# Global Motion Planning under Uncertain Motion, Sensing, and Environment Map

Hanna Kurniawati* Tirthankar Bandyopadhyay* Nicholas M. Patrikalakis*†
*Singapore–MIT Alliance for Research and Technology †Massachusetts Institute of Technology
Singapore, Republic of Singapore Massachusetts, USA
Email: {hannakur, tirtha}@smart.mit.edu Email: nmp@mit.edu

***Abstract*— Motion planning that takes into account uncertainty in motion, sensing, and environment map, is critical for autonomous robots to operate reliably in our living spaces. Partially Observable Markov Decision Processes (POMDPs) is a principled and general framework for planning under uncertainty. Although recent development of point-based POMDPs have drastically increased the speed of POMDP planning, even the best POMDP planner today, fails to generate reasonable motion strategies when the environment map is not known exactly. This paper presents *Guided Cluster Sampling (GCS)*, a new point-based POMDP planner for motion planning with uncertain motion, sensing, and environment map, when the robot has active sensing capability. It uses our observations that in this problem, the belief space $B$ can be partitioned into a collection of much smaller sub-spaces, and an optimal policy can often be generated by sufficient sampling of a small subset of the collection. GCS samples $B$ using two-stage cluster sampling, a subspace is sampled from the collection and then a belief is sampled from the subspace. It uses information from the set of sampled sub-spaces and sampled beliefs to guide subsequent sampling. Preliminary results suggest that GCS generates reasonable policies for motion planning problems with uncertain motion, sensing, and environment map, that are unsolvable by the best point-based POMDP planner today, within reasonable time. Furthermore, GCS handles POMDPs with continuous state, action, and observation spaces. We show that for a class of POMDPs that often occur in robot motion planning, GCS converges to the optimal policy, given enough time. To the best of our knowledge, this is the first convergence result for point-based POMDPs with continuous action space.**

## I. Introduction

Motion planning under uncertainty is critical for robots to operate reliably in our living spaces, such as homes, offices, and outdoor environment. Uncertainty in robot motion planning are caused by three main sources: control error, sensing error, and imperfect environment map. Control error and sensing error are common traits of controllers and sensors. Imperfect environment map is unavoidable as maps of our living spaces are acquired through sensors subject to various substantial noise. Despite the significance of all three causes of uncertainty to motion planning problems, most work take into account only one [5, 8, 16] or at most two sources of uncertainty [4, 20]. This paper presents a new motion planner that takes into account all three sources of uncertainty.

Our new planner uses the Partially Observable Markov Decision Processes (POMDPs) framework. POMDP is a mathematically principled and general framework for planning under uncertainty. To cope with uncertainty, POMDP planners reason over beliefs, where a belief is a distribution over the state space. They plan in the belief space $B$, which is the set of all possible beliefs. Although solving a POMDP is computationally intractable in the worst case [17], recent development of point-based POMDPs [15, 18, 21] have drastically increased the speed of POMDP planning. Key to point-based POMDPs is to sample a set of representative beliefs from $B$ and plan with respect to the set of sampled beliefs, instead of the entire $B$. By doing so, point-based POMDPs can generate a good approximate solution for motion planning problems with moderate difficulty, within a few minutes [10, 14, 15, 21].

However, three major challenges remain to hinder the applicability of POMDPs in motion planning under uncertain environment map. First is the curse of dimensionality. When both the map and robot's configuration are not perfectly known, the state space $S$ is a joint product between the space $E$ of possible environment maps and the configuration space $Q$, causing the dimension $dim(S)$ of $S$ to be extremely high. Consider a simplistic problem of a 2-DOFs robot operating in an environment containing two triangular obstacles where the positions of the vertices are not known perfectly. $Dim(S)$ is already 14! This dimensionality is aggravated in POMDPs, as they plan in $B$ whose size is doubly exponential in $dim(S)$.

Second is the long planning horizon typical of motion planning problems. In a motion planning task, a robot often needs to take many actions to reach the goal, resulting in a long planning horizon. The complexity of planning often grows exponentially with the horizon.

Third is continuous control space. The control space of most robots is continuous, but most POMDP planners assume discrete control space. Although sampled representation of the continuous control space can be used, it is not known whether a good approximation to the optimal policy can be guaranteed. POMDP planners compute a *max* over the control space. While sampling has been shown to approximate the *average* operator well, almost no result is known on how well sampling approximates the *max* operator.

This paper presents a new point-based POMDP planner, called *Guided Cluster Sampling (GCS)*, that alleviates the above three challenges for a robot that localizes itself through active sensing. GCS constructs a more suitable sampling distribution based on two observations. First, in many cases, the optimal policy consists of a small number of sensing actions. Second, the marginal distribution of $E$ in the robot's belief changes only after a sensing action is performed. These observations mean that, if $B$ is partitioned into a collection $\mathscr{C}$ of sub-spaces, where each sub-space consists of all beliefs with the same marginal distribution of $E$, then the set $\mathscr{R}^*(b_0)$ of beliefs reachable under an optimal policy, often lies in a small subset of $\mathscr{C}$. Since the size of a subspace in $\mathscr{C}$ is doubly exponential in $dim(Q)$, which is much smaller than $dim(S)$, and since an optimal policy can be constructed from sufficient sampling of $\mathscr{R}^*(b_0)$ [11], we only need to sufficiently sample a much smaller subspace of $B$.

Using the above observations, GCS performs two-stage cluster sampling. To sample a belief, it samples a sub-space from the collection $\mathscr{C}$ and samples a belief from the sub-space. To sample a small representative set of beliefs, GCS aims to sample from $\mathscr{R}^*(b_0)$. Of course, $\mathscr{R}^*(b_0)$ is not known a priori, as knowing $\mathscr{R}^*(b_0)$ is the same as knowing the optimal policy. Therefore, GCS uses information from the sampled sub-spaces and beliefs to guide subsequent sampling. This sampling strategy alleviates the curse of dimensionality issue.

To alleviate long planning horizon issue, GCS adopts the strategy in [14]. It reduces the effective planning horizon by using action sequences, instead of a single primitive action, to guide sampling $B$.

GCS deals with continuous state, control, and observation spaces by using sampled representations. We show that for a class of POMDPs that often occur in robot motion planning, GCS converges to the optimal policy, given enough time. To the best of our knowledge, this is the first convergence result for point-based POMDPs with continuous control space.

## II. Related Work and Background

### A. Motion planning under uncertainty

In motion planning, uncertainty arises from three main sources, i.e., imperfect control, imperfect sensing, and imperfect information about the environment. However, most work on motion planning under uncertainty takes into account only one or two causes of uncertainty: Stochastic Motion Roadmap [1] considers only control uncertainty, [5] considers only sensing uncertainty, [8, 16] consider only imperfect information about the environment, [4, 20] consider only control and sensing uncertainty, and restricts them to Gaussian. Our new planner takes into account all three sources of uncertainty and allows any type of distribution with bounded support for control, sensing, and environment map uncertainty.

A few work that take into account all three sources of uncertainty during planning [13, 22] are designed specifically for exploration task. [13] restricts control, sensing, and map uncertainty to be Gaussian. [22] finds an approximate solution using a greedy one-step lookahead method, which is often inadequate for general motion planning problems as they often require long planning horizon.

Motion planning under uncertain control, sensing, and environment map is essentially a POMDP problem, and general POMDP planners can conceptually be used. However, despite the impressive progress of such planners [15, 18, 21], they face significant difficulties when the environment map is uncertain. Recently, [7] alleviates these difficulties by using online POMDP. However, this strategy does not perform global planning in the belief space. As a result, it may not converge to the optimal policy. Our planner performs global planning in the belief space by utilizing domain specific properties.

### B. POMDP Background

Formally, a POMDP is specified as a tuple $\langle S, A, O, T, Z, R, \gamma \rangle$, where $S$ is the set of states, $A$ is the set of actions, and $O$ is the set of observations. In each time step, the agent lies in a state $s \in S$, takes an action $a \in A$, and moves from a start state $s$ to an end state $s'$. Due to the uncertainty in action, the end state $s'$ is modeled as a conditional probability function $T(s,a,s') = P(s'|s,a)$. The agent may then receive an observation. Due to the uncertainty in observation, the observation result $o \in O$ is again modeled as a conditional probability function $Z(s',a,o) = P(o|s',a)$. In each step, the agent receives a reward $R(s,a)$, if it takes action $a$ in state $s$. The goal of the agent is to maximize its expected total reward by choosing a suitable sequence of actions. When the sequence of actions has infinite length, we specify a discount factor $\gamma \in (0,1)$ so that the total reward is finite and the problem is well defined.

POMDP planning computes an *optimal policy* that maximizes the agent's expected total reward. A POMDP policy $\pi\colon B \to A$ is a mapping from $B$ to $A$, which prescribes an action $a$, given the agent's belief $b$. A policy $\pi$ induces a value function $V_\pi(b)$. The value function $V_\pi(b) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | b, \pi\right]$ specifies the expected total reward of executing policy $\pi$. To execute a policy $\pi$, an agent executes action selection and belief update repeatedly. For example, if the agent's current belief is $b$, it selects the action referred to by $a = \pi(b)$. After the agent performs action $a$ and receives an observation $o$, it updates $b$ to a new belief $b'$ given by

$$b'(s') = \tau(b,a,o) \quad = \quad \eta Z(s',a,o) \int_{s \in S} T(s,a,s') b(s) ds \quad (1)$$

where $\eta$ is a normalization constant.

A policy can be represented by various representations. GCS is general enough to be used with any policy representation for continuous $S$ and $O$, e.g., [2, 19, 23].

### C. Point-based POMDP

Point-based POMDPs trade optimality with approximate optimality in exchange for speed. It reduces the complexity of planning in $B$ by representing $B$ as a set of sampled beliefs and planning with respect to this set only. To generate a policy, most point-based POMDPs use value iteration, utilizing the fact that the optimal value function satisfies Bellman equation. They start from an initial policy, represented as a value function $V$. And iteratively perform Bellman backup on $V$ at the sampled beliefs, until the iteration converges. Over the past few years, impressive progress have been gained by improving the strategy for sampling $B$ [15, 18, 21].

Despite the impressive progress, even the best point-based POMDP planners today face significant difficulties in solving motion planning under uncertain environment map. GCS alleviates these dificulties by constructing a more suitable sampling strategy based on domain specific properties.

Furthermore, most point-based POMDPs are designed for discrete state, action, and observation spaces. Although a few [9, 19] handle continuous state, action, and observation spaces, they do not provide convergence guarantee when the action space is continuous. For a class of POMDP problems with continuous state, action, and observation spaces, that often occur in robot motion planning, GCS is guaranteed to converge to the optimal policy, given enough time.

## III. POMDP FORMULATION

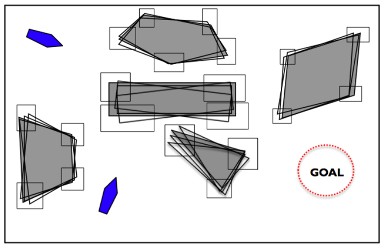


Fig. 1. The robot is a pentagon. The two blue pentagons represent the possible initial configurations of the robot. The grey polygons are obstacles. The position of each vertex is not perfectly known, and maybe anywhere within the rectangular region.

Let's first consider a robot operating in a 2D environment populated by polygonal obstacles. The number of obstacles, the number of vertices in each polygon, and the connectivity between the vertices in each polygon are perfectly known. However, the position of the vertices are not perfectly known and are represented as probability distributions with bounded support (e.g., Figure 1). Due to imperfect information about the robot's configuration and the environment it operates in, we model both robot configuration and environment map as state variables in POMDP. The map is feature-based, where the features are the obstacles' vertices and goal features. A goal feature is a mark in the environment, indicating the robot's goal position. Suppose the features are numbered sequentially from 1 to $n$. And suppose $E_i$ is the set of all possible positions of feature-$i$. Then, the POMDP state space, $S = Q \times E$, where $Q$ is the robot's configuration space and $E = E_1 \times \cdots \times E_n$ is the environment space.

The robot uses a visibility sensor with stop-and-go mechanism. Therefore, the action space $A$ consists of two subsets, control set $U$ and $\{sensing\}$. When the robot performs a control action $u \in U$, it moves according to a control law which is noisy and perceives no observation. When *sensing* is performed, the robot does not move but perceives an observation about the position of the features that lie inside the field of view of its sensor, with probability as modeled in $Z$. The function $Z$ depends only on the relative position of the features with respect to the robot. GCS can use any data association method [24]. Data association is a large domain in itself and is outside the scope of this paper. Notice that in this problem, regardless of the exact $T$ and $Z$, the marginal distribution of $E$ in the robot's belief changes only when a sensing action is performed. GCS exploits this property to generate a more suitable strategy for sampling $B$.

The robot's objective is to reach a goal region with as little cost as possible. The goal region is the set of points within a small pre-specified distance from a goal feature. The cost is the sum of moving/sensing cost and collision cost. To model this objective, for any state $s \in S$ and any action $a \in A$, we define $R(s,a) = R_{goal}(s) + R_{action}(s,a) + R_{collision}(s,a)$, where $R_{goal}(s)$ is the goal reward, i.e., a large constant if $s$ is in a goal region and 0 otherwise, $R_{action}(s,a)$ is the cost of performing $a$ from $s$, and $R_{collision}(s,a)$ is the expected collision cost of moving according to $a$ from $s$. Notice that $R$ depends only on the action performed and the relative position of environment features with respect to the robot. GCS uses this property to generate a more effective sampling guide (Section V-A).

Unlike most work in POMDP planners, in this paper, $S$, $U$, and $O$ are continuous. Continuous spaces are more natural for modeling robotics problems. To represent these continuous spaces, GCS uses sampled representations.

Now, we define the continuity properties of $R$, $Z$, and $T$ with respect to $S$ and $A$, so that convergence to the optimal policy can be guaranteed even when $A$ is approximated with its sampled representation. The approximation result is in Section VI. The properties with respect to $S$ are,

**Definition 1** *Suppose $\langle S,A,O,T,Z,R,\gamma\rangle$ is a POMDP with continuous $S$. Let $D_S$ be a metric in $S$. The POMDP is LS-continuous with parameter $(K_{RS},K_Z)$, $K_{RS},K_Z \in \mathbb{R}$, when:*

1) *The motion uncertainty is the same everywhere. For any displacement vector $d$ on $S$ and any $s,s' \in S$, $T(s,a,s') = T(s+d,a,s'+d)$.*
2) *The observation function $Z$ is Lipschitz continuous in $S$. For any $s,s' \in S$, any $a,a' \in A$, and any $o \in O$, $|Z(s,a,o) - Z(s',a,o)| \le K_Z \cdot D_S(s,s')$.*
3) *The reward $R$ is Lipschitz continuous in $S$. For any $s,s' \in S$ and any $a \in A$, $|R(s,a) - R(s',a)| \le K_{RS} \cdot D_S(s,s')$.*

Property-1 may seem odd as it means that the robot may go through an obstacle. However, notice that in motion planning, the modeling of actual physical dynamics during collision is essentially a way to generate motion strategy with low collision cost. We can generate the same strategy without modelling the effect of collision in $T$, by setting a high penalty for collision. This trick is similar to the use of potential function in deterministic motion planning. Property-1 simplifies proving the convergence result (Section VI), as transition from all states can be treated equally.
Property-2 means that the robot receives similar observation when it is at nearby configuration in similar environment. This is a common assumption in robotics.
Property-3 means that the robot receives similar immediate reward when it is at nearby configuration in similar environment. Since Lipschitz condition is closed under summation, to satisfy this property, we only need to ensure that each component of $R$ is Lipschitz. This can be satisfied easily, for instance by setting $R_{goal}$ to be linearly increasing as the robot becomes closer to the goal region, $R_{action}$ for the same action to be the same everywhere, and $R_{collision}$ to be linearly increasing as the robot becomes closer to an obstacle. This reward function is similar to potential functions used in motion planning [6].

Now, we define the properties of $R$, $Z$, and $T$ on $A$.

**Definition 2** *Suppose $\langle S,A,O,T,Z,R,\gamma\rangle$ is a POMDP where $S$ is continuous. And $A$ can be partitioned into a finite collection $\mathscr{P}$ of disjoint continuous sets, where two actions $a,a' \in A$ are* *in the same set $P \in \mathscr{P}$ whenever $O_a \bigcap O_{a'} \neq \emptyset$, $O_a$ is the set of observations that can be perceived when $a \in A$ is performed, i.e., $O_a = \{o \in O \mid \exists s \in S \;\; Z(s,a,o) > 0\}$. Suppose $D_S$ and $D_P$ are metrics on S and on set $P \in \mathscr{P}$. Then, the POMDP is LA-continuous with action partition $\mathscr{P}$ and parameters $(K_{RA}, h)$ when the three properties below hold. The parameter $K_{RA} \in \mathbb{R}$, while h is an increasing function with $h(0) = 0$, that maps distance in A to distance in S. The properties are:*

1) *For any $P \in \mathscr{P}$, any $a, a' \in P$, and any $s, s' \in S$, $T(s,a,s') = T(s,a',s'+f(a,a'))$, where f is a function that maps a pair of actions in A to a displacement vector in S, such that $D_S(s, s+f(a,a')) \leq h(D_P(a,a'))$.*
2) *The observation function is the same for any action in the same set of $\mathscr{P}$. For any $o \in O$, any $s \in S$, any $P \in \mathscr{P}$, and any $a, a' \in P$, $Z(s,a,o) = Z(s,a',o)$.*
3) *The reward R is Lipschitz continuous on each set in $\mathscr{P}$. For any $s \in S$, any $P \in \mathscr{P}$, and any $a, a' \in P$, $|R(s,a) - R(s,a')| \leq K_{RA} \cdot D_P(a,a')$.*

An example of $\mathscr{P}$ in our POMDP model is $\{U, \{sensing\}\}$. Property-1 means that when nearby actions within the same element of $\mathscr{P}$, are applied to the same state, the resulting states would be close too. The function $f$ can be linear or non-linear, which means that this property can be satisfied by both linear and non-linear control. Property-2 means that within the same action set $P \in \mathscr{P}$, the observation that can be perceived by the robot depends only on the robot's configuration and features' positions. This requirement is common in robotic system with visibility sensor that operates in a stop-and-go mode. Property-3 is similar to property-3 of Definition 1, but applied to $A$.

## IV. Overview of GCS

A key idea of point-based POMDP planners is to sample a representative set of beliefs from $B$ and use it as an approximate representation of $B$. For computational efficiency, most of the recent point-based POMDP planners sample from the set of points reachable from $b_0$, instead of the entire $B$, and consider nearby beliefs to be the same. The sampled beliefs are represented as a belief tree $T$, where each node represents a sampled belief and the root is $b_0$. To sample a new belief, these planners sample a node $b \in T$, an action $a \in A$, and an observation $o \in O$ according to suitable probability distributions or heuristics. They then compute $b' = \tau(b,a,o)$ using (1). Whenever the distance between $b'$ and its nearest node in $T$ is larger than a given threshold $\varepsilon$, $b'$ is inserted into $T$ as a child of $b$. Otherwise, $b'$ is discarded. The number of different beliefs that can be sampled is then $O(\frac{1}{\varepsilon}^{\frac{1}{\delta}^{dim(S)}})$ where $\delta$ is a given threshold used to determine if two states $s, s' \in S$ are the same. Furthermore, when the problem requires a planning horizon of $h$, $T$ needs to have at least $h$ levels. Therefore in the worst case, the size of the sampling domain is doubly exponential in $\dim(S)$ or exponential in $h$, whichever is smaller. Together, the large $\dim(S)$ and planning horizon generate a huge sampling domain that pose significant difficulty for even the best POMDP planners today.

Although the sampling domain is huge, a good sampled representation of $B$, i.e., one that enables the generation of an optimal policy, often lie in a much smaller subspace of $B$. Let's first define a partitioning $\mathscr{C}$ of $B$ as,

**Definition 3** *The partition $\mathscr{C}$ of $B$ is a collection $\{C(b_E^1), C(b_E^2), C(b_E^3), \ldots\}$ of disjoint sub-spaces, where each $C(b_E) = \{b \in B \mid \int_{q \in Q} b(\langle q,e \rangle)dq = b_E, e \in E\}$.*

We observe that the optimal motion strategies often perform a small number of sensing actions. Since a robot's belief over its environment changes only when a sensing action is performed, our observation implies that $\mathscr{R}^*(b_0)$ often lie in a small subset of $\mathscr{C}$. Since $dim(C(b_E))$ of any $C(b_E) \in \mathscr{C}$ is doubly exponential in $dim(Q)$, instead of the much larger $dim(S)$, $\mathscr{R}^*(b_0)$ lies in a much smaller subspace of $B$. This observation and the results [11] that a good sampled representation of $B$ can be constructed by sufficient sampling of $\mathscr{R}^*(b_0)$, indicate that it is sufficient to sample from a small subset of $\mathscr{C}$.

Utilizing the above observations, GCS alleviates the difficulties due to huge $B$ by performing a two-stage cluster sampling. It samples a small number of relevant subspaces from $\mathscr{C}$ and samples beliefs only from the sampled subspaces. Of course ideally, the sampled beliefs are all in $\mathscr{R}^*(b_0)$. However, since $\mathscr{R}^*(b_0)$ is not known apriori, GCS samples $B$ incrementally. It uses heuristics based on environment distributions in the sampled subspaces and the sampling history, to guide sampling subsequent subspaces and to guide sampling representative beliefs in each subspace. To further alleviate the planning complexity, GCS reduces the planning horizon by using action sequences, instead of primitive actions, to guide sample $B$.

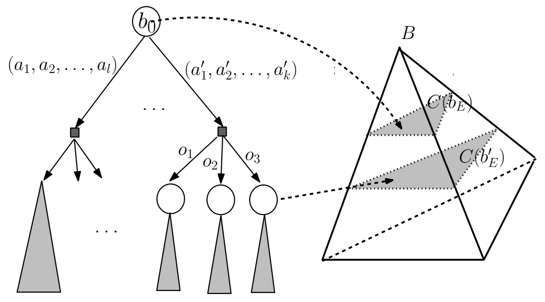


Fig. 2. The belief tree $T$ (left) and the belief space $B$ (right) it represents. Each node in $T$ lies in an element of $\mathscr{C}$. Suppose $b \in T$ lies in $C(b_E) \in \mathscr{C}$. Each out-edge of a node $b$ corresponds to a path in the $C(b_E)$.

Similar to most point-based POMDP planners, GCS constructs a belief tree $T$, but expands $T$ using the above idea. The overall algorithm is in Algorithm 1. Given a node $b \in T$ to expand, GCS finds a subspace $C(b_E) \in \mathscr{C}$ that contains $b$ (line 4). It uses the environment map distribution $b_E$ as a guide to generate an action sequence (line 5) that leads to the goal state with high probability and low collision cost, or to a belief where useful sensing data can be gained with high probability. The action sequence is either a sequence with no sensing action or a sequence with a single sensing action, located at the end. The node $b$ is then expanded using the generated action sequence. To expand $b \in T$ using an action sequence $(a_1, a_2, \ldots, a_l)$, GCS iteratively apply (1) to $b$. Specifically, $\tau(b, (a_1, a_2, \ldots, a_l), noObservation)$ (line 6) means that it computes a sequence of beliefs $(b_1, b_2, \ldots, b_l)$, where $b_1 = \tau(b, a_1, noObservation)$ and $b_i = \tau(b_{i-1}, a_i, noObservation)$ for $i \in [2, l]$, and returns the last belief $b_l$. A new belief is inserted into $T$ only when the distance between the new belief and its nearest node in $T$ is more than a given threshold. The action edge that connects $b$ with the newly inserted belief is then annotated with the action sequence. Figure 2 shows an illustration of $T$. The detailed expansion process is in line 7–16, while more detailed on GCS is in Section V.

**Algorithm 1** Guided Cluster Sampling ($N$)

```
1: Initialize $T$ by setting $b_0$ as the root.
2: for $i$ = 1 to $N$ do
3:     $b$ = Sample a node from $T$ inversely proportional to the number of
       times $b$ has been sampled.
4:     $\langle \mathbf{0}, b_E \rangle$ = Transform($b$).
5:     $(a_1, a_2, \ldots, a_l)$ = GenerateAnActSeq($b_E$, nNotImproved($b_E$)).
6:     Let $b_{l-1} = \tau(b_1, (a_1, a_2, \ldots, a_{l-1}), noObservation)$.
7:     if $a_l$ is a sensing action then
8:         Let $O' \subset O$ be the set of sampled observations.
9:         for each $o \in O'$ do
10:            $b_l = \tau(b_{l-1}, sensing, o)$.
11:            if $\min_{b' \in T} D_B(b_l, b') < \varepsilon$ then
12:                Insert $b_l$ into $T$ as a child of $b$.
13:    else
14:        Let $b_l = \tau(b_{l-1}, a_l, noObservation)$
15:        if $\min_{b' \in T} D_B(b_l, b') < \varepsilon$ then
16:            Insert $b_l$ into $T$ as a child of $b$.
17:    for all $b'$ = children of $b$ do
18:        BACKUP($b'$).
19:    if $V(b_0)$ does not improve then
20:        Increase nNotImproved($b_E$) by 1.
21:    else
22:        Reset nNotImproved($b_E$) to 0.
```

**Algorithm 2** GenerateAnActSeq($b_E$, nNotImproved)

```
1: if $T(b_E)$ has not been initialized then
2:     Initialize $T(b_E)$ by setting $\langle \mathbf{0}, b_E \rangle$ as root and nNotImproved to be 0.
3:     Constructs an uncertainty roadmap $G(b_E)$.
4: if nNotImproved > maxNotImproved then
5:     if getRandomNumber() < 0.5 then
6:         Refine($G(b_E)$) and update $T(b_E)$.
7:         $(a_1, a_2, \ldots, a_l)$ = ExpandTowardsGoal($T(b_E)$).
8:     else
9:         $(a_1, a_2, \ldots, a_l)$ = ExpandTowardsSensing($T(b_E)$).
10: else
11:    $(a_1, a_2, \ldots, a_l)$ = ExpandTowardsGoal($T(b_E)$).
12: Return $(a_1, a_2, \ldots, a_l)$.
```

## V. Guided Cluster Sampling (GCS)

### A. *Finding a subspace in $\mathscr{C}$ that contains b*

To find a subspace in $\mathscr{C}$ that contains $b$, GCS first transforms the environment map to the robot's local coordinate system (line-4 of Algorithm 1). More precisely, let $g : S \to S$ be a many to one function where $g(s) = g(\langle q, e \rangle) = \langle \mathbf{0}, \Phi(e) \rangle$, $\mathbf{0}$ is the origin of the robot's coordinate system, and $\Phi(e)$ is a function that transforms the position of each environment feature to the robot's local coordinate system. The transformation for beliefs is then defined as follows. Suppose $b' = Transform(b)$, then

$$b'(s') = \int_{s \in S} b(s) \cdot I(s, s') ds \text{ where } I(s, s') = \begin{cases} 1 & \text{if } g(s) = s'. \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

The transformed belief $b'$ has probability one that the robot's configuration is at $\mathbf{0}$. This transformation transforms the robot's uncertainty to the environment map's uncertainty, such that a sampling guide based on $b_E$ takes into account both the robot and environment uncertainty, and hence is more effective. Details on the sampling guide is in Section V-B.

Now, the question is would $Tranform$ changes the computed optimal policy. The answer is no, more precisely,

**Theorem 1** *On problems where motion uncertainty is the same everywhere, and the reward and observation functions depend only on the relative configuration of the robot with respect to the environment, $V_\pi(b) = V_\pi(Transform(b))$ for any POMDP policy $\pi$ and any belief $b \in B$.* [1]

This theorem means that the belief and its transformation can be used interchangeably in computing the optimal solution.

### B. *Generating an action sequence*

Given $b \in T$ to expand and $\langle \mathbf{0}, b_E \rangle = Transform(b)$, GCS generates action sequences using information from $b_E$. The action sequences are represented as a partial belief tree $T(b_E)$. The root of $T(b_E)$ is $\langle \mathbf{0}, b_E \rangle$. The nodes in $T(b_E)$ are beliefs in $C(b_E) \in \mathscr{C}$ or exit nodes. An exit node acts as an exit point from $C(b_E)$, and is reached after a sensing action. The nodes in $T(b_E)$ are classified into terminal nodes and non-terminal nodes. Terminal nodes consist of goal nodes, depth nodes, and exit nodes. A node $b_p$ in $T(b_E)$ is a goal node whenever $b_p(goalRegion)$ is more than a given threshold. Depth nodes are all nodes at a given maximum depth in $T(b_E)$. Each edge $\overline{b_p b'_p}$ in $T(b_E)$ is annotated by either a control sequence $(u_0, \ldots, u_k)$ where $b'_p = \tau(b_p, (u_0, \ldots, u_k), noObservation)$, or a sensing action, in which case $b'_p$ is an exit node. Each action sequence for expanding $b$ corresponds to a path from the root of $T(b_E)$ to a terminal node. It is a concatenation of the actions that annotate the edges in the path.

The tree $T(b_E)$ is constructed incrementally. Each call to *GenerateAnActSeq* (Algorithm 2) inserts a new path into $T(b_E)$, and returns the action sequence that corresponds to the newly inserted path.

To add a new path into $T(b_E)$, GCS samples a non-terminal leaf node $b_p \in T(b_E)$ and iteratively expands $b_p$ until a goal node or a depth node is reached. Since we would like $T(b_E)$ to cover $C(b_E)$ well, GCS samples a leaf node of $T(b_E)$ inversely proportional to the sampling density of $C(b_E)$. To each $b_p \in T(b_E)$, GCS assigns a weight $w(b_p)$ which is equal to the number of non-leaf nodes in $T(b_E)$ that lie within a small pre-specified distance from $b_p$. It samples a leaf node from $T(b_E)$ with probability $\sim 1/w(b_p)$.

To expand a node $b_p \in T(b_E)$, GCS uses an uncertainty roadmap $G(b_E)$ (Figure 3) constructed in the robot's C-space $Q$ with obstacles distributed as $b_E$, to quickly generate action sequences that are more likely to move the robot to the goal region with high probability and low cost. The milestones in $G(b_E)$ are the initial configuration $\mathbf{0}$, the goal configurations, and sampled configurations from $Q$. Each sampled configuration has expected collision cost lower than a given threshold $th_c$. An edge $\overline{vv'}$ in $G(b_E)$ is a straight line segment between $v$ and $v'$ with expected collision cost lower than $th_c$. It is

[1]All proofs are in the Appendix.

associated with a sequence of controls that move the robot from $v$ to $v'$ with non-zero probability.

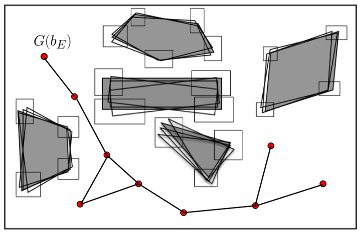


Fig. 3. An environment map where the obstacles are distributed according to $b_E$. The roadmap $G(b_E)$ is the uncertainty roadmap to guide sampling $C(b_E)$.

GCS constructs $G(b_E)$ once during initialization step (line 2 of Algorithm 2). The strategy in [8] or slight modification of any sampling based motion planner [6, 12] can be used to construct $G(b_E)$.

GCS uses the actions associated with paths in $G(b_E)$ to expand $T(b_E)$. It annotates each node $b_p \in T(b_E)$ with the milestone used to generate $b_p$. The root of $T(b_E)$ is annotated with the initial configuration $\mathbf{0}$. Suppose a node $b_p$, annotated with milestone $v$ of $G(b_E)$, is to be expanded. GCS finds the shortest path $\psi$ in $G(b_E)$ from $v$ to a goal milestone, and then expands $b_p$ according to $\psi$. Suppose $\psi$ consists of edge sequence $(\overline{vv_1}, \overline{v_1v_2}, \ldots, \overline{v_{n-1}v_n})$. GCS first expands $b_p$ with every out-edge of $v$. Let $\overline{vv'}$ be an out-edge of $v$ and is associated to a sequence of controls $(u_1, u_2, \ldots, u_k)$. GCS computes $b'_p = \tau(b_p, (u_1, u_2, \ldots, u_k), noObservation)$, inserts $b'_p$ as a child node of $b_p$, annotates the edge $\overline{b_pb'_p}$ with $(u_1, u_2, \ldots, u_k)$, and annotates $b'_p$ with $v'$. Let $b^1_p$ be the child of $b_p$ that is annotated with $v_1$, the end-milestone of the first edge in $\psi$. If $b^1_p$ is not a terminal node, GCS continues expanding $b^1_p$ with all out-edges of $v_1$ using the above expansion procedure. All other newly inserted children of $b_p$ is not expanded in the current action sequence generation. The expansion process is then repeated for the child of $b^1_p$ that is annotated by $v_2$, and so on, until the last edge in $\psi$ is used to expand $T(b_E)$ or a terminal node is reached. Suppose $b^n_p$ is the belief that is annotated with $v_n$, the last milestone in $\psi$. If $b^n_p$ is not a terminal node, GCS samples a state $s \in S$ according to $b^n_p$, and continues expanding $b^n_p$ using the shortest path from $s$ to $v_n$. This process is repeated until a terminal node is reached.

Now, when $V(b)$ does not improve after a pre-specified number of $b$'s expansions and backups, GCS expands a belief in $T(b_E)$ with a sensing action or refines $G(b_E)$. GCS selects one of them with equal probability (line 5 of Algorithm 2).

To expand $T(b_E)$ with a sensing action (line 9 of Algorithm 2), GCS first samples a belief $b_p \in T(b_E)$ that has a potential for generating useful sensing data. To sample such belief, GCS samples uniformly at random from the set of beliefs that satisfies these two conditions: (1) The belief has not been expanded with a sensing action. (2) At the belief, at least one of the environment features lies within the sensor's field of view, with probability larger than a predefined threshold. GCS inserts an exit node into $T(b_E)$ as a child of $b_p$, and annotates the edge $\overline{b_p exitNode}$ with a sensing action. GCS then finds the path from root to the newly inserted exit node, and returns the concatenation of the action sequences that annotate the edges in the path.

To refine $G(b_E)$, GCS samples additional vertices and reconstructs the roadmap. It updates $T(b_E)$ by creating new edges that correspond to new edges in the roadmap. Finally, it proceeds to expand $T(b_E)$ using the refined uncertainty roadmap (line 6–7 of Algorithm 2).

## C. Backup

The function $BACKUP(b)$ for $b \in T$ (line 18 of Algorithm 1) finds a path from $b$ to the root of $T$ and performs point-based backup at each belief in the path, starting from $b$ to the root node. It can use a slight modification of any backup computation for continuous $S$ and $O$ (e.g., [2, 19, 23]). The modification accommodates continuous control space, i.e.

$$\hat{H}_b V(b) = \max_{(a_1,\ldots,a_l)\in outEdge(b)} \Big\{ R(b,(a_1,a_2,\ldots,a_l)) + \gamma^l \int_{o\in O} P(o \mid b_l, a_l) V(\tau(b_l,a_l,o)) \Big\}$$

where $b_l = \tau(b,(a_1,\ldots,a_{l-1}), noObservation)$ and $R(b,(a_1,a_2,\ldots,a_l)) = R(b,a_1) + \sum_{i=1}^{l-1} \gamma^i T(b_i,a_i,b_{i+1}) R(b_{i+1},a_{i+1})$ with $b_1 = b$. The approximation results for using sampled representation for $A$ is in Section VI.

## D. Belief space metric

To compute distance between beliefs in $B$ (line 11 & 15 of Algorithm 1), GCS uses Wasserstein distance, a metric that is *dependent* on the underlying state space metric. The Wasserstein distance $W_D(b,b')$ between two beliefs $b,b' \in B$ is the minimum expected distance in $S$ among all possible joint densities whose marginal densities are $b$ and $b'$. More precisely we use the squared 2nd Wasserstein distance,

$$D_B(b,b') = W_D(b,b') = \inf_f \Big\{ \int_{s\in S}\int_{s'\in S} D_S(s,s') f(s,s') ds ds' \mid b = \int_{s'} f(s,s') ds', b' = \int_s f(s,s') ds \Big\} \quad (3)$$

where $D_S$ is $L2$ norm in $S$ and $f(s,s')$ is joint density function.

Compared to KL-divergence, which is commonly used in POMDPs with continuous $S$, Wasserstein distance is a true metric. This makes analysing approximation bound using Wasserstein distance easier than using KL-divergence.

Furthermore, Wassertein distance is more suitable than KL-divergence, denoted as $KL$, for goal-reaching tasks, which is common in motion planning. As an illustration, consider a 1D navigation problem where the robot's position is not known exactly. The state space here is $\mathbb{R}$. Suppose the goal is $g$ and the goal belief $b_g$ is a belief where the support is all points in $[g-1, g+1]$. Now, suppose $b$ is a belief where the support is all points in $[g-100, g-99]$ and $b'$ is a belief where the support is all points in $[g-10, g-9]$. Then, regardless of the exact distribution, $KL(b,b_g)$ and $KL(b',b_g)$ are both undefined, even though the robot is actually much closer to $g$ when it is at $b'$ than when it is at $b$, assuming the robot's belief is a good estimate of the robot's true position. On the other hand, $W_D(b',b_g) < W_D(b,b_g)$, as desired.

The Wasserstein distance satisfies the Lipschitz condition,

**Theorem 2** *In an LS-continuous POMDP with parameter $(K_{RS}, K_Z)$ and normalized observation space, for any two beliefs $b, b' \in B$ and any policy $\pi$, if* $W_D(b,b') \le \delta$, *then* $|V_\pi(b) - V_\pi(b')| \leq 2\left(\frac{K_{RS}}{1-\gamma} + \frac{\gamma K_Z R_{max}}{(1-\gamma)^2}\right)\delta$*, where* $R_{max}$ *is the maximum possible immediate reward.* [1]

This property means that two nearby beliefs under this metric have similar values, and hence with a small error, one can be used to represent the other.

## VI. Approximation bound

Our main concern is how sampled representation of the action space $A$ affects the quality of the generated policy. Approximation results are available when sampled representation of $S$ and $O$ are used [2, 19]. However, no results are available when sampled representation of $A$ is used. Here we show that when the POMDP problem satisfies $LS$-continuous and $LA$-continuous properties, the optimal policy can be approximated proportional to the sampling dispersion of $A$.

For clarity, we analyze a simplified version of GCS. We assume that the belief tree $T$ is expanded using a primitive action, instead of a sequence of actions. In this case, the partition $\mathscr{P}$ of $A$ (Definition 2) is $\{U, \{sensing\}\}$.

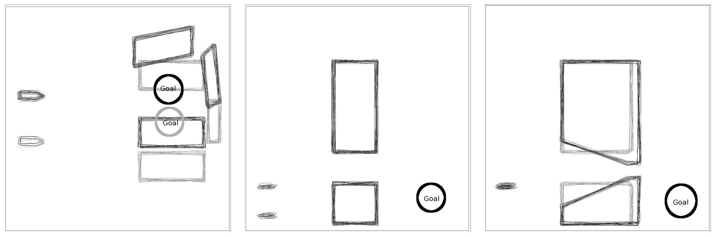
**Theorem 3** *Consider a POMDP that is LS-continuous with parameters* $(K_{RS}, K_Z)$*, and LA-continuous with parameters* $(K_{RA}, h)$ *and action partition* $\mathscr{P}$*. Suppose the distance between two nearest sampled actions in any* $P \in \mathscr{P}$ *is* $\leq \delta_A$*, the distance between two nearest sampled beliefs is* $\leq \delta_B$*, and the maximum immediate reward is* $R_{max}$*. Then,* [1]

$$|V^*(b') - V_t(b')| \leq \frac{1}{1-\gamma}\left(K_{RA}\delta_A + \left(\frac{K_{RS}}{1-\gamma} + \frac{K_Z R_{max}}{(1-\gamma)^2}\right)(4\delta_B + \gamma h(\delta_A))\right)$$

Since $h$ is an increasing function with $h(0) = 0$, $|V^*(b') - V_t(b')|$ converges to zero as $\delta_B$ and $\delta_A$ goes to zero. Similar results hold for GCS as described in Section IV and Section V. However, the terms in the approximation bound becomes much more complex, and hinders understanding.

## VII. Experimental setup and results

### A. Scenarios



ASV-parking. Navigation-1. Navigation-2.

Fig. 4. Initial beliefs of the scenarios.

We tested GCS on three scenarios (Figure 4) of an Autonomous Surface Vehicle (ASV) operation in coastal zone. The environment is populated by obstacles made of floating platforms. Weights are tied underneath, so that the platforms only move within a certain limit, despite currents and wakes. We model the obstacles as polygons, where the vertices' positions are distributed uniformly within a bounded support.

The ASV has 3DOFs and holonomic control. Each time step is 0.25s. Due to control error and currents, whenever a control $u$ is applied for 0.25s to a configuration $q$, the robot moves to a new configuration $q' = q + u + N$, where $N$ is a uniform distribution over a disc of radius 2.5cm with center $q + u$.

The ASV is equipped with a laser sensor. It has $360^0$ field of view. Due to obstruction from waves, the effective range limit is only 10m. Furthermore, the sensor has 25cm error radius, whenever it sees an object at $x$, the actual position may be anywhere within 25cm distance from $x$, with equal probability. GPS is not available. Due to obstructions from piers and other structures near coastal zones, GPS error is around 10m, which is too large for navigation purposes.

The initial belief of the environment and the robot's configuration is shown in Figure 4. In all scenarios, the goal is defined in terms of robot's position, not configuration. The cost for each control action is -1, while the cost for each sensing action is -5,000. The high sensing cost may seem odd for an ASV with laser sensor. However, the ASV is a test platform before the algorithm is used for Autonomous Underwater Vehicle (AUV) navigation using sonar. Since sonar waves disturb underwater life, the cost for sensing is high. We would like to reflect this scenario in our test. Each collision incurs a -100,000 penalty. A reward of 10,000,000 is given when the goal is reached. The discount factor is 0.995.

### B. Experimental setup

For each scenario, we ran GCS to generate 30 policies, as GCS uses random numbers. Each policy is generated for 10min. To estimate the expected total reward of each policy, we ran 100 simulation runs and compute the average total discounted rewards.

For comparison, we use BURM [8] and reactive greedy strategy [3]. BURM takes into account uncertainty in the environment map only. Comparison with BURM would show the importance of taking into account motion and sensing uncertainty. We ran BURM to generate 30 different paths. Each path is considered as an open-loop policy. To estimate the expected total reward of each path, we ran 100 simulation runs and compute the average total discounted rewards.

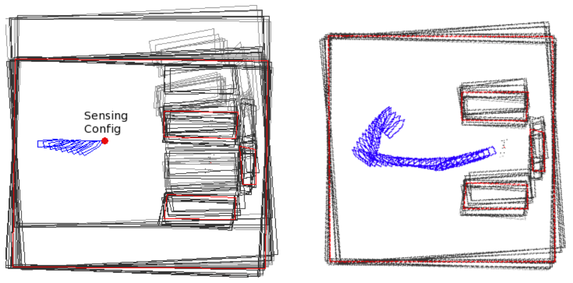
Variants of reactive greedy strategy are often used in ASVs and AUVs. Our ASV has used [3]. The strategy uses continuous sensing to handle motion, sensing, and environment uncertainty. Comparison with this strategy would show how we perform compared to current practice in marine robotics.

We can not compare with existing POMDP planners because the state space is too large for even the best planners (HSVI2 [21] and SARSOP [15]). As an illustration, if we discretize the possible position of each feature in Navigation-1 (scenario with smallest number of environment features) into only 4 cells, the possible robot's position into $10 \times 10$ cells, and the robot's possible heading into 4 cells, $|S| > 2.5 \times 10^7$, which is beyond the capability of HSVI2 and SARSOP.

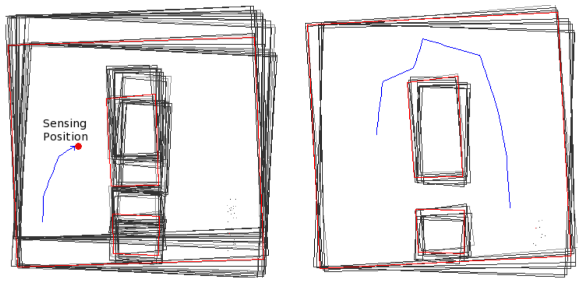
All planners were implemented in C++, and ran in a PC with 2.27GHz Intel processor and 1.5GB RAM.

### C. Experimental results

Table I shows GCS outperforms BURM and reactive greedy strategy. They suggest the importance of taking into account motion and sensing uncertainty, as well as the benefit of



a. Parking



b. Navigation-1

Fig. 5. A typical simulation run of GCS policies. Only one sensing action was performed. Left is before sensing. Right is after sensing.

**Expected Total Discounted Reward**

| Scenario | BURM | Reactive Greedy | GCS (10 min) |
|---|---|---|---|
| Parking | -395,632 | -428,140 | 14,867 |
| Navigation-1 | -1,242,902 | -662,019 | 8,030 |
| Navigation-2 | -1,697,339 | -761,415 | 8,149 |

TABLE I
COMPARISON RESULTS

global motion planning under uncertain motion, sensing, and environment map.

Figure 5 shows a typical simulation run of the policy generated by GCS. The policy for Navigation-2 is similar to that of Navigation-1, and we do not show it here due to lack of space. The simulation shows that when the ASV performs sensing, it generally perceives observation that significantly reduce its uncertainty. Therefore, the ASV can reach its goal with only a few sensing actions.

Figure 5(a) shows the benefit of global motion planning under uncertainty to Parking mission. Recall that we only require the robot position to be inside the goal region, and do not impose a particular heading. However, by taking into account uncertainty in the environment map and robot's position, the ASV always parks in a horizontal orientation, as this orientation has lower collision probability than the vertical orientation. Furthermore, to reach the chosen goal configuration, the ASV performs the necessary maneuvering outside the base station. It avoids maneuvering inside the tight space of the base station to reduce expected collision cost.

## VIII. Conclusion

This paper proposes a new global motion planner under motion, sensing, and environment map uncertainty, called Guided Cluster Sampling (GCS). GCS is a point-based POMDP planner that uses domain specific properties to construct a more suitable sampling strategy. Preliminary results show that GCS successfully solves motion planning problems with uncertain motion, sensing, and environment map that are unsolvable by the best POMDP planners today, within reasonable time.

GCS is designed for POMDPs with continuous state, action, and observation spaces. We showed that for a class of POMDPs that often occur in motion planning problems, given enough time, GCS converges to the optimal policy. To the best of our knowledge, this is the first convergence result for point-based POMDPs with continuous action space.

**Acknowledgement** The authors thank D. Hsu for the discussion and cluster computing usage, L.P. Kaelbling and J.J. Leonard for the discussion, and S. Ong for reading the early drafts of this paper.

## References

[1] R. Alterovitz, T. Simeon, and K. Goldberg. The stochastic motion roadmap: A sampling framework for planning with Markov motion uncertainty. In *RSS*, 2007.
[2] H. Bai, D. Hsu, W.S. Lee, and A.V. Ngo. Monte Carlo Value Iteration for Continuous-State POMDPs. In *WAFR*, 2010.
[3] T. Bandyopadhyay, L. Sarcione, and F. Hover. A simple reactive obstacle avoidance algorithm and its application in Singapore Harbour. In *FSR*, 2009.
[4] J.V.D. Berg, P. Abbeel, and K. Goldberg. LQG-MP: Optimized Path Planning for Robots with Motion Uncertainty and Imperfect State Information. In *RSS*, 2010.
[5] B. Burns and O. Brock. Sampling-Based Motion Planning with Sensing Uncertainty. In *ICRA*, 2007.
[6] H. Choset, K.M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L.E. Kavraki, and S. Thrun. *Principles of Robot Motion : Theory, Algorithms, and Implementations*. The MIT Press, 2005.
[7] A. Guez and J. Pineau. Multi-Tasking SLAM. In *ICRA*, 2010.
[8] L. Guibas, D. Hsu, H. Kurniawati, and E. Rehman. Bounded Uncertainty Roadmaps for Path Planning. In *WAFR*, 2008.
[9] K. Hauser. Randomized Belief-Space Replanning in Partially-Observable Continuous Spaces. In *WAFR*, 2010.
[10] K. Hsiao, L.P. Kaelbling, and T. Lozano-Perez. Grasping POMDPs. In *ICRA*, pages 4685–4692, 2007.
[11] D. Hsu, W.S. Lee, and N. Rong. What makes some POMDP problems easy to approximate? In *NIPS*, 2007.
[12] S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for optimal motion planning. In *RSS*, 2010.
[13] T. Kollar and N. Roy. Efficient Optimization of Information-Theoretic Exploration in SLAM. In *AAAI*, pages 1369–1375, 2008.
[14] H. Kurniawati, Y. Du, D. Hsu, and W.S. Lee. Motion planning under uncertainty for robotic tasks with long time horizons. *IJRR*, 30(3):308–323, 2011.
[15] H. Kurniawati, D. Hsu, and W.S. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *RSS*, 2008.
[16] P. Missiuro and N. Roy. Adapting probabilistic roadmaps to handle uncertain maps. In *ICRA*, 2006.
[17] C.H. Papadimitriou and J.N. Tsitsiklis. The Complexity of Markov Decision Processes. *Math. of Operation Research*, 12(3):441–450, 1987.
[18] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *IJCAI*, pages 1025–1032, 2003.
[19] J.M. Porta, N. Vlassis, M.T.J. Spaan, and P. Poupart. Point-Based Value Iteration for Continuous POMDPs. *JMLR*, 7(Nov):2329–2367, 2006.
[20] S. Prentice and N. Roy. The Belief Roadmap: Efficient Planning in Linear POMDPs by Factoring the Covariance. In *ISRR*, 2007.
[21] T. Smith and R. Simmons. Point-based POMDP algorithms: Improved analysis and implementation. In *UAI*, July 2005.
[22] C. Stachniss, G. Grisetti, and W. Burgard. Information Gain-based Exploration Using Rao-Blackwellized Particle Filters. In *RSS*, pages 65–72, 2005.
[23] S. Thrun. Monte carlo POMDPs. In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *NIPS 12*, pages 1064–1070. MIT Press, 2000.
[24] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, 2005.

## Appendix

To prove Theorem 1 and Theorem 2, we use $\alpha$-function as the policy representation. The policy $\pi$ is represented by a set of $\alpha$-functions $\Gamma$, where $\pi(b) = \arg\max_{\alpha\in\Gamma} \int_{s\in S} \alpha(s)\cdot b(s)ds$. Each $\alpha$-function corresponds to a policy tree $T_\alpha$. Each node in $T_\alpha$ corresponds to an action and each edge corresponds to an observation. The value $\alpha(s)$ is the expected total reward of executing $T_\alpha$ from $s$. Let $a_0$ denotes the root of $T_\alpha$ and let's use the same notation to denote a node and its corresponding action. Then, executing $T_\alpha$ starting from $s$ means that the robot at state $s$ starts execution by performing $a_0$. An arc from $a_0$ to a node at the next level of $T_\alpha$ is followed, based on the observation perceived. Suppose the arc points to node $a_1$, then at the next step, the robot performs $a_1$. This process is repeated until a leaf node is reached. The value $\alpha(s)$ can be written as,

$$\begin{aligned}\alpha(s) &= R(s,a_0) + \\ &\gamma \int_{s_1\in S}\int_{o\in O} T(s,a_0,s_1)Z(s_1,a_0,o_1)\alpha_{a_0o}(s_1)dods_1 \quad (4)\end{aligned}$$

where $\alpha_{a_0o}$ is the $\alpha$-function that corresponds to the sub-tree of $T_\alpha$ whose root is the child of $a_0$ via edge $o$.

### A. Proof of Theorem 1

To prove the theorem, we first show that for any $\alpha$-vector, $\alpha(s) = \alpha(g(s))$. For this purpose, we show that for any function $f: S \to S$ that does not change the robot's relative configuration with respect to the environment, $\alpha(s) = \alpha(f(s))$. Since $g$ is an instance of such function, $\alpha(s) = \alpha(g(s))$, too.

We prove $\alpha(s) = \alpha(f(s))$ by induction on the levels of $T_\alpha$. When $T_\alpha$ has only one level, $\alpha(s) = R(s,a_0)$. Since the reward function depends only on the relative configuration of the robot and since applying $f$ to $s$ does not change this relative configuration, $\alpha(s) = R(s,a_0) = R(f(s),a_0) = \alpha(f(s))$. Assume that for any $f$, any $\alpha$, and any $s\in S$, $\alpha(s) = \alpha(f(s))$ when $T_\alpha$ has $i$ levels. Now, we show that $\alpha(s) = \alpha(f(s))$ when $T_\alpha$ has $(i+1)$ levels. The key is to show that applying $f$ to $s$ does not change the integration term in (4). Let's first look at the transition function. Based on property-1 of LS-continuous, we have $T(s,a_0,s_1) = T(f(s),a_0,s'_1)$ where $s'_1 = s_1 + (f(s)-s)$. Since, the displacement vector $(f(s)-s)$ does not change the relative robot's configuration with respect to the environment, the relative robot's configuration at $s_1$ is the same as that at $s'_1$. Since $Z$ depends only on the relative robot's configuration, $Z(s_1,a_0,o) = Z(s'_1,a_0,o)$ for any $o\in O$. Using the result from level-$i$, $\alpha(s_1) = \alpha(s'_1)$. Hence, the integration term of (4) for $\alpha(s)$ and $\alpha(f(s))$ are the same. This result and the fact that the reward function depends only on the robot's relative configuration, gives us $\alpha(s) = \alpha(f(s))$.

Now, we prove that for any policy $\pi$, $V_\pi(b) = V_\pi(Transform(b))$. Let $\mathscr{R}$ be a partition of $S$, such that each set $R_s \in \mathscr{R}$ consists of all states in $S$ where the robot's relative configurations with respect to the environment, is the same as that of $s$. This means that each state in the same set of $\mathscr{R}$ has the same $\alpha$-value. And hence we can write,

$$V_\pi(b) = \max_{\alpha\in\Gamma}\int_{s\in S}\alpha(s)\cdot b(s) = \max_{\alpha\in\Gamma}\int_{R_s\in\mathscr{R}}\alpha(s)\int_{s'\in R_s} b(s'). \quad (5)$$

From the definition of $Transform$ in (2), $(Transform(b))(s) = \int_{s'\in R_s} b(s')$. Therefore, (5) can be rewritten as $V_\pi(b) = \max_{\alpha\in\Gamma}\int_{R_s\in\mathscr{R}}\alpha(s)\cdot(Tranform(b))(s) = V_\pi(Transform(b))$, which is the result we want. □.

### B. Proof of Theorem 2

To prove Theorem 2, we first need the following lemma.

**Lemma 1** *In an LS-continuous POMDP with parameter $(K_{RS},K_Z)$ and normalized observation space, for any $\alpha$-function and any state $s,s'\in S$, $|\alpha(s)-\alpha(s')| \le \left(\frac{K_{RS}}{1-\gamma}+\frac{\gamma K_Z R_{\max}}{(1-\gamma)^2}\right)D_S(s,s')$.*

Proof of Lemma 1. Using the definition of $\alpha$ value in (4) and the triangle inequality, we have

$$\begin{aligned}|\alpha(s)-\alpha(s')| \le\ & \left|R(s,a_0)-R(s',a_0)\right| + \\ &\gamma\left|\int_{s_1\in S}\int_{o\in O}T(s,a_0,s_1)Z(s_1,a_0,o)\alpha_{a_0o}(s_1)dods_1 - \right. \\ &\left.\int_{s_1\in S}\int_{o\in O}T(s',a_0,s_1)Z(s_1,a_0,o)\alpha_{a_0o}(s_1)dods_1\right| \quad (6)\end{aligned}$$

Based on property-3 of *LS*-continuous, we can bound the first absolute term on the right hand side of (6) as $|R(s,a_0)-R(s',a_0)| \le K_{RS}D_S(s,s')$.

Now, we bound the second absolute term on the right hand side of (6). Let $d = s'-s$. Property-1 of *LS*-continuous gives us $T(s,a_0,s_1) = T(s',a_0,s_1+d)$. Hence, we can rewrite the last absolute term in (6) as,

$$\left|\int_{s_1\in S}T(s,a_0,s_1)\int_{o\in O}\big(Z(s_1,a_0,o)\alpha_{a_0o}(s_1) - Z(s_1+d,a_0,o)\alpha_{a_0o}(s_1+d)\big)dods_1\right|.$$

Using property-2 of *LS*-continuous, we have $Z(s_1+d,a_0,o) \ge Z(s_1,a_0,o) - K_Z\cdot D_S(s_1,s_1+d)$.

Substituting the above bounds to (6) gives

$$\begin{aligned}|\alpha(s)-\alpha(s')| \le\ & K_{RS}\cdot D_S(s,s') + \gamma\left|\int_{s_1\in S}T(s,a_0,s_1)\right. \\ &\int_{o\in O}Z(s_1,a_0,o)\left(\alpha_{a_0o}(s_1)-\alpha_{a_0o}(s_1+d)\right)do + \\ &\left.\int_{o\in O}K_Z\cdot D_S(s_1,s_1+v)\alpha_o(s_1+d)do\right| \\ \le\ & \left(\frac{K_{RS}}{1-\gamma}+\frac{\gamma K_Z R_{\max}}{(1-\gamma)^2}\right)D_S(s,s')\end{aligned}$$

The last inequality holds, after $\alpha_{a_0o}$ is expanded recursively and assuming that $O$ is normalized. □.

Now, we proof Theorem 2. Let $V_\pi(b) = \alpha\cdot b$ and $V_\pi(b') = \alpha'\cdot b'$. Then, there must always be a point $b_c = ab+(1-a)b'$ such that $\alpha\cdot b_c = \alpha'\cdot b_c$, as $\alpha\cdot b \ge \alpha'\cdot b$ and $\alpha'\cdot b' \ge \alpha\cdot b'$

$$\begin{aligned}|V^*(b)-V^*(b')| &= |\alpha\cdot b - \alpha'\cdot b'| \\ &\le |\alpha\cdot b-\alpha\cdot b_c| + |\alpha'\cdot b_c - \alpha'\cdot b'| \quad (7)\end{aligned}$$

Suppose $f$ is the joint density function used in computing $W_D(b,b_c)$ with $b(s) = \int_{s'\in S}f(s,s')ds'$ and $b_c(s') = \int_{s\in S}f(s,s')ds$. And suppose $g$ is the joint density function used in computing $W_D(b_c, b')$ with $b_c(s) = \int_{s' \in S} g(s,s')ds'$ and $b'(s') = \int_{s \in S} g(s,s')ds$. Then, we can rewrite (7) as,

$$\begin{aligned}
&|V^*(b) - V^*(b')| \quad \leq \\
&\left| \int_{s\in S} \alpha(s) \int_{s'\in S} f(s,s')ds'ds - \int_{s'\in S} \alpha(s') \int_{s\in S} f(s,s')dsds' \right| + \\
&\left| \int_{s\in S} \alpha'(s) \int_{s'\in S} g(s,s')ds'ds - \int_{s'\in S} \alpha'(s') \int_{s\in S} g(s,s')dsds' \right| \\
&\quad \leq \int_{s\in S}\int_{s'\in S} f(s,s') \left|\alpha(s) - \alpha(s')\right| ds'ds + \\
&\quad\quad \int_{s\in S}\int_{s'\in S} g(s,s') \left|\alpha'(s) - \alpha'(s')\right| ds'ds
\end{aligned}$$

Substituting the difference between $\alpha$ values in the above inequality with the result of Lemma 1, and using the definition of Wasserstein distance give us,

$$|V^*(b) - V^*(b')| \quad \leq \quad \left( \frac{K_{RS}}{1-\gamma} + \frac{\gamma K_Z R_{max}}{(1-\gamma)^2} \right) \left( W_D(b, b_c) + W_D(b_c, b') \right)$$

Using the convexity property of $W_D$, we get the desired result. $\square$.

*C. Proof of Theorem 3*

To proof Theorem 3, we first need the following lemma that bounds the error generated by a single backup operation.

**Lemma 2** *Consider a POMDP that satisfies LS-continuous with parameter $(K_{RS}, K_Z)$ and LA-continuous with parameter $(K_{RA}, h)$. Suppose the sampling dispersion in each element of $\mathscr{P}$ is $\leq \delta_A$. Then, the error generated by a single simplified GCS backup at a belief b is bounded as,* $|HV(b) - \hat{H}_b V(b)| \leq K_{RA}\delta_A + \gamma \left( \frac{K_{RS}}{1-\gamma} + \frac{K_Z R_{max}}{(1-\gamma)^2} \right) h(\delta_A)$.

Proof of Lemma 2. To shorten the proof writing, let's use the $Q$-value notation. For any $b \in B$ and any $a \in A$,

$$\begin{aligned}
Q(b,a) = &\int_{s\in S} R(s,a)b(s)ds + \\
&\gamma \int_{s\in S}\int_{s'\in S}\int_{o\in O} T(s,a,s')Z(s',a,o)b(s)\alpha(s')dods'ds \quad (8)
\end{aligned}$$

The single backup error is then,

$$\begin{aligned}
\left|HV(b) - \hat{H}_b V(b)\right| \quad &= \quad \max_{P\in\mathscr{P}} \max_{a\in P} Q(b,a) - \max_{P\in\mathscr{P}} \max_{a\in Samp(P)} Q(b,a) \\
&\leq \quad \max_{P\in\mathscr{P}} \left( Q(b, a^*_P) - Q(b, \hat{a}^*_P) \right) \quad (9)
\end{aligned}$$

where $Samp(P)$ is the sampled representation of $P \in \mathscr{P}$, $a^*_P = \arg\max_{a\in P} Q(b,a)$, and $\hat{a}^*_P = \arg\min_{a\in Samp(P)} D_P(a, a^*_P)$.

Let's compute $Q(b, a^*_P) - Q(b, \hat{a}^*_P)$ for an element $P$ of $\mathscr{P}$. For writing compactness, we drop the $P$ subscript. Using (8) and triangle inequality, we get

$$\begin{aligned}
Q(b,a^*) - Q(b,\hat{a}^*) \quad \leq \quad &\int_{s\in S} |R(s,a^*) - R(s,a)|\, b(s)ds + \\
\gamma \int_{s\in S} b(s) &\left| \int_{s'\in S}\int_{o\in O} T(s,a^*,s')Z(s',a^*,o)b(s)\alpha(s')dods' - \right. \\
&\left. \int_{s'\in S}\int_{o\in O} T(s,\hat{a}^*,s')Z(s',\hat{a}^*,o)\alpha(s')dods' \right| ds \quad (10)
\end{aligned}$$

Using property-3 of *LA*-continuous, we bound the first term in the right hand side as $\int_{s\in S} |R(s,a^*) - R(s,a)|\, b(s)ds \leq K_{RA}\delta_A$. Using property-1 of *LA*-continuous, $T(s,a^*,s') = T(s,\hat{a}^*,s' + f(a^*,\hat{a}^*)$. Therefore, (10) can be rewritten as,

$$\begin{aligned}
Q(b,a^*) - Q(b,\hat{a}^*) \quad \leq \quad K_{RA}\delta_A + \gamma &\left| \int_{s\in S}\int_{s'\in S} b(s)T(s,a^*,s') \right. \\
&\int_{o\in O} \Big( Z(s',a^*,o)\alpha(s') - \\
&\left. Z(s' + f(a^*,\hat{a}^*), \hat{a}^*, o)\alpha(s' + f(a^*,\hat{a}^*)) \Big) dods'ds \right|
\end{aligned}$$

Since $a^*$ and $\hat{a}^*$ belong to the same element of $\mathscr{P}$, using property-2 of *LA*-continuous, we get $Z(s' + f(a^*,\hat{a}^*), \hat{a}^*, o) = Z(s' + f(a^*,\hat{a}^*), a^*, o)$. Using property-2 of *LS*-continuous, we get $Z(s' + f(a^*,\hat{a}^*), a^*, o) \geq Z(s',a^*,o) - K_Z D_S(s', s' + f(a^*,\hat{a}^*))$. Using these properties and the assumption that $O$ is normalized, rearranging the above inequality gives us

$$\begin{aligned}
Q(b,a^*) - Q(b,\hat{a}^*) \quad \leq \quad K_{RA}\delta_A + \gamma &\left| \int_{s\in S}\int_{s'\in S} b(s)T(s,a^*,s') \right. \\
&\Big( K_Z D_S(s', s' + f(a^*,\hat{a}^*))\alpha(s' + f(a^*,\hat{a}^*)) + \\
&\left. \int_{o\in O} Z(s',a^*,o) \left( \alpha(s') - \alpha(s' + f(a^*,\hat{a}^*)) \right) do \Big) ds'ds \right| \\
&\leq K_{RA}\delta_A + \gamma \left( \frac{K_{RS}}{1-\gamma} + \frac{K_Z R_{max}}{(1-\gamma)^2} \right) h(D_P(a^*,\hat{a}^*))
\end{aligned}$$

The last inequality holds based on three properties: (1) any $\alpha$ value does not exceed $\frac{R_{max}}{1-\gamma}$, (2) property-1 of *LA*-continuous, i.e., $D_S(s', s' + f(a^*,\hat{a}^*)) \leq h(D_P(a^*,\hat{a}^*))$, and (3) Lemma 1.

Using the above inequality and the fact that for any $P \in \mathscr{P}$, $D_P(a^*_P, \hat{a}^*_P) \leq \delta_A$, and $h$ is an increasing function, $\left|HV(b) - \hat{H}_b V(b)\right| \leq K_{RA}\delta_A + \gamma \left( \frac{K_{RS}}{1-\gamma} + \frac{K_Z R_{max}}{(1-\gamma)^2} \right) h(\delta_A)$. $\square$.

Now, we can prove Theorem 3. The difference between the optimal value $V^*$ and the value $V_t$ computed by the simplified GCS after $t$ steps are,

$$|V^*(b) - V_t(b)| \leq \left|V^*(b) - V^*(b')\right| + \left|V^*(b') - V_t(b')\right| + \left|V_t(b') - V_t(b)\right|.$$

Applying Theorem 2 to $V^*$ and $V_t$, and bounding $|V^*(b') - V_t(b')| \leq \varepsilon_t$, we get

$$|V^*(b) - V_t(b)| \leq 4 \left( \frac{K_{RS}}{1-\gamma} + \frac{K_Z R_{max}}{(1-\gamma)^2} \right) \delta_B + \varepsilon_t. \quad (11)$$

To compute $\varepsilon_t$, notice that by definition, $V^*(b') = HV^*(b')$ and $V_t(b') \leq \hat{H}_b V_{t-1}(b')$. Hence, $|V^*(b') - V_t(b')| \leq \left|HV^*(b') - \hat{H}_b V_{t-1}(b')\right|$ and the following holds

$$\begin{aligned}
\left|V^*(b') - V_t(b')\right| \leq &\left|HV^*(b') - HV_{t-1}(b')\right| + \\
&\left|HV_{t-1}(b') - \hat{H}_b V_{t-1}(b')\right|. \quad (12)
\end{aligned}$$

Using the contraction property of $H$ and (11), we can bound the first absolute term on the right hand side of (12) as $|HV^*(b') - HV_{t-1}(b')| \leq \gamma \left( 4 \left( \frac{K_{RS}}{1-\gamma} + \frac{K_Z R_{max}}{(1-\gamma)^2} \right) \delta_B + \varepsilon_{t-1} \right)$. The last absolute term of (12) can be bounded using Lemma 2. As a result, we get

$$\left|V^*(b') - V_t(b')\right| \quad \leq \quad \gamma\left(4\left(\frac{K_{RS}}{1-\gamma} + \frac{K_Z R_{max}}{(1-\gamma)^2}\right)\delta_B + \varepsilon_{t-1}\right)$$
$$+\left(K_{RA}\delta_A + \gamma\left(\frac{K_{RS}}{1-\gamma} + \frac{K_Z R_{max}}{(1-\gamma)^2}\right)h(\delta_A)\right)$$

Expanding the recursion gives us,

$$\left|V^*(b') - V_t(b')\right| \leq \frac{1}{1-\gamma}\left(K_{RA}\delta_A + \left(\frac{K_{RS}}{1-\gamma} + \frac{K_Z R_{max}}{(1-\gamma)^2}\right)(4\delta_B + \gamma h(\delta_A))\right),$$

which is the result we want. □.