EG-RRT: Environment-Guided Random Trees for Kinodynamic Motion Planning with Uncertainty and Obstacles

Léonard Jaillet, Judy Hoffman, Jur van den Berg, Pieter Abbeel, Josep M. Porta, Ken Goldberg

***Abstract*— Existing sampling-based robot motion planning methods are often inefficient at finding trajectories for kinodynamic systems, especially in the presence of narrow passages between obstacles and uncertainty in control and sensing. To address this, we propose EG-RRT, an Environment-Guided variant of RRT designed for kinodynamic robot systems that combines elements from several prior approaches and may incorporate a cost model based on the LQG-MP framework to estimate the probability of collision under uncertainty in control and sensing. We compare the performance of EG-RRT with several prior approaches on challenging sample problems. Results suggest that EG-RRT offers significant improvements in performance.**

I. Introduction

This paper presents an efficient motion planning algorithm for robotic systems subject to both internal dynamical constraints and external environmental or task specific constraints. This appears, for example, with non-holonomic robots moving in cluttered scenes with narrow passages or when optimizing a path quality metric such as probability of collision. Motion planning for such systems is known to be very challenging and exact solvers have PSPACE complexity making them unusable for practical applications [1].

An emerging class of sampling-based approaches have potential for efficiently generating solutions that satisfy system constraints [2], [3]. For systems with dynamical constraints, path conversion techniques have been developed for some non-holonomic systems, that first find a geometric path from specific local methods and then impose time constraints on velocities and accelerations [4]. For more general systems, kinodynamic motion planning [5], which considers kinematic and dynamic constraints within the same framework, is more suitable. In particular, Rapidly-exploring Random Tree (RRT) based methods [6] or the Expansive-Space Tree (EST) planner [7], are promising as they provide a direct way to incorporate the dynamics of the system by the use of forward integration to search the state space. It has, however, been shown that even RRT methods can become inefficient in certain scenarios [8].

This work has been partially supported by the Spanish Ministry of Science and Innovation under project DPI2010-18449 and by the U.S. National Institute of Health under Award 1R01EB-006435-01A1 and by the U.S. National Science Foundation under Award 0905344.

L. Jaillet and J. M. Porta are with the Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Barcelona, Spain. {ljaillet, porta}@iri.upc.edu

J. Hoffman, P. Abbeel and K. Goldberg are with the University of California, Berkeley, USA. {judyhoffman, pabbeel, goldberg}@berkeley.edu

J. van den Berg is with the University of North Carolina, Chapel Hill, USA. berg@cs.unc.edu

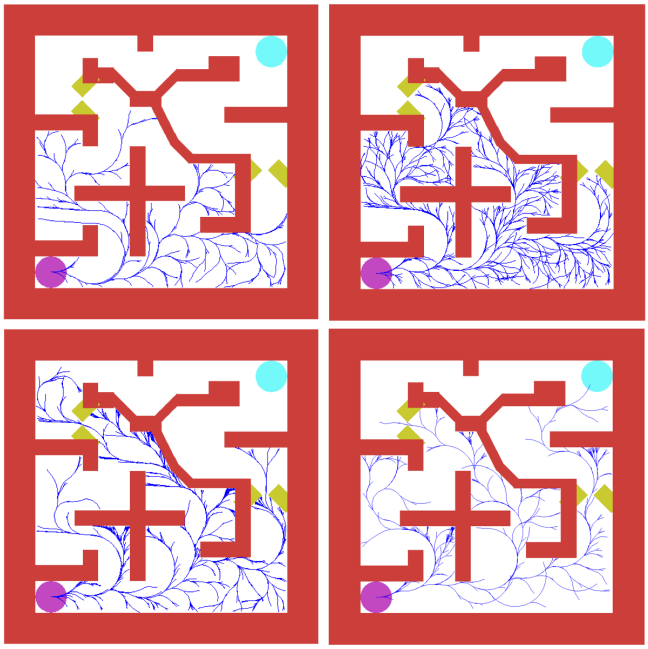

Fig. 1. Performance comparison of four RRT-based planners in a maze-like environment with two narrow passages (between yellow obstacles). Each planner was allowed to generate up to 1500 nodes. In the upper left, standard RRT produces poor coverage of the space. In the upper right, Reachability-Guided RRT improves coverage but gets stuck in the narrow passages. In lower-left, Resolution-Complete RRT fails to find a solution after generating 1500 nodes. In the lower-right, EG-RRT covers homogeneously the space and finds a solution after generating in average less than 800 nodes.

A central issue with kinodynamic systems with substantial environmental constraints comes from the disagreement between the distance evaluation based on the available metric and the true cost-to-go [9]. Indeed, the cost-to-go from one state to another depends on the system dynamics, the position of the obstacles, as well as other possible constraints imposed by the environment. None of these tend to be easy to account for and are typically not accounted for at all in the metric.

To limit the role of the metric, several methods use space decomposition and select the states to be expanded according to the sample density [7], [10], [11]. It has also been shown that a considerable speed-up can be obtained if a workspace decomposition is used to guide the search [12]–[14]. However the efficiency of such approach does not take benefit of the potential exploratory strength of RRT methods biased toward yet unexplored regions of the space.

When using the RRT framework, the imperfect metric can cause the same node to be selected many times although not participating, or poorly, to the extension of the tree. The two

typical causes are (i) points chosen for extension based on the imperfect metric are simply not able to grow the tree in the direction the sample suggests it should grow per the kinodynamic constraints; (ii) points close to obstacles that only (or mostly) extend into obstacles are repeatedly chosen for expansion.

Accordingly, there are two groups of variants of the basic RRT [5]: the first group adapts the distribution bias to further take into account the system dynamics. In [9], a heuristic based on the Affine Quadratic Regulator (AQR) is designed to approximate the exact minimum distance-time pseudometric. The Reachability-Guided RRT (RG-RRT) planner [15] explicitly accounts for the local limitations of the system dynamics to bias sampling towards states that have a high probability of favoring diffusion of the tree.

The second group of RRT variants focuses on reducing the weight of repetitive unfruitful expansions. In RRT-blossom [16], potential new edges intruding upon already-explored space are eliminated as the expansion is not considered useful enough. The Resolution-Complete RRT planner[1] (RC-RRT) proposed by Cheng et al. [18] records exploration successes and failures to avoid similar expansions and favors the most promising states. A similar principle based on expansion failures can also be found in [19].

In this paper we combine elements of RG-RRT and RC-RRT into a new framework. In the resulting planner, which we call Environment-Guided RRT (EG-RRT), the Voronoi bias allocated to the nodes and inherited from the RRT expansion mechanism is modified such that it integrates also the local limitations of the system dynamics as well as the risk of leading to unproductive expansions (Figure 1).

We then explore how EG-RRT can plan safe paths in the presence of uncertainty. Few works have addressed planning under uncertainty based on the RRT framework. Particle-RRT [20] produces distributions of states derived from the uncertainty rather than single states from which cost based policy is built to select the most promising nodes. In [21], a mobility-based extension of RRT is proposed where state distributions are estimated based on the Stochastic Response Surface Method. In our case, the EG-RRT planner is enhanced by a guiding mechanism that builds on the LQG-MP framework [22], [23] to estimate incremental path quality. LQG-MP is a tool that allows to evaluate a-priori distribution of states along paths based on stochastic dynamics and sensory models and assuming a Linear Quadratic Gaussian (LQG) control policy, which consists of a Kalman filter and an Linear Quadratic Regulator (LQR) feedback controller.

The remainder of the paper is organized as follows. Section II, presents the two variants that inspired our planner and analyzes advantages and limitations of each method through the presentation of challenging scenarios. Section III introduces the core of EG-RRT that is experimentally evaluated in Section IV. Finally, Section V extends the method to the problem of planning safe paths under uncertainty.

[1]The method is called Resolution-Complete in [17] since it is shown that it can comprise such property under some given conditions.

II. Improving RRT exploration

A. Standard RRT

All RRT methods [5] build trees of nodes and edges that correspond to states and small amplitude motions, respectively. The exploration mechanism iterates over three main steps, covering progressively the reachable regions of the free state space: First, a random state is sampled in the state space. Second, the state of the tree closest to the sample is identified. Third, this state is expanded by simulating the dynamics forward in time for a given input. If the expansion is valid, i.e., it respects the state bounds and is free of collision, a new node with its corresponding edge is inserted in the tree.

RRTs have an (implicit) Voronoi bias that steers them towards yet unexplored regions of the space [6]. However in case of kinodynamic systems, the imperfection of the underlying metric can compromise such behavior. Typically, the metric relies on the Euclidean distance between points which does not necessary reflect the true cost-to-go between states. Finding a good metric is known to be a difficult problem [9]. Simple heuristics can be designed to improve the choice of the tree state to be expanded and to improve the input selection mechanism without redefining a specific metric. In the following, two variants of the literature based on RRT are presented.

B. Reachability-Guided RRT

The Reachability-Guided RRT (RG-RRT) proposed by Shkolnik et al. [15], [24] provides a way to improve tree state selection by taking into account the system dynamics. A reachability set is associated with each state in the tree. This reachability set is a region of the space that can be reached by the node within a given finite time, simulating the dynamics forward in time using the set of available inputs U. This set is computed (or approximated) each time a new node is inserted in the tree.

The node selected for expansion is the one that has the closest reachable point to the randomly sampled tree state. If this reachable point, however, is at a larger distance than the current closest node, the extension is discarded and a new state is sampled to attempt a new expansion. Thus, the method selects the most promising expansions by filtering those which do not participate effectively to the tree expansion toward the sampled state.

By default the reachable set is built by simulating the dynamics forward for the entire set of possible inputs but without taking into account the obstacles. The possibility of designing reachability sets taking into account the obstacles is also mentioned in [15], however the computational cost of such operation would be high since it would require testing all potential expansions when inserting a new node. A different RRT variant that takes into account obstacles during the node selection/extension is presented below.

C. Resolution-Complete RRT

The RC-RRT planner proposed by Chang et al. [8], [18] discretizes the input space U and considers expansion

failures in order to redefine the most promising nodes. A data structure associated with each node stores the set of inputs already applied to the node. When a given input has been applied once to the state (with success or not), it is not applied anymore. Moreover, when all possible inputs have been applied, the node is discarded for future extension.

In addition, nodes are penalized to estimate when extensions attempts are likely to fail, based on the notion of Constraint Violation Frequency (CVF). Each new node is initialized with a CVF of 0. Then, when the input leads to a collision or a violation of the state restrictions (e.g. it exceeds the speed bound) the CVF of the node is increased by $\frac{1}{m}$, where m is the number of possible input of the discretized set U. Moreover, the CVF of the parent state is increased by $\frac{1}{m^2}$ and by applying a recursion mechanism, the CVF of its k^{th} parent state is increased of $\frac{1}{m^{k+1}}$. Thus, each node's CVF is bounded between 0 and 1, where the value 1 is reached when no valid descendant can be generated. Then, during the node selection, the probability of discarding a node of the list of nearest neighbors is equal to the value of its CVF.

D. Analysis of Efficiency of the Variants

We first consider the exploration tree of Figure 2 for a car-like robot with second order dynamics restricted to forward motions. Solid lines are RRT edges, dots are RRT nodes, dotted lines represent the Voronoi region boundaries based on Euclidean distance, and colored quadrilaterals are the reachables sets for the nodes of that color. The two red nodes have large Voronoi region and thus a high probability to be selected so an expansion scheme based on the basic RRT will lead to a collision. These nodes decrease the probability of finding a way through the narrow passage since they reduce the Voronoi region of the more promising green node. If RG-RRT is used instead of RRT, the bias from the reachability regions of the red nodes will remain predominant and the probability to extend the green node will still be low. A penalty-based strategy, such as used in RC-RRT, is better at overcoming this difficulty. As the failed expansion attempts for the red nodes are recorded, the probability of selecting them for extension decreases compared to the benefit of the green node.

Consider now the same problem but with an exploration tree as shown in Figure 3. Once again, a basic RRT will have significant difficulty to go through the narrow passages since the Voronoi bias associated with the two red nodes is predominant, even though an extension of the green node is more promising. RG-RRT highly increases the probability of selecting the green node. Conversely, RC-RRT is unlikely to improve the situation since red nodes which do not lead to direct collisions will take long to be penalized by the method.

For each of the two challenging scenarios only one of the RRT variants was well suited to overcome the difficulty. In the following section, we describe a novel approach which fuses the strengths of both variants.

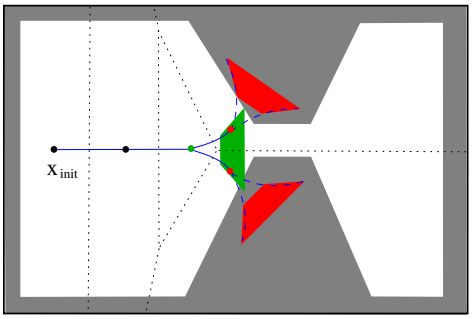

Fig. 2. Exploration tree for a car-like robot with 2nd-order dynamics restricted to forward motions. With a basic RRT extension scheme, nodes in red with strong Voronoi bias reduce the chance to find a way through the narrow passage. In this situation, the reachability sets of RG-RRT (colored quadrilaterals) do not change significantly the bias and the situation remains difficult. In the case of RC-RRT, failures of expansion attempts are recorded, increasing the chance for the green node to be selected, so the planner will be more efficient.

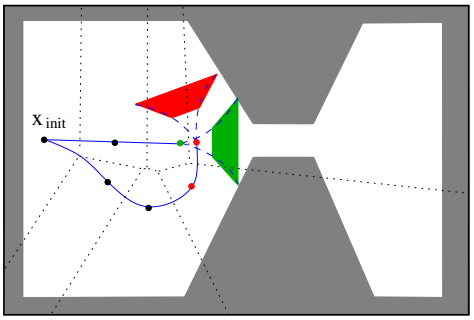

Fig. 3. Exploration tree for a car-like robot with 2nd-order dynamics restricted to forward motions. With a basic RRT extension scheme, nodes in red with strong Voronoi bias reduce the chance to find a way through the narrow passage. In this situation, RG-RRT has good chance to overcome the difficulty since the reachability set of the green node which is rightmost, highly increases the chance of the node to be selected. On the contrary, RC-RRT will hardly improve the situation, since red nodes which do not lead to direct collisions will take long to be penalized by the method.

III. Environment-Guided RRT

Algorithm 1 gives a high-level description of our hybrid planner called Environment-Guided RRT (EG-RRT), which combines the strengths of RG-RRT and RC-RRT. The method is initialized with the `InsertNode` function which computes the reachable set of x_{init} and adds it to the tree. Then, similarly to the basic RRT, an iterative procedure is performed until a stop condition is achieved (typically a state generated inside X_{goal} or a maximum number of nodes reached). First, a random state x_{rand} is sampled. Then, the `BestState` function selects the most promising candidate state, x_{best} as well as the input u_{near} generating the state the closest to x_{rand} (see details in subsection III-A). `BestState` possibly returns a null state, meaning that no relevant expansion has been found. In that case, the method reiterates the sampling and the search of another promising state and input. Finally, when a state x_{best} has been found, `BestInput` returns an input u_{best} and the associated expanded state x_{new}, such as x_{new} respects the state restrictions, is free of collision, and is the closest feasible state to x_{rand} that could be generated from x_{best} (see details in subsection III-B). If

Algorithm 1: BUILD_EG_RRT(x_{init})

```
input : the state space X;
the root state x_init, and the goal region X_goal;
output: the tree T;
T ← InsertNode (x_init, T);
while not StopCondition(T, X_goal) do
    x_rand ← RandomState() ;
    (x_best, u_near) ← BestState(x_rand, T) ;
    if x_best ≠ ∅ then
        (u_best, x_new) ← BestInput(x_best, u_near, x_rand);
        if u_best ≠ ∅ then
            T ← InsertNode(x_new, T );
            T ← InsertEdge(x_best, x_new, u_best, T );
```

no admissible u_{best} has been found, a new random state has to be sampled. Otherwise, the node with its reachable set is inserted in the tree. Finally, the `InsertEdge` function labels the input u_{best} of x_{best} as expanded before adding an edge linking x_{best} to x_{new} within the tree.

A. Best State Selection

The `BestState` function is described in Algorithm 2. A node of the tree is candidate for expansion only if it is not fully expanded (some inputs have not been tried yet) and if its CVF is lower than a random number between 0 and 1. This corresponds to the penalty-based part of the node selection. If the node is a candidate, its distance to x_{rand} is evaluated as well as the distance of the reachable set to x_{rand}. This process is repeated for all the nodes in order to estimate both the distance to the closest node and the distance to the closest point of the reachable sets. Then, if no node has been selected or if the distance to the closest reachable state is higher than the distance to the closest node, the expansion is filtered and the function returns null elements. Otherwise, following the reachability-guided principle, it returns the state x_{best} and the input u_{near} that leads to the closest reachable point.

Algorithm 2: BestState(x_{rand}, $\mathcal{T}$)

```
d^r_min ← ∞, d^n_min ← ∞;
x_best ← ∅ ;
for all x ∈ T do
    if not FullyExpanded(x) then
        if CVF(x)< Random(0,1) then
            (d^r, u^r) ← ReachSetDist(x_rand, x);
            d^n ← Dist(x_rand, x);
            if d^r < d^r_min then
                d^r_min ← d^r;
                x_best ← x;
                u_near ← u^r;
            if d^n < d^n_min then
                d^n_min ← d^n;
if x_best = ∅ or d^n_min < d^r_min then
    return (∅,∅);
return (x_best, u_near);
```

B. Best Input Selection

The `BestInput` function described in Algorithm 3 finds among the input generating valid states (valid meaning that it respects the state bounds and it is collision free), the one that leads to the closest state to x_{rand}. Since u_{near} and the associated state x_{best} generated by the `BestState` function correspond to the expansion that gets the closest to x_{rand} without considering bounds and collision checks, the function tests first the feasibility of u_{best}. If it leads to a valid state then u_{near} is returned as u_{best}. Otherwise, all the inputs of U not already expanded are considered and the function returns among the valid one closest to x_{rand}. Moreover, each time an invalid input u is detected, the `UpdateCVF` function updates the CVF of the states similarly to what is done in [18], and labels as expanded the input u for the state x. Finally, if all the node's inputs are expanded, the node is labeled as `FullyExpanded`.

Algorithm 3: BestInput(x_{best}, u_{near})

```
x_cur ← Integrate (x_best, u_near);
if IsValid (x_best, u_near, x_cur) then
    return (u_near, x_cur);
d_min ← ∞;
u_best ← ∅, x_new ← ∅ ;
for all u ∈ U do
    if u ≠ u_near and not Expanded(x_best, u) then
        x_cur ← Integrate (x_best, u);
        d ← Dist(x_cur, x_rand);
        if d < d_min then
            if IsValid(x_best, u, x_cur) then
                d_min ← d;
                u_best ← u;
                x_new ← x_cur;
            else
                UpdateCVF(x_best, u);
return (u_best, x_new);
```

C. Remarks on the Algorithm

In order to efficiently combine RG-RRT and RC-RRT, some elements of each variant have been adapted. In RG-RRT, the distance between x_{rand} and the closest reachable point is compared to the distance between x_{rand} and the closest node. In our method the test is limited to the nodes passing the CVF test. It reduces the influence of the discarded nodes and limits the distance computations. Moreover, in RC-RRT, when all the nodes are discarded by the CVF filtering process (`BestState` function), the closest node not fully expanded is returned, which ensures the selection of at least one node. In our case, the function returns empty elements and a new random state is sampled which filters even further low interest expansions and again avoids involving all nodes for the distance computation. Finally, when an input fails to generate a valid state, the reachability set of the associated node could be updated by cropping the corresponding region. However, since the cost of updating the set appeared to be more expensive than the gain due to a better evaluation of the reachability sets we

discard this strategy. Moreover the importance of such nodes is already decreased through the updating of the CVFs.

IV. Experiments

We applied the EG-RRT planner to a non-holonomic car-like robot with 2nd-order dynamics restricted to forward motions, evolving in environments of various difficulty. The planner is compared to the basic RRT, RG-RRT and RC-RRT. The dynamics is propagated using forward integration with a time step of $0.5s$. Regarding the sampling process, and following [2], we sample states for extension within the goal region with a probability of 0.05. Computational times in seconds are presented in the tables and were averaged over 100 runs. Numbers in parentheses indicate the percentage of failures to find a solution after a maximal running time of $t = 600s$.

A. Model Description

The vehicle is represented by a 4 dimensional state vector $\mathbf{x} = (x, y, \theta, v)$ consisting of its position (x, y), its orientation θ, and its speed v. Its control input $\mathbf{u} = (a, \Phi)$ is a 2-D vector consisting of the acceleration a and the steering wheel angle Φ. The model satisfies the kinodynamic constraints according to the state transition function $\dot{\mathbf{x}} = f(\mathbf{x}, u)$ of the form:

$$\begin{pmatrix} v\cos\theta \\ v\sin\theta \\ \frac{v}{d}\tan(\Phi) \\ a \end{pmatrix} \quad (1)$$

where d is the distance between the front and rear axle of the car [6]. The set of input U is discretized into 25 elements such that five accelerations and five steering wheel angles are possible for each integration step. The reachable set is estimated by taking into account the bounds of U as suggested in [15]. In the results presented, we used the 2-D Euclidean distance based on (x, y) coordinates to compute distance between the states. Note that we also performed experiments introducing the orientation θ in the metric. It resulted in a loss of performance in all RRT variants except for the basic RRT where the performance remained rather similar to those presented next.[2]

B. Cluttered Environment

The method was first tested on an environment cluttered by circular obstacles. Three levels of difficulty are considered that are function of the radius of the obstacles (see Figure 4). Table I gives computational times for EG-RRT and the other methods it is compared to.

A first remark is that all variants outperform the basic RRT. In this scenario, RC-RRT gets better results than RG-RRT when the difficulty increases. EG-RRT is faster than both RC-RRT and RG-RRT, except in the easy scenario where the cost of the additional operations in EG-RRT slightly exceeds the performance gain due to a better exploration.

[2]This phenomenon can be explained by a lower Voronoi bias when introducing θ, which partially alleviates the blocking situations appearing in Figure 2 and Figure 3 but at the cost of a much lower exploratory strength.

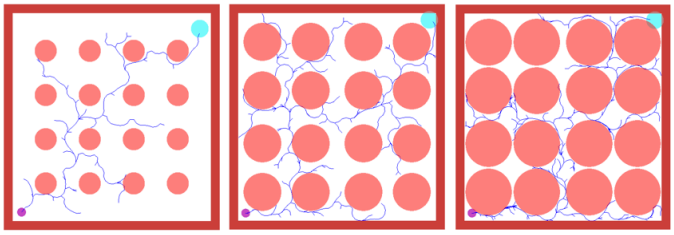

Fig. 4. EG-RRT trees for a car-like robot in environments cluttered by circular obstacles. The benefit of such a planner gets even stronger when the difficulty of the scene increases.

	RRT	RG-RRT	RC-RRT	EG-RRT
Easy	2.3	0.5	1.2	0.7
Medium	133.1 (20)	15.6	4.6	2.5
Difficult	598.0 (99)	253.1 (10)	22.1	12.6

TABLE I

Computational times in seconds for RRT variants for the cluttered scenes of Figure 4. (Numbers in parentheses indicate number of times in 100 runs that planner failed to find a solution after 600 seconds).

C. Maze Environment

The second environment is shown in Figure 1. It involves a maze where two narrow passages (due to obstacles in yellow) can be added which increases the difficulty of the motion planning problem. Table II gives the computational time of the RRT variants as a function of the number of narrow passages involved (in case of single narrow passage, the one on the right of the scene is considered). Contrarily to the previous case, the scene contains regions where the tree can be locally trapped and the path leading to the goal requires significant detours.

In this environment, RG-RRT outperforms RC-RRT. Again the combination of both methods increases the exploration efficiency. For the more challenging environments (in which the scenario presented in Figures 2 and 3 gets more probable), the benefit of using EG-RRT is even stronger.

Nb. narrow pass.	RRT	RG-RRT	RC-RRT	EG-RRT
0	52.6	10.8	32.6	10.4
1	138.3	29.6	67.9	15.9
2	251.2 (23)	52.6	109.4 (6)	19.5

TABLE II

Computational time in seconds of RRT variants for the Maze of Figure 1, with 0, 1 or 2 narrow passages. (Numbers in parentheses indicate number of times in 100 runs that planner failed to find a solution after 600 seconds).

V. Planning under Uncertainty

Taking advantage of the efficiency of EG-RRT, we propose an extension of the method to find safe paths in the presence of sensing and motion uncertainty. First, the LQG-MP method [22] is applied to each path of the RRT to characterize the a-priori probability distributions over states in that path. Then, the distributions are used to assign costs to the nodes of the tree based on evaluations of the probability of collision for each portion of the path.

A. LQG-MP Costs

LQG-MP is a tool that predicts a-priori distributions of states along paths based on stochastic dynamics and sensory models and assuming a Linear Quadratic Gaussian (LQG) control policy, which consists of a Kalman filter and an Linear Quadratic Regulator (LQR) feedback controller. In our case, the distributions are evaluated over a limited time horizon which is sufficient to estimate the collision risk when passing through a given state while limiting the computational effort (in our experiments, we fixed a maximal backward horizon equivalent to 12 propagation steps). Moreover, Kalman filter matrices that are used for evaluating a-priori probability distributions are computed only once for each state and stored in the node data structure which reduces even further the computational effort (see Figure 5).

From the LQG-MP distributions, it is possible to compute at each step the number of standard deviations c_s, that one can deviate from the path before the robot may collide with an obstacle[3]. Then, a lower bound on the probability of avoiding a collision at a stage s is given by $\Gamma(n/2, c_s^2/2)$, where n is the dimension of the Gaussian distribution and Γ is the regularized Gamma function. This gives us the following lower bound on the probability of avoiding a collision along a portion of path of length $l+1$:

$$\Pi = \prod_{s=0}^{l} \Gamma(n/2, c_s^2/2). \tag{2}$$

Finally, we set the cost of a node to $c = 1 - \Pi$.

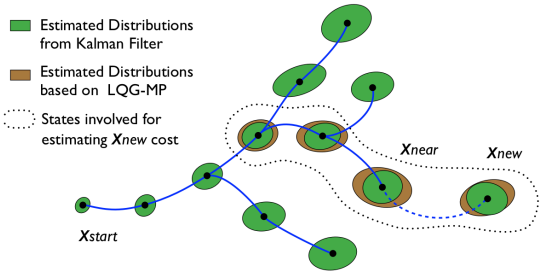

Fig. 5. The cost of a new state depends on the risk of collision of the last states traveled (dot region), for which LQG-MP can be applied to estimate their a-priori probability distributions within the space. Estimated distribution based on Kalman filter matrices are computed only once per state and stored in the nodes.

B. Cost-Guided Exploration

Our planner uses the costs computed from uncertainty to guide the search in a similar way as the Transition-RRT approach [25]. During the search, an additional test is performed before inserting a new state to the tree that aims to filter highest cost states. The probability p_{ij} to accept a new state j of cost c_j reached from a parent state i of cost c_i is set as follows:

$$p_{ij} = \begin{cases} exp(-\frac{c_i - c_j}{K \cdot T}) & \text{if } c_i > c_j, \\ 1 & \text{otherwise,} \end{cases} \tag{3}$$

where K is a fixed parameter and T is a variable called Temperature that is automatically tuned during the search in order to adaptively control the filtering strength, and ensures a constant growth of the tree (see [25] for further details).

C. Experimental Evaluation

We integrated the cost-guided exploration mechanism into all RRT variants for comparison. These variants are identified using the "+" superscript. Experiments were performed on the same car-like robot as used in Section IV, where additional process uncertainty is modeled by corrupting the steering wheel angle and the acceleration with Gaussian process noise as in [22]. The adaptive temperature variable that appears in Equation 3 is initialized to $T_0 = 10^{-4}$ and is increased by 5% every 50 extension failures.

In the first scenario, represented in Figure 6, the robot receives feedback on its position from 8 sensors (in yellow). The noise in the measurement increases quadratically with the distance from the sensors. Results are presented in Table III, with time in seconds in the top row. *Quality* is the estimate of the probability of avoiding collision along the whole solution path, which is computed once the exploration reached the goal by applying Equation 2 on the entire path. *% Success* is the percentage averaged over 1000 simulations of paths successfully executed in presence of uncertainty. Note that in this problem, there is only an 8% probability of successfully executing a path from a basic RRT without cost-guidance. The data suggests that EG-RRT$^+$ is significantly faster and finds significantly safer paths.

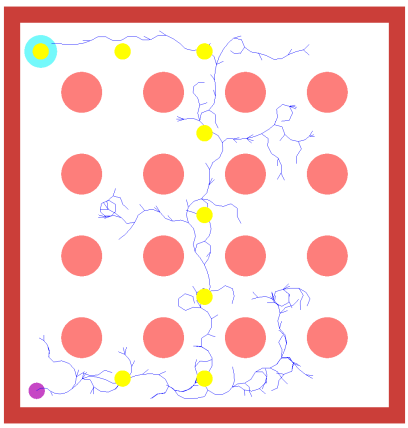

Fig. 6. EG-RRT$^+$ tree leading to a solution path in a problem with uncertainty where 8 sensors represented in yellow are distributed in the space. The noise of the measurement increases quadratically with the distance from the sensors. The solution found with EG-RRT$^+$ remains close to sensors and far from obstacles to limit the risk of collision.

In the second scenario (Figure 7) two cameras provide partial information on the position of the robot in the regions of same color. The orange camera measures the x-coordinate whereas the green one measures the y-coordinate. In this problem, there is only a 2% probability of successfully executing a path from a basic RRT instance if the cost-guidance

[3]To compute costs efficiently, we used the method proposed in [22], that first scales the environment and the uncertainty ellipse such that this last one becomes a unit disc and then estimates the distance between the disc and the scaled obstacles.

	RRT$^+$	RC-RRT$^+$	RG-RRT$^+$	EG-RRT$^+$
time (s)	67.1	109.2	49.7	25.0
Quality	0.07	0.41	0.50	0.71
% Success	36	66	75	87

TABLE III

COMPARATIVE RESULTS FOR COST-GUIDED VARIANTS OF RRT IN THE PROBLEM ILLUSTRATED IN FIGURE 6.

scheme is not incorporated. Results presented in Table IV show that EG-RRT$^+$ outperforms other variants by providing much safer paths while requiring less computational effort.

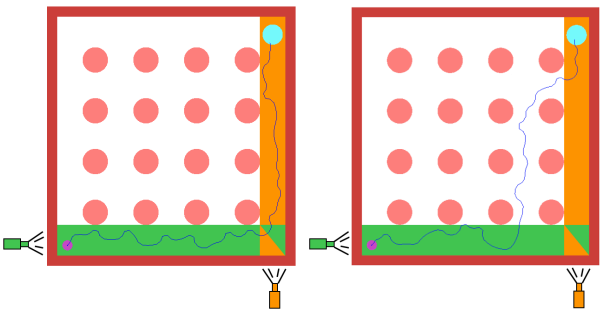

Fig. 7. Comparison of solution paths in a second problem with uncertainty. Two cameras at lower left and lower right provide partial information on the position of the robot. The orange camera measure the x-coordinate whereas the green one measures the y-coordinate. The path at left found by EG-RRT$^+$ is safer than the path at right found by RRT$^+$ (right).

	RRT$^+$	RC-RRT$^+$	RG-RRT$^+$	EG-RRT$^+$
time (s)	187.7 (2)	46.9	68.3	17.6
Quality	0.36	0.50	0.56	0.72
% Success	50	68	69	81

TABLE IV

COMPARATIVE RESULTS FOR COST-GUIDED VARIANTS OF RRT IN THE PROBLEM ILLUSTRATED IN FIGURE 7.

VI. CONCLUSION

This paper presents EG-RRT, an Environment-Guided variant of RRT designed for kinodynamic systems that combines elements from several prior approaches and may incorporate a cost model based on the LQG-MP framework to estimate the probability of collision under uncertainty in control and sensing. We compare the performance of EG-RRT with several prior approaches on challenging sample problems. Results suggest that EG-RRT offers significant improvements in performance. We also describe how to enhance the method by a guiding mechanism that builds on the LQG-MP framework to plan in the presence of uncertainty. Results suggest that the better exploration strength of EG-RRT produces paths faster and with a lower probability of collision.

We would like as future work to investigate how the size of the discretized input set affects the performance of EG-RRT. Also, we plan to compare performance with other planners and on other kinodynamic systems and scenarios. In particular we would like to further investigate how the method scales with problems of higher dimensionality.

REFERENCES

[1] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic motion planning," *J. ACM*, vol. 40, pp. 1048–1066, November 1993.

[2] S. LaValle, *Planning Algorithms*. New York: Cambridge University Press, 2006.

[3] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge: MIT Press, 2005.

[4] J.-P. Laumond, *Robot Motion Planning and Control*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1998.

[5] S. M. LaValle and J. J. K. Jr., "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[6] S. LaValle and J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," in *Algorithmic and Computational Robotics: New Directions*, B. Donald, K. Lynch, and D. Rus, Eds. Boston: A.K. Peters, 2001, pp. 293–308.

[7] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *International Journal of Robotics Research*, vol. 21, pp. 233–255, 2000.

[8] P. Cheng, "Sampling-based motion planning with differential constraints," Ph.D. dissertation, University of Illinois, 2005.

[9] E. L. Glassman and R. Tedrake, "A quadratic regulator-based heuristic for rapidly exploring state space," in *IEEE International Conference on Robotics and Automation*, 2010, pp. 5021–5028.

[10] A. M. Ladd and L. E. Kavraki, "Fast tree-based exploration of state space for robots with dynamics," *International Workshop on Algorithmic Foundations of Robotics VI*, pp. 297–312, 2005.

[11] ——, "Motion planning in the presence of drift, underactuation and discrete system changes," in *Robotics: Science and Systems*. MIT Press, 2005, pp. 233–241.

[12] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Discrete search leading continuous exploration for kinodynamic motion planning," in *Robotics: Science and Systems*. MIT Press, 2008, pp. 326–333.

[13] ——, "Impact of workspace decompositions on discrete search leading continuous exploration (DSLX) motion planning," *IEEE International Conference on Robotics and Automation*, pp. 3751–3756, 2008.

[14] ——, "Motion planning with dynamics by a synergistic combination of layers of planning," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 469–482, 2010.

[15] A. Shkolnik, M. Walter, and R. Tedrake, "Reachability-guided sampling for planning under differential constraints," in *IEEE international conference on Robotics and Automation*, Piscataway, NJ, USA, 2009, pp. 4387–4393.

[16] M. Kalisiak, "Toward more efficient motion planning with differential constraints," Ph.D. dissertation, University of Toronto, 2007.

[17] P. Cheng and S. M. LaValle, "Resolution complete rapidly-exploring random trees," in *IEEE International Conference on Robotics and Automation*, 2002, pp. 267–272.

[18] P. Cheng and S. M. Lavalle, "Reducing metric sensitivity in randomized trajectory design," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 43–48, 2001.

[19] J. Kim and J. M. Esposito, "An RRT-based algorithm for testing and validating multi-robot controllers," in *Robotics: Science and Systems*, 2005, pp. 249–256.

[20] N. Melchior and R. Simmons, "Particle RRT for path planning with uncertainty," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 1617–1624.

[21] G. Kewlani, G. Ishigami, and K. Iagnemma, "Stochastic mobility-based path planning in uncertain environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 1183–1189.

[22] J. van den Berg, P. Abbeel, and K. Goldberg, "LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information," *Robots, Science and Systems*, 2010.

[23] J. van den Berg, S. Patil, R. Alterovitz, P. Abbeel, and K. Goldberg, "LQG-based planning, sensing and control of steerable needles," *Proc. Workshop on Algorithmic Foundation of Robotics*, pp. 373–389, 2010.

[24] A. Shkolnik, "Sample-based motion planning in high-dimensional and differentially-constrained systems," Ph.D. dissertation, Massachusetts Institute of Technology, 2010.

[25] L. Jaillet, J. Cortés, and T. Siméon, "Sampling-based path planning on configuration-space costmaps," *IEEE Transactions on Robotics*, vol. 26, pp. 635–646, 2009.