning problem in the state space and returns a policy, $\pi : \mathcal{X} \rightarrow \mathcal{U}$, while the active-sensing module handles active sensing. The coordinator which maintains a belief over the states and interacts with the environment.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

When the planner is initialized, the task planner solves the planning problem in the state space for a policy, which is passed on to the active-sensing module. The coordinator passes its internal belief to the active-sensing module. Then the active-sensing module maps the belief particles through the policy to obtain task-action particles, calculates the task-action entropy, and returns the sensing action that minimizes the entropy of the task action. The coordinator obtains the sensing action, performs it, receives a measurement, and updates its belief according to the measurement update in (3). Next, the coordinator selects the task action 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 (2).

In this work, sensing actions are used to minimize the uncertainty of the future task action via entropy minimization. Since the measurement is not known when the sensing action is being chosen, the entropy is conditioned on the measurement. The sensing action is the minimizer of the entropy, i.e.,

$$v=\underset{v\in\mathcal{V}}{\operatorname{argmin}}\, h(U|Z;v,\bar{b}), \tag{9}$$

where U denotes the random variable of the future action and Z denotes the random variable of the measurement. The belief $\bar{b}$ is the prior belief the coordinator has before a measurement is taken. The conditional entropy is calculated for each sensing action v, and the sensing action that minimizes it is chosen. The sensing-action space is assumed to be discrete. If it is continuous, sampling or discretization can be employed.

According to (6), the entropy of the future task action conditioned on the measurement is given by

$$h(U|Z;v,\bar{b})=\int_{\mathcal{Z}} p(z|v,\bar{b})h(U|z,v,\bar{b})dz. \tag{10}$$

The first term in the integration is the PDF of the measurements z given a sensing action v and belief $\bar{b}$. The PDF $p(z|v, \bar{b})$ can be written as

$$\begin{aligned} p(z|v,\bar{b})&=\int_{\mathcal{X}} p(z|v,\bar{b},x)p(x|v,\bar{b})dx,\\ &=\int_{\mathcal{X}} p(z|v,x)\bar{b}(x)dx. \end{aligned} \tag{11}$$

The first line in (11) follows from the law of total probability. From the first to the second line, $\bar{b}$ disappears from $p(z|v, \bar{b}, x)$ because z is independent of $\bar{b}$ once x is given. So, the $p(z|v, \bar{b}, x)$ becomes $p(z|v, x)$ which simply is the measurement model. $p(x|v,\bar{b})$ in the first line becomes $\bar{b}$ in the second line because prior to receiving a measurement, $\bar{b}$ contains all the information about the state.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

Algorithm 1

Minimum Task-Action Entropy Active Sensing

1: **procedure** GetSensingAction(π,$\hat{\mathcal{P}}_x$)
2: **for all** $v \in \mathcal{V}$ **do**
3: $h_v = 0$
4: **for** $i = 1, \ldots, M$ **do**
5: sample x from $\hat{\mathcal{P}}_x$
6: sample z from $p(z|v, x)$
7: $\mathscr{P}'_x = \mathrm{MeasurementUpdate}(\mathscr{P}_x, v, z)$
8: $\mathscr{P}_u = \pi(\mathscr{P}'_x)$
9: h_v += $\wedge h_{\hat{\mathcal{P}}_u}(u)$
10: **end for**
11: **end for**
12: **return** $\mathrm{argmin}_v\, h_v$
13: **end procedure**

The remaining term in (10) is the entropy $h(U|z, v, \bar{b})$, which is estimated as follows. First, the belief particles $\bar{b}$ are updated using v and z according to the measurement update in (3). Then the task-action particles are obtained by mapping the belief particles through the policy π. Finally, the entropy of the task action is estimated from the task-action particles using the nearest neighbor method described in (8).

The active-sensing algorithm is shown in Algorithm 1. The inputs include the policy π and the particles $\hat{\mathcal{P}}_x$, which represent the current belief of the coordinator, $\bar{b}$. The algorithm estimates the task-action entropy of each sensing action and returns the sensing action with minimum entropy. The integration in (10) is approximated using Monte Carlo simulation in the inner loop, where the integral in (11) is replaced by sampling in line 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, and the entropy of the task-action particles are estimated using (8). Then the sensing action that results in the lowest conditional entropy is returned.

C. Analysis

Let the number of available sensing actions be denoted by n_v. Monte Carlo simulation is performed M times for each sensing action, so Line 5 to 9 in Algorithm 1 are executed $n_v M$ times. 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 algorithms that preprocess the data so that nearest neighbor query can be made more efficiently. A widely used nearest algorithm is k-d tree, which has construction time complexity $O(dN \log N)$, where d is the dimension of particles, and query time complexity $O(\log N)$ under some assumptions [18]. If

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

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 $O(dn_v MN \log N)$.

V. Simulation Results

Two examples are presented in this section. The purpose of the examples is to compare the action-entropy active-sensing algorithm with the state-entropy active-sensing algorithms in a simple setting. State-entropy active sensing [11, 10] is chosen for comparison because it is widely used and it is closely related to the proposed method. In the first example, a robot has to reach a moving goal without colliding with stationary obstacles, while in the second example, a robot has to go reach a stationary goal while avoiding moving obstacles. In both examples, the robots can choose to observe the position of itself or one of the moving objects at a time. Since each object effects the robot differently at different stages of the task, the robot's ability to choose the object that is most relevant will be important. The two examples capture the kind of problems where the task actions depend on the states of the robot as well as the environment, while the sensing capability is limited. Therefore, each sensing action should give us the most relevant information.

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 Python. For brevity, the action-entropy and the state-entropy active-sensing algorithms shall be denoted by AE and SE in this section.

A. Example: Moving Goal

In this example, a robot tries to reach a moving goal on the other side of the map. This example is analogus to problems encountered in robotic surgical intervention where the target is often embedded in deformable tissues. So, the exact location of the target has to be obtained via sensing.

The setup is depicted in Fig. 2. The robot is on the left, while the goal is on the right. There are two stationary obstacles in the middle. The robot navigates using a potential function [3], and it can only sense the goal's or its own location at a time. When the robot is close to the obstacles, the position of the goal is less important because the repulsive forces from the obstacles dominate the policy. On the other hand, when the robot is sufficiently far away from the obstacles, the position of the goal is as important as the robot's position because both of them are required to calculate the attractive force, which dominates the policy in this case. The uncertainty of the robot's or the goal's positions are indistinguishable for state entropy. However, since action entropy measures the uncertainty of the task action, the effect of the policy and the robot's position on sensing action selection is apparent.

In this example, the state space is $[0, 1]^4$. The state is a vector containing the position of the robot and the goal, i.e., $x=[x_{\text{robot}}^T, x_{\text{goal}}^T]^T$, where $x_{\text{robot}}, x_{\text{goal}} \in \mathbb{R}^2$. The initial belief is Gaussian with mean $[0.1, 0.5, 0.9, 0.5]^T$ and covariance matrix $0.001 I_4$, where I_n denotes an n-by-n identity matrix. The initial position of the robot and the goal are $[0.1, 0.5]^T$ and $[0.9, 0.5]^T$, respectively. The task-action space is $\mathcal{U} = \{u \in \mathbb{R}^2 : \|u\| \leq 0.1\}$. The state-transition

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

and the measurement models are also Gaussian. The measurement model's covariance matrix is $0.02I_2$. In this example, we increase the difficulty of the problem by making the transition model less certain. In Trial 1, 2, 3, and 4, the state-transition model's covariance matrices are $0.001I_4$, $0.002I_4$, $0.005I_4$, and $0.01I_4$ respectively. The particle filter uses 100 particles to model belief propagation. The reward for reaching the goal is +10, while hitting the obstacles has -10 reward, and each move action has negative distance reward.

The average cumulative rewards and run time measurements from 1000 simulations are shown in Table I. The rewards of AE and SE are listed in the second and the third columns. The times AE and SE take to select a sensing action are listed in the forth and fifth columns. In all trials, the AE performs better than SE. Moreover, AE takes less time to select a sensing action compared to SE. This is because the action space's dimension is 2 while the state space's dimension is 4.

B. Example: Moving Obstacles

This example represents decision making problems with dynamic environment. The robot has to reach the goal without colliding with the obstacles that move around randomly. The setup is depicted in Fig. 3. In this example, the effect of the number of moving obstacles on the performance is studied. There is one obstacle in Trial 1, and one obstacle is added in each subsequent trial. The robot navigates using a potential function, and it can observe either its own position or one of the obstacles' position at a time.

In this example, the state space is $[0, 1]^{2(n+1)}$, where n is the number of obstacles in each trail. The state is a vector containing the position of the robot and the obstacles, i.e., $x=[x_{\text{robot}}^T, x_{\text{obs1}}^T, \cdots, x_{\text{obs}n}^T]^T$, where x_{robot}, x_{obs1}, ⋯, $x_{\text{obs}n} \in \mathbb{R}^2$. The initial belief is Gaussian. The mean and the covariance matrix of the initial belief corresponding to the robot are $[0.1, 0.5]^T$ and $0.001I_2$, respectively. The obstacles 1 through 4 have initial beliefs with mean $[0.75, 0.75]^T$, $[0.25, 0.25]^T$, $[0.25, 0.75]^T$, and $[0.75, 0.25]^T$, respectively. All obstacles' initial beliefs have covariance matrix $0.001I_2$. The state-transition and measurement models are also Gaussian. The covariance matrix of the measurement model is $0.001I_2$. The covariance matrix of the state-transition model is $0.001I_{2(n+1)}$. The particle filter uses 200 particles to model belief propagation in the first trial, and 100 particles are added in each subsequent trials. The reward for reaching the goal is +10, while hitting the obstacles has -10 reward, and each move action has negative distance reward.

Simulation results are shown in Table II, where each trial is repeated 1000 times. The rewards of AE and SE are listed in the second and the third columns. The times AE and SE take to select a sensing action are listed in the forth and fifth columns. The action-entropy active-sensing algorithm has higher average reward in all trails. Moreover, run time for the action-entropy method does not grow quite as fast as the state-entropy method. This is because the action-entropy method maps state particles to task-action particles, which has lower dimension, before performing entropy estimation.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

VI. Discussion

The action-entropy algorithm has higher rewards than the state-entropy algorithm in both examples. The action-entropy method is also significantly faster when the dimension of the state space increases. This is because the action-entropy method maps the state particles to task-action particles, which have lower dimension. Entropy estimation is faster in lower dimensions, so the overall algorithm is faster. Note that this is true for the potential field method, and state-feedback control in general, since mapping a state to a task action takes $O(d)$, which is less than $O(d \log N)$ required by a nearest neighbor search. If a sampling-based planner is employed, nearest-neighbor searches may be required for finding the closest node to a particle. In this case, time complexity of the action-entropy method is of the same order as the state-entropy method. Collaborations between the action-entropy active-sensing method and other state-space planner is a part of future work.

Scipy's implementation of k-d tree is chosen for nearest-neighbor search in entropy estimation because it is implemented purely in Python. This makes speed comparison possible since the rest of the algorithms are written in Python. Entropy estimation can be made faster by using faster nearest-neighbor library such as FLANN [19]. A comparison between Scipy and FLANN in entropy estimation is shown in Table III. Therefore, the overall speed of the algorithm is expected to be much faster when implemented in C++.

It should be noted that Gaussian distributions are used in the simulations for simplicity. Since the active sensing algorithm uses particle filters to represent beliefs and does not rely on a parametric filter, the method will work with other distributions.

VII. Conclusions

This paper presents the action-entropy active-sensing method that is designed to be used in conjunction with a task planner in continuous state and action spaces. The active-sensing method reduces the uncertainty of the future task action by choosing the sensing action that minimizes the uncertainty of the future task action. In this work, particle filters are used to model belief propagation in continuous state spaces. The entropy is estimated from the particles using the k-nearest neighbor method. Two examples are presented demonstrating how the action-entropy active-sensing method can be used in decision making under uncertainty problems.

Acknowledgments

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

References

1. Kaelbling LP, Littman ML, Cassandra AR. Planning and acting in partially observable stochastic domains. Artificial Intelligence. May; 1998 101(1–2):99–134.
2. Pineau, J., Gordon, G., Thrun, S. Point-based Value Iteration: An Anytime Algorithm for POMDPs. Proceedings of the 18th International Joint Conference on Artificial Intelligence, ser. IJCAI'03; San Francisco, CA, USA: Morgan Kaufmann Publishers Inc; 2003. p. 1025-1030.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

3. Choset, H., Lynch, KM., Hutchinson, S., Kantor, GA., Burgard, W., Kavraki, LE., Thrun, S. Principles of Robot Motion: Theory, Algorithms, and Implementations (Intelligent Robotics and Autonomous Agents series). A Bradford Book; May. 2005
4. Ajgl, J., Šimandl, M. Differential entropy estimation by particles. 18th IFAC World Congress; 2011; p. 11 991-11 996.
5. Kavraki LE, Svestka P, Latombe JC, Overmars MH. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation. 1996; 12:566–580.
6. LaValle, SM., Kuffner, JJ. Randomized Kinodynamic Planning. Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA'99); 1999; p. 1-7.
7. Elbanhawi M, Simic M. Sampling-Based Robot Motion Planning: A Review. IEEE Access. 2014; 2:56–77.
8. Manyika, JM., Durrant-Whyte, HF. Applications in Optical Science and Engineering. International Society for Optics and Photonics; 1992. Information-theoretic approach to management in decentralized data fusion; p. 202-213.
9. Schmaedeke, WW. Optical Engineering and Photonics in Aerospace Sensing. International Society for Optics and Photonics; 1993. Information-based sensor management; p. 156-164.
10. Porta, J., Terwijn, B., Krose, B. Efficient entropy-based action selection for appearance-based robot localization. 2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422); 2003.
11. Denzler J, Brown CM. Information theoretic sensor data selection for active object recognition and state estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence. 24(2):2002.
12. Kwok, C., Fox, D. Reinforcement learning for sensing strategies," in Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on; 2004.
13. Guerrero P, Ruiz-Del-Solar J, Romero M, Angulo S. Task-Oriented Probabilistic Active Vision. International Journal of Humanoid Robotics. 2010; 7(03):451–476.
14. Thrun, S., Burgard, W., Fox, D. Intelligent robotics and autonomous agents. The MIT Press; Aug. 2005 Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series), ser.
15. Porta JM, Vlassis N, Spaan MTJ, Poupart P. Point-Based Value Iteration for Continuous POMDPs. The Journal of Machine Learning Research. Dec.2006 7:2329–2367.
16. Bai, H., Hsu, D., Lee, W., Ngo, VA. Monte Carlo Value Iteration for Continuous-State POMDPs. In: Hsu, D.Isler, V.Latombe, J-C., Lin, MC., editors. Algorithmic Foundations of Robotics IX, ser. Springer Tracts in Advanced Robotics. Vol. 68. Berlin, Heidelberg: Springer Berlin Heidelberg; 2011. p. 175-191.ch. 11
17. Cover, TM., Thomas, JA. Elements of Information Theory. 1991.
18. Freidman JH, Bentley JL, Finkel RA. An algorithm for finding best matches in logarithmic expected time. ACM Transactions on Mathematical Software. 1977; 3(3):209–226.
19. Muja M, Lowe DG. Scalable Nearest Neighbour Algorithms for High Dimensional Data. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2014; 36(11):2227–2240. [PubMed: 26353063]

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

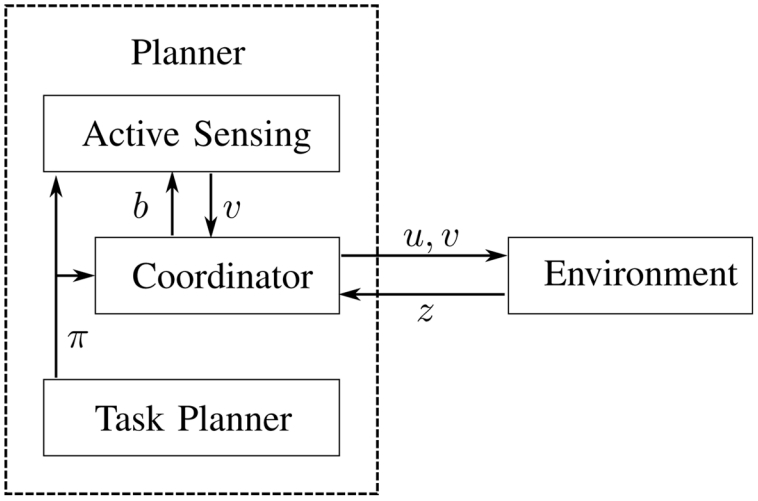

Fig. 1.

The planner is divided into three submodules. The sensing module handles active sensing, while the task planner solves the planning problem and provides a policy for the other parts. The coordinator maintains the internal belief and interacts with the environment.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

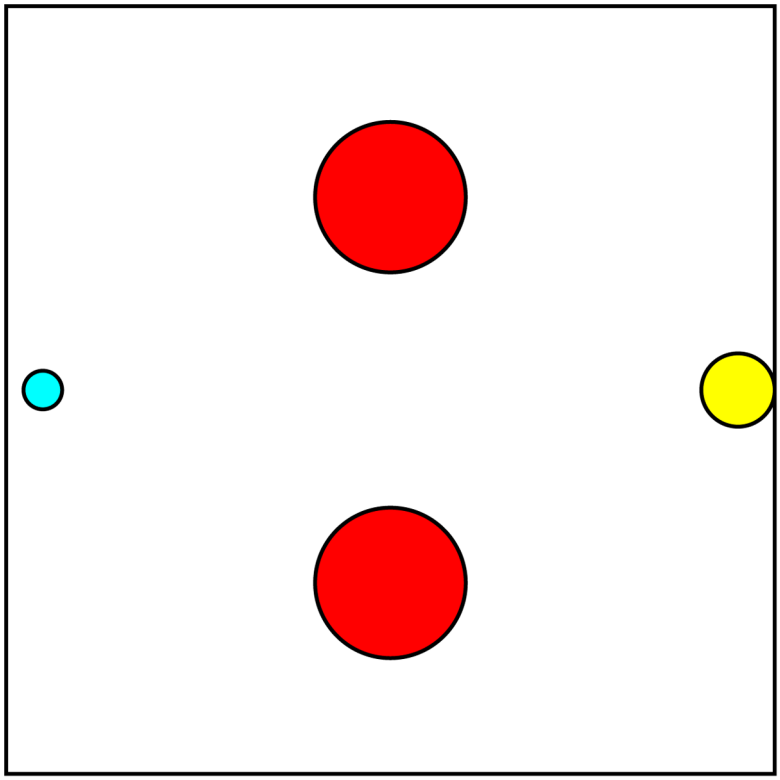

Fig. 2.

In this example, the robot (blue) has to reach the goal (yellow) while avoid collision with the obstacles (red). The positions of the robot and the goal are the states of the problem. Only the robot can be controlled, while the goal moves around randomly. Sensing action space is observing the position of robot or the goal.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

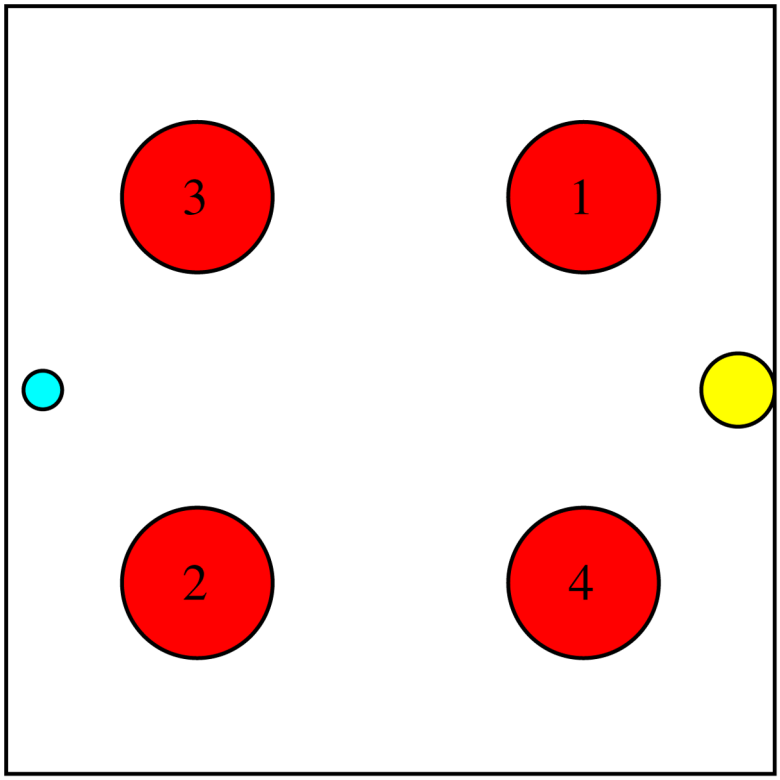

Fig. 3.

In this example, the robot (blue) has to reach the goal (yellow) while avoid colliding with the obstacles (red). The positions of the robot and the obstacles are the states of the problem. Only the robot can be controlled, while the obstacles move around randomly. Sensing action space is observing the position of the robot or one of the obstacles. The numbers on the obstacles indicate the trial the obstacles are first added to the simulation.

Author Manuscript

Author Manuscript

Author Manuscript

Author Manuscript

Table I

Moving goal simulation results

	Trial Number			
Results	**1**	**2**	**3**	**4**
AE Reward	8.621	6.666	1.750	-2.505
SE Reward	7.994	6.034	0.805	-3.310
AE Time (s)	1.19	1.15	1.17	1.17
SE Time (s)	1.65	1.63	1.65	1.66

Author Manuscript

Author Manuscript

Author Manuscript

Author Manuscript

Table II

Moving obstacles simulation results

	Trial Number			
Results	**1**	**2**	**3**	**4**
AE Reward	8.532	7.572	4.871	0.644
SE Reward	8.352	7.295	4.840	-0.654
AE Time (s)	2.51	6.22	11.52	51.12
SE Time (s)	4.62	18.55	51.12	112.35

Author Manuscript

Author Manuscript

Author Manuscript

Author Manuscript

Table III

This table shows how long entropy estimation takes when using Scipy's *k*-d tree versus FLANN. Dimension (d) and number of particles (N) are varied. The results from Scipy (t_s) and FLANN (t_f) are shown as (t_s, t_f). Time is in millisecond.

	$N = 100$	$N = 200$	$N = 300$
$d = 2$	(20.44, 0.78)	(45.21, 1.67)	(69.62, 2.63)
$d = 4$	(33.20, 1.39)	(78.78, 3.28)	(130.17, 5.39)
$d = 6$	(52.47, 2.30)	(149.18, 5.79)	(251.39, 8.93)