2008 IEEE/RSJ International Conference on Intelligent Robots and Systems
Acropolis Convention Center
Nice, France, Sept, 22-26, 2008

RRT-SLAM for Motion Planning with Motion and Map Uncertainty for Robot Exploration

Yifeng Huang
yhuangf@cs.sfu.ca
RAMP Lab, Engineering Science,
Simon Fraser University

Kamal Gupta
kamal@cs.sfu.ca
RAMP Lab, Engineering Science,
Simon Fraser University

Abstract— **We address the motion planning (MP) subproblem that arises in a robotic exploration and mapping task. We consider sensing, localization and mapping uncertainties in the motion planning subproblem. The robot is holonomic with known size and shape, and is equipped with a laser range sensor. We use a rapidly exploring randomized tree (RRT) in conjunction with a simulated particle based Simultaneous Localization and Mapping (SLAM) algorithm to expand the tree. The simulated SLAM explicitly accounts for sensor, localization and mapping uncertainty in the planning stage. Moreover, the RRT itself is represented in the augmented configuration space where an extra dimension of uncertainty is used. The collision likelihood along a planned path is explicitly computed and is used to select a planned path. Preliminary simulations show the effectiveness and benefits of our integrated approach.**

I. INTRODUCTION

Autonomous exploration has been an active research area in robotics. In general, the exploration process is a loop[1], which consists of mainly three stages in each iteration: I) plan for robot's next (reachable) configuration to further explore its environment; II) execute the plan, i.e., move to the planned configuration and sense; and III) update the environment model and the robot's own position. Two subproblems are involved in stage I, the planning stage: I-a) where to go, also called the next best view (NBV) (view planning subproblem [1], [2], [3], [4]; and I-b) how to get there, i.e., determine a reachable path to get to the next viewpoint (motion planning subproblem).

Our proposed system is essentially a holonomic (the base can turn with a very small radius) mobile-manipulator[1] with known geometry. It is equipped with an on-board laser range sensor and moves in a static indoor environment. To plan its next motion, robot needs the knowledge/information about the environment (current map) and its state (location and orientation in the map), which are continually updated in stage III of each iteration in the exploration process mentioned before. Simultaneous Localization and Mapping (SLAM) algorithms [5], a key advance in the mobile robotics literature in past decade, solve this problem, and we use FastSLAM [6], a particle based algorithm to update and build the current map and the robot state. The map, the robot state, and the sensor, all have inherent uncertainty which the motion plan must take into account and that is the focus of this paper. We assume that an NBV algorithm such as the one presented in [4] is used to determine a viewpoint - a robot configuration (location and orientation) - in the global frame. The problem we address is how to plan robot's motion that will guide the robot from its current location to the viewpoint in the presence of localization, mapping and sensing uncertainty.

[1]Although, as a first step, the results in this paper are reported with the mobile base only.

We use a rapidly exploring random tree (RRT) to plan for a path because of the high dimensionality of the configuration space of the proposed mobile-manipulator system (eight degrees of freedom). Since we use FastSLAM to build and update the map and robot state, the uncertainty is inherently represented as a set of particles and hence it is natural to extend this particle based uncertainty representation to build and extend the RRT. Therefore, in order to extend the tree toward a randomly placed node, our RRT planner executes the very same, albeit simulated, particle based SLAM algorithm, in the process fully accounting for the localization, sensing and mapping uncertainties inherent in the robot-sensor system. Note that a planned path is no longer simply collision-free/in collision, instead a collision likelihood is associated with the path and used as travel cost. A key contribution of this paper is the formulation of this collision likelihood of a planned path within the particle filter based framework.

There exist planning approaches that take into account either localization uncertainty or mapping uncertainty, but not both at the same time. For instance, in [7], a simplified ellipse model of the localization uncertainty is used in conjunction with a modified A* search. Note that the collision status of a path is still binary (collision-free/in collision). A key contribution of their work is to show that the search space needs to incorporate extra uncertainty dimension(s) (towers of uncertainty) on top of C-space. [8] uses a similar approach, but with expected energy consumed as the travel cost. Our work differs in that we use particle based representation of uncertainty in the planning stage. Often, the distribution of robot location is complex and may not be well represented by the ellipse model [9]. In addition, they do not concern themselves with collision likelihood of the path.

[10] also addresses localization uncertainty, but searches only in C-space, mainly because they use vision based sensors pointing to the ceiling, such that the localization uncertainty at a given configuration is fixed and does not depend on the path travelled to reach that configuration. [11] considers only map uncertainty and explicitly evaluates the chance of collision in terms of collision likelihood for a given path. They do not take the localization uncertainty into consideration.

[12] proposes an integrated approach that uses sensor-based Random Tree (SRT) to build a roadmap of the explored area. The sensor covers the entire body of the robot, hence is not applicable to our system. In addition, the map uncertainty and collision probabilities are not addressed. The approach in [12] uses localization potential (which presumably would provide a fast approximation of simulated localization) to evaluate view candidates (in the local mode).

[13] presents an interesting extension of RRT, called particle RRT, to search for a feasible motion plan to the goal with robot motion uncertainty (due to unknown friction parameter, in their case). Each extension to the tree corresponds to a given command and is attempted several times under different conditions (corresponding to different friction co-efficient values drawn from a priori distribution). Nodes in the tree are created by clustering the results from these simulations. Our system is mounted with an on-board range sensor, which means we have to consider explicitly the sequence of observations in our planning stage, which is not addressed in [13]. Secondly, we address the travel cost in terms of collision likelihood along the path, which is not a concern in [13].

Our approach has similarity to recent work that incorporates exploration within SLAM framework, in particular, [3] and [14]. [14] also simulates SLAM for a given set of commands and chooses the one that maximizes information gain. However, the set of potential paths considered (the Voronoi diagram of the environment) is often small and the simulated SLAM is not integrated within an overall planning framework. For instance, when following a Voronoi diagram based path from A to B in Fig. 1 (shown as $path_1$), the robot's localization uncertainty may accumulate quickly along the path (since the robot can not localize itself well due to absence of objects within the sensing range along the path), and the robot could get "lost" and then remain lost as it reaches close to B and may collide with B. Our RRT-SLAM planner will return an alternative solution such as $path_2$, which, although longer, is a better choice, since it allows the robot to localize itself when traveling along the path with its on-board range sensor, thereby reducing the chance of collision caused by the localization uncertainty when approaching the goal area near position B. In addition, they are not concerned with collision likelihood of a planned path.

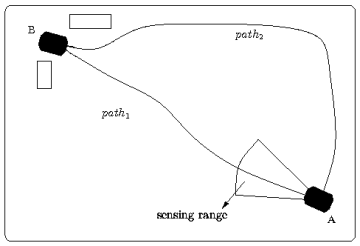


Fig. 1. Robot's motion planning in wide open area

It should also be noted that another general framework for planning in the presence of robot localization uncertainty is that of Partially Observable Markov Decision Problem (POMDP) and the solution is a "policy", which maps from belief states to actions over the belief space. Searching for the optimal policy is computationally expensive, and up until recently, was feasible for only a small number of states (few tens of states). Recently, two works [9][15] have been done that make higher dimensional POMDP solvable for practical problems. [15] shows a successful example in target tracking problem in known environment. The size of the belief space for realistic motion planning problems for exploration in unknown environments is prohibitively large for current POMDP algorithms.

II. Determining Collision likelihood along a planned path

We now describe how we evaluate the collision likelihood along a planned path while taking into account the localization, mapping and sensing uncertainty.

A. Simulated SLAM Framework

Let π denote a planned path. Let ν_π denote a possible trail the robot would follow, were it to execute the planned path π. So essentially, the planned path π corresponds to a distribution of a set of potential trails. This distribution depends on the possible potential maps and sensor readings. The collision likelihood of π is simply an expectation of collisions over these trails. We now set up the framework to calculate this collision likelihood within a particle filter based SLAM. We use the standard SLAM notation [5]. Let t denote the current time. Let κ_ν be the sequence of potential sensor measurements along trajectory ν. At times, we may omit the subscript ν (or π) for simplicity of notation. We use $q_{subscript}$ to denote the random variable associated with the robot state at the indicated $subscript$ (time or location), and $\overline{q}_{subscript}$ to denote an actual configuration indicated by the $subscript$. Denote m as the environment model. Let $\mathcal{V}$ (or Π) $= \{0, 1\}$ be a binary random variable to indicate the event that robot trail ν (or π) is collision-free (0) or not (1). We then have:

$$\begin{aligned} & p(\Pi = 0) \\ &= \iiint_{\nu m \kappa} p(\mathcal{V} = 0 | \nu, m, \pi, \kappa) p(\nu, m, \kappa | \pi) d\nu dm d\kappa \\ &= \iiint_{\nu m \kappa} p(\mathcal{V} = 0 | \nu, m) p(\nu, m | \pi, \kappa) p(\kappa | \pi) d\nu dm d\kappa \end{aligned} \quad (1)$$

Put in word, we have to consider the joint distribution of robot's potential trails, sensor readings, and environment model updates along the given path π in order to acquire the collision likelihood of following path π. The integrand comprises two terms, the first one where $p(\mathcal{V} = 0 | \nu, m)$ is the likelihood of robot trail ν being free in the environment model m, which is essentially a geometric calculation; and $p(\nu, m | \pi, \kappa) \cdot p(\kappa | \pi)$, which is to be acquired by running the particle filter based SLAM algorithm in simulation over possible observations.

As in FastSLAM [5], we use particles to represent the joint distribution of the environment model m and the trail ν. Let $m^{[i]}$ and $\nu^{[i]}$ be the environment model and trail in the i^{th} particle, respectively. Let $\sharp_S$ be the number of SLAM particles. Eq. 1 then becomes

$$\begin{aligned} & p(\Pi = 0) \\ &\doteq \int_\kappa \left[\sum_{i=1}^{\sharp_S} p(\mathcal{V} = 0 | \nu^{[i]}, m^{[i]}) p(\nu^{[i]}, m^{[i]} | \pi, \kappa) \right] p(\kappa | \pi) d\kappa, \end{aligned} \quad (2)$$

where $p(\mathcal{V} = 0 | \nu^{[i]}, m^{[i]})$ is the likelihood of robot trail $\nu^{[i]}$ being free in the environment model $m^{[i]}$; $p(\nu^{[i]}, m^{[i]} | \pi, \kappa)$, is acquired by running the particle filter based SLAM algorithm in simulation. Note that the sequence of control commands at each time instant t (denoted as a_t^π) is not explicitly specified, but instead it is derived from the planned path π, and previous robot state at $t - 1$, i.e.,

$a_t^\pi = \overline{q}_{next} - \overline{q}_t$, where $\overline{q}_t$ is the sample mean configuration of the particle set representing the robot state at time t, and $\overline{q}_{next}$ is robot's intermediate goal, for example the next node in the path, if the path were composed of a sequence of nodes connected by straight line segments. Thus, the "motion command model" in classic SLAM is replaced by a "path following model".

To determine $p(\Pi = 0)$, we consider a "test-and-evaluate" algorithm (similar to [14], [5]) that conducts several simulated SLAM sequences, each of which consists of a sequence of "simulated SLAM" steps: 1) draw a particle from the SLAM particle set with a draw probability equal to its weight; 2) apply the command a_t^π using the "path following model" and simulate SLAM, i.e., a) use the map and the robot position in the drawn particle to generate sensor measurement by ray-casting; and b) update the map and the robot position with SLAM particle filter.

The above two simulated SLAM steps are repeated until the robot is deemed to reach the end of the path π, i.e., the sample mean configuration of the particle set representing robot state is within a threshold distance of the configuration at the end of the path. This corresponds to one simulated SLAM sequence.

By repeating the simulated sequence M times, we can substitute the integration step over κ in Eq. 2 with summation. Let $\hat{\kappa}^{[j]}$ be the sequence of sensor measurements in j^{th} simulated sequence, and $\hat{\nu}^{[i,j]}, \hat{m}^{[i,j]}$ be the i^{th} particle in the j^{th} simulated SLAM sequence.

$$p(\Pi = 0) \doteq \sum_{j=1}^{M} \left[\sum_{i=1}^{\sharp_S} p(\mathcal{V} = 0 | \hat{\nu}^{[i,j]}, \hat{m}^{[i,j]}) p(\hat{\nu}^{[i,j]}, \hat{m}^{[i,j]} | \pi, \hat{\kappa}^{[j]}) \right] \cdot p(\hat{\kappa}^{[j]} | \pi) \quad (3)$$

The likelihood $p(\hat{\kappa}^{[j]}|\pi)$ is given by[14]:

$$p(\hat{\kappa}^{[j]}|\pi) \doteq \sum_{i=1}^{\sharp_S} p(\hat{\kappa}^{[j]} | \hat{\nu}^{[i,j]}, \hat{m}^{[i,j]}) . p(\hat{\nu}^{[i,j]}, \hat{m}^{[i,j]} | \pi) \quad (4)$$

The first term in the summation in Eq. 4 can be approximated by ray-casting operations performed in the map of each particle and the second term represents the weight of that particle.

We now discuss how to compute the collision likelihood for a given trail, i.e., the first factor term $p(\mathcal{V} = 0 | \hat{\nu}^{[i,j]}, \hat{m}^{[i,j]})$ in Eq. 3. We use occupancy grids, even though it demands extensive memory, primarily because it is particularly useful for path finding and facilities collision likelihood computation[2]. The value associated with each cell indicates the probability of the cell being occupied. The probability of a trail being collision free in a given occupancy grid map m is simply the product of probabilities of being free for all those cells swept by the robot along the trail in a given map.

Using Eqs. 3 and 4, we can now determine the collision likelihood for a planned path. The computation time will be M times the time taken to simulate particle based SLAM algorithm and the embedded collision computation costs. For grid based map, there exist SLAM algorithms that run at $O(\sharp_S{}^2 |A|)$ at each time instant [16], where $|A|$ is the size of area swept by the range finder.

In practice, the simulated SLAM steps, however, are relatively expensive in terms of computational cost, and are not practical for searching over a large amount of possible paths (such as in RRT) in an on-line manner. In our implementation, we propose two simplifications for on-line implementation on a real system: (i) we do not update the environment model during the simulated SLAM, and plan robot's motion based only on our current knowledge about the environment by ignoring possible information/knowledge gain about the environment through simulations, and (ii) we use the environment model, denoted as $\hat{m}^*$, in the most weighted SLAM particle (essentially the best estimate of the environment and the robot's location in it), instead of using all the maps in the SLAM particles. Effectively, this implies that instead of "simulated SLAM" we execute "simulated localization" with respect to the map in the most weighted SLAM particle. Such simplification saves the time cost on maps updating step in each time instant, which is the most time consuming part of the particle based SLAM algorithm.

B. Particle Based Simulated Localization

Let $\sharp_L$ be the number of particles from the simulated localization. Eq. 3 simplifies to

$$p(\Pi = 0) \doteq \sum_{j=1}^{M} \left[\sum_{i=1}^{\sharp_L} p(\mathcal{V} = 0 | \hat{\nu}^{[i,j]}, \hat{m}^*) p(\hat{\nu}^{[i,j]} | \hat{m}^*, \pi, \hat{\kappa}^{[j]}) \right] \cdot p(\hat{\kappa}^{[j]} | \hat{m}^*, \pi) \quad (5)$$

Although faster than Eq. 3, running multiple simulated localization sequences is still computationally expensive. We now discuss an alternative formulation (and the mathematical justification behind it) in which the trail particle sets could be sampled sequentially over time from the particle sets that represent robot state at each time instant in one simulated sequence. The collision likelihood when robot travels along the planned path could also be updated sequentially over time as we update the trail particle set.

From Eq. 5, summing over $\hat{\kappa}_j$, we have:

$$p(\Pi = 0) \doteq \sum_{i}^{\sharp_L} p(\mathcal{V} = 0 | \hat{\nu}^{[i]}, \hat{m}^*) p(\hat{\nu}^{[i]} | \hat{m}^*, \pi) \quad (6)$$

A key issue is to acquire particles that represent the distribution $p(\hat{\nu}|\hat{m}^*, \pi)$ in a sequential manner. First, we start with an approximation (similar to [14],[10]) of $p(q_t|\hat{m}^*, \pi)$ with $p(q_t|\hat{m}^*, \pi, \hat{\kappa}_t^*)$ ($\hat{\kappa}_t^*$ is the sequence of range readings received from time 0 to t in the simulated sequence):

$$p(q_t|\hat{m}^*, \pi) \doteq p(q_t|\hat{m}^*, \pi, \hat{\kappa}_t^*) \quad (7)$$

From now on we need to distinguish between the particles representing the whole trail (we will call them trail particles) and the particles representing the robot state (pose) at a single time instant t (we will call them state particles, and denote the particles set by $\mathcal{L}_t$). If Eq. 7 holds, we could run the simulated localization sequence once, and use the returned

[2]Other commonly used map representations, such as feature based map, could also be used, but the collision likelihood computation[11] is not as straightforward as in the grid based map case.

sets of robot state particles at each time instant as approximated representations of $p(q_0|\hat{m}^*, \pi), \cdots p(q_t|\hat{m}^*, \pi)$. A possible trail ν is then consists of a sequence of states $\nu = (q_0, \cdots, q_t)$, with each state q_t as a particle drawn from the state particle set $\mathcal{L}_t$ at successive time instants.

We now need to calculate the likelihood $p(\hat{\nu}|\hat{m}^*, \pi)$ for a single trail ν. Under the commonly used assumption of Markov property about robot localization [5], we have $p(q_0 \cdots q_t|\hat{m}^*, \pi) = p(q_t|q_{t-1}, \pi, \hat{m}^*) \cdots p(q_1|q_0, \hat{m}^*, \pi) p(q_0|\hat{m}^*, \pi,)$, which means we need to calculate the conditional likelihood $p(q_t|q_{t-1}, \hat{m}^*, \pi)$ for all the time instants to acquire the likelihood of the whole trail.

To determine $p(q_t|q_{t-1}, \hat{m}^*, \pi)$, consider two extreme cases: 1) no sensor reading, at time t, is available or the reading does not supply extra information about robot location. $p(q_t|q_{t-1}, \hat{m}^*, \pi)$ is then simply the "path following motion model", i.e., $p(q_t|q_{t-1}, a_t^{\pi})$; 2) the sensor reading gives enough information to localize the robot accurately at time t. In this case, the q_t is independent of q_{t-1}. Given that the range sensor is quite accurate, it tends to localize the robot quite well. Hence, we consider the two extreme cases as reasonable approximations in our case.

This is incorporated in our simulated localization algorithm (similar to [14]) as follows. Each trail particle is extended by applying the motion model. Then, the robot registers the observation at the latest robot state corresponding to the extended trail particle with the environment model and updates the likelihood of the trail particle based on the result of registration. Finally, the extended trail particles are re-sampled based on their updated likelihoods. As before, $\mathcal{L}_t$ denotes the state particle set representing robot state at t in the simulated sequence. If the range sensor reading localizes the robot accurately (case 2 mentioned in above paragraph), only a small number of particles will have descendants in the re-sample step. Let the number of particles in $\mathcal{L}_{t-1}$, who have decedents in $\mathcal{L}_t$ be ϕ_{t-1}. We consider the ratio $\eta_{t-1} = \frac{\phi_{t-1}}{\sharp_L}$. If it is lower than a threshold τ_p, we will assume q_{t-1} is independent of q_t. Otherwise, $p(q_t|q_{t-1}, \hat{m}^*, \pi)$ is determined by the "motion model". Formally,

$$p(q_t|q_{t-1}, \pi, \hat{m}^*) \doteq \begin{cases} p(q_t|q_{t-1}, a_{t-1}), & \eta_{t-1} \leq \tau_p \\ p(q_t|\hat{m}^*, \pi), & otherwise \end{cases} \quad (8)$$

where a_{t-1} are the sequence control commands to follow the path between $t-1$ and t.

Given the approximation techniques mentioned above, i.e., Eq. 7 and Eq. 8, we could sample the trail particles $\nu^{[i]}$ (used in Eq. 6) that represent the distribution $p(\hat{\nu}|\hat{m}^*, \pi)$, sequentially over time. Coupled with the collision likelihood calculation as in Sec-II-B.1, one can determine the collision likelihood for a planned path, (details in Section-II-B.3).

1) Collision Likelihood for a Trail: Since the laser range finder is quite accurate, in practice, we binarize the map $\hat{m}^*$ (denote the binarized version of $\hat{m}^*$ as m_b^*). Then, $p(\mathcal{V} = 0|\hat{\nu}^{[i]}, \hat{m}_b^*)$ becomes a binary distribution (collision/no-collision) and could be simply determined with a collision detector.

2) Collision likelihood for a Path: After binarizing the map, the collision likelihood for a path π (Eq. 6) then becomes:

$$p(\Pi = 0) \doteq \sum_i p(\hat{\nu}_f^{[i]}|\hat{m}_b^*, \pi) \quad (9)$$

where $\hat{\nu}_f^{[i]}$ is the i^{th} trail particle that is collision-free in map $\hat{m}_b^*$. Eq. 9 basically says that the likelihood of following a planned path without collisions is the summation of the normalized likelihood (also called normalized weights) for all those trails that are collision-free.

Algorithm 1: SimLocal

input : $\mathcal{L}_{n_i}$, $\mathcal{T}_{n_i}$, π , m_b^*, $P_{free}^{n_i}$.
output: $\mathcal{L}_{n_{i+1}}$, $\mathcal{T}_{n_{i+1}}$, $P_{free}^{n_{i+1}}$.
begin
  $t = 0$, $\mathcal{L}_t = \mathcal{L}_{n_i}$, $P_t = P_{free}^{n_i}$;
  while IsNotNextNode($\mathcal{L}_t$, π) **do**
    $\mathcal{L}'_{t+1} \sim p(q_{t+1}|\mathcal{L}_t, a_t^{\pi})$;
    extend the trail particles to the state particles in $\mathcal{L}'_{t+1}$;
    check collision and update the likelihood for each extended trail based on the "path following model";
    update P_{t+1}, i.e., the collision likelihood of following the path π up to time $t+1$;
    simulate one localization step, i.e., acquire $\mathcal{L}_{t+1}$;
    sample $\sharp_L$ trail particles based on Eq. 8;
    for each particle in $\mathcal{L}_{t+1}$, record its index along with weight of the trail particle associated with it into $\mathcal{T}_{t+1}$;
    t= t+1;
  end
  return $< \mathcal{L}_t, \mathcal{T}_t, P_{free}^t >$
end
notation:
$\mathcal{L}_{n_i}$: localization particles set at graph node n_i.
$\mathcal{T}_{n_i}$: a data set stored at graph node n_i. $|\mathcal{T}_{n_i}| = |\mathcal{L}_{n_i}|$.
$P_{free}^{n_i}$: the likelihood that robot is collision free when arriving n_i along the path $\pi_{[q_{n_0}, q_{n_{i+1}}]}$.
$\sharp_L$: the size of the particles set $\mathcal{L}_t$.
a_t^{π}, the control command to follow the path at time t.

3) Algorithm for Sequentially Updating Collision Likelihood Along a Path: We consider a path π that consists of many segments in C-space, over a graph structure, such that each end point of the segments along the path corresponds to a graph node. When robot moves from graph node n_i to n_{i+1}, the trail particle set as well as the collision likelihood is continuously updated at each time instant t (we set $t = 0$, when robot is at graph node n_i), as stated in Alg. 1.

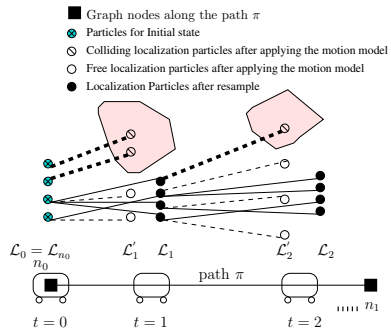


Fig. 2. Sample trail particles sequentially over each time instant.

Fig. 2 schematically illustrates the algorithm for a simple scenario with state particle sets $\mathcal{L}_t$ of size 4, and three consecutive robot states returned by the simulated localization sequence at time $t = 0, 1, 2$, when robot moves from graph node n_0 to n_1. At each time instant t, we

apply the motion command (line-4, Alg. 1), a_t^{π}, to $\mathcal{L}_t$ to acquire the initial guess of the robot state at $t+1$, denoted by $\mathcal{L}_{t+1}^{'}$) (shown as circles). We approximate the true trail between state particles in two consecutive sets with straight lines between them. Computing collision checks and trail likelihood update (line-5 and 6 in Alg. 1) is $O(\sharp_L{}^2)$ at each time instant t, and therefore is slow for a real time solution. For efficiency, we only test the collision of the straight line between localization particles in $\mathcal{L}_t$ and their descendants in $\mathcal{L}_{t+1}^{'}$, shown as dashed lines (both bold and thin) in Fig. 2. The updated likelihood of a particular extended trail (line-6, Alg. 1) is simply the previous trail likelihood multiplied by the likelihood of the motion model used for extending the trail.

At line-7, the likelihood that robot is collision-free when following the path π is updated sequentially over each time instant. Let $p(\Pi_t = 0)$ be the likelihood that robot is collision-free when robot follows the path up to time t. Based on Eq. 9, we then have,

$$p(\Pi_{t+1} = 0) = p(\Pi_t = 0) \sum_i p(\hat{\nu}_{f_{t+1}}^{[i]} | m^*, \pi), \qquad (10)$$

where $p(\hat{\nu}_{f_{t+1}}^{[i]} | m^*, \pi)$ is the normalized weight of the i^{th} extended trails particle that is free of collision up to time $t+1$.

The simulated localization step (line-8) for acquiring $\mathcal{L}_{t+1}$ (shown as solid circles) is similar to the "test-and-evaluate" steps mentioned in Section-II-A (except that it runs simulated localization instead of SLAM).

The re-sampling step for trail particles (line-9) is based on Eq. 8 and could be performed using widely used standard sequential sampling (without replacement) [5]. Again, for efficiency, we sample updated trails from a subset of all possible updated trails (to acquire all the possible updated trails, we need to extend each trail particle at time t to all the localization particles in $\mathcal{L}_{t+1}$). This subset includes only those trails that are extended from fathers (in $\mathcal{L}_t$) of the localization particles in $\mathcal{L}_{t+1}$. In addition, we do not re-test the updated trails for collision. The updated trail particle set is shown as solid lines in Fig. 2.

Finally, since the collision-free likelihood is updated in a sequential fashion, we do not need to explicitly memorize the trail particles. Only the trail likelihood along with localization particle index is updated and recorded at each graph node along the path (line-10, Alg. 1).

III. Planning Motion with Localization, Sensing and Mapping Uncertainty, RRT-SLAM

We introduce an extra dimension, that of uncertainty, to C-space (similar to [7], except that they used A*), and the RRT is built over the composite uncertainty-Configuration space (UC-space).

The detailed algorithm, which incorporates our collision likelihood estimation algorithm, is shown in Alg. 2. We have two criteria for the planned path: 1) the collision likelihood along the path is lower than a user defined threshold; and 2) the uncertainly of the robot at goal position is smaller than another user defined threshold. Note that the major modification of the basic RRT is in the "Extend" step (Alg. 3).

Algorithm 2: RRT-SLAM

input: $\mathcal{Y}$ - SLAM particle set before robot planning the motion.
output: tree - A random tree.

1 **begin**
2 $< \overline{q}^*, m_b^* >$:= the most weighted particle in $\mathcal{Y}$;
3 **for** $i = 1$ to $\sharp_L$ **do**
4 $\overline{q}_{n_s}^{[i]} = q^*$; $\omega_{n_s}^{[i]} = \varpi_{n_s}^{[i]} = \frac{1}{\sharp_L}$
5 **end**
6 $P_{free}^{n_s} = 1$;
7 tree.init($\overline{q}_s, u_s = 0, \mathcal{L}_{n_s}, \mathcal{T}_{n_s}^L, P_{free}^{n_s}$);
8 **while** !TimeUp() and !($\overline{q}_g \in$ tree) **do**
9 $< \overline{q}_r, u_r >$:= RandomState() ;
10 n_c := NearestNeighbor($< \overline{q}_r, u_r >$, tree);
11 Extend(tree, $< \overline{q}_r, u_r >, n_c, m^*$) ;
12 **end**
13 **end**

notation: $\overline{q}_{n_j}^{[i]}$: the i_{th} particle in $\mathcal{L}_{n_j}$.
$\omega_{n_j}^{[i]}$: the weight of $\overline{q}_{n_j}^{[i]}$.
$\varpi_{n_j}^{[i]}$: the weight of the a trail particle, stored in $\mathcal{T}_{n_j}$, which index is i.

Algorithm 3: Extend

input: n_c - a tree node.
$< \overline{q}_r, u_r >$ - a node state the tree will extend from n_c towards to.
tree - the random tree
m^* - a map.

1 **begin**
2 $\overline{q}_{new}$ = ExtendTo($\overline{q}_{n_c}, \overline{q}_r$);
3 $< \mathcal{L}_{n_{new}}, \mathcal{T}_{n_{new}}, P_{free}^{n_{new}} >$ =
4 SimLocal($\mathcal{L}_{n_c}, \mathcal{T}_{n_c}, m^*, \pi_{[\overline{q}_{n_s}, \overline{q}_{n_{new}}]}, P_{free}^{n_c}$,);
5 **if** $\mathcal{L}_{n_{new}} \neq$ NULL and $P_{free}^{n_{new}} > 1 - \tau_{col}$ **then**
6 u_{new} = UncertaintyScalar($\mathcal{L}_{n_{new}}$);
7 $n_{new} := \{\overline{q}_{new}, u_{new}, \mathcal{L}_{n_{new}}, \mathcal{T}_{n_{new}}, P_{free}^{n_{new}}\}$;
8 add n_{new} to tree as son of n_c;
9 **end**
10 **else**
11 return;
12 **end**
13 **end**

notation: τ_{col}: the threshold of collision likelihood of following a path leading to the goal.

We extend our tree as follows: first, we generate a random configuration $\overline{q}_r$ in C-space, and apply a distance metric to choose the nearest one among the tree nodes (denoted as n_c) to the $\overline{q}_r$. The distance metric used in our RRT is:

$$|n_r, n_c| = \varphi |\overline{q}_r, \overline{q}_c| + (1 - \varphi) * u_c, \qquad (11)$$

where $0 < \varphi < 1$ is a weight, and u_c is the uncertainty at node n_c.

The basic idea is that the distance metric tends to bias to tree nodes with lower localization uncertainty. Fig. 3 shows a typical tree in 2D with extra dimension of localization uncertainty(shown by an ellipse around the node). Tree branches are marked by light gray segments, and the planned path is shown with black segments. The goal point is marked by the dark square. Note that in this specific example, the planned path passes through the goal point twice. When robot reaches the goal point at the first time the localization uncertainty is higher than the threshold for localization uncertainty at the goal point. The robot needs further motions around goal area to localize itself better, and finally reaches the goal again with the constraint on localization uncertainty at the goal less than the required threshold.

IV. Simulations

We have run preliminary tests in a simulated environment in 2D. We use the MobileSim [17] program as our mobile

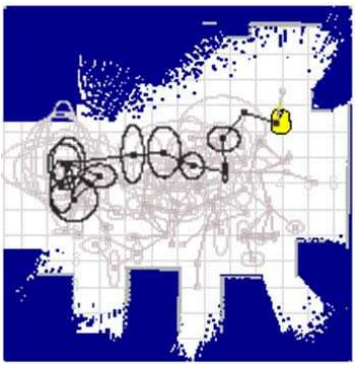


Fig. 3. A typical random tree returned by Algorithm-2

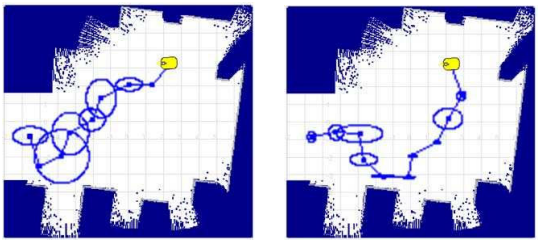
robot simulator. The robot is represented with a rectangle. The simulated range sensor, with sensing range of 4.0 meters and 180^o span (approximately the same field of view as the Hokuyo URG-04LX range sensor), is fixed on the mobile robot and points to the front. We also simulate the motion uncertainty of the mobile robot. Our planner program is built on top of the Stage [18], and MPK [19] system. The planner interacts, i.e., sends control commands and receives "on-board" sensor readings, with the mobile simulator through TCP/IP. We run our simulation under Linux on a Pentium-III 1.0Ghz computer with 500MB memory.

A typical scenario, which indicates the effectiveness of our RRT-SLAM is shown in Fig. 4, where the robot needs to go across the open free area with a collision likelihood less than 0.15 (chosen somewhat arbitrarily). The goals in Fig. 4(a) and Fig. 4(b) are at the same location but the uncertainty threshold at goal in Fig. 4(a) (0.8) is much larger than in Fig. 4(b) (0.2). In UC-space, it corresponds to two different points separated along the uncertainty dimension. Our algorithm returns paths that satisfy the collision likelihood threshold along the path(s) and the localization uncertainty at the goal. Note that even though path1 is shorter than path2, the localization uncertainty along the first path is higher than the second path, and the collision likelihood for the first path (0.023) is higher than the second path (0.0). Here, the higher likelihood of collision for the first path is mainly due to the localization uncertainty around the third last path node. We do not explicitly show the localization particles sets along the path in Fig. 4, but use the ellipse to demonstrate the localization uncertainty at each tree node along the path.

Intuitively, allowing a path with non-zero likelihood of collision and higher localization uncertainty tends to reduce the difficulty level of the planning problem. In practice, the specific threshold would be experimentally determined by the user. We need further efforts to determine the effect of threshold on the path planned.

V. Conclusion and Future Work

In this work, we considered sensing, localization and mapping uncertainties in the motion planning sub-problem that arises in a robotic exploration and mapping task. We used RRT in conjunction with a simulated particle based SLAM algorithm to expand the tree in UC-space. The collision likelihood along a planned path is explicitly computed and is used to select a planned path. Preliminary simulations show



(a) path1 (b) path2

Fig. 4. Paths based on different goal uncertainties

the effectiveness and benefits of our integrated approach. In near future, we would like to extend our approach to a mobile-manipulator exploring its environment.

References

[1] Y. Yong and K. Gupta. C-space entropy: A measure for view planning and exploration for general robot-sensor systems in unknown environments. *International Journal of Robotics Research*, 23(12):1197–1223, Feb. 2004.

[2] P. K. Allen, M. K. Reed, and I. Stamos. View planning for site modeling. In *Proceedings of the DARPA Image Understanding Workshop*, Monterey, CA, Nov. 1998.

[3] F. Bourgault, A.A. Makarenko, S.B. Williams, B. Grocholsky, and H.F Durrant-Whyte. Information based adaptive robotic exploration. In *Proceedings of the 2002 IEEE International conferenece on intelligent Robots and System*, pages 540–545, May 2002.

[4] Lila Torabi and Kamal Gupta. An approach to occupancy grid based method for navigation and exploration. *IEEE International Conference on Intelligent Robots and Systems (IROS 2007)*, 2007.

[5] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, 2005.

[6] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, 2002. AAAI.

[7] Alain Lambert and Dominique Gruyer. Safe path planning in an uncertain-configuration space. In *Proceedings of the 2002 IEEE International conferenece on Robotics and Automation*, pages 4185–4190, Sept 2003.

[8] J. P. Gonzalez and A. Stentz. Planning with uncertaitny in position: an optimal and efficient planner. In *Proceedings 2005 IEEE International Conference on Intelligent Robots and Systems (IROS 2005)*, pages 1327–1332, Edmenton, CANADA, May 2005.

[9] N. Roy and G. Gordon. Exponential family pca for belief compression in pomdps. In *Proceedings. Advances in Neural Information Processing (15) NIPS*, Vancouver, BC. CANADA, June 2002.

[10] N. Roy and S. Thrun. Coastal navigation with mobile robots. In *Advances in Neural Processing Systems 12*, volume 12, pages 1043–1049, 2000.

[11] P. E. Missiuro and N. Roy. Adapting probabilitic roadmaps to handle uncertain maps. In *Proceedings 1995 IEEE International Conference on Robotics and Automation*, pages 1261–1267, Orlando, US, May 2006.

[12] L. Freda, F. Loiudice, and G. Oriolo. A randomized method for integrated exploration. In *In Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS2006)*, 2006.

[13] Nik A. Melchior and Reid Simmons. Particle rrt for path planning with unceratinty. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, Roma, Italy, April 2007.

[14] Cyrill Stachniss, Giorgio Grisetti, and Wolfram Burgard. Information gain-based exploration using rao-blackwellized particle filters. In *Proceedings of Robotics: Science and Systems*, June 2005.

[15] D. Hsu, W.S. Lee, and N. Rong. A point-based pomdp planner for target tracking. In *IEEE International Conference on Robotics and Automation 2008 (ICRA 2008)*, 2008.

[16] A. Eliazar and R. Parr. Dp-slam 2.0. In *IEEE International Conference on Robotics and Automation 2004 (ICRA 2004)*, 2004.

[17] http://www.mobileworld.com.

[18] Richard T. Vaughan, Brian Gerkey, and Andrew Howard. On device abstractions for portable, resuable robot code. In *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robot Systems (IROS 2003)*, pages 2121–2427, Las Vegas, USA, October 2003.

[19] Ian Gipson, Kamal Gupta, and Michael Greenspan. Mpk: An open extensible motion planning kernel. *Journal of Robotic Systems*, 18:433–443, 2001.