LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information

Jur van den Berg, Pieter Abbeel and Ken Goldberg

The International Journal of Robotics Research 2011 30: 895 originally published online 3 June 2011
DOI: 10.1177/0278364911406562

The International Journal of Robotics Research
30(7) 895–913

DOI: 10.1177/0278364911406562
ijr.sagepub.com

LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information

Jur van den Berg[1], Pieter Abbeel[2] and Ken Goldberg[2]

Abstract

In this paper we present 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 the 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 required ensemble of candidate paths from which the best path is selected; in this paper we report results using rapidly exploring random trees (RRT). We study the performance of LQG-MP with simulation experiments in three scenarios: (A) a kinodynamic car-like robot, (B) multi-robot planning with differential-drive robots, and (C) a 6-DOF serial manipulator. We also present a method that applies Kalman smoothing to make paths C_k-continuous and apply LQG-MP to precomputed roadmaps using a variant of Dijkstra's algorithm to efficiently find high-quality paths.*

Keywords

Planning, control, uncertainty

1. Introduction

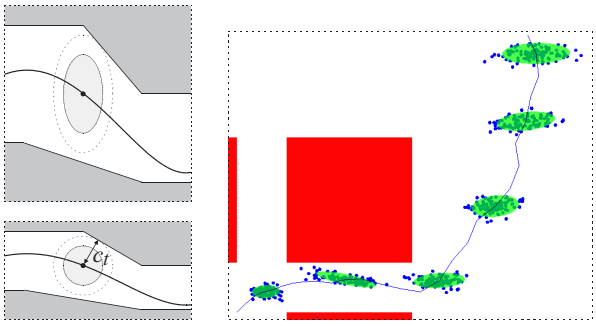
Motion uncertainty, e.g. due to unmodeled external influences on the motion of the robot, and imperfect state information due to partial or noisy measurements of the robot's state, arise in many real-world robotic tasks ranging from guiding mobile robots over uneven terrain to performing robotic surgery with high-degree-of-freedom (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 the safety and accuracy are of critical importance for many robotic tasks, these uncertainties will have a 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 (LaValle and Kuffner 2001; Kavraki et al. 1996), and leave issues of uncertainty to the *control* phase in which the path may be executed using a feedback controller (Kuwata et al. 2008). Planning and control are related but distinct fields. While recent work on path planning has addressed motion and/or sensing uncertainty (see Section 2), most planning methods do not take control into account during execution and most control methods take the path as given. LQG-MP (linear-quadratic Gaussian motion planning) 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 Gaussian (LQG) controller with Gaussian models of the motion and sensing uncertainty, as it provides *optimal* control for guiding a robot along a planned path (Bertsekas 2001). We show that for a given stochastic model of the

[1]University of North Carolina at Chapel Hill, Chapel Hill, NC, USA
[2]University of California, Berkeley, CA, USA

Corresponding author:
Jur van den Berg, Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC 27599-3175, USA.
Email: berg@cs.unc.edu



(a) (b)

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). Here 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 1,000 candidate paths for scenario A. The samples result from performing 100 simulations.

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 Figure 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.

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 rapidly exploring random trees (RRT) motion planning algorithm (LaValle and Kuffner 2001) for representative path planning problems, and validated our approach using simulation experiments. We show that the quality of candidate paths 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 suboptimal paths.

Paths obtained with RRT motion planning can be non-smooth. Standard smoothing techniques tend to short-cut paths or can even result in paths that are not dynamically feasible (e.g. through the use of splines for smoothing). We describe a Kalman smoothing approach, which can produce paths that are C_k continuous and illustrate its performance in conjunction with LQG-MP.

While the LQG-MP approach does not directly lend itself to efficient roadmap-based planning, we describe an approximation similar to the approximations made in past work on planning in information spaces (Prentice and Roy 2009) for the LQG-MP setting. This approximation enables efficient planning over pre-computed roadmaps.

This paper is organized as follows. We start by discussing related work in Section 2. We formally define the problem addressed in this paper in Section 3. In Section 4 we show how LQG-MP computes the *a priori* probability distributions for a given path. In Section 5 we discuss application examples and simulation results of LQG-MP for several motion and sensing models and planning objectives. In Section 6 we discuss a Kalman smoothing approach to make paths C_k-continuous while avoiding obstacles. In Section 7 we describe an approximation to the LQG-MP evaluation which enables using a variant of Dijkstra's algorithm on a pre-computed roadmap to efficiently find good paths. We conclude in Section 8.

2. 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 of the robot's own state, and (iii) uncertainty about the environment. Our approach focuses on the first two, but is also applicable to the latter if distributions of the position of obstacles in the environment are available *a priori*, as we show in one of our experiments. Our approach does not explicitly account for sensing of the environment.

Planners that specifically take into account motion uncertainty include those of Kewlani et al. (2009), Melchior and Simmons (2007), and Tedrake (2009). These planners plan paths that avoid rough terrain, but do not consider partial observability and sensing uncertainty. In Huang and Gupta (2009), 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 Roy et al. (1999), which aims to optimize the information content along a path. Planners in Bouilly et al. (1995), Fraichard and Mermond (1998), and Lazanas and Latombe (1995) 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. Markov decision processes (MDPs), for instance, can be used with motion uncertainty to optimize probability of success (Thrun et al. 2005; Alterovitz et al. 2007). However, they require discretization of the state and control input spaces. The MDP concept can be extended to partially observable Markov decision processes (POMDPs) to also include sensing uncertainty (Kaelbling et al. 1998; Porta et al. 2006; Kurniawati et al. 2008). While POMDPs suffer from issues of scalability (Papadimitriou and Tsisiklis 1987), recent advances have shown considerable success in applying approximate sample-based solutions to POMDPs in reasonable computation times. The method of LaValle and Hutchinson (1998) 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 (Missiuro and Roy 2006; Burns and Brock 2007; Guibas et al. 2008; Nakhaei and Lamiraux 2008). 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 (Pepy and Lambert 2006; Gonzalez and Stentz 2009; Huynh and Roy 2009; Prentice and Roy 2009; Platt et al. 2010). 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. In addition to 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. Also, in addition to maximizing the likelihood of arrival at the goal, LQG-MP can also be used to minimize the probability of collisions with obstacles in order to maximize the probability of successful execution of the path.

3. 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:

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

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, the path will be executed using a feedback controller that aims to keep the robot close to the path. We assume that noisy sensors provide us with partial information about the state according to a given stochastic observation model:

$$\mathbf{z}_t = h(\mathbf{x}_t, \mathbf{n}_t), \quad \mathbf{n}_t \sim \mathcal{N}(\mathbf{0}, N_t), \tag{2}$$

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 and the stochastic observation model, 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.

4. *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 LQG control (Bertsekas 2001). 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 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.

4.1. Linear(ized) dynamics and observation model

In principle, our approach applies to linear dynamics and observation models f and h. However, since the robot will be 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:

$$\begin{aligned}\mathbf{x}_t &= f(\mathbf{x}^\star_{t-1}, \mathbf{u}^\star_{t-1}, \mathbf{0}) + A_t(\mathbf{x}_{t-1} - \mathbf{x}^\star_{t-1}) \\ &\quad + B_t(\mathbf{u}_{t-1} - \mathbf{u}^\star_{t-1}) + V_t\mathbf{m}_t, \qquad (3)\\ \mathbf{z}_t &= h(\mathbf{x}^\star_t, \mathbf{0}) + H_t(\mathbf{x}_t - \mathbf{x}^\star_t) + W_t\mathbf{n}_t, \qquad (4)\end{aligned}$$

where

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

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

$$\begin{aligned}\bar{\mathbf{x}}_t &= \mathbf{x}_t - \mathbf{x}^\star_t,\\ \bar{\mathbf{u}}_t &= \mathbf{u}_t - \mathbf{u}^\star_t,\\ \bar{\mathbf{z}}_t &= \mathbf{z}_t - h(\mathbf{x}^\star_t, \mathbf{0}),\end{aligned} \qquad (6)$$

as the state deviation, control input deviation, and measurement deviation, respectively, we can formulate the dynamics and observation model of Equations (3) and (4) as

$$\bar{\mathbf{x}}_t = A_t\bar{\mathbf{x}}_{t-1} + B_t\bar{\mathbf{u}}_{t-1} + V_t\mathbf{m}_t, \quad \mathbf{m}_t \sim \mathcal{N}(\mathbf{0}, M_t), \qquad (7)$$

$$\bar{\mathbf{z}}_t = H_t\bar{\mathbf{x}}_t + W_t\mathbf{n}_t, \quad \mathbf{n}_t \sim \mathcal{N}(\mathbf{0}, N_t). \qquad (8)$$

This is a standard formulation of a model for LQG-control problems.

4.2. 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:

$$\tilde{\mathbf{x}}^-_t = A_t\tilde{\mathbf{x}}_{t-1} + B_t\bar{\mathbf{u}}_{t-1}, \qquad (9)$$

$$P^-_t = A_tP_{t-1}A^{\mathrm{T}}_t + V_tM_tV^{\mathrm{T}}_t, \qquad (10)$$

Measurement update:

$$K_t = P^-_tH^{\mathrm{T}}_t(H_tP^-_tH^{\mathrm{T}}_t + W_tN_tW^{\mathrm{T}}_t)^{-1}, \qquad (11)$$

$$\tilde{\mathbf{x}}_t = \tilde{\mathbf{x}}^-_t + K_t(\bar{\mathbf{z}}_t - H_t\tilde{\mathbf{x}}^-_t), \qquad (12)$$

$$P_t = (I - K_tH_t)P^-_t. \qquad (13)$$

These are the standard Kalman filter equations for optimal estimation given the dynamics and observation model of Equations (7) and (8) (Welch and Bishop 2006). 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 $\bar{\mathbf{z}}_t$.

4.3. 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 a quadratic cost function defined over the execution. As the linearizations of the motion and observation models are good approximations only when the actual states and control inputs are close to those along the path, we define the cost function as

$$\mathbb{E}\left(\sum_{t=0}^{\ell}(\bar{\mathbf{x}}^{\mathrm{T}}_tC\bar{\mathbf{x}}_t + \bar{\mathbf{u}}^{\mathrm{T}}_tD\bar{\mathbf{u}}_t)\right), \qquad (14)$$

such that deviations from the path are quadratically penalized, given positive-definite weight matrices C and D.

For the dynamics model of Equation (7), the cost function is minimal when $\bar{\mathbf{u}}_t = L_{t+1}\bar{\mathbf{x}}_t$, where L_t is the *feedback matrix*, which is computed in advance for all $t \in 1, \ldots, \ell$ using backward recursion:

$$S_\ell = C, \qquad (15)$$

$$L_t = -(B^{\mathrm{T}}_tS_tB_t + D)^{-1}B^{\mathrm{T}}_tS_tA_t, \qquad (16)$$

$$S_{t-1} = C + A^{\mathrm{T}}_tS_tA_t + A^{\mathrm{T}}_tS_tB_tL_t. \qquad (17)$$

These are the standard equations for a finite-horizon discrete-time LQR controller (Bertsekas 2001).

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

$$\bar{\mathbf{u}}_t = L_{t+1}\tilde{\mathbf{x}}_t. \qquad (18)$$

It follows from the *separation theorem*, which states that the estimator (Kalman filter) and the controller can be optimized independently (for linear systems with quadratic cost) (Bertsekas 2001), that this control policy is optimal. 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.

4.4. 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 (12)). This gives the following equations:

$$\bar{\mathbf{x}}_t = A_t\bar{\mathbf{x}}_{t-1} + B_tL_t\tilde{\mathbf{x}}_{t-1} + V_t\mathbf{m}_t, \tag{19}$$

$$\begin{aligned}\tilde{\mathbf{x}}_t &= A_t\tilde{\mathbf{x}}_{t-1} + B_tL_t\tilde{\mathbf{x}}_{t-1} + K_t\big(\bar{\mathbf{z}}_t - H_t(A_t\tilde{\mathbf{x}}_{t-1} + B_tL_t\tilde{\mathbf{x}}_{t-1})\big) \\ &= A_t\tilde{\mathbf{x}}_{t-1} + B_tL_t\tilde{\mathbf{x}}_{t-1} + K_t\big(H_t\bar{\mathbf{x}}_t + W_t\mathbf{n}_t \\ &\quad - H_t(A_t\tilde{\mathbf{x}}_{t-1} + B_tL_t\tilde{\mathbf{x}}_{t-1})\big) \\ &= A_t\tilde{\mathbf{x}}_{t-1} + B_tL_t\tilde{\mathbf{x}}_{t-1} + K_t\big(H_t(A_t\bar{\mathbf{x}}_{t-1} + B_tL_t\tilde{\mathbf{x}}_{t-1} + V_t\mathbf{m}_t) \\ &\quad + W_t\mathbf{n}_t - H_t(A_t\tilde{\mathbf{x}}_{t-1} + B_tL_t\tilde{\mathbf{x}}_{t-1})\big) \\ &= A_t\tilde{\mathbf{x}}_{t-1} + B_tL_t\tilde{\mathbf{x}}_{t-1} + K_tH_tA_t\tilde{\mathbf{x}}_{t-1} + K_tH_tV_t\mathbf{m}_t + K_tW_t\mathbf{n}_t \\ &\quad - K_tH_tA_t\tilde{\mathbf{x}}_{t-1},\end{aligned} \tag{20}$$

Equation (19) follows from substituting Equation (18) into Equation (7). The first equality of (20) follows from substituting Equation (18) into Equation (9) and Equation (9) into Equation (12); 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:

$$\begin{bmatrix}\bar{\mathbf{x}}_t \\ \tilde{\mathbf{x}}_t\end{bmatrix} = \begin{bmatrix}A_t & B_tL_t \\ K_tH_tA_t & A_t + B_tL_t - K_tH_tA_t\end{bmatrix}\begin{bmatrix}\bar{\mathbf{x}}_{t-1} \\ \tilde{\mathbf{x}}_{t-1}\end{bmatrix} + \begin{bmatrix}V_t & 0 \\ K_tH_tV_t & K_tW_t\end{bmatrix}\begin{bmatrix}\mathbf{m}_t \\ \mathbf{n}_t\end{bmatrix},$$

where

$$\begin{bmatrix}\mathbf{m}_t \\ \mathbf{n}_t\end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \begin{bmatrix}M_t & 0 \\ 0 & N_t\end{bmatrix}).$$

We write these equations in 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 using forward recursion:

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

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

The upper-left block of R_t provides the *unconditional a priori* variance of the state (deviation) $\bar{\mathbf{x}}_t$. In contrast, the matrix P_t computed in the Kalman filter is the *conditional* variance of the state deviation $\bar{\mathbf{x}}_t$ given its estimate $\tilde{\mathbf{x}}_t$. Since the estimate $\tilde{\mathbf{x}}_0$ of the initial state deviation is fully known *a priori*, R_0 is initialized with zero variance for the estimate and variance P_0 for the true state.

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

$$\begin{bmatrix}\mathbf{x}_t \\ \mathbf{u}_t\end{bmatrix} = \begin{bmatrix}I & 0 \\ 0 & L_{t+1}\end{bmatrix}\begin{bmatrix}\bar{\mathbf{x}}_t \\ \tilde{\mathbf{x}}_t\end{bmatrix} + \begin{bmatrix}\mathbf{x}_t^\star \\ \mathbf{u}_t^\star\end{bmatrix}, \tag{24}$$

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

$$\begin{bmatrix}\mathbf{x}_t \\ \mathbf{u}_t\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\mathbf{x}_t^\star \\ \mathbf{u}_t^\star\end{bmatrix}, \Lambda_tR_t\Lambda_t^{\mathrm{T}}\right), \quad \Lambda_t = \begin{bmatrix}I & 0 \\ 0 & L_{t+1}\end{bmatrix}. \tag{25}$$

The covariance between the states and control inputs $\left[\begin{smallmatrix}\mathbf{x}_i \\ \mathbf{u}_i\end{smallmatrix}\right]$ and $\left[\begin{smallmatrix}\mathbf{x}_j \\ \mathbf{u}_j\end{smallmatrix}\right]$ at stages i and j along the path is given by

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

These *a priori* distributions (Equation (25)) and their covariances (Equation (26)) are correct regardless of the *controllability* and *observability* of the dynamics and observation model of Equations (7) and (8). Even for formally uncontrollable and/or unobservable systems, the LQR-controller and Kalman filter provide the optimal control policy and state estimate, respectively. The controller or observer may not converge in this case, but this will be reflected in the computed *a priori* probability distributions: the *a priori* uncertainty in the state will be larger the further along the path.

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. In the experiments we present next, we use the *a priori* distributions of the state to approximate the probability of collisions with obstacles. We have not used the *a priori* distributions of the control input, nor the covariances between the states at different stages along the path, but we envision that the former can be used to compute the probability that the applied control inputs remain within their bounds, and the latter to compute the conditional distributions of the remainder of the path after each application of a control input during the execution.

5. 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 2 GHz with 4 GB RAM.

For each scenario, we use the RRT algorithm (LaValle and Kuffner 2001) 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.

5.1. Car-like robot

In the first scenario, we apply LQG-MP to a non-holonomic car-like robot with second-order dynamics in a two-dimensional 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 Figure 2(a)).

5.1.1. Dynamics model The state $\mathbf{x} = (x, y, \theta, v)$ of the robot is a four-dimensional vector consisting of its position (x, y), its orientation θ, and its speed v (see Figure 2(b)). Its control input $\mathbf{u} = (a, \phi)$ is a two-dimensional 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 (LaValle 2006).

5.1.2. 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 and renders the system formally *unobservable*, some information about the other variables is still obtained through the interplay with the dynamics model.

5.1.3. 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 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 by 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 (see http://en.wikipedia.org/wiki/Chi_square). It provides a lower bound of the probability of avoiding collisions at stage t. We now define the quality of a path Π by multiplying these probabilities at all stages:

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

Even though this (erroneously) assumes the probabilities to be independent between stages along the path, the quality measure 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 $\mathcal{N}(\left[\begin{smallmatrix} \mathbf{x}_t^\star \\ \mathbf{u}_t^\star \end{smallmatrix}\right], \Lambda_t R_t \Lambda_t^{\mathrm{T}})$ as computed in Equation (25). Let U_t be a matrix such that $U_t U_t^{\mathrm{T}} = \Sigma_t$. 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 Figure 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 (van den Bergen 2004). Transforming the environment (including the robot) by U_t^{-1} (such that the uncertainty ellipse becomes a unit disc, see Figure 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.

5.1.4. Results We randomly generated 1,000 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 1,000 is shown in Figure 2(a). It can be seen that the 'lower-right' passage is chosen

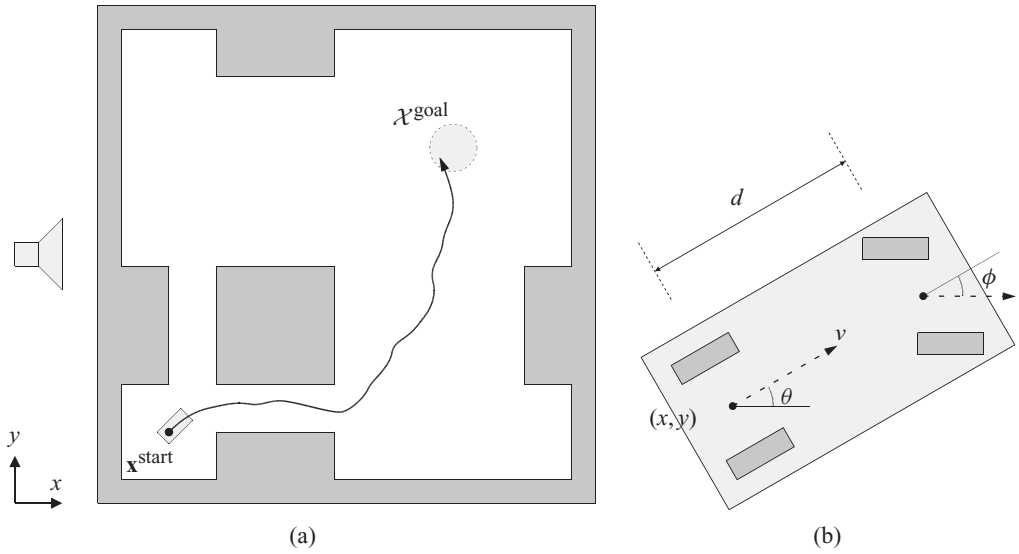


Fig. 2. (a) The environment of scenario A, in which a car-like robot has to move between a start state and a goal region without colliding with obstacles. Sensors can only measure the *y*-coordinate of the position of the robot. The best path according to LQG-MP among the 1,000 generated by RRT is shown. (b) The state **x** of a car-like robot.

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 10,440 seconds, which is almost 4,000 times as long as the time needed by LQG-MP to evaluate the paths. The results are summarized in Table 1. It turns out that the path selected by LQG-MP has a 99% probability of success. The average probability of success over the 1,000 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 and traditional metrics such as path length and clearance cannot discriminate between the upper-left and lower-right passage.

Table 1. Results for scenario A (1,000 paths, 10,000 simulations per path).

Path	Success rate
Best overall	99%
Average overall	61%
Worst overall	13%
LQG-MP	99%
Best upper-left	88%

In Figure 1(b) the samples of 100 simulations are shown for the best among the 1,000 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.

5.1.5. Analyzing linearization and Gaussian effects To analyze the effect of the linearization on the distributions as computed by LQG-MP versus the true distributions resulting from performing simulations, we performed experiments for varying levels of initial, process, and sensing noise. When these noise levels are high and samples are expected to deviate far from the path, then they may fall outside the 'tube' around the path where the linearization is valid, which will result in different distributions than those computed by LQG-MP. To measure the 'distance' between distributions, we use the symmetric Kullback–Leibler (KL)

Table 2. Linearization and Gaussian effects (10,000 simulations).

Noise factor x	KL divergence	α	KL divergence
1	0.001	0.1	0.007
2	0.002	0.2	0.007
3	0.007	0.3	0.008
4	0.047	0.4	0.008
5	21	0.5	0.03
6	971	0.6	0.007
7	3×10^5	0.7	0.006
8	3×10^4	0.8	0.006
9	3×10^7	0.9	0.007
10	4×10^4	1.0	0.006

divergence (see http://en.wikipedia.org/wiki/Kullback-Leibler_divergence) between the Gaussian distribution computed by LQG-MP and the distribution computed from performing 10,000 simulations. To compute the KL divergence, we fit a Gaussian to the distribution of the 10,000 samples by computing their mean and variance. The symmetric KL divergence between two Gaussians $\mathcal{N}(\mathbf{m}_0, \Sigma_0)$ and $\mathcal{N}(\mathbf{m}_1, \Sigma_1)$ of dimension n in *nats* is then given by

$$\begin{aligned} KL = \frac{1}{4}\Bigg(&\mathrm{tr}(\Sigma_1^{-1}\Sigma_0) + (\mathbf{m}_1 - \mathbf{m}_0)^{\mathrm{T}}\,\Sigma_1^{-1}(\mathbf{m}_1 - \mathbf{m}_0) \\ &- \log\frac{\det\Sigma_0}{\det\Sigma_1} - n\Bigg) \\ +\frac{1}{4}\Bigg(&\mathrm{tr}(\Sigma_0^{-1}\Sigma_1) + (\mathbf{m}_0 - \mathbf{m}_1)^{\mathrm{T}}\,\Sigma_0^{-1}(\mathbf{m}_0 - \mathbf{m}_1) \\ &- \log\frac{\det\Sigma_1}{\det\Sigma_0} - n\Bigg), \end{aligned} \tag{30}$$

which is the average of the asymmetric KL divergences in either direction.

The results are given in the left two columns of Table 2 for the best path found by LQG-MP shown in Figure 1(b). The noise levels are varied by multiplying the matrices P_0, M, and N with a factor x^2, and the average KL divergence is shown of the distributions of each of the stages along the path.

As can be seen, the linearization is valid for all 10,000 simulations for noise levels up to a factor of four. The distributions computed by LQG-MP closely match the distributions computed from the simulations. For a noise factor of five some of the simulation samples escape the tube around the path where the linearization is valid, and then show random behavior. It is to be noted that the vast majority of the 10,000 simulations follow a distribution according to that computed by LQG-MP, but the few that do not significantly change its variance resulting in a relatively high divergence. For noise factors of six and above the controller cannot be trusted for a significant number of samples, resulting in very high divergences. In these cases, a LQR-controller computed around the path is not an adequate control approach to handle the amount of uncertainty. As a result, LQG-MP does not precompute accurate probability distributions in these cases.

To study the effect of the Gaussian assumption on the distributions of the uncertainty in LQG-MP, we perform simulations where the actual noise is increasingly non-Gaussian. To this end, we blend a uniform distribution with a Gaussian distribution for the initial uncertainty, the motion noise, and the sensor noise. For each of these, the uniform distribution $\mathcal{U}$ has an identical mean and variance as the Gaussian distribution $\mathcal{N}$ (with a noise factor of three), and a parameter $0 \leq \alpha \leq 1$ gives the weighting factor between the uniform and the Gaussian distributions. The resulting combined distribution is then given by

$$\sqrt{\frac{1}{\alpha^2(1-\alpha)^2}}(\alpha\mathcal{U} + (1-\alpha)\mathcal{N}), \tag{31}$$

where the normalizing factor ensures that the variance remains constant. The right two columns of Table 2 show the average KL divergence between the distributions computed by LQG-MP (which assumes Gaussian distributed noise) and the distributions resulting from the 10,000 simulations for increasingly non-Gaussian noise along the path selected by LQG-MP. The results suggest that the fact that the noise is actually non-Gaussian has very little effect on the resulting distributions. The distributions predicted by LQG-MP based on the assumption that the noise is Gaussian are almost the same, which is reflected by a low value for the divergence between the distributions for any value of α (the relatively high divergence for $\alpha = 0.5$ is due to the randomness among the 10,000 simulations). It should be noted that LQG-MP was aware of the correct mean and variance of the noise distributions, and it seems that this is more important than the specific nature of the distribution. Obviously, if LQG-MP assumes a different mean and variance than the actual noise distributions, it is not able to predict the correct probability distributions of the state of the robot along the path.

5.2. *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 Figure 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

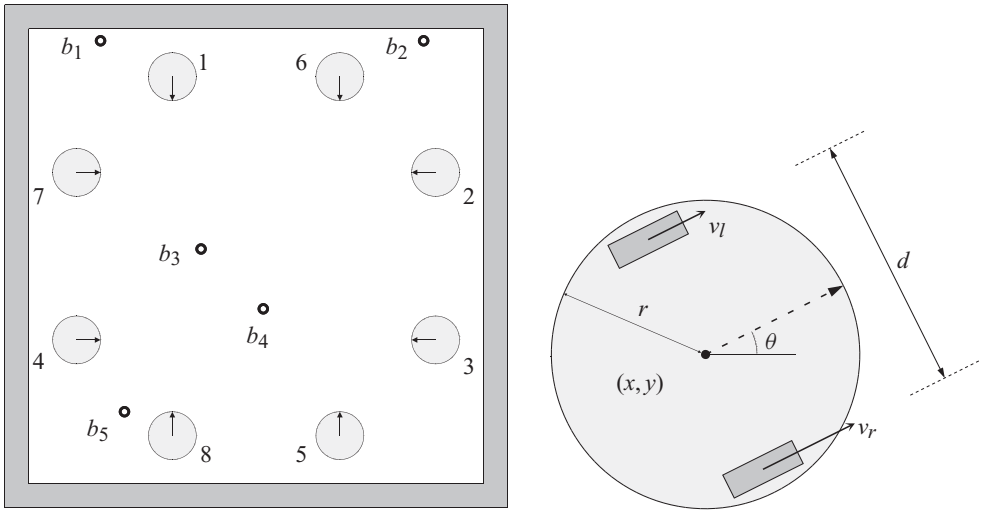


Fig. 3. (a) The environment of scenario B, in which eight robots have to move to their antipodal position in the environment without mutual collisions. The numbers indicate the priority rank assigned to each robot. Five beacons $b_1, \ldots, b_5$ send out a signal whose strength decays quadratically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

obstacles (van den Berg and Overmars 2005). 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).

5.2.1. Dynamics model The state $\mathbf{x} = (x, y, \theta)$ of each robot is a three-dimensional vector consisting of its position (x, y) and its orientation θ (see Figure 3(b)). Its control input $\mathbf{u} = (v_l, v_r)$ is a two-dimensional 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:

$$f(\mathbf{x}, \mathbf{u}, \mathbf{m}) = \begin{bmatrix} x + \frac{1}{2}\tau(v_l + \tilde{v}_l + v_r + \tilde{v}_r)\cos\theta \\ y + \frac{1}{2}\tau(v_l + \tilde{v}_l + v_r + \tilde{v}_r)\sin\theta \\ \theta + \tau(v_r + \tilde{v}_r - v_l - \tilde{v}_l)/d \end{bmatrix}, \quad (32)$$

where τ is the time step and d the distance between the left and right wheel of the robot (LaValle 2006).

5.2.2. 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:

$$h(\mathbf{x}, \mathbf{n}) = \begin{bmatrix} 1/((x - \check{x}_1)^2 + (y - \check{y}_1)^2 + 1) + \tilde{b}_1 \\ \vdots \\ 1/((x - \check{x}_5)^2 + (y - \check{y}_5)^2 + 1) + \tilde{b}_5 \end{bmatrix}. \quad (33)$$

5.2.3. 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 that 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

$$\int_{\|\mathbf{p}\|<2r} \frac{\exp(-\frac{1}{2}(\mathbf{p} - \mathbf{p}_t^{ij})^{\mathrm{T}} (\Sigma_t^i + \Sigma_t^j)^{-1} (\mathbf{p} - \mathbf{p}_t^{ij}))}{2\pi \det(\Sigma_t^i + \Sigma_t^j)^{1/2}} \, \mathrm{d}\mathbf{p}, \quad (34)$$

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

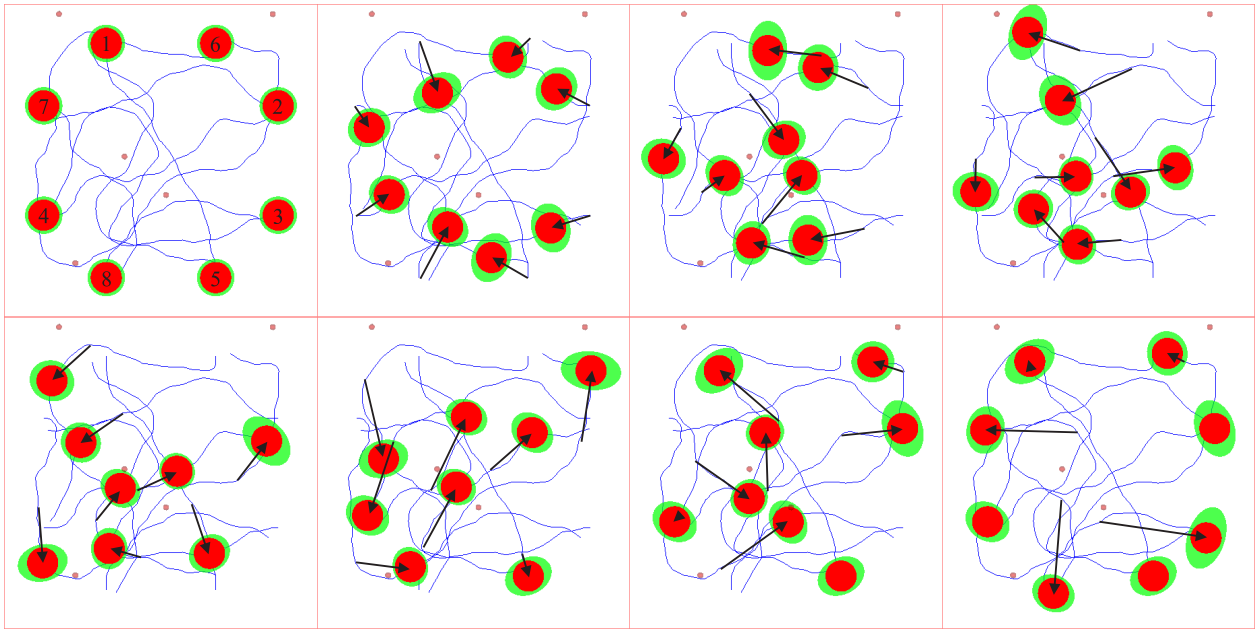


Fig. 4. The paths resulting from consecutively applying LQG-MP to each of the robots in scenario B (snapshots at $t = 0, 3, 6, 9, 12, 16, 20, 28$). The numbers in the top-left image indicate the priority rank of the robots. The arrows show the movement with respect to the previous image. The robots enlarged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

the probability that robot j does not collide with any robot at any stage along its path is:[1]

$$\prod_{t=0}^{\ell} \prod_{i=1}^{j-1} (1 - \mathbb{P}_t(i \otimes j)) . \tag{35}$$

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

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.

5.2.4. Results For each of the robots in turn, we planned 1,000 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 Figure 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.

Table 3. Results for scenario B (1,000 paths per robot).

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

The quantitative results are given in Table 3. The second column shows the time needed to plan 1,000 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 1,000 paths. This is the path that LQG-MP selects for the particular robot. The fourth column shows the average probability of success of the 1,000 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.

5.3. 6-DOF manipulator

In the third experiment, we apply LQG-MP to a holonomic 6-DOF articulated robot in a three-dimensional environment. The robot needs to move from its initial state $\mathbf{x}^{\text{start}}$ to a configuration in which the end-effector is inside a goal region on the other side of the environment.

5.3.1. Dynamics model The state $\mathbf{x} = (\theta_1, \ldots, \theta_6)$ of the robot is a six-dimensional vector consisting of the angles of rotation at each of the joints (see Figure 5(a)). The control input $\mathbf{u} = (\omega_1, \ldots, \omega_6)$ is a six-dimensional 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:

$$f(\mathbf{x}, \mathbf{u}, \mathbf{m}) = \begin{bmatrix} \theta_1 + \tau(\omega_1 + \tilde{\omega}_1) \\ \vdots \\ \theta_6 + \tau(\omega_6 + \tilde{\omega}_6) \end{bmatrix}. \tag{36}$$

5.3.2. 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 onto 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 Figure 5(b)). Hence, the measurement $\mathbf{z}$ is a four-dimensional 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:

$$h(\mathbf{x}, \mathbf{n}) = \begin{bmatrix} (g_x(\mathbf{x}) - \check{x}_1)/(g_z(\mathbf{x}) - \check{z}_1) \\ (g_y(\mathbf{x}) - \check{y}_1)/(g_z(\mathbf{x}) - \check{z}_1) \\ (g_x(\mathbf{x}) - \check{x}_2)/(g_z(\mathbf{x}) - \check{z}_2) \\ (g_y(\mathbf{x}) - \check{y}_2)/(g_z(\mathbf{x}) - \check{z}_2) \end{bmatrix} + \mathbf{n}. \tag{37}$$

5.3.3. 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 $\text{tr}(\Sigma_\ell)$ is minimal. Σ_ℓ can be approximated from the variance X_ℓ of the state $\mathbf{x}_\ell$ computed by LQG-MP as $\Sigma_\ell = T_\ell X_\ell T_\ell^{\mathrm{T}}$, 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.

5.3.4. Results We planned 1,000 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 Figure 6(a). The robot chooses to move in a plane parallel to the viewing direction of the camera while being fully stretched out. This brings the end-effector closer to the camera, where it can be positioned precisely. Interestingly, the worst paths are those in the plane perpendicular to the camera. Indeed, an experiment in which the camera is placed above the robot results in best (and worst) paths with similar characteristics (see Figure 6(b)).

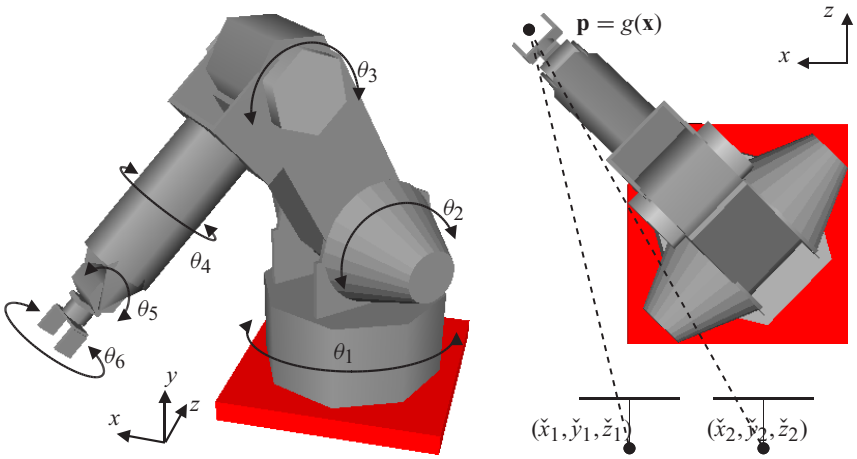
6. Path smoothing

Owing to the randomized nature of the RRT algorithm, the paths it produces can be non-smooth. This is particularly apparent in scenario C (see Figure 6) as the dynamics are of first-order, i.e. the velocity is controlled directly. This can be addressed by *smoothing*. However, most smoothing techniques in randomized motion planning repeatedly attempt to reduce path length between two randomly chosen states along the path (Geraerts and Overmars 2007). These techniques tend to bring the path closer to obstacles, which can be suboptimal with respect to the planning objective, for instance when minimizing the probability of collisions, and tend to map multiple paths to the same smoothed result, hence they are not ideal for LQG-MP.

In this section we introduce a technique based on *Kalman smoothing* that can produce a path that is C_k-continuous, for any value of k, while staying close to the original path. The measure of smoothness is the magnitude of the kth-order difference of the control input $\mathbf{u}$ between consecutive stages of the path. The Kalman smoother places a *stochastic* constraint on the magnitude of the kth-order differences, which is set by the user in the form of a variance matrix.

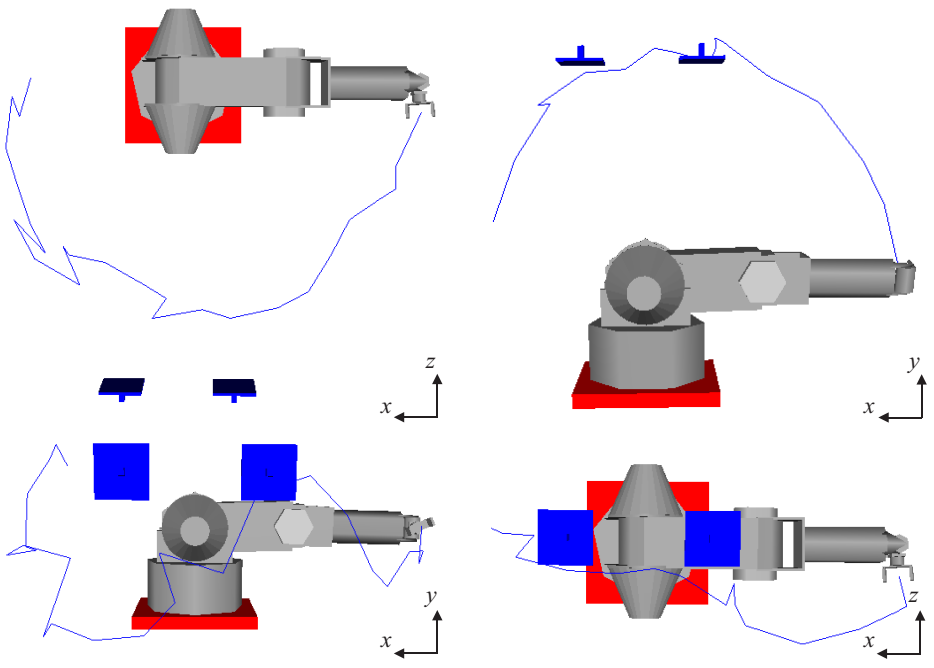
6.1. Kalman smoothing

In general, the Kalman smoother is able to infer the optimal estimate of the state at any stage of a dynamics system given all past and all 'future' measurements of the state. Whereas the Kalman filter produces a real-time estimate of the current state given all measurements so far, the Kalman



(a) (b)

Fig. 5. (a) The state **x** of the articulated robot of scenario C. (b) A stereo camera pair provides noisy observations of the position **p** of the end-effector of the robot. These observations are non-uniform; positions nearer the cameras can be observed more precisely and motion parallel to the camera planes can be observed more precisely than motion orthogonal to the camera planes. We consider how two placements of the cameras affect the resulting plans.



(a) (b)

Fig. 6. Resulting paths for two placements of the cameras. In the left column (a), the cameras are placed beside the robot and the path is shown from the top and side views. In the right column (b), the cameras are placed above the robot and the path is shown from the side and top views.

smoother is typically used offline to infer the most likely trajectory of the robotic system in hindsight given all measurements along the trajectory. Here, we use the Kalman smoother to smooth a path produced by the RRT algorithm by defining a pseudo-dynamics model that enforces smoothness on the path by stochastically constraining the magnitude of the kth-order difference of the control input, and a pseudo-observation model that lets the states and control inputs along the unsmooth RRT path be 'noisy observations' of the smooth path.

Let the dynamics model $f(\mathbf{x}, \mathbf{u})$ be linear and deterministic and given by

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) = A_t \mathbf{x}_{t-1} + B_t \mathbf{u}_{t-1}. \tag{38}$$

In scenario C for example, $A = I$ and $B = \tau I$ (see Section 5.3.1). Let $\Pi = (\mathbf{x}_0^\star, \mathbf{u}_0^\star, \ldots, \mathbf{x}_\ell^\star, \mathbf{u}_\ell^\star)$ be the original unsmoothed path generated by the RRT algorithm. We now smooth the path by generating a new path Π' using a Kalman smoother that complies with a higher-order pseudo-dynamics model f', and uses the states and control inputs along the original path Π as noisy observations of the new smoothed path Π'. The pseudo-dynamics model is given by

$$\mathbf{x}'_t = f'(\mathbf{x}'_t, \mathbf{m}'_t) = A'_t \mathbf{x}'_{t-1} + V'_t \mathbf{m}'_t, \quad \mathbf{m}'_t \sim \mathcal{N}(\mathbf{0}, M'), \tag{39}$$

where a pseudo-state $\mathbf{x}'$ is defined as the concatenation of the state $\mathbf{x}$, the control input $\mathbf{u}$, and differences $\Delta\mathbf{u}, \Delta^2\mathbf{u}, \ldots, \Delta^{k-1}\mathbf{u}$ of the control input up to order $k-1$. The process matrix A'_t integrates all of the control input differences, and assumes the $(k-1)$th-order difference $\Delta^{k-1}\mathbf{u}$ stays the same:

$$\mathbf{x}'_t = \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \\ \Delta\mathbf{u}_t \\ \vdots \\ \Delta^{k-2}\mathbf{u}_t \\ \Delta^{k-1}\mathbf{u}_t \end{bmatrix},$$

$$A'_t = \begin{bmatrix} A_t & B_t & \frac{1}{2!}B_t & \cdots & \frac{1}{(k-1)!}B_t & \frac{1}{k!}B_t \\ 0 & I & \frac{1}{1!}I & \cdots & \frac{1}{(k-2)!}I & \frac{1}{(k-1)!}I \\ 0 & 0 & I & \ddots & & \frac{1}{(k-2)!}I \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \vdots & & \ddots & I & \frac{1}{1!}I \\ 0 & 0 & \cdots & \cdots & 0 & I \end{bmatrix} \tag{40}$$

The only change allowed in the $(k-1)$th-order difference $\Delta^{k-1}\mathbf{u}$ comes from the zero-mean random variable $\mathbf{m}'_t = \Delta^k \mathbf{u}_t$ with variance matrix M'_t, which is the kth-order difference of the control input, which is also integrated into the other state variables:

$$V'_t = \begin{bmatrix} \frac{1}{(k+1)!}B_t \\ \frac{1}{k}I \\ \frac{1}{(k-1)!}I \\ \vdots \\ \frac{1}{2!}I \\ I \end{bmatrix}, \quad \mathbf{m}'_t = \Delta^k \mathbf{u}_t, \quad \Delta^k \mathbf{u}_t \sim \mathcal{N}(\mathbf{0}, M'). \tag{41}$$

The pseudo-observation model $h'(\mathbf{x}', \mathbf{n}')$ is given by

$$\mathbf{z}'_t = h'(\mathbf{x}'_t, \mathbf{n}'_t) = H'_t \mathbf{x}'_t + \mathbf{n}'_t, \quad \mathbf{n}'_t \sim \mathcal{N}(\mathbf{0}, N'_t), \tag{42}$$

where the pseudo-observation $\mathbf{z}'_t$ at stage t consists of the state and control input of the original path produced by the RRT algorithm:

$$\mathbf{z}'_t = \begin{bmatrix} \mathbf{x}_t^\star \\ \mathbf{u}_t^\star \end{bmatrix}, \qquad H'_t = \begin{bmatrix} I & 0 & 0 & \cdots & 0 \\ 0 & I & 0 & \cdots & 0 \end{bmatrix}. \tag{43}$$

The variance matrix N'_t is constant and equal to N' for all t except for $t = 0$ and $t = \ell$, when its upper-left portion is 0 to ensure that the smoothed path Π' precisely begins and ends in the start and goal state.

The Kalman smoother produces the most 'likely' path given the stochastic constraint defined by the pseudo-dynamics model f' and the stochastic constraint defined by the 'noisy observations'. The pseudo-dynamics model enforces the path to be kth-order smooth, while the observation model ensures that the path follows the original unsmoothed path Π. The smoothed path Π' can be generated by applying the Rauch–Tung–Striebel two-pass filter to the pseudo-dynamics and observation model. The forward pass is similar to the Kalman filter. The backward pass incorporates the 'future' measurements into the estimate of the pseudo-state:

Forward pass: $0 < t \leq \ell$

$$\hat{\mathbf{x}}'_{t|t-1} = A'_t \hat{\mathbf{x}}'_{t-1|t-1} \tag{44}$$

$$P'_{t|t-1} = A'_t P'_{t-1|t-1} A_t'^{\mathrm{T}} + V'_t M'_t V_t'^{\mathrm{T}}, \tag{45}$$

$$K'_t = P'_{t|t-1} H_t'^{\mathrm{T}} (H'_t P'_{t|t-1} H_t'^{\mathrm{T}} + N'_t)^{-1} \tag{46}$$

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + K'_t (\mathbf{z}'_t - H'_t \hat{\mathbf{x}}_{t|t-1}) \tag{47}$$

$$P'_{t|t} = (I - K'_t H'_t) P_{t|t-1}. \tag{48}$$

Backward pass: $\ell > t \geq 0$

$$L'_t = P'_{t|t} A_t'^{\mathrm{T}} P'^{-1}_{t+1|t} \tag{49}$$

$$\hat{\mathbf{x}}'_{t|\ell} = \hat{\mathbf{x}}'_{t|t} + L'_t (\hat{\mathbf{x}}'_{t+1|\ell} - \hat{\mathbf{x}}'_{t+1|t}) \tag{50}$$

$$P'_{t|\ell} = P'_{t|t} + L'_t (P'_{t+1|\ell} - P'_{t+1|t}) L_t'^{\mathrm{T}}, \tag{51}$$

where $\hat{\mathbf{x}}'_{s|t}$ and $P'_{s|t}$ are the mean and variance of the pseudo-state at stage s given 'measurements' up to time t. The pseudo-states $\hat{\mathbf{x}}'_{t|\ell}$ are the final pseudo-states along the smoothed path Π'.

The order k, the process noise M', and the observation noise N' are parameters of this smoothing technique. It ensures that the state $\mathbf{x}$ along path Π' is $k-1$ times differentiable. Varying the relative magnitude of M' and N' provides a trade-off between smoothness in the kth order and closeness to the original path.

6.2. Results

We implemented a first-order smoother (i.e. $k = 1$) and experimented with it on scenario C. Prior to evaluating each

Table 4. Kalman smoothing of paths in Figure 6. Quality of paths with varying noise in the pseudo-observation matrix (σ'); the trace of the variance of the end-effector position ($\mathrm{tr}(T_\ell X_\ell T_\ell^{\mathrm{T}})$) at the goal state is shown (lower is better).

	Cameras next to robot	Cameras above robot	
σ'	Path 1	Path 2	Path 3
0	4.00	2.37	2.73
1	4.10	2.43	2.63
10	4.21	2.55	2.35
100	4.28	2.56	2.38

of the 1,000 paths that were randomly generated by the RRT algorithm, we first smoothed them using various relative magnitudes of $M' = I$ and $N' = \sigma'^2 I$. The results are given in Table 4.[2] The first row ($\sigma' = 0$) shows results for the unsmoothed path. The subsequent rows show results for increasingly smoother paths.

For the setup where the cameras are placed next to the robot (second column), the same path turned out to be best for increasing degrees of smoothing (labeled path 1 in the table). Interestingly, though, the quality of the path as defined by the likelihood of arriving at the goal position decreases slightly as the path gets smoother (the trace gets larger). Apparently, the smoother pulls the path away from the (local) optimum. In Figure 7, the smoothed path is shown for $\sigma' = 1$ and $\sigma' = 10$ (note that the non-smoothed path is shown in Figure 6(a)).

For the setup where the cameras are placed above the robot (third and fourth column), different paths are best for different degrees of smoothing. These paths showed the same global characteristics, though. Also in this case the quality of the path that is best when no smoothing takes place (labeled path 2 in the table) decreases for increasing degrees of smoothness. On the other hand, the quality of path that is best when the paths are maximally smoothed (labeled path 3 in the table) increases with the degree of smoothing. For this particular path the smoother pulls the path towards a local optimum. For $\sigma' = 10$, it has a higher quality than the non-smoothed best path.

7. LQG-MP on roadmaps

In the LQG-MP technique as presented so far, we have created a large set of candidate paths and evaluated each of them to find a good path. A limitation of this approach is that it takes a significant amount of computation time to construct all of the candidate paths. This limitation can be partly overcome by considering paths contained in a *roadmap* that is pre-constructed for the environment.

Since the Kalman gain matrices K_t along a path are computed using forward recursion (from the start state, see Section 4.2), and the LQR feedback matrices L_t using backward recursion (from the goal state, see Section 4.3), the entire path needs to be given in order to compute the *a priori* probability distributions along the path using LQG-MP. As a general roadmap contains an exponential number of paths between any pair of start and goal vertices, iterating over all of them is typically infeasible. By defining a separate controller for each edge in the roadmap, we can use a variant of Dijkstra's algorithm to find a path within the roadmap nearly instantaneously.

7.1. Finding a path within a roadmap

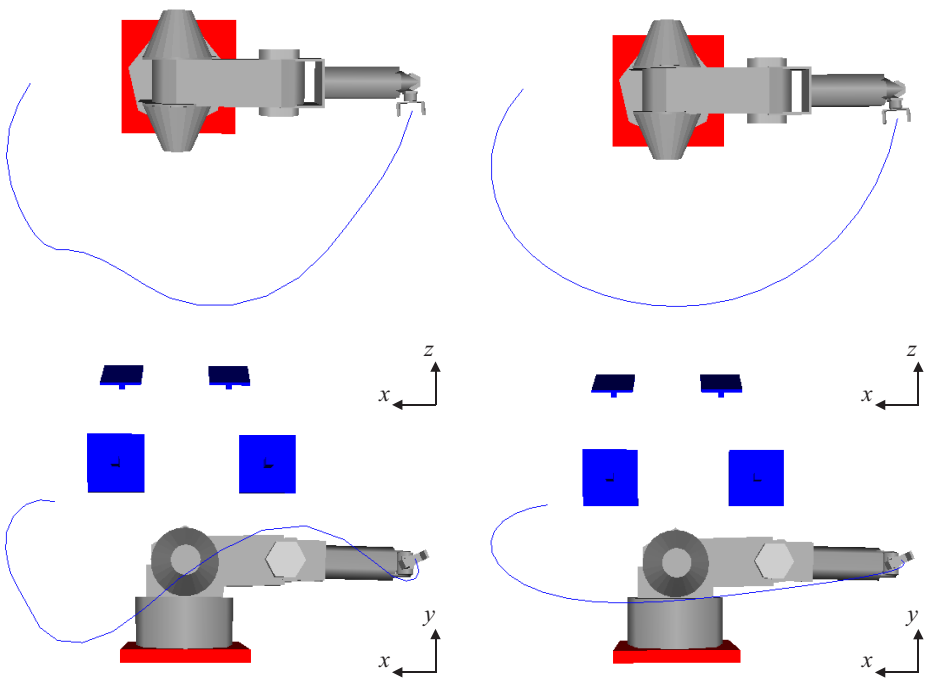
Let the roadmap be defined by a set of vertices $\mathcal{V}$, of which each vertex v is associated with a state $\mathbf{x}^v \in \mathcal{X}$, and a set of edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$, of which each edge $e = (u, v)$ is associated with a path $\Pi_e = (\mathbf{x}_0^e, \mathbf{u}_0^e, \ldots, \mathbf{x}_\ell^e, \mathbf{u}_\ell^e)$ such that $\mathbf{x}_0^e = \mathbf{x}^u$, $\mathbf{x}_\ell^e = \mathbf{x}^v$, and $\mathbf{x}_t^e = f(\mathbf{x}_{t-1}^e, \mathbf{u}_{t-1}^e, \mathbf{0})$ for all $t \in 1, \ldots, \ell$. That is, the path starts in vertex u, ends in vertex v, and complies with the dynamics model f.

Given a start vertex s and the variance matrix P_0 of the initial uncertainty about the state of the robot, we can use the LQG-MP method without adaptation to evaluate the quality of the paths associated with each of the outgoing edges $e = (s, v)$ of s. The LQG-MP method computes the variance matrices R_t^e of the uncertainty of the true state and estimated state along each of these paths (see Equation (23)). The variance $R_\ell^{(s,v)}$ at the last state of a path between the start vertex s and a neighboring vertex v is used as the initial variance $R_0^{(v,w)}$ of any path outgoing from v that is evaluated. To compute the Kalman gain matrices $K_t^{(v,w)}$ along this outgoing path, the initial uncertainty $P_0^{(v,w)}$ of the state in the Kalman filter needs to be given. Note that this uncertainty is not contained directly in the $R_0^{(v,w)}$ matrix; the R matrix contains the *a priori* variance of the true state and the *a priori* variance of the estimated state. The P-matrices in the Kalman filter signify the variance of the true state *conditioned* on the fact that the estimated state is known. Therefore, $P_0^{(v,w)}$ is computed as follows:

$$P_0^{(v,w)} = R_{\tilde{\mathbf{x}}} - R_{\tilde{\mathbf{x}}\hat{\mathbf{x}}} R_{\hat{\mathbf{x}}}^{-1} R_{\tilde{\mathbf{x}}\hat{\mathbf{x}}}, \quad R_0^{(v,w)} = \begin{bmatrix} R_{\tilde{\mathbf{x}}} & R_{\tilde{\mathbf{x}}\hat{\mathbf{x}}} \\ R_{\tilde{\mathbf{x}}\hat{\mathbf{x}}} & R_{\hat{\mathbf{x}}} \end{bmatrix}, \quad (52)$$

which is the standard equation for conditional variance of Gaussians. With these adaptations, the quality of an edge e can be evaluated using LQG-MP for any initial variance matrix R_0^e.

A variant of Dijkstra's algorithm to find a path within the roadmap to a goal vertex g can be implemented as shown in Algorithm 1. It takes as input the start vertex s, the initial state uncertainty P_0 and the roadmap defined by $\mathcal{V}$ and $\mathcal{E}$. The algorithm calls a function LQGMP(e, R_0^e) that computes all covariance matrices R_t^e along edge e given the initial variance R_0^e using LQG-MP (see Equation (23)).



(a) (b)

Fig. 7. The best path among the candidates for scenario C when the cameras (blue squares) are placed next to the robot after smoothing with (a) $\sigma' = 1$ and (b) $\sigma' = 10$. Each path is shown from two angles. Greater noise in the pseudo-observation model allows a looser fit to the nominal path.

The function $p_e = \text{COMPUTEPROBABILITY}(R_0^e, \ldots, R_{\ell-1}^e)$ computes the probability that the execution of edge e will be collision-free based on the variance matrices along the edge. Hence, the probability that the execution of an entire path is collision-free is the *product* of the probabilities along each of the constituent edges, which should be maximized. To fit this within the additive non-negative cost-minimizing framework of Dijkstra's algorithm, the additive cost of each edge is $-\log p_e$. Note that this cost is non-negative, and that a path with minimal accumulated cost has a maximal probability of collision-free execution:

$$\underset{\Pi=(e_0,\ldots,e_n)}{\text{argmax}} \{\prod_{i=0}^{n} p(e_i)\} = \underset{\Pi=(e_0,\ldots,e_n)}{\text{argmin}} \{\sum_{i=0}^{n} -\log p(e_i)\}. \quad (53)$$

The path can then be extracted by following backpointers from the goal back to the start vertex.

Algorithm 1 Dijkstra$(s, P_0, \mathcal{V}, \mathcal{E})$.

for all vertices $v \in \mathcal{V}$ **do**
  v.cost $\leftarrow \infty$
$s.R \leftarrow \left[\begin{smallmatrix} P_0 & 0 \\ 0 & 0 \end{smallmatrix}\right]$.
s.cost $\leftarrow 0$
$\mathcal{Q} \leftarrow \{s\}$
while not priority queue $\mathcal{Q}$ is empty **do**
  Pop vertex u from $\mathcal{Q}$ with minimal u.cost.
  for all edges (u, v) in $\mathcal{E}$ **do**
    $R_0^{(u,v)}, \ldots, R_\ell^{(u,v)} \leftarrow \text{LQGMP}((u, v), u.R)$
    $c \leftarrow -\log \text{COMPUTEPROBABILITY}(R_0^{(u,v)}, \ldots, R_{\ell-1}^{(u,v)})$
    if u.cost $+ c < v$.cost **then**
      v.cost $\leftarrow u$.cost $+ c$
      $v.R \leftarrow R_\ell^{(u,v)}$
      v.backpointer $\leftarrow u$
      if $v \notin \mathcal{Q}$ **then**
        $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{v\}$

7.2. Discussion

The above algorithm is an approximation to finding an optimal path in two ways. Firstly, the a-priori probability distributions are based on an LQR-controller that is split up into multiple controllers; one for each edge. This means that the distributions are based on a controller that does not take into account future expected cost beyond the end of each edge. Normally, the LQR-controller is defined over an entire path, which would result in (slightly) different *a priori* probability distributions.

Second, for Dijkstra's algorithm to return an optimal path, the cost of any subpath should be independent of what succeeds or precedes it. This is not the case in our

setup: the cost incurred when traversing an edge along a path depends on the initial covariance matrix when starting to traverse this edge. This initial covariance matrix in turn depends on the entire history leading up to that point. This breaks the basic dynamic programming assumption of 'optimal substructure' of paths.

In most practical cases, this suboptimality is likely to be negligible, and Dijkstra's algorithm will provide a good approximation. We note this assumption is also made (implicitly) in Prentice and Roy's work on belief roadmaps (Prentice and Roy 2009). There, the likelihood of arriving at the goal is maximized, and their variant of Dijkstra's algorithm evaluates paths based on the trace of the covariance matrix at intermediate nodes. Similar to the cost function in our algorithm, the trace of the covariance does not observe the optimal substructure property. A potential advantage of our approach compared with that of Prentice and Roy (2009) is that we do not assume maximum likelihood observations along the path, and can therefore infer the true *a priori* probability distributions along the path.

7.3. Experiment

We experimented with the roadmap approach of LQG-MP on a 'hovercraft'-type robot with second-order dynamics in a two-dimensional environment with obstacles. The robot needs to move from a start state $\mathbf{x}^{\text{start}}$ to a goal state $\mathbf{x}^{\text{goal}}$ without colliding with the obstacles in the environment (see Figure 8(a)).

7.3.1. Dynamics model The state $\mathbf{x} = (x, y, v_x, v_y)$ of the robot is a four-dimensional vector consisting of its position (x, y) and its velocity (v_x, v_y). Its control input $\mathbf{u} = (a_x, a_y)$ is a two-dimensional acceleration vector corrupted by process noise $\mathbf{m} = (\tilde{a}_x, \tilde{a}_y) \sim \mathcal{N}(\mathbf{0}, \sigma_a^2 I)$. This gives the following linear dynamics model:

$$f(\mathbf{x}, \mathbf{u}, \mathbf{m}) = \begin{bmatrix} x + \tau v_x + \tau^2 (a_x + \tilde{a}_x)/2 \\ y + \tau v_y + \tau^2 (a_y + \tilde{a}_y)/2 \\ v_x + \tau (a_x + \tilde{a}_x) \\ v_y + \tau (a_y + \tilde{a}_y) \end{bmatrix}, \quad (54)$$

where τ is the duration of a stage (time step).

7.3.2. Observation model The robot receives feedback on its position from 10 sensors in the environment. Each sensor has a known location $(\check{x}_i, \check{y}_i, 1)$. Hence, the measurement vector $\mathbf{z}$ is 20-dimensional and consists of 10 measurements of the robot's position. The noise $\mathbf{n} = (\tilde{x}_1, \tilde{y}_1, \ldots, \tilde{x}_{10}, \tilde{y}_{10})$ in the measurement increases quadratically with the distance from the sensor. This gives the following observation model:

$$h(\mathbf{x}, \mathbf{n}) = \begin{bmatrix} x + ((x - \check{x}_1)^2 + (y - \check{y}_1)^2 + 1)\tilde{x}_1 \\ y + ((x - \check{x}_1)^2 + (y - \check{y}_1)^2 + 1)\tilde{y}_1 \\ \vdots \\ x + ((x - \check{x}_{10})^2 + (y - \check{y}_{10})^2 + 1)\tilde{x}_{10} \\ y + ((x - \check{x}_{10})^2 + (y - \check{y}_{10})^2 + 1)\tilde{y}_{10} \end{bmatrix}. \quad (55)$$

7.3.3. Planning objective We aim to find the path for the robot with a minimal probability of colliding with obstacles. This probability is approximated in the same way as for scenario A (see Section 5.1.3).

7.3.4. Results We constructed a roadmap (by hand) for the environment consisting of 80 vertices and edges corresponding to unit-speed paths between neighboring vertices (see Figure 8(a)). We ran our algorithm for various placements of the sensors. The results are shown in Figures 8(b)–(d). Not surprisingly, the robot chooses a path close to the sensors to minimize the uncertainty about its position, which in turn enables it to avoid collisions with the obstacles with high probability. For the sensor placement of Figure 8(c), the robot chose to not precisely follow the sensors, as this allows for a shorter path with less narrow passages. In Figure 8(d), the sensors were randomly placed. On average, it took 0.48 seconds to compute the path within the roadmap.

8. 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 with 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 permit real-time application of LQG-MP. Even though using roadmaps partly addresses this issue, it is a major objective of future work to bring planning times down, for instance by devising

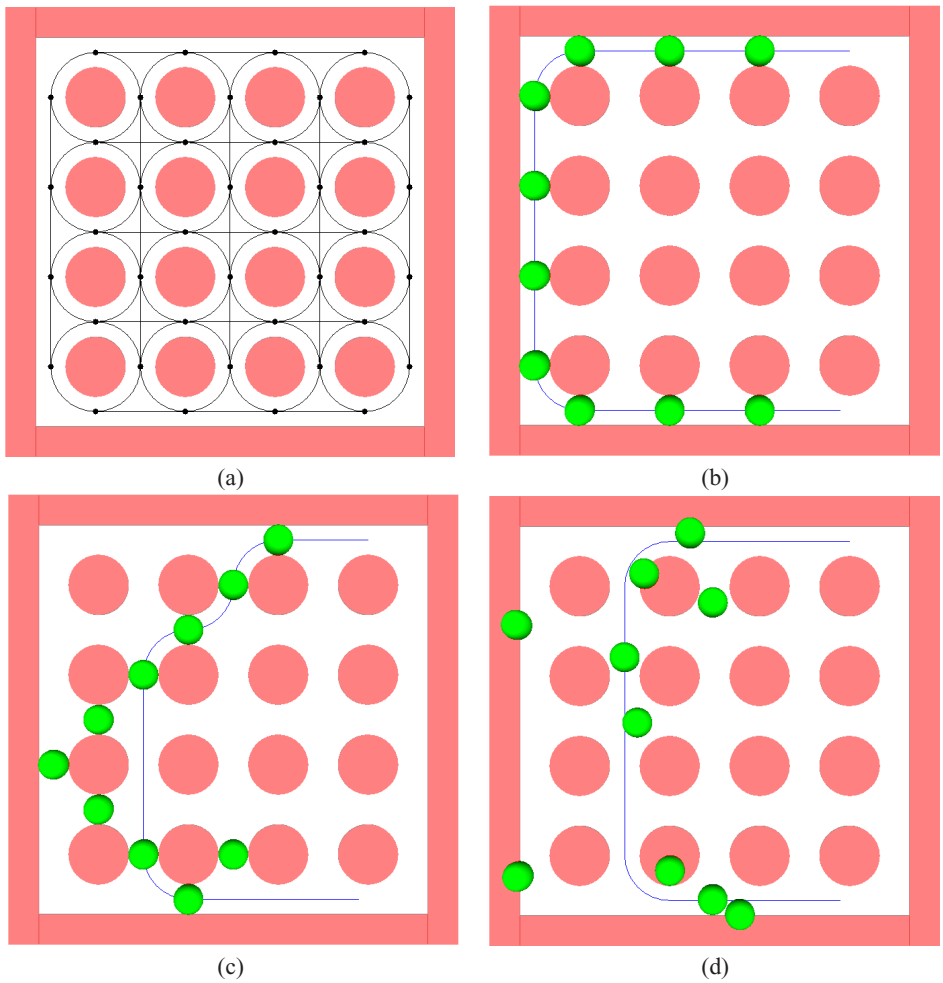


Fig. 8. (a) The environment with obstacles (light red), and the roadmap created for the environment. (b), (c), and (d) Paths between two vertices of the roadmap. Sensors (green spheres) measure the position of the robot with noise increasing quadratically with the distance to the sensor. The best path in the roadmap according to LQG-MP is shown.

a focused planner such that planning a large set of candidate paths is not required. Ideally, a smoother that directly optimizes the chosen planning objective is integrated into this planner. Other limitations, such as the fact that the candidate paths may not constitute a representative sample in high-dimensional state spaces might then also be resolved.

In the experiments we have performed, we have not included sensor models that are conditional in the sense that they only provide measurements in case there is no occlusion or the robot is within a field of view. Our approach naturally handles spatially varying sensor models, but the linearized observation model is based on the mean state of the robot on the path: it would assume that a measurement is obtained if the mean is within the field of view, and no measurement if the mean is outside the field of view, even if part of the distribution brings the robot outside (or inside) the field of view. This may be an appropriate approximation in many cases, but our approach does not handle such discontinuities in general since the assumption that the observation model is 'well linearizable' is locally violated. We also assumed that the motion and observation models have noise with Gaussian distributions. While our experiments suggest that our approach works as well for the more general case of distributions of which the mean and variance are sufficient statistics, exploring noise models such as multi-modal mixtures of distributions remains as future work.

We applied Kalman smoothing to make paths C_k-continuous while avoiding obstacles and in future work will explore and apply this technique to other contexts. We also applied LQG-MP to precomputed roadmaps using a variant of Dijkstra's algorithm to efficiently find good paths and will continue to refine this approach. We also apply LQG-MP to optimizing accuracy and safety in challenging robotic applications, such as autonomous

helicopter flight, needle steering for prostate brachytherapy, and robotic-assisted surgery.

Notes

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.
2. Not too much should be read into the higher quality for the setup with the camera above the robot; the end-effector is able to come closer to the cameras in this case.

Funding

This work was supported in part by the NSF (award number 0905344) and the NIH (award number 1R01EB-006435-01A1).

References

Alterovitz R, Siméon T and Goldberg K (2007) The stochastic motion road-map: a sampling framework for planning with Markov motion uncertainty. In: *Proceedings of Robotics: Science and Systems*, 2007.

Bertsekas D (2001) *Dynamic Programming and Optimal Control*. Nashua, NH: Athena Scientific.

Bouilly B, Simeon T and Alami R (1995) A numerical technique for planning motion strategies of a mobile robot in presence of uncertainty. *Proceedings of the IEEE International Conference on Robotics and Automation*, 1995.

Burns B and Brock O (2007) Sampling-based motion planning with sensing uncertainty. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, 2007.

Fraichard T and Mermond R (1998) Path planning with uncertainty for car-like robots. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, 1998.

Geraerts R and Overmars M (2007) Creating high-quality paths for motion planning. *The International Journal of Robotics Research* 26: 845–863.

Gonzalez J and Stentz A (2009) Using linear landmarks for path planning with uncertainty in outdoor environments. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.

Guibas L, Hsu D, Kurniawati H and Rehman E (2008) Bounded uncertainty roadmaps for path planning. In: *Proceedings of the Workshop on Algorithmic Foundations of Robotics*, 2008.

Huang Y and Gupta K (2009) Collision-probability constrained PRM for a manipulator with base pose uncertainty. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.

Huynh V and Roy N (2009) icLQG: combining local and global optimization for control in information space. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, 2009.

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

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

Kewlani G, Ishigami G and Iagnemma K (2009) Stochastic mobility-based path planning in uncertain environments. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.

Kuwata Y, Teo J, Karaman S, Fiore G, Frazzoli E and How J (2008) Motion planning in complex environments using closed-loop prediction. In: *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2008.

Kurniawati H, Hsu D and Lee W (2008) SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. *Proceedings of the Robotics: Science and Systems*, 2008.

LaValle S (2006) *Planning Algorithms*. Cambridge: Cambridge University Press.

LaValle S and Hutchinson S (1998) An objective-based framework for motion planning under sensing and control uncertainties. *The International Journal of Robotics Research* 17: 19–42.

LaValle S and Kuffner J (2001) Randomized kinodynamic planning. *The International Journal of Robotics Research* 20: 378–400.

Lazanas A and Latombe J (1995) Motion planning with uncertainty: a landmark approach. *Artificial Intelligence* 76: 285–317.

Melchior N and Simmons R (2007) Particle RRT for path planning with uncertainty. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, 2007.

Missiuro P and Roy N (2006) Adapting probabilistic roadmaps to handle uncertain maps. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, 2006.

Nakhaei A and Lamiraux F (2008) A framework for planning motions in stochastic maps. In: *Proceedings of the International Conference on Control, Automation, Robotics and Vision*, 2008.

Papadimitriou C and Tsisiklis J (1987) The complexity of Markov decision processes. *Mathematics of Operations Research* 12: 441–450.

Pepy R and Lambert A (2006) Safe path planning in an uncertain-configuration space using RRT. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.

Platt R, Tedrake R, Kaelbling L and Lozano-Perez T (2010) Belief space planning assuming maximum likelihood observations. In: *Proceedings of the Robotics: Science and Systems*, 2010.

Porta J, Vlassis N, Spaan M and Poupart P (2006) Point-based value iteration for continuous POMDPs. *Journal of Machine Learning Research* 7: 2329–2367.

Prentice S and Roy N (2009) The belief roadmap: efficient planning in belief space by factoring the covariance. *The International Journal of Robotics Research* 28: 1448–1465.

Rauch H, Tung F and Striebel C (1965) Maximum likelihood estimates of linear dynamic systems. *AIAA Journal* 3: 1445–1450.

Roy N, Burgard W, Fox D and Thrun S (1999) Coastal navigation—mobile robot navigation with uncertainty in dynamic environments. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, 1999.

Tedrake R (2009) LQR-trees: Feedback motion planning on sparse randomized trees. In: *Proceedings of the Robotics: Science and Systems*, 2009.

Thrun S, Burgard W and Fox D (2005) *Probabilistic Robotics*. Cambridge, MA: The MIT Press.

van den Berg J and Overmars M (2005) Prioritized motion planning for multiple robots. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005.

van den Bergen G (2004) *Collision Detection in Interactive 3D Environments*. San Mateo, CA: Morgan Kaufmann.

Welch G and Bishop G (2006) *An Introduction to the Kalman Filter*. Techical Report TR 95-041, University of North Carolina at Chapel Hill, 2006.