LQG-MP: Optimized Path Planning for Robots with Motion Uncertainty and Imperfect State Information

Jur van den Berg Pieter Abbeel Ken Goldberg

***Abstract*— This paper presents LQG-MP (linear-quadratic Gaussian motion planning), a new approach to robot motion planning that takes into account the sensors and the controller that will be used during execution of the robot's path. LQG-MP is based on the linear-quadratic controller with Gaussian models of uncertainty, and explicitly characterizes in advance (i.e., before execution) the a-priori probability distributions of the state of the robot along its path. These distributions can be used to assess the quality of the path, for instance by computing the probability of avoiding collisions. Many methods can be used to generate the needed ensemble of candidate paths from which the best path is selected; in this paper we report results using the RRT-algorithm. We study the performance of LQG-MP with simulation experiments in three scenarios involving a kinodynamic car-like robot, multi-robot planning with differential-drive robots, and a 6-DOF manipulator.**

I. Introduction

Motion uncertainty, i.e. the fact that the motion of the robot unpredictably deviates from what a dynamics model predicts, and imperfect state information due to partial or noisy measurements of the robot's state, arise in many real-world robotic tasks ranging from guiding mobile robots over uneven terrain to performing robotic surgery with high-DOF manipulators. The amount of motion and sensing uncertainty may depend on the particular motion that is executed and the state the robot is in, so different paths for the robot will have different uncertainties associated with them. Because safety and accuracy are of critical importance for many robotic tasks, these uncertainties will have significant influence on which path is best for the task at hand. The challenge we discuss in this paper is to precisely quantify these uncertainties in advance, such that the best path can be selected for the robot.

Many traditional path planners assume deterministic motion and full knowledge of the state [18], [13], and leave issues of uncertainty to the *control* phase in which the path may be executed using a feedback controller [15]. Planning and control are related but distinct fields. While recent work on path planning has addressed motion and/or sensing uncertainty (see Section II), most planning methods do not account for control during execution and most control methods take the path as given. LQG-MP builds a bridge between these disciplines and draws from results in both.

The key insight of LQG-MP is that the a-priori knowledge of the sensors and controller that will be used during the execution of the path can be used to optimize the path in the planning phase. We base our approach on the linear-quadratic controller (LQG-controller) with Gaussian models of the motion and sensing uncertainty, as it provides *optimal* control for guiding a robot along a planned path [4]. We will show that for a given stochastic model of the motion dynamics, and a stochastic model of the sensor measurements obtained during execution, it is possible to derive *in advance* (i.e. before execution) the a-priori probability distributions of the states and the control inputs of the robot along a given path (see Fig. 1). These distributions can be used to compute, for example, the probability that collisions will be avoided, the likelihood that the robot will arrive at the goal, or any other measure defining the quality of the path. We can then use any motion planning method to generate a large set of candidate paths, and select the path that is best with respect to the chosen planning objective.

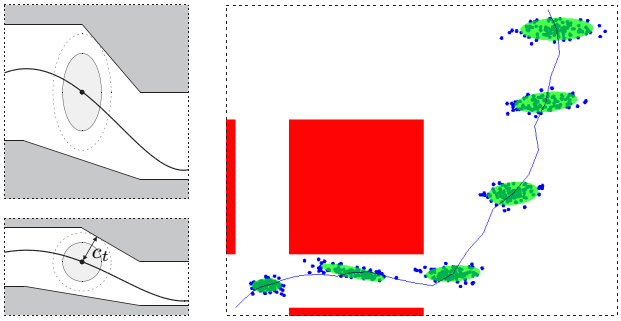


Fig. 1. (a) The maximum factor c_t by which the ellipse containing the positions within one standard deviation can be scaled before it intersects obstacles gives an indication of the probability that collisions will be avoided (top). c_t is computed as the Euclidean distance to the nearest obstacle in the environment transformed such that the ellipse becomes a unit disc (bottom). (b) The ellipses show the a-priori distributions as computed by LQG-MP along the best among the 1000 candidate paths for Scenario A. The samples result from performing 100 simulations.

Our approach is generally applicable to both holonomic and non-holonomic robots with state spaces of arbitrary dimension and kinematics and dynamics constraints. We assume that the stochastic dynamics model of the robot and the stochastic observation model are given explicitly, and that their stochasticity can be modeled by Gaussian noise. Our approach is designed for linear models, but can also be applied to non-linear models if they are locally well approximated by their linearizations.

We implemented our approach using the RRT motion planning algorithm [18] for representative path planning problems, and validated our approach using simulation experiments. We will show that the quality of candidate paths

This work was supported in part by NSF Award 0905344 and NIH Award 1R01EB-006435-01A1.

The authors are with the University of California at Berkeley, Berkeley, CA, USA. E-mail: {berg, pabbeel, goldberg}@berkeley.edu.

can differ starkly based on the uncertainty, even if traditional planning criteria such as path length or clearance from obstacles are similar, and that the type of sensors used during execution of the path has a significant influence on which path is best. A path planner that is unaware of the sensors, the controller and their uncertainties would not be able to make this distinction, and may produce sub-optimal paths.

The remainder of this paper is organized as follows. We start by discussing related work in Section II. We formally define the problem addressed in this paper in Section III. In Section IV we show how LQG-MP computes the a-priori probability distributions for a given path. In Section V, we discuss application examples and simulation results of LQG-MP for several motion and sensing models and planning objectives. We conclude in Section VI.

II. Related Work

A substantial body of work has addressed uncertainty in motion planning. The uncertainty typically originates from three sources: (i) motion uncertainty, (ii) sensing uncertainty and partial observations, and (iii) uncertainty about the environment. Our approach focuses on the first two, but is to some extent also applicable to the latter, as we will show in one of our experiments.

Planners that specifically take into account motion uncertainty include [14], [21], [29]. These planners plan paths that avoid rough terrain, but do not consider partial observability and sensing uncertainty. In [10], the probability of collisions is minimized for the specific case of a manipulator with base pose uncertainty. The sensing uncertainty is taken into account in the planner of [28], which aims to optimize the information content along a path. Planners in [5], [7], [20] assume that landmark regions exist in the environment where the accumulated motion uncertainty can be "reset".

Other approaches blend planning and control by defining a global control policy over the entire environment. MDPs, for instance, can be used with motion uncertainty to optimize probability of success [1], [30]. However, they require discretization of the state and control input spaces. The MDP concept can be extended to POMDPs to also include sensing uncertainty [12], [16], [26], but these suffer from issues of scalability [24]. The method of [17] also provides a global control policy in case of motion and sensing uncertainty.

Another class of planners considers the uncertainty about the environment and obstacles, rather than motion and sensing uncertainty [6], [9], [22], [23]. They typically aim to plan paths for which the probability of collisions is minimal.

Existing planners that are most directly related to LQG-MP take into account the available sensing capability to maximize the probability of arriving at the goal or to minimize expected cost [8], [11], [25], [27]. However, these algorithms implicitly assume to receive maximum-likelihood measurements from the sensors, which does not result in the true probability distributions of the state of the robot, but rather a measure of how well one will be able to infer the state. Besides the sensors, LQG-MP also takes into account the controller that will be used for executing the path, and computes the true a-priori probability distributions of the state of the robot along its future path.

III. Problem Definition

Let $\mathcal{X} = \mathbb{R}^n$ be the *state space* of the robot, and let $\mathcal{U} = \mathbb{R}^m$ be the *control input space* of the robot. We assume that time is discretized into stages of equal duration, and that applying a control input $\mathbf{u}_t \in \mathcal{U}$ at stage t brings the robot from state $\mathbf{x}_t \in \mathcal{X}$ at stage t to state $\mathbf{x}_{t+1} \in \mathcal{X}$ at stage $t+1$ according to a given stochastic dynamics model:

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}, \mathbf{m}_t), \qquad \mathbf{m}_t \sim \mathcal{N}(\mathbf{0}, M_t), \quad (1)$$

where $\mathbf{m}_t$ is the process noise at stage t drawn from a zero-mean Gaussian distribution with variance M_t that models the motion uncertainty. We assume that the function f is either linear or locally well approximated by its linearization.

Let us be given a start state $\mathbf{x}^{\text{start}} \in \mathcal{X}$ where the robot begins and a goal region $\mathcal{X}^{\text{goal}} \subset \mathcal{X}$ where the robot needs to go. A *path* Π for the robot is defined as a series of states and control inputs $(\mathbf{x}_0^\star, \mathbf{u}_0^\star, \ldots, \mathbf{x}_\ell^\star, \mathbf{u}_\ell^\star)$, such that $\mathbf{x}_0^\star = \mathbf{x}^{\text{start}}$, $\mathbf{x}_\ell^\star \in \mathcal{X}^{\text{goal}}$, and $\mathbf{x}_t^\star = f(\mathbf{x}_{t-1}^\star, \mathbf{u}_{t-1}^\star, \mathbf{0})$ for $0 < t \leq \ell$, where ℓ is the number of stages of the path. That is, a path connects the start state and the goal region, and is consistent with the dynamics model if there were no process noise.

During execution of the path, the robot will deviate from the path due to motion uncertainty. To compensate for unexpected motions, we assume that the path will be executed using a feedback controller that aims to keep the robot close to the path by minimizing the cost function

$$\mathbb{E}\Big(\sum_{t=0}^{\ell}((\mathbf{x}_t - \mathbf{x}_t^\star)^T C(\mathbf{x}_t - \mathbf{x}_t^\star) + (\mathbf{u}_t - \mathbf{u}_t^\star)^T D(\mathbf{u}_t - \mathbf{u}_t^\star))\Big), \quad (2)$$

which quadratically penalizes deviations from the path. C and D are given positive-definite weight matrices.

We assume that noisy sensors provide us with partial information about the state during execution of the path according to a given stochastic observation model:

$$\mathbf{z}_t = h(\mathbf{x}_t, \mathbf{n}_t), \qquad \mathbf{n}_t \sim \mathcal{N}(\mathbf{0}, N_t), \quad (3)$$

where $\mathbf{z}_t$ is the measurement obtained at stage t that relates to state $\mathbf{x}_t$ through function h, and $\mathbf{n}_t$ is the measurement noise drawn from a zero-mean Gaussian with variance N_t. We assume that the function h is either linear or locally well approximated by its linearization.

We define our problem in two parts; (i) given the stochastic dynamics model, the stochastic observation model, and the cost function, compute the a-priori distributions of the state and control input along a given path, and (ii) given a planning objective based on the probability distributions, select the best path among a large set of candidates.

IV. A-priori Probability Distributions

In this section we describe how to compute the a-priori probability distributions of the state and control input of the robot along a given path Π. For this, we use the fact that we know in advance what controller will be used to execute

the path: for linear dynamics and observation models with Gaussian noise and a quadratic cost function, the optimal approach for executing the path is to use an LQR feedback controller in parallel with a Kalman filter for state estimation, which is called linear-quadratic Gaussian (LQG) control [4]. A Kalman filter provides the optimal estimate of the state given previous state estimates, measurements and control inputs, and an LQR controller provides the optimal control input given the estimate of the state.

We will first discuss how to linearize the dynamics and observation model, and then review the Kalman filter and LQR controller. From these, we compute the a-priori probability distributions of the states and the control inputs of the robot along the path.

A. Linear(ized) Dynamics and Observation Model

In principle, our approach applies to linear dynamics and observation models f and h. However, since the robot is controlled to stay close to the path during execution, we can approximate non-linear models with local linearizations (i.e. first-order Taylor expansions) around the path Π. This gives the following linear(ized) stochastic dynamics and observation model:

$$\begin{aligned}\mathbf{x}_t &= f(\mathbf{x}^\star_{t-1}, \mathbf{u}^\star_{t-1}, \mathbf{0}) + A_t(\mathbf{x}_{t-1} - \mathbf{x}^\star_{t-1}) + \\ &\quad B_t(\mathbf{u}_{t-1} - \mathbf{u}^\star_{t-1}) + V_t\mathbf{m}_t, \qquad (4)\\ \mathbf{z}_t &= h(\mathbf{x}^\star_t, \mathbf{0}) + H_t(\mathbf{x}_t - \mathbf{x}^\star_t) + W_t\mathbf{n}_t, \qquad (5)\end{aligned}$$

where

$$A_t = \frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}^\star_{t-1}, \mathbf{u}^\star_{t-1}, \mathbf{0}),\ B_t = \frac{\partial f}{\partial \mathbf{u}}(\mathbf{x}^\star_{t-1}, \mathbf{u}^\star_{t-1}, \mathbf{0}),$$
$$V_t = \frac{\partial f}{\partial \mathbf{m}}(\mathbf{x}^\star_{t-1}, \mathbf{u}^\star_{t-1}, \mathbf{0}),\ H_t = \frac{\partial h}{\partial \mathbf{x}}(\mathbf{x}^\star_t, \mathbf{0}),\ W_t = \frac{\partial h}{\partial \mathbf{n}}(\mathbf{x}^\star_t, \mathbf{0})$$

are the Jacobian matrices of f and h along path Π.

It is convenient to express the control problem in terms of the *deviation* from the path. By defining

$$\bar{\mathbf{x}}_t = \mathbf{x}_t - \mathbf{x}^\star_t, \quad \bar{\mathbf{u}}_t = \mathbf{u}_t - \mathbf{u}^\star_t, \quad \bar{\mathbf{z}}_t = \mathbf{z}_t - h(\mathbf{x}^\star_t, \mathbf{0}), \quad (6)$$

as the state deviation, control input deviation and measurement deviation, respectively, we can formulate the dynamics and observation model of Equations (4) and (5) as

$$\begin{aligned}\bar{\mathbf{x}}_t &= A_t\bar{\mathbf{x}}_{t-1} + B_t\bar{\mathbf{u}}_{t-1} + V_t\mathbf{m}_t, \quad \mathbf{m}_t \sim \mathcal{N}(\mathbf{0}, M_t), \qquad (7)\\ \bar{\mathbf{z}}_t &= H_t\bar{\mathbf{x}}_t + W_t\mathbf{n}_t, \qquad \mathbf{n}_t \sim \mathcal{N}(\mathbf{0}, N_t), \qquad (8)\end{aligned}$$

and the cost function of Equation (2) as

$$\mathbb{E}\big(\sum_{t=0}^{\ell}(\bar{\mathbf{x}}_t^T C\bar{\mathbf{x}}_t + \bar{\mathbf{u}}_t^T D\bar{\mathbf{u}}_t)\big). \qquad (9)$$

This is the standard formulation of an LQG-control problem.

B. Kalman Filter for Optimal State Estimation

The Kalman filter keeps track of the estimate $\tilde{\mathbf{x}}_t$ and variance P_t of the true state $\bar{\mathbf{x}}_t$ during the execution of the path. It continually performs two steps; a process update to propagate the applied control input $\bar{\mathbf{u}}_t$, and a measurement update to incorporate the obtained measurement $\bar{\mathbf{z}}_t$:

Process update:

$$\begin{aligned}\tilde{\mathbf{x}}_t^- &= A_t\tilde{\mathbf{x}}_{t-1} + B_t\bar{\mathbf{u}}_{t-1} \qquad (10)\\ P_t^- &= A_tP_{t-1}A_t^T + V_tM_tV_t^T, \qquad (11)\end{aligned}$$

Measurement update:

$$\begin{aligned}K_t &= P_t^-H_t^T(H_tP_t^-H_t^T + W_tN_tW_t^T)^{-1} \qquad (12)\\ \tilde{\mathbf{x}}_t &= \tilde{\mathbf{x}}_t^- + K_t(\bar{\mathbf{z}}_t - H_t\tilde{\mathbf{x}}_t^-) \qquad (13)\\ P_t &= (I - K_tH_t)P_t^-. \qquad (14)\end{aligned}$$

These are the standard Kalman filter equations for optimal estimation given the dynamics and observation model of Equations (7) and (8) [31]. Note that the Kalman-gain matrices K_t can be computed *in advance* (i.e. before execution) given the initial variance P_0, without knowledge of the actual control inputs $\bar{\mathbf{u}}_t$ and measurements $\bar{\mathbf{z}}_t$.

C. LQR for Optimal Control

The control inputs $\bar{\mathbf{u}}_t$ that are optimal to apply during execution of the path are determined by the control policy that minimizes the cost function of Equation (9). For the dynamics model of Equation (7), the cost function is minimal when $\bar{\mathbf{u}}_t = L_t\bar{\mathbf{x}}_t$, where L_t is the *feedback matrix*, which is computed in advance for all $t \in 0, \ldots, \ell - 1$ using:

$$\begin{aligned}S_\ell &= C \qquad (15)\\ L_t &= -(B_{t+1}^TS_{t+1}B_{t+1} + D)^{-1}B_{t+1}^TS_{t+1}A_{t+1} \qquad (16)\\ S_t &= C + A_{t+1}^TS_{t+1}A_{t+1} + A_{t+1}^TS_{t+1}B_{t+1}L_t. \qquad (17)\end{aligned}$$

These are the standard equations for a finite-horizon discrete-time LQR controller [4].

As the true state $\bar{\mathbf{x}}_t$ is unknown, the estimate $\tilde{\mathbf{x}}_t$ of the state which is obtained from the Kalman filter is used to determine the control input $\bar{\mathbf{u}}_t$ at each stage t during execution of the path. Hence, the control policy is:

$$\bar{\mathbf{u}}_t = L_t\tilde{\mathbf{x}}_t. \qquad (18)$$

After application of the control input, the Kalman filter produces the estimate of the next state from which in turn a new control input is determined. This cycle repeats until the execution of the path is complete.

D. A-priori Distributions of State and Control Input

Given the LQR control policy and the Kalman filter, we can analyze in advance how the true state $\bar{\mathbf{x}}_t$ and the estimated state $\tilde{\mathbf{x}}_t$ will evolve during execution of the path as functions of each other. The evolution of the true state $\bar{\mathbf{x}}_t$ is dependent on the estimated state through the LQR control policy (Equation (18)) and the evolution of the estimated state $\tilde{\mathbf{x}}_t$ is dependent on the true state through the measurement obtained in the Kalman filter (Equation (13)). This gives the following equations:

$$\begin{aligned}\bar{\mathbf{x}}_t &= A_t\bar{\mathbf{x}}_{t-1} + B_tL_{t-1}\tilde{\mathbf{x}}_{t-1} + V_t\mathbf{m}_t, \qquad (19)\\ \tilde{\mathbf{x}}_t &= A_t\tilde{\mathbf{x}}_{t-1} + B_tL_{t-1}\tilde{\mathbf{x}}_{t-1} + K_t\big(\bar{\mathbf{z}}_t - \\ &\quad H_t(A_t\tilde{\mathbf{x}}_{t-1} + B_tL_{t-1}\tilde{\mathbf{x}}_{t-1})\big) \qquad (20)\end{aligned}$$

$$
\begin{aligned}
&= A_t\tilde{\mathbf{x}}_{t-1} + B_tL_{t-1}\tilde{\mathbf{x}}_{t-1} + K_t\big(H_t\bar{\mathbf{x}}_t + W_t\mathbf{n}_t - \\
&\quad H_t(A_t\tilde{\mathbf{x}}_{t-1} + B_tL_{t-1}\tilde{\mathbf{x}}_{t-1})\big) \\
&= A_t\tilde{\mathbf{x}}_{t-1} + B_tL_{t-1}\tilde{\mathbf{x}}_{t-1} + K_t\big(\\
&\quad H_t(A_t\bar{\mathbf{x}}_{t-1} + B_tL_{t-1}\tilde{\mathbf{x}}_{t-1} + V_t\mathbf{m}_t) + W_t\mathbf{n}_t - \\
&\quad H_t(A_t\tilde{\mathbf{x}}_{t-1} + B_tL_{t-1}\tilde{\mathbf{x}}_{t-1})\big) \\
&= A_t\tilde{\mathbf{x}}_{t-1} + B_tL_{t-1}\tilde{\mathbf{x}}_{t-1} + K_tH_tA_t\bar{\mathbf{x}}_{t-1} + \\
&\quad K_tH_tV_t\mathbf{m}_t + K_tW_t\mathbf{n}_t - K_tH_tA_t\tilde{\mathbf{x}}_{t-1},
\end{aligned}
$$

Equation (19) follows from substituting Equation (18) into Equation (7). The first equality of (20) follows from substituting Equation (18) into Equation (10) and Equation (10) into Equation (13); the second and third equalities follow after substituting Equations (8) and (19), respectively, and the fourth equality follows after expanding the terms.

Combining Equations (19) and (20) gives the matrix form:

$$
\begin{bmatrix}\bar{\mathbf{x}}_t\\ \tilde{\mathbf{x}}_t\end{bmatrix} = \begin{bmatrix} A_t & B_tL_{t-1} \\ K_tH_tA_t & A_t + B_tL_{t-1} - K_tH_tA_t\end{bmatrix}\begin{bmatrix}\bar{\mathbf{x}}_{t-1}\\ \tilde{\mathbf{x}}_{t-1}\end{bmatrix} + \begin{bmatrix} V_t & 0 \\ K_tH_tV_t & K_tW_t\end{bmatrix}\begin{bmatrix}\mathbf{m}_t\\ \mathbf{n}_t\end{bmatrix}, \quad \begin{bmatrix}\mathbf{m}_t\\ \mathbf{n}_t\end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \begin{bmatrix} M_t & 0 \\ 0 & N_t\end{bmatrix}),
$$

which we write shorthand (for the appropriate definitions of $\mathbf{y}_t$, $\mathbf{q}_t$, F_t, G_t and Q_t) as:

$$
\mathbf{y}_t = F_t\mathbf{y}_{t-1} + G_t\mathbf{q}_t, \qquad \mathbf{q}_t \sim \mathcal{N}(\mathbf{0}, Q_t). \tag{21}
$$

From this, we can compute the mean $\hat{\mathbf{y}}_t$ and the variance R_t of $\mathbf{y}_t = \left[\begin{smallmatrix}\bar{\mathbf{x}}_t\\ \tilde{\mathbf{x}}_t\end{smallmatrix}\right]$ for any stage t of the execution of the path:

$$
\hat{\mathbf{y}}_t = F_t\hat{\mathbf{y}}_{t-1}, \qquad \hat{\mathbf{y}}_0 = \mathbf{0}, \tag{22}
$$

$$
R_t = F_tR_{t-1}F_t^T + G_tQ_tG_t^T, \quad R_0 = \begin{bmatrix} P_0 & 0 \\ 0 & 0\end{bmatrix}. \tag{23}
$$

Note that the mean $\hat{\mathbf{y}}_t$ is zero for all stages t. Hence, $\left[\begin{smallmatrix}\bar{\mathbf{x}}_t\\ \tilde{\mathbf{x}}_t\end{smallmatrix}\right] \sim \mathcal{N}(\mathbf{0}, R_t)$. As it follows from Equations (18) and (6) that

$$
\begin{bmatrix}\mathbf{x}_t\\ \mathbf{u}_t\end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & L_t\end{bmatrix}\begin{bmatrix}\bar{\mathbf{x}}_t\\ \tilde{\mathbf{x}}_t\end{bmatrix} + \begin{bmatrix}\mathbf{x}_t^\star\\ \mathbf{u}_t^\star\end{bmatrix}, \tag{24}
$$

the a-priori distribution of the state $\mathbf{x}_t$ and the control input $\mathbf{u}_t$ at stage t of the execution of the path is:

$$
\begin{bmatrix}\mathbf{x}_t\\ \mathbf{u}_t\end{bmatrix} \sim \mathcal{N}(\begin{bmatrix}\mathbf{x}_t^\star\\ \mathbf{u}_t^\star\end{bmatrix}, \Lambda_tR_t\Lambda_t^T), \qquad \Lambda_t = \begin{bmatrix} I & 0 \\ 0 & L_t\end{bmatrix}. \tag{25}
$$

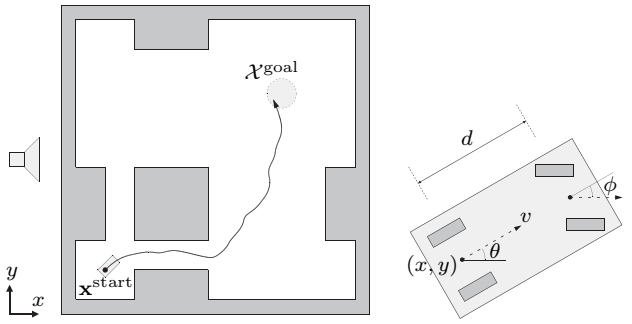
The covariance between $\left[\begin{smallmatrix}\mathbf{x}_i\\ \mathbf{u}_i\end{smallmatrix}\right]$ and $\left[\begin{smallmatrix}\mathbf{x}_j\\ \mathbf{u}_j\end{smallmatrix}\right]$ is given by:

$$
\mathrm{cov}(\begin{bmatrix}\mathbf{x}_i\\ \mathbf{u}_i\end{bmatrix}, \begin{bmatrix}\mathbf{x}_j\\ \mathbf{u}_j\end{bmatrix}) = \Lambda_iR_iF_{i+1}^TF_{i+2}^T\cdots F_j^T\Lambda_j^T, \quad i < j. \tag{26}
$$

Using the a-priori distributions, the quality of path Π can be computed with respect to the chosen planning objective. We can then use any motion planner to generate a large set of candidate paths, from which the best one is selected.

V. Example Applications and Results

In this section, we report simulation results for three scenarios in which LQG-MP is used to select a path. In each of the three scenarios, we use a different dynamics model, observation model and planning objective, and provide comparative analysis with a brute-force approach. We report results for an Intel P7350 2GHz with 4GB RAM.



(a) (b)

Fig. 2. (a) The environment of Scenario A, in which a car-like robot has to move between a start state and a goal region without colliding with obstacles. Sensors can only measure the y-coordinate of the position of the robot. The best path according to LQG-MP among the 1000 generated by RRT is shown. (b) The state $\mathbf{x}$ of a car-like robot.

For each scenario, we use the random rapidly-exploring tree (RRT) algorithm [18] to generate a large set of candidate paths. The RRT algorithm is well suited for our context as it can handle any dynamics model (without process noise) of the form of Equation (1) well. Even though it only plans a single path between the start state and the goal region, the path is generated randomly and will thus be different each time the algorithm is run. Hence, to generate multiple different paths, we run the RRT algorithm multiple times.

A. Car-Like Robot

In the first scenario, we apply LQG-MP to a non-holonomic car-like robot with 2nd-order dynamics in a 2-D environment with obstacles. The robot needs to move from a start state $\mathbf{x}^{\text{start}}$ to a goal region $\mathcal{X}^{\text{goal}}$ without colliding with the obstacles in the environment (see Fig. 2(a)).

1) Dynamics model: The state $\mathbf{x} = (x, y, \theta, v)$ of the robot is a 4-D vector consisting of its position (x, y), its orientation θ, and its speed v (see Fig. 2(b)). Its control input $\mathbf{u} = (a, \phi)$ is a 2-D vector consisting of an acceleration a and the steering wheel angle ϕ, corrupted by process noise $\mathbf{m} = (\tilde{a}, \tilde{\phi}) \sim \mathcal{N}(\mathbf{0}, \left[\begin{smallmatrix}\sigma_a^2 & 0\\ 0 & \sigma_\phi^2\end{smallmatrix}\right])$. This gives the following non-linear dynamics model:

$$
f(\mathbf{x}, \mathbf{u}, \mathbf{m}) = \begin{bmatrix} x + \tau v\cos\theta \\ y + \tau v\sin\theta \\ \theta + \tau v\tan(\phi + \tilde{\phi})/d \\ v + \tau(a + \tilde{a})\end{bmatrix}, \tag{27}
$$

where τ is the duration of a stage (time step), and d the distance between the front and rear axle of the car [19].

2) Observation model: To show the effect of partial sensing, the robot only receives feedback on the y-coordinate of its position. Hence, the measurement vector $\mathbf{z}$ is univariate and consists of a measurement of the y-coordinate of the robot corrupted by measurement noise $\mathbf{n} = \tilde{y} \sim \mathcal{N}(0, \sigma_y^2)$. This gives the following linear observation model:

$$
h(\mathbf{x}, \mathbf{n}) = y + \tilde{y}. \tag{28}
$$

Even though the sensor feedback is very partial, information about the other variables is still obtained through the interplay with the dynamics model.

3) Planning objective: We aim to find the path for the robot with a minimal probability of colliding with obstacles. Instead of computing this probability exactly, we will use an approximation that can be computed efficiently given the probability distributions along the path. To this end, we look at the number of standard deviations that one can deviate from the path before the robot may collide with an obstacle. Let this number be denoted c_t for stage t along the path. For a multivariate Gaussian distribution of dimension n, the probability that a sample is within c_t standard deviations is given by $\Gamma(n/2, c_t^2/2)$, where Γ is the regularized Gamma function [32]. It provides a lower bound of the probability of avoiding collisions at stage t. We now define the quality of a path Π as:

$$\prod_{t=0}^{\ell} \Gamma(n/2, c_t^2/2), \tag{29}$$

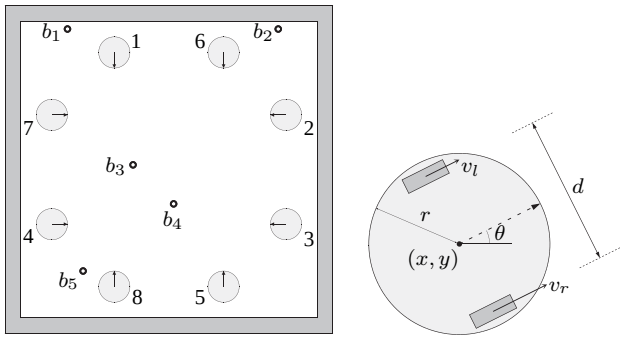
which is indicative of the probability that collisions will be avoided during execution. It is the planning objective to find a path for which this measure is maximal.

The value of c_t for stage t is computed as follows. For simplicity, we approximate the geometry of the car by a bounding disc, such that its orientation has no influence on whether or not the car is colliding. Also its speed does not influence its collision status. Hence, c_t is determined by the distribution $\mathcal{N}(\mathbf{p}_t, \Sigma_t)$ of the position of the car (i.e. $n = 2$), which is the marginal distribution of the first two variables of $\mathcal{N}(\left[\begin{smallmatrix}\mathbf{x}_t^\star \\ \mathbf{u}_t^\star\end{smallmatrix}\right], \Lambda_t R_t \Lambda_t^T)$ as computed in Equation (25). Let U_t be a matrix such that $U_t U_t^T = \Sigma_t$. The set of positions within one standard deviation is then an ellipse centered at the mean $\mathbf{p}_t$ obtained by transforming a unit disc by U_t, and c_t is the maximum factor by which the ellipse can be scaled such that it does not intersect with obstacles (see Fig. 1(a)).

Computing c_t can efficiently be implemented using a collision-checker that is capable of performing distance calculations and linear transformations on the geometry, for instance SOLID [3]. Transforming the environment (including the robot) by U_t^{-1} (such that the uncertainty ellipse becomes a unit disc, see Fig. 1(a)), and calculating the Euclidean distance between the robot and the nearest obstacle in the transformed environment gives the value of c_t for stage t.

4) Results: We randomly generated 1000 paths using the RRT algorithm, which took 56.8 seconds. For each of the paths, we computed the a-priori probability distributions and the measure of Equation (29), which took in total 2.67 seconds. The best path among the 1000 is shown in Fig. 2(a). It can be seen that the "lower-right" passage is chosen to get to the goal. This can be explained as the uncertainty will mainly be in the x-coordinate given that the sensors only provide feedback on the y-coordinate. The geometry of the lower-right passage allows for more deviation in the x-direction than the upper-left passage. Indeed, changing the observation model such that only the x-coordinate is measured results in a path that takes the upper-left passage.

To validate our results, we used a brute-force approach to estimate for each path the "ground-truth" probability that it will be executed without collisions. We performed 10,000 simulations of executions of the path using the LQR-controller and an extended Kalman Filter with artificially generated process and measurement noise, and counted the number of collision-free executions. This took in total 10440 seconds, which is almost 4000 times as much as the time needed by LQG-MP to evaluate the paths. It turns out that the path selected by LQG-MP has a 99% probability of success. The average probability of success over the 1000 paths is 61%, and the worst path has a probability of success of 13%. This is an indication of the typical and worst-case success rate of paths planned by a planner unaware of the uncertainties. Among the paths taking the upper-left passage, the best one has a success rate of 88% (versus 99% for the best path overall). This shows that the type of sensors used during execution has a significant influence on which path is optimal, even as the environment is symmetric.



(a) (b)

Fig. 3. (a) The environment of Scenario B, in which eight robots have to move to their antipodal position in the environment without mutual collisions. The numbers indicate the priority rank assigned to each robot. Five beacons $b_1, \ldots, b_5$ send out a signal whose strength decays quadratically with distance. (b) The state $\mathbf{x}$ of the differential-drive robot.

In Fig. 1(b) the samples of 100 simulations are shown for the best among the 1000 paths, along with the uncertainty ellipses of the a-priori probability distributions as computed by LQG-MP. As can be seen, the samples indeed follow the a-priori distributions computed by LQG-MP. This shows that any error introduced into LQG-MP by the linearization of the dynamics model is insignificant for this example.

B. Multi-Robot Planning with Differential-Drive Robots

In the second experiment, we apply LQG-MP to multi-robot motion planning with disc-shaped differential-drive robots (e.g. Roomba vacuum cleaners). Eight robots need to move simultaneously to their antipodal position in the environment without mutual collisions (see Fig. 3(a)). We use a prioritized approach to the multi-robot planning problem: the robots are planned for one by one in order of a priority assigned to them, and aim to avoid collisions with robots of higher priority, which are treated as moving obstacles [2]. This means that for each robot we apply LQG-MP to a dynamic environment in which not only the robot itself is subject to uncertainty, but also the obstacles (i.e. the robots of higher priority).

1) Dynamics model: The state $\mathbf{x} = (x, y, \theta)$ of each robot is a 3-D vector consisting of its position (x, y) and its

orientation θ (see Fig. 3(b)). Its control input $\mathbf{u} = (v_l, v_r)$ is a 2-D vector consisting of the speeds of the left and right wheel, respectively, corrupted by process noise $\mathbf{m} = (\tilde{v}_l, \tilde{v}_r) \sim \mathcal{N}(\mathbf{0}, \sigma_v^2 I)$. This gives the following non-linear dynamics model:

$$f(\mathbf{x}, \mathbf{u}, \mathbf{m}) = \begin{bmatrix} x + \frac{1}{2}\tau(v_l + \tilde{v}_l + v_r + \tilde{v}_r)\cos\theta \\ y + \frac{1}{2}\tau(v_l + \tilde{v}_l + v_r + \tilde{v}_r)\sin\theta \\ \theta + \tau(v_r + \tilde{v}_r - v_l - \tilde{v}_l)/d \end{bmatrix}, \quad (30)$$

where τ is the time step and d the distance between the left and right wheel of the robot [19].

2) Observation model: The robots receive feedback on their state from five beacons $b_1, \ldots, b_5$ scattered around the environment that each send out an identifiable signal of unit strength that decays quadratically with the distance to the beacon. Each beacon b_i has a known location $(\check{x}_i, \check{y}_i, 1)$. Hence, the measurement vector $\mathbf{z}$ consists of five readings of signal strengths, one from each beacon, corrupted by measurement noise $\mathbf{n} = (\tilde{b}_1, \ldots, \tilde{b}_5) \sim \mathcal{N}(\mathbf{0}, \sigma_b^2 I)$. This gives the following non-linear observation model:

$$h(\mathbf{x}, \mathbf{n}) = \begin{bmatrix} 1/((x-\check{x}_1)^2 + (y-\check{y}_1)^2 + 1) + \tilde{b}_1 \\ \vdots \\ 1/((x-\check{x}_5)^2 + (y-\check{y}_5)^2 + 1) + \tilde{b}_5 \end{bmatrix}. \quad (31)$$

3) Planning objective: For each robot, we aim to minimize the probability that it will collide with a robot of higher priority along its path. In this experiment, we approximate this probability more directly than we did for the first scenario. Let us assume we are planning for robot j, and that a path has already been planned for robots $1, \ldots, j-1$. As the robots are disc-shaped, only their position influences whether or not they collide. Let $\mathcal{N}(\mathbf{p}_t^i, \Sigma_t^i)$ be the marginal probability distribution of the position of robot i at stage t along i's path as computed by LQG-MP. Then, the distribution of the *relative* position of robot j and robot i (for $i \in 1, \ldots, j-1$) at stage t is $\mathcal{N}(\mathbf{p}_t^i - \mathbf{p}_t^j, \Sigma_t^i + \Sigma_t^j)$. The probability $\mathbb{P}_t(i \otimes j)$ that robot j collides with robot i at stage t is then given by:

$$\int_{\|\mathbf{p}\|<2r} \frac{\exp(-\frac{1}{2}(\mathbf{p}-\mathbf{p}_t^{ij})^T(\Sigma_t^i+\Sigma_t^j)^{-1}(\mathbf{p}-\mathbf{p}_t^{ij}))}{2\pi\det(\Sigma_t^i+\Sigma_t^j)^{1/2}}\,\mathrm{d}\mathbf{p}, \quad (32)$$

where $\mathbf{p}_t^{ij} = \mathbf{p}_t^i - \mathbf{p}_t^j$. This is the integral over the set of relative positions $\mathbf{p}$ for which the robots collide (that is when $\|\mathbf{p}\| < 2r$, where r is the radius of the robots) of the probability density function of the distribution of relative positions, and can be evaluated numerically. It follows that the probability that robot j does not collide with any robot at any stage along its path is:[1]

$$\prod_{t=0}^{\ell}\prod_{i=1}^{j-1}(1 - \mathbb{P}_t(i \otimes j)). \quad (33)$$

It is the planning objective for robot j to maximize this probability.

[1]Note that we assume here that the probabilities of avoiding collisions at different stages along the path are independent. This is not the case, but it will for practical purposes be a reasonable assumption.

TABLE I

RESULTS FOR SCENARIO B (1000 PATHS PER ROBOT)

	Computation time		Success rate	
robot	RRT	LQG-MP	Best path	Avg. path
1	22.3s	0.23s	100%	100%
2	28.2s	0.99s	100%	70.3%
3	29.5s	1.75s	100%	69.2%
4	30.5s	2.79s	100%	60.9%
5	57.0s	2.92s	99.2%	10.6%
6	49.8s	3.90s	99.8%	21.0%
7	39.2s	5.26s	99.9%	24.8%
8	77.8s	6.85s	99.7%	13.0%
total	334s	24.7s	98.6%	2.13%

As a secondary objective, we aim to minimize the uncertainty around the robot's path to leave maximal "space" for the other robots. That is, in case of equal probabilities of success, we aim to minimize the function $\sum_{t=0}^{\ell} \mathrm{tr}(\Sigma_t^j)$. This is equivalent to maximizing the likelihood that the robot will exactly follow the path Π during execution. The robot with the highest priority does not need to avoid other robots, so it will select its path purely based on the secondary objective.

4) Results: For each of the robots in turn, we planned 1000 paths using the RRT algorithm and selected the path that is best according to the planning objective. Note that the paths were planned such that, if there were no uncertainty, they are collision-free with respect to the robots of higher priority for which a path has already been selected. The result is shown in Fig. 4, along with the uncertainty ellipses of the a-priori probability distributions along the paths. It can be seen that the robots need to get close to the beacons to be able to estimate their position accurately. Almost all of the robots move through the region around the central beacons b_3 and b_4. At the same time, the robots aim to stay far away from each other, in order to minimize the probability of collisions. Robot 2, for instance, makes a wide detour around robot 1. Robot 3 first avoids robot 1 and then robot 2, causing its path to have a wide S-shape.

The quantitative results are given in Table I. The second column shows the time needed to plan 1000 paths for each robot, and the third column shows the time needed by LQG-MP to compute the probabilities of success for all paths. It shows that these probabilities can be computed efficiently. Per path, it takes an order of magnitude less time than planning the path itself. The third column shows the probability of success of the best path among the 1000 paths. This is the path that LQG-MP selects for the particular robot. The fourth column shows the average probability of success of the 1000 paths. This provides an indication of what an uncertainty-unaware planner would typically achieve. The probability that all eight robots successfully reach their goal is the product of the robot's individual probabilities of success, and is shown in the bottom row. This is 98.6% for LQG-MP, whereas an uncertainty-unaware planner would on average only have a 2.13% probability of success.

C. 6-DOF Manipulator

In the third experiment, we apply LQG-MP to a holonomic 6-DOF articulated robot in a 3-D environment. The robot needs to move from its initial state $\mathbf{x}^{\mathrm{start}}$ to a configuration

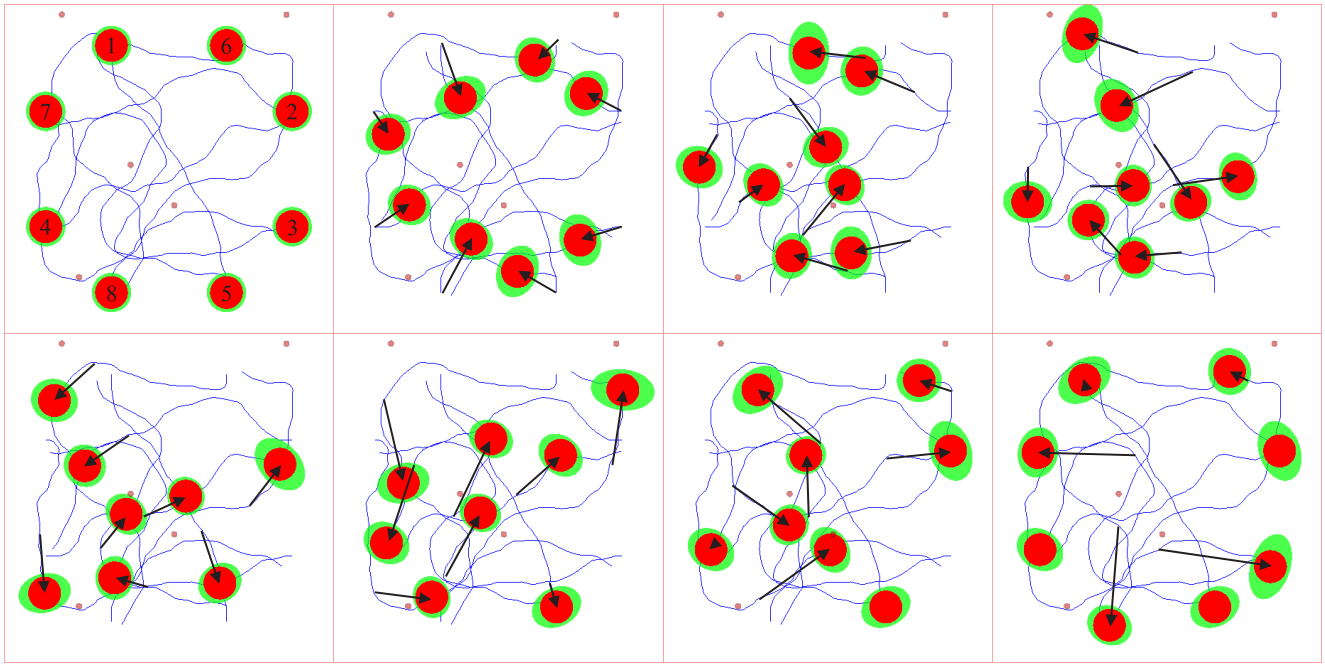
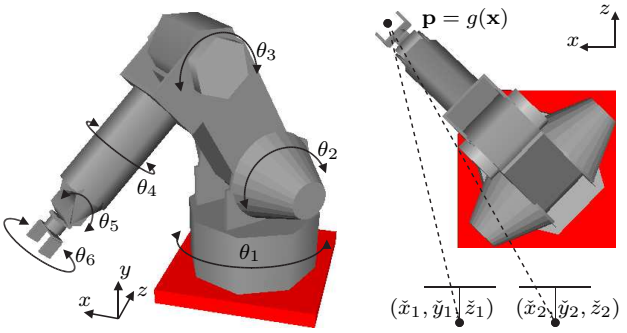


Fig. 4. The paths resulting from consecutively applying LQG-MP to each of the robots in Scenario B (snapshots at $t = 0, 3, 6, 9, 12, 16, 20, 28$). The numbers in the top-left image indicate the priority rank of the robots. The arrows show the movement with respect to the previous image. The robots enlarged by the uncertainty ellipses of their a-priori probability distributions are shown in green.



(a) (b)

Fig. 5. (a) The state $\mathbf{x}$ of the articulated robot of Scenario C. (b) A stereo camera tracks the position $\mathbf{p}$ of the end-effector of the robot.

in which the end-effector is inside a goal region on the other side of the environment.

1) Dynamics model: The state $\mathbf{x} = (\theta_1, \ldots, \theta_6)$ of the robot is a 6-D vector consisting of the angles of rotation at each of the joints (see Fig. 5(a)). The control input $\mathbf{u} = (\omega_1, \ldots, \omega_6)$ is a 6-D vector consisting of the angular speeds at each of the joints, corrupted by process noise $\mathbf{m} = (\tilde{\omega}_1, \ldots, \tilde{\omega}_6) \sim \mathcal{N}(\mathbf{0}, \sigma_\omega^2 I)$. Ignoring higher order dynamics, this results in the following linear dynamics model:

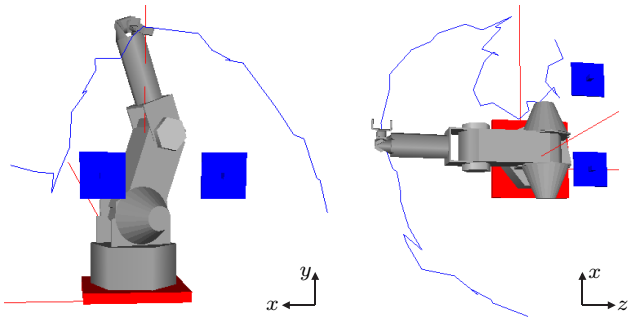
$$f(\mathbf{x}, \mathbf{u}, \mathbf{m}) = \begin{bmatrix} \theta_1 + \tau(\omega_1 + \tilde{\omega}_1) \\ \vdots \\ \theta_6 + \tau(\omega_6 + \tilde{\omega}_6) \end{bmatrix}. \tag{34}$$

2) Observation model: The robot receives feedback from a stereo camera that tracks the position of the end-effector of the robot. Let $\mathbf{p} = g(\mathbf{x})$ be the function relating the set of joint angles of the state $\mathbf{x}$ to the position $\mathbf{p} \in \mathbb{R}^3$ of the end-effector. This point is projected on the imaging plane of each camera i, which has a unit focal distance and a known location $(\check{x}_i, \check{y}_i, \check{z}_i)$ (see Fig. 5(b)). Hence, the measurement $\mathbf{z}$ is a 4-D vector consisting of the pixel coordinates of the end-effector on the imaging planes of both cameras, corrupted by measurement noise $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma_n^2 I)$. Ignoring occlusions, this gives the following non-linear observation model:

$$h(\mathbf{x}, \mathbf{n}) = \begin{bmatrix} (g_x(\mathbf{x}) - \check{x}_1)/(g_z(\mathbf{x}) - \check{z}_1) \\ (g_y(\mathbf{x}) - \check{y}_1)/(g_z(\mathbf{x}) - \check{z}_1) \\ (g_x(\mathbf{x}) - \check{x}_2)/(g_z(\mathbf{x}) - \check{z}_2) \\ (g_y(\mathbf{x}) - \check{y}_2)/(g_z(\mathbf{x}) - \check{z}_2) \end{bmatrix} + \mathbf{n}. \tag{35}$$

3) Planning objective: We aim to maximize the likelihood that the end-effector arrives at its goal position. Let $\mathcal{N}(\mathbf{p}_\ell, \Sigma_\ell)$ be the distribution of the position of the end-effector at the last stage of the path, then this likelihood is maximal when $\mathrm{tr}(\Sigma_\ell)$ is minimal. Σ_ℓ can be approximated from the variance X_ℓ of the state $\mathbf{x}_\ell$ computed by LQG-MP as $\Sigma_\ell = T_\ell X_\ell T_\ell^T$, where $T_\ell = \frac{\partial g}{\partial \mathbf{x}}(\mathbf{x}_\ell^\star)$, i.e. the Jacobian matrix of function g at the goal position.

4) Results: We planned 1000 paths for the robot using the RRT algorithm, and computed for each the likelihood of arriving at the goal. Constructing the paths took 192 seconds, and evaluating them using LQG-MP took 1.16 seconds. The path found best is shown in Fig. 6(a). Interestingly, the robot chooses to move in the plane perpendicular to the viewing direction of the camera while being fully stretched out. In such configurations, the position of the end-effector contains most information about the angles at the joints. This apparently outweighs the benefit of more precise positioning when the end-effector is closer to the camera. Indeed, an experiment in which the camera is placed above the robot



(a) (b)

Fig. 6. The best path among the candidates for Scenario C when the cameras (blue squares) are placed (a) next to the robot and (b) above the robot. The jaggedness of the paths is due to the random nature of the RRT algorithm.

results in a path with similar characteristics (see Fig. 6(b)).

VI. Conclusion and Future Work

We have presented LQG-MP, a new approach to evaluate paths in motion planning for robots subject to motion and sensing uncertainty. LQG-MP precisely characterizes the a-priori probability distributions of the state of the robot along a given path, based on which the path can be optimized for the particular task. We have shown that this considerably increases the probability of a successful execution when compared to uncertainty-unaware planners. The key of LQG-MP is that it takes into account the a-priori knowledge of both the sensors and controller in the planning phase.

In the experiments we performed, we have not used the a-priori distributions of the control input that LQG-MP also computes, nor the covariances between the states at different stages along the path. We envision that these could be used to compute the conditional distributions of the remainder of the path after each application of a control input during the execution. If the new distributions indicate that the quality has dropped below a threshold, we might opt to *replan*. Current planning times, though, do not allow for real-time application of LQG-MP. It is a major objective of future work to bring planning times down, for instance by devising a focused planner such that planning a large set of candidate paths is no longer required. Other limitations, such as the fact that the candidate paths may not constitute a representative sample in high-dimensional state spaces, and the jaggedness of the paths that RRT produces, might then also be resolved.

We plan to use LQG-MP in future work on optimizing accuracy and safety in challenging robotic applications, such as autonomous helicopter flight, needle steering for prostate brachytherapy, and robotic-assisted surgery.

References

[1] R. Alterovitz, T. Siméon, K. Goldberg. The stochastic motion roadmap: a sampling framework for planning with Markov motion uncertainty. *Proc. Robotics: Science and Systems*, 2007.

[2] J. van den Berg, M. Overmars. Prioritized motion planning for multiple robots. *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2005.

[3] G. van den Bergen. *Collision detection in interactive 3D environments*. Morgan Kaufmann Publishers, 2004.

[4] D. Bertsekas. *Dynamic programming and optimal control*. Athena Scientific, 2001.

[5] B. Bouilly, T. Simeon, R. Alami. A numerical technique for planning motion strategies of a mobile robot in presence of uncertainty. *Proc. IEEE Int. Conf. on Robotics and Automation*, 1995.

[6] B. Burns, O. Brock. Sampling-based motion planning with sensing uncertainty. *Proc. IEEE Int. Conf. on Robotics and Automation*, 2007.

[7] T. Fraichard, R. Mermond. Path planning with uncertainty for car-like robots. *Proc. IEEE Int. Conf. on Robotics and Automation*, 1998.

[8] J. Gonzalez, A. Stentz. Using linear landmarks for path planning with uncertainty in outdoor environments. *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2009.

[9] L. Guibas, D. Hsu, H. Kurniawati, E. Rehman. Bounded uncertainty roadmaps for path planning. *Proc. Workshop on Algorithmic Foundations of Robotics*, 2008.

[10] Y. Huang, K. Gupta. Collision-probability constrained PRM for a manipulator with base pose uncertainty. *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2009.

[11] V. Huynh, N. Roy. icLQG: combining local and global optimization for control in information space. *Proc. IEEE Int. Conf. on Robotics and Automation*, 2009.

[12] L. Kaelbling, M. Littman, A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101(1-2):99–134, 1998.

[13] L. Kavraki, P. Svestka, J.-C. Latombe, M. Overmars. Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Trans. on Robotics and Automation* 12:4(566–580), 1996.

[14] G. Kewlani, G. Ishigami, K. Iagnemma. Stochastic mobility-based path planning in uncertain environments. *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2009.

[15] Y. Kuwata, J. Teo, S. Karaman, G. Fiore, E. Frazzoli, J. How. Motion planning in complex environments using closed-loop prediction. *Proc. AIAA Guidance, Navigation, and Control Conf. and Exhibit*, 2008.

[16] H. Kurniawati, D. Hsu, W. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. *Proc. Robotics: Science and Systems*, 2008.

[17] S. LaValle, S. Hutchinson. An objective-based framework for motion planning under sensing and control uncertainties. *Int. J. of Robotics Research* 17(1):19–42, 1998.

[18] S. LaValle, J. Kuffner. Randomized kinodynamic planning. *Int. Journal on Robotics Research* 20(5):378–400, 2001.

[19] S. LaValle. *Planning algorithms*. Cambridge University Press, 2006.

[20] A. Lazanas, J. Latombe. Motion planning with uncertainty: a landmark approach. *Artificial Intelligence*, 76(1-2):285–317, 1995.

[21] N. Melchior, R. Simmons. Particle RRT for path planning with uncertainty. *Proc. IEEE Int. Conf. on Robotics and Automation*, 2007.

[22] P. Missiuro, N. Roy. Adapting probabilistic roadmaps to handle uncertain maps. *Proc. IEEE Int. Conf. on Robotics and Automation*, 2006.

[23] A. Nakhaei, F. Lamiraux. A framework for planning motions in stochastic maps. *Proc. Int Conf. on Control, Automation, Robotics and Vision*, 2008.

[24] C. Papadimitriou, J. Tsisiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[25] R. Pepy, A. Lambert. Safe path planning in an uncertain-configuration space using RRT. *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2006.

[26] J. Porta, N. Vlassis, M. Spaan, P. Poupart. Point-based value iteration for continuous POMDPs. *Journal of Machine Learning Research* 7:23292367, 2006.

[27] S. Prentice, N. Roy. The belief roadmap: efficient planning in linear POMDPs by factoring the covariance. *Proc. Int. Symp. of Robotics Research*, 2008.

[28] N. Roy, W. Burgard, D. Fox, S. Thrun. Coastal navigation - mobile robot navigation with uncertainty in dynamic environments. *Proc. IEEE Int. Conf. on Robotics and Automation*, 1999.

[29] R. Tedrake. LQR-trees: Feedback motion planning on sparse randomized trees. *Proc. Robotics: Science and Systems*, 2009.

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

[31] G. Welch, G. Bishop. An introduction to the Kalman filter. *Tech. Report TR 95-041*, University of North Carolina at Chapel Hill, 2006.

[32] Wikipedia. Chi-square distribution. http://en.wikipedia.org/wiki/Chi_square, 2010.