Multi-tasking SLAM

Arthur Guez[†] and Joelle Pineau[‡]
School of Computer Science, McGill University, Montréal, Canada
[†]*aguez@cs.mcgill.ca,* [‡]*jpineau@cs.mcgill.ca*

***Abstract*— The problem of simultaneous localization and mapping (SLAM) is one of the most studied in the robotics literature. Most existing approaches, however, focus on scenarios where localization and mapping are the only tasks on the robot's agenda. In many real-world scenarios, a robot may be called on to perform other tasks simultaneously, in addition to localization and mapping. These can include target-following (or avoidance), search-and-rescue, point-to-point navigation, refueling, and so on. This paper proposes a framework that balances localization, mapping, and other planning objectives, thus allowing robots to solve sequential decision tasks under map and pose uncertainty. Our approach combines a SLAM algorithm with an online POMDP approach to solve diverse navigation tasks, without prior training, in an unknown environment.**

I. INTRODUCTION

The simultaneous localization and mapping (SLAM) algorithms that have been developed in recent years allow a robot to map an environment in the absence of global position information. Generally, the purpose of applying a SLAM algorithm is to build an accurate map of the environment, in an autonomous manner or not, that can later be used to do motion planning under full map observability [1], [2], [3], [4], [5], [6], [7], [8]. There are several limitations with that paradigm. An obvious one is that pure map exploration must take place before the robot can do anything useful in the environment. This might not be an issue if the robot is confined to a small bounded working environment that can be explored quickly before starting a task, or if the robot can depend on another robot to perform the mapping process. However, many situations arise in which the robot has no means of determining the extent of its working environment *a priori*, and in which no map is available (e.g. a robot deployed for a search-and-rescue operation). Furthermore, in some scenarios, the robot's working environment may be task-specific, or can change dynamically. In such settings, it becomes difficult to justify a separate exploration period to get a map estimate. Moreover, it might not be possible to let the robot run freely everywhere in order to map the entire environment at once; there could be constraints on where the robot can go at any moment in time.

In an attempt to address some of these challenges, researchers have studied the problem of autonomous exploration for SLAM [9], [10], [11], [12], [13], [14]. However most of the work in this area focuses on producing exploration strategies that are designed to efficiently acquire the map, but are not able to handle parallel tasks, such as target-following (or avoidance), point-to-point navigation, or refueling, to name just a few.

Those issues suggest that the mapping process should be fully integrated with general task planning, providing a model of the environment which is sufficient for the robot's current task list, but which is not acquired at the expense of these other tasks. This paper proposes a decision-theoretic framework that is capable of solving a sequential decision-making task when the map, or part of it, is unavailable. In this framework, the set of possible tasks is not limited to motion planning tasks (e.g. reaching a predefined goal location). Instead, a task is defined in terms of a cost function that expresses the robot's task priority.

We frame this problem in the Partially Observable Markov Decision Process (POMDP) framework. This allows us to express a broad class of tasks in the context of planning under map and pose uncertainty. The planning balances the need for exploration with exploitation of the current model to solve a task. We conduct planning online at every step. This implies that no training time is necessary and that the robot's task and environment can be modified at any time. The robot can be dropped in an unknown environment and start acting immediately towards achieving its current goal; and multiple goals can be balanced or interleaved automatically.

Our framework can achieve optimal behavior under some assumptions (e.g. infinite resources). For more complex scenarios, we leverage well-known approximation techniques to make our planning algorithm computationally tractable. We employ a Rao-Blackwellized particle filter (RBPF) to maintain our posterior distribution over the map and past trajectories. We select actions at every time step using an online POMDP search method, and we direct our belief search using Rapidly-Exploring Random Trees (RRT). Our approach is evaluated in a rich simulation environment on a set of contrasting task domains.

II. METHODS

A. Problem Definition

Partially Observable Markov Decision Processes (POMDPs) provide a rich decision-theoretic framework to model our planning problem [15]. A POMDP is defined by a set of states S, a set of actions A, and a set of observations Z. The transition function $T : S \times A \times S \to [0, \infty]$ defines the discrete dynamics of the system as a conditional probability density $T(s,a,s') = p(s' \mid s,a)$, the probability of the next state s' given the current state s and an action a. The observations about the current state are generated according to an observation function $O : S \times A \times Z \to [0, \infty]$,

where $O(s',a,z) = p(z \mid s',a)$ is the conditional probability density over the observations given that we enter state s' after executing action a. If an agent (or robot) is acting in that environment, a reward function $R : S \times A \rightarrow \mathbb{R}$ can be defined that specifies the cost/reward obtained by the agent at each time step. The goal of the robot in that setting is to find an action selection strategy, denoted by π, that maximizes its expected sum of discounted rewards. In order to do that, the robot has to maintain a conditional probability distribution over its current state, given its history of observations and actions, $\{a_0, z_1, ..., a_{t-1}, z_t\}$, and an initial distribution over states, b_0. Formally, we define a belief state $b_t(s) = p(s_t = s \mid b_0, a_0, z_1, ..., a_{t-1}, z_t)$. The optimal policy π^* is defined by the following equation:

$$\pi^* = \arg\max_{\pi \in \Pi} E\left[\sum_{t=0}^{\infty} \gamma^t \sum_{s \in S} b_t(s) \sum_{a \in A} R(s,a)\pi(b_t,a) \mid b_0\right], \quad (1)$$

where $E[]$ denotes the mathematical expectation, γ is the discount factor, and $\pi(b_t,a)$ is the probability that action a is executed by policy π for the current belief b_t. The optimal policy can be computed by solving Bellman's equation [16].

In multi-tasking SLAM, the state space S is the product of the set of possible maps $\mathbf{M}$, the set of possible trajectories $\mathbf{X}$, and the set of additional planning states $\mathbf{P}$,

$$S = \mathbf{M} \times \mathbf{X} \times \mathbf{P}, \quad (2)$$

where each pose x_i in a trajectory $x_{1:t}$ is defined by (x, y, θ). We will limit ourselves in this work to a circular holonomic robot operating in a planar environment. Our map is represented as an occupancy grid [17]. We consider planning tasks such as target-following, point-to-point navigation and the like, which are commonly defined by a set of discrete planning states (the extension to continuous domains is mathematically straightforward, though computationally challenging except under simple distributional assumptions, e.g. linear-Gaussian). An action $a = (\Delta d, \theta)$ is executed by first rotating in place by an angle of θ and moving directly forward for a distance d. A local obstacle avoidance algorithm [18] is used to control the robot during the forward movement to provide real-time collision avoidance. Other local actions could be incorporated in our framework such as a low-level controller to perform a particular local task (e.g. grab an object).

The observations are obtained from laser readings, l, collected by a laser range-finder mounted on the robot, and from the relative odometry measurements, u. The observation function is defined by $p(l,u \mid m, x_{1:t+1}, a_t) = p(l \mid x_{t+1}, m)p(u \mid a, x_t, m)$. Here again, richer measurements can be incorporated in a straightforward manner.

The state-to-state transition function is defined by $p(x_{t+1} \mid m, x_t, a)$. It is a combination of the robot's motion model and the handling of obstacles from the map. We assume the possibility to sample from that transition function when no obstacle is present; we do not try to model the position of the robot when a collision occurs, as we always try to avoid collisions.

The reward function $R : S \times A \rightarrow \mathbb{R}$ is task dependent. It could give a positive reward for going to some goal location, exploring some part of the map, and so forth. Negative rewards can be incorporated (e.g. for traversing dangerous zones, for performing expensive maneuvers, etc.)

The prior belief $b_0(x,m)$ incorporates a distribution over the starting position of the robot, and a prior over each grid cell being occupied (e.g. $\Pr(m_i) = 0.5$ in the case of an unexplored environment). The prior distribution over maps can be adapted depending on the kind of environment the robot is expected to visit, or when an approximate or partial map of the environment is available.

B. State Estimation

The problem of state estimation is to maintain a posterior distribution over states given the past observations; in our framework this corresponds to tracking the belief state, b_t. To track the component of the state describing the map and the pose, we need to solve the SLAM problem. Given the state representation outlined in the previous section, it is not possible to do this exactly. Rather, we approximate the distribution over pose and map using a Rao-Blackwellized particle filter (RBPF) [2]. The RBPF is a popular particle filter that takes advantage of the structure of a problem to reduce the variance of the estimation process. In a RBPF, each particle represents a possible trajectory $x_{1:t}$ and map m. The key idea is that the structure of the problem is used to decompose the posterior over maps and trajectories using the chain rule and the independence of the map and odometry measurements given the past trajectory:

$$p(x_{1:t}, m \mid l_{1:t}, u_{0:t}) = p(m \mid x_{1:t}, l_{1:t})p(x_{1:t} \mid l_{1:t}, u_{0:t}), \quad (3)$$

where $l_{1:t}$ is the history of laser observations and $u_{0:t}$ is the history of odometry measurements. Then the conditional posterior distribution $p(m \mid x_{1:t}, l_{1:t})$ is analytically tractable and the marginal posterior distribution, $p(x_{1:t} \mid l_{1:t}, u_{0:t})$, which needs to be estimated, lies in a space of reduced dimension. We follow the approach in [5], which uses a Sampling Importance Resampling (SIR) filter to estimate $p(x_{1:t} \mid z_{1:t}, u_{0:t})$ and to maintain the particles. In that approach, an improved proposal distribution is used by approximating the optimal proposal distribution with respect to the variance of the particle weights under the Markov assumption. This proposal distribution is computed on the fly and depends on the particle it is used for. This leads to a more robust filter compared to approaches that only use a fixed proposal distribution. Furthermore, resampling of the importance weights $w^{(i)}$ is only carried out if the effective number of particles N_{eff} drops below a threshold; where N_{eff} is defined as

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} (w^{(i)})^2}, \quad (4)$$

with N being the number of particles. This improved RBPF requires fewer particles to accurately update the posterior based on new observations.

For the task-related planning state, p, we assume that it can be computed analytically when conditioned on the map, the past trajectory, and past observations. Therefore an estimate of p can be associated with each particle of the RBPF filter. In scenarios where this assumption is not realistic, a separate particle filter can be associated with each of the RBPF particles to approximate the planning state distribution. In the tasks presented in Section III, the planning state space, $\mathbf{P}$, has some discrete components, that encode the status of the robot with respect to its different targets, and some continuous components such as the position of the other robot. Note that, even if $\mathbf{P}$ is finite, the global state space S on which the planning is executed is continuous and high-dimensional.

C. Action Selection

Recall that the goal of our framework is to help the robot select actions that allow an optimal trade-off between localization, mapping, and task planning. Now that we have defined a method for estimating the state of the robot and environment, the only remaining problem is to define the algorithm used to optimize the policy π. A common method to select optimal decisions in a POMDP is to rely on an *offline* algorithm that finds the best action to execute in all possible situations before acting in the environment, i.e. that finds a near-optimal policy $\pi(b,a)$, for all b and a[19]. In large environments, solving for the entire policy offline is intractable. Other *online* POMDP search algorithms only consider the current situation and search within a limited horizon for the best plan to execute. Those online methods scale better to larger problems and have the advantage of being able to adapt to dynamic environments. We refer the reader to [20] for a survey on the subject. Yet few online POMDP methods are able to handle continuous domains. In [21], a practical online algorithm to tackle Bayesian reinforcement learning in continuous POMDPs is described and applied to a simple robotic planning task. At each step, using the current posterior distribution, their online planning algorithm samples sequences of actions and observations to recursively expand a search tree in belief space. They estimate the value of the leaf nodes using a heuristic and select the action that maximizes the sum of discounted rewards over the fixed horizon defined by the tree.

Our planning approach is inspired by this work. In our case, we maintain the posterior distribution as described in Section II-B, which implies that our belief updates are computationally expensive operations. The complexity of the planning algorithm in [21] is exponential in the depth of the search tree and depends heavily on the complexity of the particle filter update; that means that our planning algorithm would not be computationally tractable if we wanted a deep search tree. In many environments with obstacles, shallow search trees will provide poor estimates of global navigation values. Therefore, we need a method that handles complex belief representations, while preserving the ability to search deeply in belief space.

As in [20], [21], we build a search tree in which the top node is the current belief, and which branches on actions and observations. We limit the branching factor by sampling a single observation (at random) from the observation function. This has the effect of increasing the variance of the search process, but has substantial computational advantages.

We further limit the branching factor by selecting very few actions at each node (with the exception of the root node, where more actions are considered). This pruning operation can introduce a substantial bias. To limit this bias, in the context of navigation tasks, it seems important that the pose estimate of all the reached belief nodes be distributed evenly in the set of all collision-free poses, C_{free}. Thus we need to decide which actions to select at each belief node in order to avoid obstacles while preserving the desired spreading effect over the belief.

Searching for control actions in the belief space is impractical, so instead we extract the *expected* map and pose from the current posterior distribution, and use these to generate conventional (deterministic, fully observable) search trees in this expected space. Those search trees are then used as a heuristic to direct the search in belief space.

Using the particle filter described above, we define the expected pose and map as follows. Let the probability that cell $\bar{m}_{xy}$ is occupied be defined, as in [22], by

$$p(\bar{m}_{xy} \mid l_{1:t}, u_{0:t}) = \int p(m_{xy} \mid x_{1:t}, l_{1:t}) p(x_{1:t} \mid l_{1:t}, u_{0:t}) dx_{1:t}, \tag{5}$$

$$p(\bar{m}_{xy} \mid l_{1:t}, u_{0:t}) \approx \sum_{i=1}^{N} w^{(i)} p(m_{xy} \mid x_{1:t}^{(i)}, l_{1:t}). \tag{6}$$

The expected pose at step t is defined by

$$\bar{x}_t = \int x_t p(x_{1:t} \mid l_{1:t}, u_{0:t}) dx_{1:t}, \tag{7}$$

$$\bar{x}_t \approx \sum_{i=1}^{N} w^{(i)} x_t^{(i)}. \tag{8}$$

Using this expected pose and map estimate, we can leverage existing algorithms for planning in observable, deterministic environments to select the (heuristically) best action for the current belief. For tasks requiring motion planning, a good choice to plan in C_{free} are the Rapidly-Exploring Random Trees (RRTs) [23]. They are fast to compute and are able to cover the space with a small amount of nodes.

We grow M RRTs of K points using the expected map $\bar{m}$ and starting from the expected pose $\bar{x}_t$, as extracted from the current posterior distribution. The operation of growing a RRT is referred to as BUILD_RRT in Alg. 1, and is similar to the one presented in [23]. We assume this sub-routine returns a graph G, representing the RRT, that defines collision-free paths from the current (expected) pose, given the current (expected) map. We then utilize the paths in the RRTs to select action branches when recursively expanding the search tree over belief space.

Given a current belief, in order to go towards an RRT node at position (x,y), we select an action $a = (\Delta d, \theta)$ that would move the robot from its expected pose $\bar{x}$ towards (x,y) and check (independently for all particles) that this action

does not lead to a collision. We then sample a particle i according to the current weight distribution and sample the laser readings l and odometry measurements u that would be obtained if a was executed in the model defined by the ith particle. Next, we run a belief update with those new observations, add the new belief node to the search tree, and repeat the same steps until $\bar{x}$ is close enough to (x,y). Those steps are described in Alg. 3. Throughout the tree expansion process, we collect the trajectories up to all belief nodes and backup their estimated rewards to the root. A heuristic estimate of a belief value is provided by $\hat{V}()$. We then apply the max operator on all the sampled action values Q to obtain the best action, *bestA*, to execute.

With the method described above, we are able to achieve a deeper search with less belief updates, but at the cost of getting less accurate estimates of the expected value of each action. However, we believe that this trade-off is well-adapted for many navigation tasks. Furthermore, because the planning is done online, the system is able to correct for errors at a later step.

For cases where the robot's planning task, represented in the state space by **P**, is not of a navigation type, another deterministic planning algorithm can be leveraged in a similar way. For example, when the task deals with discrete entities, a STRIPS-type planner [24] could be integrated instead of the RRTs.

Algorithm 1: Online planning algorithm

Input: Belief represented as particles p and weights w, K, M
Output: *bestA*

for $i = 1$ *to* M **do**
  Extract $\bar{m}$ and $\bar{x}$ from belief represented by p and w
  $G_i \leftarrow BUILD\_RRT(K, \bar{m}, \bar{x})$
  Create (global) array Q of sampled action values
  Create empty trajectory t, a list of actions
  $EVAL\_TREE\_PATH(G_i.root, Q, t, 0, 0, p, w)$
end
$maxQ \leftarrow -\infty$
for $i = 1$ *to* $|Q|$ **do**
  if $Q_i.q > maxq$ **then**
    $maxq \leftarrow Q_i.q$
    $bestA \leftarrow Q_i.a$
  end
end

Algorithm 2: EVAL_TREE_PATH

Input: Graph node n, Q, t, q, $step$, p, w
forall *children* n' *of* n **do**
  $\{step', t', q', p', w'\} \leftarrow$ EVAL_EDGE($Q, t, step, p, w, q$,position of n')
  EVAL_TREE_PATH($n', Q, t', q', step', p', w'$)
end

Algorithm 3: EVAL_EDGE

Input: Q, t, $step$, p, w, q, $target$
Extract $\bar{x}$ from belief represented by p and w.
while $|\bar{x} - target| > \Delta d$ **do**
  $\theta \leftarrow ANGLE\_TO\_TARGET(\bar{x}, target)$
  $a \leftarrow (\Delta d, \theta)$
  if *collision*(p, w, a) **then**
    break
  end
  $step \leftarrow step + 1$
  Sample i based on w distribution
  Sample observation l, u from $p(l, u \mid p^i.m, p^i.x, a)$
  $p, w \leftarrow UpdatePosterior(p, w, l, u)$
  $\hat{V} \leftarrow \hat{V}(p, w)$
  Add a to t
  Add $(t_0, q + \gamma^{step}\hat{V})$ to Q
  $q \leftarrow q + \gamma^{step} \sum_{l=1}^{N} w^{(l)} R(p^{(l)}, a)$
  Extract $\bar{x}$ from belief represented by p and w.
end
return $\{step, t, q, p, w\}$

D. Dealing with Uncertainty

When computing expected rewards over the search tree, the uncertainty in the belief should be taken into account automatically during the decision process. However, the approximations introduced in the planning process (sparse sampling of actions, particle filter tracking, finite planning horizon) can introduce some errors in how the uncertainty over the expected reward is calculated in this framework. For example, consider the comparison of two sampled trajectories in the search tree with respect to their distance to a goal location. If the localization uncertainty at the end of one trajectory is high but its expected distance to the goal is smaller than for the other trajectory, then the first trajectory will be chosen even though the second trajectory has lower localization uncertainty. Moreover, the finite planning horizon does not allow the algorithm to realize the long-term consequence of localization uncertainty for the decision process. If the localization uncertainty grows too large so that there are not enough particles to keep track of all the hypotheses, then considerable errors in the map can appear, which in turn might translates to poor decision making and poor rewards.

Therefore, it is helpful to use additional statistics, obtained while propagating beliefs in the search tree, in order to handle the uncertainty explicitly. According to [22], the mean information of the expected map (EMMI) is a robust measure of uncertainty for RBPFs. While it seems to be a good candidate for pure state estimation scenarios, it is too expensive to compute many times, as is required for planning (in the expansion of the search tree). Instead, we use a computationally cheaper measure, the standard deviation σ_{dist} of the distance to the expected pose, which we find to

be correlated in most cases to the EMMI. Let σ_{dist} be defined by:

$$\sigma_{\text{dist}} = \sqrt{\int p(x_{1:t} \mid l_{1:t}, u_{0:t})(x_t - \bar{x}_t)^2 dx_{1:t}}, \quad (9)$$

$$\approx \sqrt{\sum_{i=1}^{N} w^{(i)} \left((x_{t_x}^{(i)} - \bar{x}_{t_x})^2 + (x_{t_y}^{(i)} - \bar{x}_{t_y})^2 + (\alpha(x_{t_\theta}^{(i)} - \bar{x}_{t_\theta}))^2 \right)}, \quad (10)$$

with α a scaling factor to account for the fact that the angle θ is not in the same unit as the 2D coordinates. We integrate this into our planning algorithm by pruning paths that lead to high uncertainty (in visited areas). Moreover, if the localization uncertainty is already high before planning, such that $\sigma_{\text{dist}} > \sigma_{\max}$, we force the robot to perform a task of active localization until an acceptable level of uncertainty is reached again ($\sigma_{\text{dist}} < \sigma_{\min}$). This is similar to the place-revisiting and loop-closure actions that occur in autonomous exploration planning methods [11][22].

III. EXPERIMENTS

We investigate performance of the multi-tasking SLAM framework on two contrasting task domains. The experiments are performed in the CARMEN [25] and Player/Stage [26] simulated environments, assuming a Pioneer robot equipped with a laser range finder with 10m range. Throughout the experiments, we use the following parameters; the RBPF has $N = 30$ particles, the N_{eff} threshold is $N/2$ and the grid cell size is 0.1m, we build $M = 8$ RRTs of $K = 150$ points each, the distance of each move action is $\Delta d =$ 1m. Those parameters are chosen to roughly balance the computation time with the planning performance on an Intel Xeon QuadCore 2.66Ghz processor. The planning time to compute each action is a few seconds. The discount factor γ is set to 0.99. The active localization parameters are set to $\sigma_{\max} = 0.06$ and $\sigma_{\min} = 0.015$, set according to the number of particles. In order to save on computations, we approximate the *expected* map $\bar{m}$ and pose $\bar{x}$ with the most likely map and pose from the RBPF when constructing the RRT search trees.

A. Target-Following Experiment

In the first experiment, illustrated in Fig. 1, our robot (in light gray) needs to be as close as possible to a moving target (in black), but only when that moving target is in a certain region of the map, as specified by the black arrows on Fig. 1(a). This target is modeled as another robot with identical sensors moving randomly while avoiding obstacles. Note that this experiment happens in real-time, the target moves even when our robot pauses to select actions. Our robot gets a reward of $10 - d$ at every step, with d the euclidean distance between the two robots, when the moving target is in the specified region. When the moving target is not in the specified area, our robot needs to go to fixed target #1 to collect a reward of 10, then to fixed target #2 where it waits and collects a reward of 10 at every step (unless the moving target is in the specified area, in which case the reward is 0). Each action, except for the wait action, costs -0.1. To simplify the experiment, the robot can observe the exact position of the moving target at all times. Noise and motion tracking could be added for a more realistic experiment.

We assume the task takes place in the California Science Center (mezzanine level), using a map from the Radish data set [27]. The position of the targets and the region definition are represented as clickable objects on the Stage interface (left column in Fig. 1) and can be changed dynamically during the experiment. The left column in Fig. 1 shows the true state of the experiment from the simulator, with footprints of the robot and moving target from recent history. The right column in Fig. 1 displays the past trajectory estimate from the best particle along with its associated map estimate. The best sampled planning trajectory is displayed along with the estimated σ_{dist} at every step of that planning trajectory. A video of an experiment similar to the one in Fig. 1 is attached to this paper [28]. Fig. 2 shows the evolution of the information of the expected map (EMI)[1] and the sum of rewards from that experiment. The EMI of the map $\bar{m}$ is defined, like in [22], as:

$$\text{EMI}(\bar{m}) = \sum_{x,y} (1 - H(\bar{m}_{xy})), \quad (11)$$

where H is the entropy of a random variable. The EMI increases with exploration as long as the trajectory uncertainty is kept low. The time step of the three snapshots in Fig. 1 are indicated by dotted vertical lines in Fig. 2.

In Fig. 1(a), the moving target is not in the region defined by the arrows anymore, so our robot goes towards target #1. Notice that the planned trajectory in the corresponding Fig. 1(b) goes through target #1 and then in direction of target #2, but that trajectory is only collision-free in the current map estimate of the robot. The robot then acquires more map information and discovers a wall that blocks the sampled path, so it chooses another path to target #2, as can be seen in Fig. 1(d). During its travel from target #1 to target #2, the sum of rewards only decreases (see step 14 to 44 in Fig. 2) because the robot only gets the negative reward from the moving actions. However, the EMI increases between step 26 and 44 because the robot is exploring new parts of the map to get to target #2. Right after target #2 is reached, the moving target re-enters the area defined by the arrows, which causes the robot to start travelling back to that area in order to follow the other robot (see Fig. 1(e,f)).

B. Active Localization Experiment

This experiment showcases the active localization of the robot in order to keep the localization uncertainty within safe margins, as described in Section II-D. To maintain short planning time, we cannot afford a large number of particles in our RBPF. Therefore, we are particularly prone to particle depletion problems in the search process, as a result of losing the uncertainty in the trajectory estimate. The nodes in the

[1]Note that the EMI is different from the EMMI defined in Section II-D.

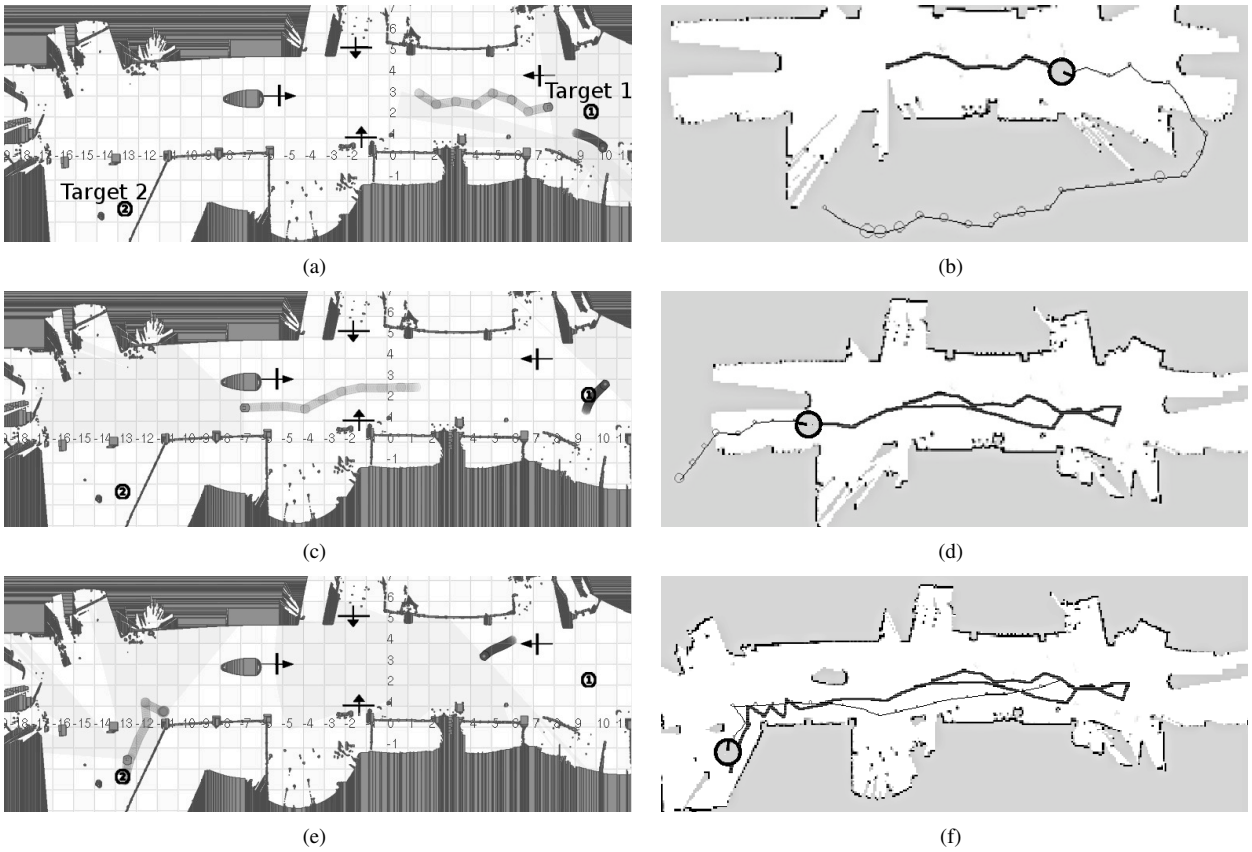


Fig. 1. Target following experiment. In the left column, the real state of the experiment from the simulator is displayed. Our robot is in light gray and the moving target is in black. The footprints show the recent past positions of the robot and the moving target. The black arrows define the region in which the moving target needs to be followed. The circled 1 and 2 are the two fixed targets. In the right column is the past trajectory of the robot represented by the thick black line, the robot represented by an oriented circle, and the best sampled planning trajectory represented by a thin black line. The σ_{dist} estimates are represented by circles along the sampled trajectory. The time steps corresponding to each of these three snapshots are indicated as vertical dotted lines in Fig. 2.

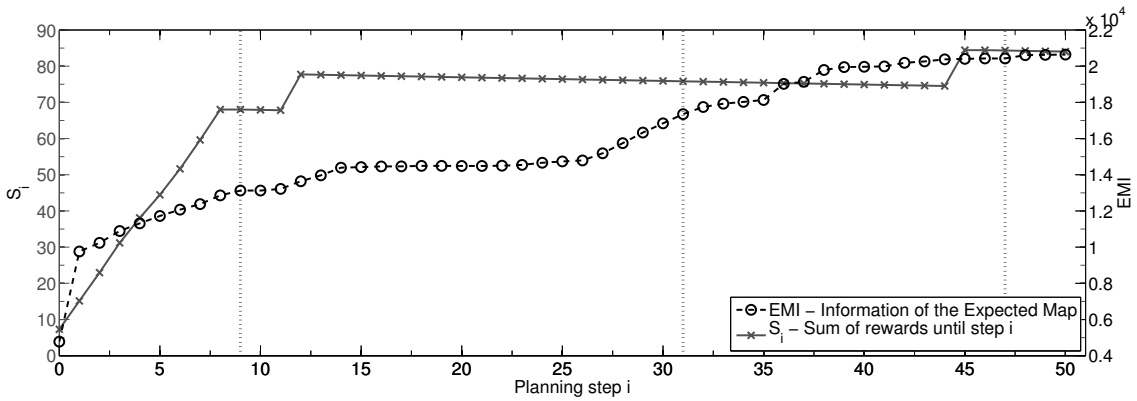
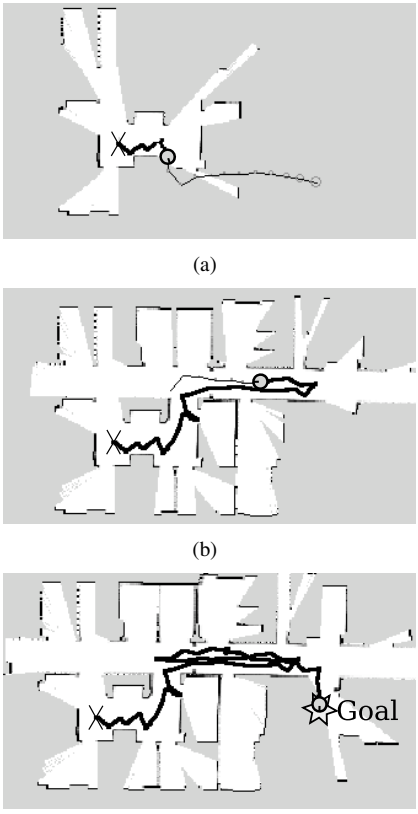


Fig. 2. Plot of the EMI and sum of rewards from the multitask experiment in Fig. 1. The vertical dotted lines correspond to the time steps of the three snapshots in Fig. 2

search tree that most suffer from particle depletion are difficult to distinguish from nodes at which proper relocalization is achieved, because most uncertainty measures will look alike in these situations. For this experiment, we overcome this problem by directing the robot towards its start position when selecting paths that reduce σ_{dist}. The experiment takes place in the hospital section map from the Stage simulator. Good localization is difficult to achieve in this map because of the presence of long corridors and perfectly straight walls.

In this task, illustrated in Fig. 3, the robot simply needs to reach a goal location (while mapping and localizing with sufficient accuracy to achieve this goal). The robot starts on the left in Fig. 3, and after a short exploration period, the robot enters the main corridor. At step 20 in Fig. 4, the task is



(a)

(b)

(c)

Fig. 3. Active localization experiment.

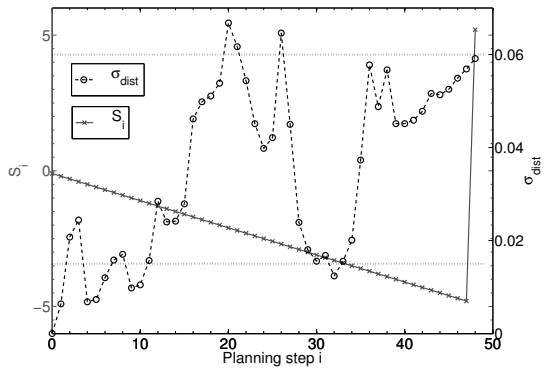


Fig. 4. Plot of the sum of rewards and σ_{dist} for the active localization experiment. The upper and lower dotted horizontal lines correspond to the σ_{max} and σ_{min} values, respectively.

switched to an active localization task because $\sigma_{\text{dist}} > \sigma_{\text{max}}$. The robot then selects actions to relocalize as can be seen in Fig. 3(b). At step 33 in Fig. 4, the active localization task ends because $\sigma_{\text{dist}} < \sigma_{\text{min}}$. The robot is then able to reach the goal (See Fig. 3(c)).

IV. RELATED WORK

Since SLAM is a challenging problem, much of the work is on passive approaches that focus on accurate belief estimation [1], [2], [3], [4], [5], [6], [7], [8]. The problem of autonomous exploration for SLAM has been visited by several researchers that proposed online [9], [10], [11], [12], [13] and offline [14] approaches. That problem can be considered a special case of the more general problem of planning under map and pose uncertainty which we are considering in this paper. Although our framework can handle autonomous exploration tasks quite well, we are not attempting to compete with those approaches since their planning algorithms are designed specifically for the autonomous exploration task.

Online POMDP planning applied to robot navigation was explored in [21] and [29], but not in the case of map uncertainty. Dealing with map uncertainty is a major obstacle to developing fast and robust planning methods for unknown and/or dynamic environments, where the planning algorithm has to deal with the computationally expensive map estimation process and a state space augmented with a high-dimensional component. Our contribution in that context is to provide a mathematically-principled framework that handles the map uncertainty and approximation methods to keep planning tractable.

Our approach of using RRTs in mean space to direct the belief search is similar in flavor to the Belief RoadMaps (BRMs) in which probabilistic roadmaps are used to plan in mean space of an Extended Kalman Filter [30]. Probabilistic RRTs for autonomous navigation in dynamic environment have been explored in [31] for local navigation to handle moving obstacles.

V. CONCLUSION

This paper proposes a novel framework for performing SLAM when the mapping and localization are not the primary focus of the robot. We describe a decision-theoretic framework capable of handling the map and pose uncertainty in conjunction with an online planning algorithm capable of solving diverse planning tasks in this setting. The experiments we conducted in simulated settings suggest that our approach would transfer to a real-world setting with minimal effort.

However, there are still open research problems relating to our approach. Most limitations of our work lie in the way we approximate the decision making process in order to keep the planning tractable. It would be useful to investigate how these approximation methods can be refined to improve the accuracy of the action selection process while conserving the tractability of the algorithm. Particle depletion problems could be alleviated by incorporating existing methodology on the subject such as in [32], but it remains unclear if this problem can be dealt with effectively within reasonable computational requirements. Other possible extensions include dealing with continuous planning states and a more diverse set of actions and sensors. This would augment the capabilities of the robot and allow it to learn a richer model of the environment, though likely at the expense of strong parametric assumptions.

VI. ACKNOWLEDGMENTS

The authors gratefully acknowledge support from the Natural Sciences and Engineering Council of Canada (NSERC) and the Fonds Québécois de la Recherche sur la Nature et les Technologies (FQRNT). We would also like to thank Gregory Dudek, Ioannis Rekleitis, Jordan Frank, and Robert West for their helpful suggestions and comments.

REFERENCES

[1] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," *Autonomous robot vehicles*, vol. 1, pp. 167–193, 1990.

[2] K. Murphy, "Bayesian map learning in dynamic environments," 1999, pp. 1015–1021.

[3] D. Hahnel, W. Burgard, D. Fox, and S. Thrun, "An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements," in *In Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003, pp. 206–211.

[4] A. Eliazar and R. Parr, "DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks," in *In Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2003, pp. 1135–1142.

[5] G. Grisetti, C. Stachniss, and W. Burgard, "Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling," in *In Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005, pp. 2432–2437.

[6] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *In Proc. of the National Conference on Artificial Intelligence (AAAI)*, 2002, pp. 593–598.

[7] J. Leonard and H. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 376–382, 1991.

[8] M. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, 2001.

[9] R. Sim and N. Roy, "Global a-optimal robot exploration in slam," in *In Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005, pp. 661–666.

[10] R. Sim and G. Dudek, "Online control policy optimization for minimizing map uncertainty during exploration," in *In Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2004, pp. 1758–1763.

[11] C. Stachniss, G. Grisetti, and W. Burgard, "Information gain-based exploration using rao-blackwellized particle filters," in *In Proc. of Robotics: Science and Systems (RSS)*, 2005, pp. 65–72.

[12] R. Martinez-Cantin, N. de Freitas, A. Doucet, and J. Castellanos, "Active policy learning for robot planning and exploration under uncertainty," in *In Proc. of Robotics: Science and Systems (RSS)*, 2007.

[13] R. Martinez-Cantin, N. de Freitas, E. Brochu, J. Castellanos, and A. Doucet, "A Bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot," *Autonomous Robots*, vol. 27, no. 2, pp. 93–103, 2009.

[14] T. Kollar and N. Roy, "Trajectory optimization using reinforcement learning for map exploration," *The International Journal of Robotics Research*, vol. 27, no. 2, p. 175, 2008.

[15] E. Sondik, "The optimal control of partially observable markov processes," Ph.D. dissertation, Stanford University, 1971.

[16] L. Kaelbling, M. Littman, and A. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[17] H. Moravec, "Sensor fusion in certainty grids for mobile robots," *AI magazine*, vol. 9, no. 2, p. 61, 1988.

[18] I. Ulrich and J. Borenstein, "VFH+: Reliable obstacle avoidance for fast mobile robots," in *In Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, 1998.

[19] J. Pineau, G. Gordon, and S. Thrun, "Anytime point-based approximations for large POMDPs," *Journal of Artificial Intelligence Research*, vol. 27, pp. 335–380, 2006.

[20] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa, "Online planning algorithms for POMDPs," *Journal of Artificial Intelligence Research*, vol. 32, pp. 663–704, 2008.

[21] S. Ross, B. Chaib-draa, and J. Pineau, "Bayesian reinforcement learning in continuous POMDPs with application to robot navigation," in *In Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2008, pp. 2845–2851.

[22] J. Blanco, J. Fernandez-Madrigal, and J. Gonzalez, "A Novel Measure of Uncertainty for Mobile Robot SLAM with Rao Blackwellized Particle Filters," *The International Journal of Robotics Research*, vol. 27, no. 1, p. 73, 2008.

[23] J. Kuffner Jr and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *In Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2000.

[24] R. Fikes and N. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, vol. 2, no. 3/4, pp. 189–208, 1971.

[25] M. Montemerlo, N. Roy, and S. Thrun, "Perspectives on standardization in mobile robot programming: The Carnegie Mellon navigation (CARMEN) toolkit," in *In Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003, pp. 2436–2441.

[26] B. P. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *In Proc. of the International Conference on Advanced Robotics (ICAR)*, 2003, pp. 317–323.

[27] A. Howard and N. Roy, "The robotics data set repository (radish)," 2003. [Online]. Available: http://radish.sourceforge.net/

[28] A. Guez and J. Pineau, "Multi-task slam experiment," [Video recording] Submitted to ICRA 2010 as an attachment to this paper.

[29] A. Foka and P. Trahanias, "Real-time hierarchical POMDPs for autonomous robot navigation," *Robotics and Autonomous Systems*, vol. 55, no. 7, pp. 561–571, 2007.

[30] S. Prentice and N. Roy, "The belief roadmap: Efficient planning in linear POMDPs by factoring the covariance," in *In Proc. of the International Symposium of Robotics Research (ISRR)*, 2007.

[31] C. Fulgenzi, A. Spalanzani, and C. Laugier, "Probabilistic rapidly-exploring random trees for autonomous navigation among moving obstacles," in *Workshop on safe navigation, IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[32] C. Stachniss, G. Grisetti, and W. Burgard, "Recovering particle diversity in a Rao-Blackwellized particle filter for SLAM after actively closing loops," in *In Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005, pp. 655–660.