A State Lattice Approach for Motion Planning under Control and Sensor Uncertainty[⋆]

Adrián González-Sieira[⋆⋆], Manuel Mucientes[⋆⋆⋆], and Alberto Bugarín

Centro de Investigación en Tecnoloxías da Información (CITIUS),
University of Santiago de Compostela, Spain
{adrian.gonzalez,manuel.mucientes,alberto.bugarin.diz}@usc.es

Abstract. Reliable motion planners have to take into account not only the kinematic constraints of the robot but, also, the uncertainty of both the motion and sensor models. In this way, it is possible to evaluate a motion plan based not just on the maximum likelihood path, but also in deviations from that path that have a non-negligible probability. As a result, motion plans are more robust and require a lower number corrections during the online implementation of the plan. In this paper we address the problem of motion planning under uncertainty in both motion and sensor models using a state lattice. The approach manages a very efficient representation of the state space, calculates on-demand the a-priori probability distributions of the most promising states with an Extended Kalman Filter, and executes an informed forward exploration of the state space with Anytime Dynamic A*. We provide results with a differential drive robot under different scenarios, showing the ability of the planner to calculate optimal solutions that minimize the probability of collision and the time to reach the goal state.

1 Introduction

Motion planning algorithms have experimented an impressive evolution in the last years. While the first approaches tried to solve planning and control problems separately, current proposals pose more realistic planners that take into account the kinematic restrictions of the vehicle dynamics and, therefore, can guarantee the generation of feasible solutions. The most successful approaches in this field are planners based on stochastic sampling sampling methods —probabilistic roadmaps (PRM), rapidly-exploring randomized trees (RRT) and their variants— and those based on deterministic sampling —state lattices.

[⋆] This work was supported by the Spanish Ministry of Economy and Competitiveness under grants TIN2011-22935 and TIN2011-29827-C02-02. This work was also supported in part by the European Regional Development Fund (ERDF/FEDER) under the project CN2012/151 of the Galician Ministry of Education.

[⋆⋆] A. González-Sieira is supported by a FPU grant (ref. AP2012-5712) from the Spanish Ministry of Education, Culture and Sports.

[⋆⋆⋆] M. Mucientes is supported by the Ramón y Cajal program of the Spanish Ministry of Economy and Competitiveness.

DOI: 10.1007/978-3-319-03653-3_19, © Springer International Publishing Switzerland 2014

These planners assume a full knowledge of the state and motions, leaving uncertainty issues to the controller that implements the generated paths, which is typically a feedback controller.

Motion uncertainty in autonomous robots originates from the inaccuracy of control actions, unmodeled external influences, and the usage of partial or noisy information about the state. The evolution of the uncertainty depends on the executed motions and the current state of the vehicle, i.e., different generated paths can have very different uncertainties. Planning without taking into account the uncertainty produces unrealistic paths, as the full knowledge of the states and actions assumed by traditional planners can only be reproduced in simulation conditions. Uncertainty causes failures or deviations from the planned path and, frequently, it is necessary to obtain a new plan when the feedback controller cannot return the system to a state belonging to the planned path.

Under the assumption that both the motion and the sensor models are known, it is possible to predict the a-priori —at planning time— probability distributions of the states given the control commands to follow a path. This prediction can be done using a Kalman filter when the stochasticity of the models can be described with Gaussian distributions. With the information provided by these distributions, a planner can be designed to obtain solutions that optimize one or several objectives: minimal probability of collision, minimal covariance along the path, maximum likelihood of reaching the goal state, etc.

In this paper, we present a motion planning algorithm based on a deterministic sampling technique, the state lattice [1], that can obtain optimal and sub-optimal bounded paths taking into account the motion and sensing uncertainty. This approach extracts from the vehicle motion model a set of discrete actions connecting states belonging to the lattice. As the state lattice uses a regular sampling scheme, these actions are position-independent and can be used to connect every pair of states equally arranged. This provides a very efficient representation of the state space which cannot be achieved with stochastic sampling methods.

We use an Extended Kalman Filter (EKF) to estimate the a-priori distributions, as we assume that both the motion and sensor models follow Gaussian distributions. The estimated probability density functions (PDFs) are used to approximate the probability of collision along a path. Then, the algorithm obtains the safest and optimal path to the goal using a discrete search algorithm, Anytime Dynamic A* (AD*) [2], over the state lattice. Our proposal executes an informed forward exploration of the state space, calculating on-demand the a-priori distributions of the most promising states according to a heuristic function that takes into account the vehicle dynamics and the map information, improving the efficiency of the planner.

2 Related Work

The most successful approaches to motion planning are sample based techniques. Among all the approaches, three of them have been widely used: i) PRM [3], that obtains random samples in free space, building a graph that connects the nearest ones; ii) RRT [4], that builds a tree in free space by iteratively obtaining random samples and connecting them with the nearest existing one; iii) state lattices [1], that exploit the benefits of a regular and deterministic sampling scheme to represent the problem as a directed graph. The first two approaches have many interesting properties, but they assume that the trajectories connecting every pair of states are generated online from the system dynamics —which depending on its complexity can become a difficult problem— while state lattices can work with a set of actions generated offline, which favours an efficient search.

Uncertainty in motion planning comes from different sources: control noise, sensor noise, partial information from the environment, and map uncertainty. Some planners only consider motion uncertainty, as [5], that tries to avoid rough terrain because the result of a control may differ from the expected. The proposal in [6] uses Markov Decision Process (MDP) theory to maximize the probabilities of collision avoidance and of successfully reaching the goal. A generalization of MDP, Partially Observable Markov Decision Process (POMDP), can be used to include sensor uncertainty, but this approach is not scalable to realistic problems due to its high computational complexity [7].

For systems that can be modelled with Gaussian PDFs, the state can be estimated using a Kalman filter, which has proved to be a successful approach for autonomous vehicles equipped with on-board sensors. This is the approach for Belief Roadmap (BRM) [8], which generates paths minimizing the state uncertainty at the goal, but regardless their total cost. Another approach is LQG-MP [9], which obtains the trajectory with the lowest cost and with a probability of collision under a threshold. However this proposal uses an RRT, which makes not possible to guarantee that the optimal path will be found [10]. Other proposals use variations of RRT modified with pruning strategies, but they also inherit the lack of guarantee of finding the optimal solution. The planner presented in [11] uses an extension of RRT which provably converges to the optimal solution when the number of samples tend to infinite, but as all random sampling techniques assumes that the action connecting every pair of states can be efficiently computed online.

3 Problem Formulation

The motion planner obtains optimal collision-free paths taking into account both control and sensor uncertainty. The robot dynamics (f) and sensor models (h) are explicitly given and they are linear or can be locally approximated by their linearization:

$$\begin{aligned} x_t &= f\left(x_{t-1}, u_t, m_t\right), & m_t &\sim (0, M_t) \\ z_t &= h\left(x_t, n_t\right), & n_t &\sim (0, N_t) \end{aligned} \tag{1}$$

where $x_t \in \mathcal{X} = \mathcal{X}^{free} \cup \mathcal{X}^{obs}$ is the state of the robot, $u_t \in \mathcal{U}$ is the control input, m_t is the random process noise, z_t is the sensor measurement and n_t its associated random noise. $\mathcal{X}^{obs}$ is formed by all states where the robot collides with obstacles in the environment.

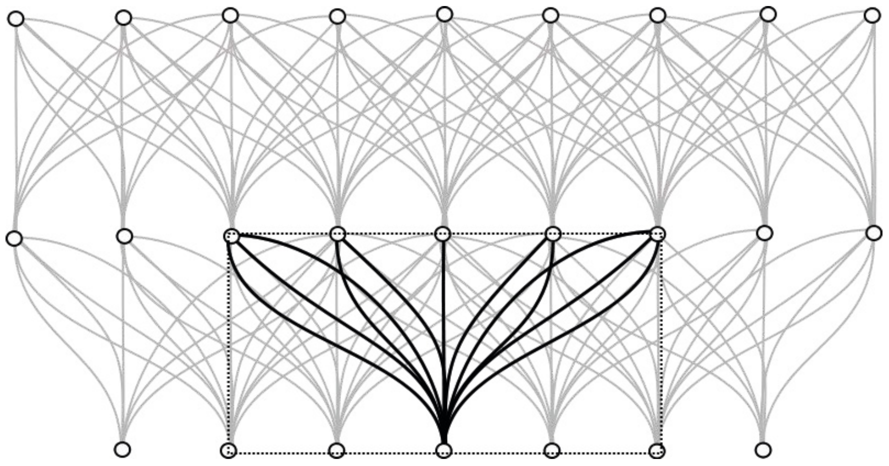


Fig. 1. Rectangular arrangement of the lattice. The canonical control set (black) is independent of the beginning position and is replicated in all equivalent states allowing a very efficient representation of the problem.

Planning relies on a state lattice, which samples the state space $\mathcal{X}$ in a regular form, obtaining a set of states $x_t \in \mathcal{X}^{lat} \subset \mathcal{X}$, named lattice states. These states are connected by a finite set of actions extracted from the vehicle dynamics, the canonical control set $\mathcal{U}$. The generation of $\mathcal{U}$ follows an iterative optimization method based on Newton-Raphson detailed in [12].

The state lattice is generated using a regular discretization scheme. Because of this regularity the canonical actions are independent of the beginning position and the same control commands connect every pair of states in $\mathcal{X}^{lat}$ equally arranged, as shown in Fig. 1.

Each canonical control $u^{a:b} \in \mathcal{U}$ is a composition of control commands that drive the robot from $x^a \in \mathcal{X}^{lat}$ to $x^b \in \mathcal{X}^{lat}$:

$$\begin{aligned} u^{a:b} &= \left(u_1^{a:b}, u_2^{a:b}, ..., u_{t^{a:b}}^{a:b}\right) \\ x^{a:b} &= \left(x_1^{a:b}, x_2^{a:b}, ..., x_{t^{a:b}}^{a:b}\right) \\ x_1^{a:b} &= x^a, x_{t^{a:b}}^{a:b} = x^b \\ x_t^{a:b} &= f(x_{t-1}^{a:b}, u_t^{a:b}, 0) \in \mathcal{X}, \forall t \in [1, t^{a:b}] \end{aligned} \tag{2}$$

where the intermediate states —which may not belong to $\mathcal{X}^{lat}$— are generated with the motion model f with no noise, and $t^{a:b}$ is the total time of the canonical action. Given an initial state of the vehicle ($x^{start} \in \mathcal{X}^{lat}$), and a goal state ($x^g \in \mathcal{X}^{lat}$), a valid path is a combination of N canonical actions, $(u^{start:1}, ..., u^{N-1:g})$, that drive the robot to the goal state without collisions.

4 Uncertainty Prediction

If both the prior probability and the motion and sensor noises are Gaussian, an EKF can be used to estimate the a-priori PDFs for each state $x_t, \forall t \in [1, t^{a:b}]$

given the control inputs along the followed path. The PDFs estimated with the EKF are also Gaussian, so a state is described as:

$$x_t \sim N(\bar{x}_t, \Sigma_t) \tag{3}$$

The a-priori PDFs are calculated when the search algorithm expands a new lattice state and its neighbourhood is generated. The complete process of uncertainty propagation is detailed in algorithm 1.

Algorithm 1. Uncertainty prediction: $uncertainty(x^a)$

```
for all $x^b \in succ(x^a)$ do
  $\bar{x}^0 = \bar{x}^a$
  $\Sigma^0 = \Sigma^a$
  for all $t \in [1, t^{a:b}]$ do
    $\widetilde{x}_t = f(\bar{x}^{a:b}_{t-1}, u^{a:b}_t, 0)$
    $\widetilde{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + V_t M_t V_t^T$
    $K = \widetilde{\Sigma}_t H_t^T (H_t \widetilde{\Sigma}_t H_t^T + W_t N_t W_t^T)^{-1}$
    $\bar{x}_t = \widetilde{x}_t + K(z_t - h(\widetilde{x}_t))$
    $\Sigma_t = (I - K H_t)\widetilde{\Sigma}_t$
  end for
  $\bar{x}^b = \bar{x}_t$
  $\Sigma^b = \Sigma_t$
end for
return $x^{a:b}_t, \forall t \in [1, t^{a:b}], \forall x^b \in succ(x^a)$
```

Given a state $x^a \in \mathcal{X}^{lat}$, and for each successor $x^b \in \mathcal{X}^{lat}$, the uncertainty is estimated applying an EKF iteratively to obtain the intermediate a-priori PDFs along the motion primitive. Each execution of the EKF performs two steps: a prediction step that uses the dynamics model and the current command $u^{a:b}_t$ of the canonical action, and an update step that incorporates the information given by the maximum likelihood measurement z_t. As the prediction of the PDFs is done at planning time without any information of the future execution of the path, the EKF is applied considering that both controls and measurements are those with maximum likelihood.

During the execution of the planned path the vehicle could deviate from the predicted trajectory. The optimal approach to minimize this deviation is to use a feedback controller based in a Linear Quadratic Regulator (LQR) in conjunction with a Kalman estimation of the state, a technique named Linear Quadratic Gaussian (LQG) controller.

5 Search Algorithm

Motion planning is a search problem that can be expressed as a directed graph where the nodes are the discrete states of the lattice ($\mathcal{X}^{lat}$), and the arcs connecting them are the actions of the canonical control set $\mathcal{U}$. As the canonical control set is generated from the vehicle dynamics, it is clear that the state lattice observes by construction the differential constraints of the robot.

5.1 Anytime Dynamic A*

The search algorithm we have used in this proposal is Anytime Dynamic A* (AD*) [2]. This algorithm is very adequate for motion planning problems because it combines both replanning and anytime search. It is possible to calculate suboptimal bounded solutions adjusting an heuristic inflation parameter, ϵ, depending on the time available to obtain a solution.

The main operations done by the search algorithm are summarized in algorithm 2. While the solution is not found, the algorithm selects the most promising lattice state x^a. This state is the one that minimizes the cost of the the path from the beginning state $c(x^{start:a})$ plus the estimated cost to the goal $e(x^a)$, which is calculated by the heuristic function inflated by the parameter ϵ. Then, the expansion of the state x^a is performed, generating all the paths to its successor states $x^b \in succ(x^a)$ (each point in the path is represented by a Gaussian distribution) with algorithm 1.

Algorithm 2. Main loop of AD* with uncertainty

```
while solution not found do
    select $\min_{x^a \in \mathcal{X}^{lat}}(c(x^{start:a}) + \epsilon \cdot e(x^a))$
    $uncertainty(x^a)$
    for all $x^b \in succ(x^a)$ do
        $c(x^{start:b}) = c(x^{start:a}) + c(x^{a:b})$
    end for
end while
```

In its typical form, AD* executes a backwards search. This makes possible to change the beginning state and obtain new solutions without replanning from scratch. Nevertheless, if we have to estimate the PDF of each state in the path at planning time, the PDFs are propagated along the outgoing trajectories of the most promising state x^a in each iteration of the algorithm, using the procedure introduced in algorithm 1. This function needs the prior PDF of x^a. This means that the distribution of the initial state $x^{start} \in \mathcal{X} \sim N(\bar{x}^{start}, \Sigma^{start})$ needs to be known in order to propagate it along the paths generated by the search algorithm. For this reason, a variant of AD* that executes a forward exploration of the state space was used.

After the PDFs are propagated, the cost of the path from the starting state to each successor state $x^b \in succ(x^a)$ must be updated. This operation just requires the calculation of the cost from x^a to x^b (algorithm 3).

In a planner without uncertainty the cost is the time to drive the robot throughout the path without collisions. However, if the planner takes into account the uncertainty, the paths have to be, at the same time, safe and optimal,

Algorithm 3. Cost evaluation: $c(x^{a:b})$

```
$c_s = 0$
for all $t \in [1, t^{a:b}]$ do
    $S =$ set of samples from $x_t^{a:b} \sim N(\bar{x}_t^{a:b}, \Sigma_t^{a:b})$
    $p_c = \frac{\sum_S checkCollision(S_i)}{|S|}$
    $c_s = c_s - \ln(1 - p_c)$
end for
return $\begin{bmatrix} c_s \\ t^{a:b} \\ \Sigma^b \end{bmatrix}$
```

i.e., the algorithm has to minimize the probability of collision and, at the same time, get an optimal cost —minimizing the time— to reach the goal state. For this reason, the cost function of a path between states $x^a \in \mathcal{X}^{lat}$ and $x^b \in \mathcal{X}^{lat}$ is defined as a vector of three elements that should be minimized: a safety cost, c_s, related to the collision probability along the path, the total time of the path, $t^{a:b}$ and the covariance at the final state, Σ^b.

The probability of collision of a path between the states x^a and x^b is obtained as a combination of the probabilities of collision for each intermediate state, $x_t^{a:b}$ (algorithm 3). Given a state $x_t^{a:b}$, its PDF is sampled, and for each sample in the set S, a collision check is performed. The collision check is done by applying the hyperplane separation theorem between the obstacles in the map and the vehicle shape at the sampled pose. This theorem requires the convexity of the checked forms to success. This is guaranteed by approximating the vehicle shape with a polygon and decomposing the obstacle information in squared cells stored in a grid map.

The motion planner propagates the PDFs and evaluates the cost of an action between a state and its successors on-demand when the search algorithm expands a new state x^a. As the expanded states are the most promising ones, this saves a lot of computation time without the need to applying external pruning strategies as in those approaches based on RRT, being simply integrated in the search.

Our proposal has to compare the cost of the different lattice states in order to select the most promising one, and this cost is a vector of three components. The comparison of the cost of two states, x^a and x^b is solved as:

$$\begin{aligned} c(x^{start:a}) < c(x^{start:b}) \Leftrightarrow (c_s^{start:a} = c_s^{start:b}) \vee \\ (c_s^{start:a} = c_s^{start:b} \wedge t^{start:a} < t^{start:b}) \vee \\ (c_s^{start:a} = c_s^{start:b} \wedge t^{start:a} = t^{start:b} \wedge \Sigma^a = \Sigma^b) \end{aligned} \tag{4}$$

thus, there is a hierarchy in the components of the cost vector: the most relevant element is the safety cost, then the total time of the path and, finally, the covariance of the final state.

The minimization of the cost function calculated in algorithm 3 and this comparison criterion generate safe paths with the lowest possible execution time and, between paths with equal safety cost and execution time, the one with lowest uncertainty at the final state.

5.2 Heuristic

AD* executes an informed search over the state space. The estimation of the cost of the path between a lattice state and the goal state is generated by the heuristic function e, that influences the order in which the states are expanded. In [13] introduced an heuristic function for state lattices that was defined as the combination of two values: the cost of the path taking into account the kinematic constraints of the vehicle and considering free space (e_{FSH}), and the cost of the path regardless the motion model but using the obstacles information, e_{H2D}:

$$e(x_t) = \max(e_{H2D}(\bar{x}_t), e_{FSH}(\bar{x}_t)), \tag{5}$$

where the heuristic function uses the mean of the PDF ($\bar{x}$), which is a reasonable assumption attending to the optimistic nature of the heuristic.

e_{H2D} is calculated executing a Dijkstra search over an 8-connected 2D grid where the positions match the states in $\mathcal{X}^{lat}$. The exploration begins in the goal state and it is stopped when the explored cost reaches 1.5 times the cost between the initial state and the goal state. This process is executed only once, at the beginning of the planning process. For states not explored by the Dijkstra search the heuristic takes its maximum value, providing a stop condition for the planner when a solution up to the maximum Dijkstra explored cost cannot be found.

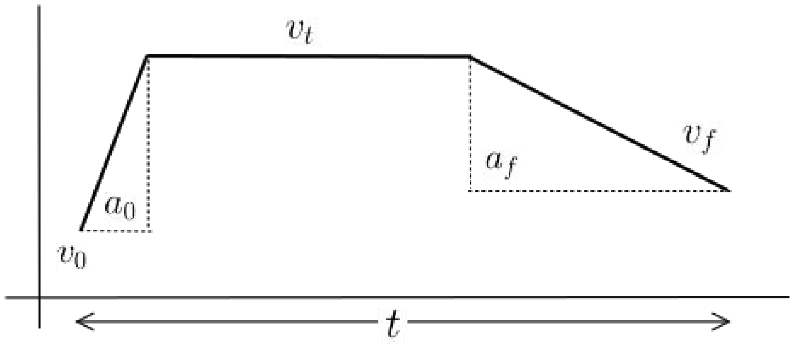
On the other hand, e_{FSH} uses the motion model to estimate the cost to the goal, which makes it a motion planning problem itself and, therefore it cannot be computed online. As this heuristic considers always free space, its values can be precomputed offline and stored in a Heuristic Look-Up Table, following the construction process detailed in [13].

6 Experimental Results

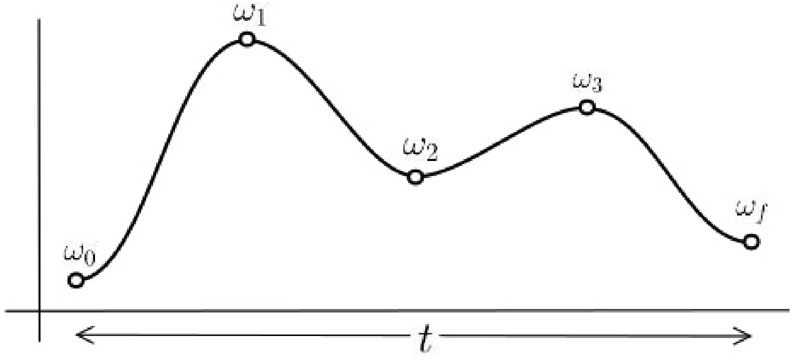
The proposed motion planner was tested with a differential drive robot in a 2D environment using Player/Stage. We have selected different scenarios (landmark positions) and, also, different uncertainty degrees for the sensor model.

The state $x_t = (x_x\, x_y\, x_\theta\, x_v\, x_\omega)^T$ is a 5-dimensional vector containing the vehicle pose and the current linear and angular speed, and the control command $u_t = (u_v\, u_\omega)^T$ is a 2-dimensional vector that defines the linear and angular speeds.

The lattice was built with the following discretization resolutions: 0.5 m in both x_x and x_y; x_θ contains the orientation values of the neighbours of a 16-connected grid, $x_v = \{0, 0.2, 0.5\}$ m/s, and $x_\omega = 0$ rad/s. This means that the lattice states can only take these values as part of the discretization scheme, but for any other state $x_t \notin \mathcal{X}^{lat}$, the state vector can take any value. In fact, the evolution of the linear and angular speeds in the canonical control set is defined by trapezoidal function for the linear velocity and cubic spline function for the



(a) Trapezoidal linear velocity profile.



(b) Cubic-spline angular velocity profile.

Fig. 2. Definition of the variable profiles of u_v and u_ω for a canonical control action

angular velocity. As it can be seen in Fig. 2(a) and Fig. 2(b), x_v and x_ω can take any intermediate value, while the beginning and final ones must observe the state lattice restrictions.

The canonical control set $\mathcal{U}$ used in the experiments was built connecting neighbours of distances (in number of states) 0, 1, 4 and 8, allowing motion primitives up to 4 m long.

The non-linear motion model $f(x_{t-1}, u_t)$ used in the tests is defined as:

$$\begin{bmatrix} x_x + \frac{u_v}{u_\omega}(-\sin(x_\theta) + \sin(x_\theta + u_\omega \Delta t)) \\ x_y + \frac{u_v}{u_\omega}\cos(x_\theta) - \frac{u_v}{u_\omega}\cos(x_\theta + u_\omega \Delta t) \\ x_\theta + u_\omega \Delta t \\ u_v \\ u_\omega \end{bmatrix} \tag{6}$$

where Δt is the time of application of the action, and the control noise covariance is calculated as a percentage of the control command u_t:

$$M_t = 0.1 I u_t \tag{7}$$

The sensor measurements come from the nearest landmark in the environment, $b = (b_x, b_y)$, according to the following observation model $h(x_t, b)$:

$$\begin{pmatrix} h_d \\ h_\theta \end{pmatrix} = \begin{pmatrix} \sqrt{(x_x - b_x)^2 + (x_y - b_y)^2} \\ \arctan(x_y - b_y, x_x - b_x) - x_\theta \end{pmatrix} \tag{8}$$

The measurement noise covariance is also defined as a percentage of the value of the measurement:

$$N = \begin{cases} 0.05 I h & \text{if } h_d <= 5 \\ 0.3 I h & \text{if } h_d > 5 \end{cases}, \tag{9}$$

which introduces a variable measurement noise depending on the distance between the landmark and the vehicle.

Collision check is a very frequent operation and could be very time consuming without a careful implementation (specially for a grid map with cell resolution of 0.125 m). To optimize this operation, offline we have centered the vehicle shape in a set of poses around the central point of a cell, calculating for each pose the list of adjacent cells occupied by the shape of the robot. When a collision check has to be performed online, the difference between the central position of the shape and the central point of the nearest cell is calculated. Given that difference, the closest sample is picked and the collision is checked with the list of adjacent cells that was generated offline. To guarantee that this loss of precision in the collision check does not affect the safety of the generated paths, the vehicle shape is enlarged with a safety margin that matches the sampling resolution of the offline cells list.

For the example in Fig. 3, a motion planner that obtains the optimal path without taking into account the motion and sensor uncertainty, obtains solutions with a high collision probability. This may cause failures when the vehicle deviates from the planned path, even in the case of executing it with a feedback controller.

The configuration of the landmarks in the environment directly affects the evolution of the covariance and, therefore, the solution obtained by the planner. In the example of Fig. 3(a) the localization is good near the upper doors, so the solution is similar to the optimal path without uncertainty but avoiding the proximity to the corners. When the landmarks are moved down, as in Fig. 3(b) and Fig. 3(c), the safest solution avoids the first narrow door, navigates through zones with good measurements and safely passes through the second one. More detailed results for these executions are given in Table 1, which contains the average values for 5 executions of the planner in the same conditions. Stochasticity comes from the sampling of the PDF to estimate the collision probability, p_c, which is obtained by combination of the collision probabilities of all states along the path. Even in the case of similar solutions as shown in Fig. 3(a), the proximity of a single state to obstacles significantly increases p_c.

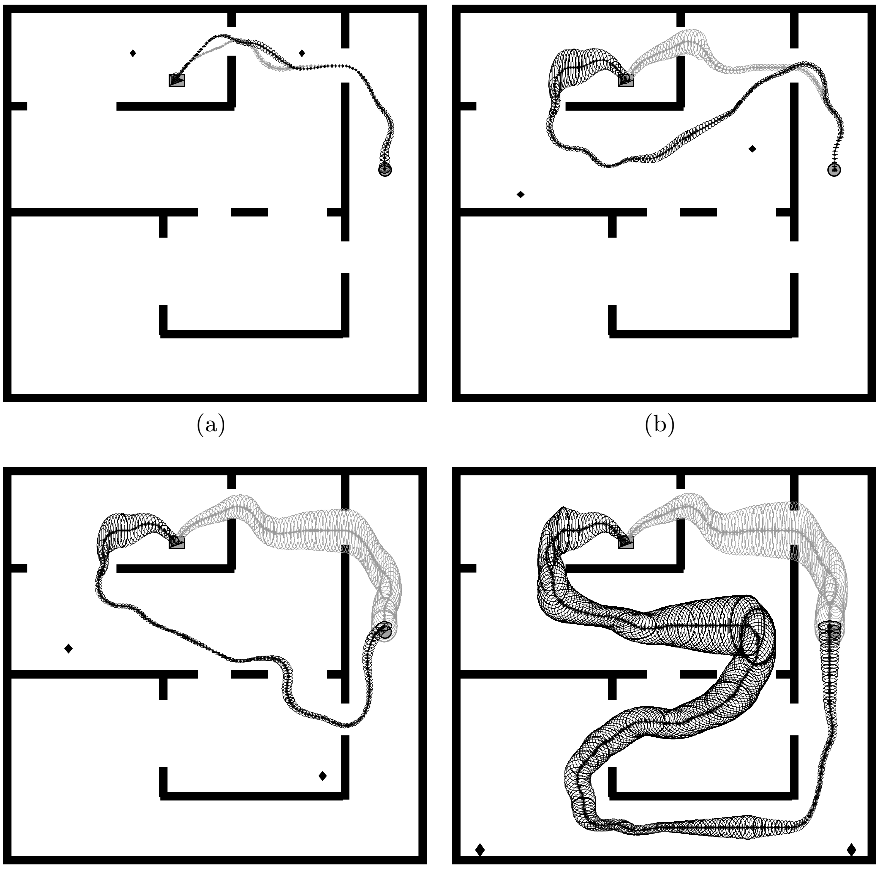


Fig. 3. Comparative between the best paths with (in black) and without (in gray) taking into account the uncertainty. Each figure shows a different positioning of the landmarks (diamond symbols). The ellipses represent the double of the deviation (of x_x and x_y) for the states on the path.

"O-" prefix denotes the optimal solution using a planner that does not take uncertainty into account, while "U-" indicates the solution obtained when planning with uncertainty.

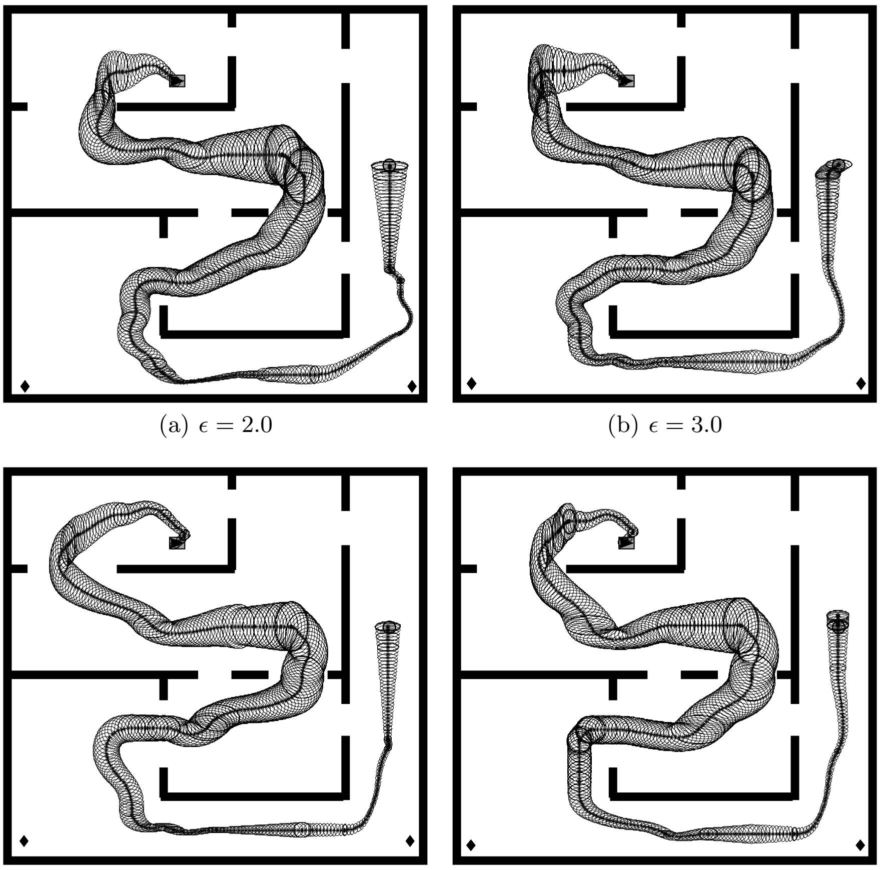
For the worst case, in Fig. 3(d), the measurements are very noisy in the upper region of the map, so the covariance grows fast and passing trough narrow areas has a high probability of collision. If the landmarks are set in the lower region of the environment, the safest solution is to take the longest path, as it obtains measurements with a lower uncertainty to safely reach the goal.

The number of lattice states expanded by the search algorithm and the planning time depend on the degree of uncertainty with respect to the optimal path without uncertainty. The reason is that the heuristic function does not contain information about the probability of collision or the covariance evolution, so if the optimal path is blocked due to a high probability of collision, the heuristic may underestimate the cost to the goal region, causing an increase in the number of expansions.

Figure 4 shows a comparative between different sub-optimal bounded solutions. The landmark positions are the same as in Fig. 3(d), but in this case the algorithm was run for different values of the heuristic inflation parameter, ϵ. As shown in Table 1, the higher the value of ϵ, the lower the execution time and the number of expansions of the algorithm. These values have an upper bound given by the loss of information caused by estimating the cost to the goal without uncertainty, i.e., for high values of ϵ the heuristic function does not provide useful information and, therefore, the positive effect of a lower number of state expansions disappears.

Table 1. Details of the experiments

Problem	**Planning**			**Solution**	
	ϵ	**Time (s)**	**Expansions**	p_c	**Cost (s)**
O-3(a)	1.0	1	146	0.51	31.02
U-3(a)	1.0	1	121	0	31.16
O-3(b)	1.0	1	146	0.97	31.02
U-3(b)	1.0	48	4,344	0	57.40
O-3(c)	1.0	1	146	1	31.02
U-3(c)	1.0	114	10,699	0	65.17
O-3(d)	1.0	1	146	1	31.02
	1.0	299	25,771	0	123.48
	2.0	204	16,183	0	132.82
U-3(d)	3.0	198	15,779	0	128.71
	5.0	198	15,856	0	139.29
	10.0	196	15,634	0	152.99



(a) $\epsilon = 2.0$

(b) $\epsilon = 3.0$

(c) $\epsilon = 5.0$

(d) $\epsilon = 10.0$

Fig. 4. Comparative of different sub-optimal bounded paths for several values of ϵ

7 Conclusions

We have presented a motion planning algorithm that takes into account the uncertainty of the motion and sensor models. The proposal is based on a search algorithm that obtains safe and optimal paths over a state lattice. The planner uses an EKF to predict the PDFs of the different states throughout the possible paths, and assigns them a probability of collision. Moreover, the planner can also get anytime solutions calculating safe sub-optimal bounded paths.

The performance of the motion planner was tested with several examples with different uncertainty conditions. All the results show a good performance both in terms of the probability of reaching the goal state without collisions and the time to travel throughout the path. Further work is still necessary to improve the PDFs prediction and, also to reduce the planning time. As many components of the planer are computed offline (like the canonical control set), the same approach could be applied to the PDFs if they can be expressed as a function of the initial covariance, thus improving the efficiency.

References

1. Pivtoraiko, M., Knepper, R.A., Kelly, A.: Differentially constrained mobile robot motion planning in state lattices. Journal of Field Robotics 26(3), 308–333 (2009)
2. Likhachev, M., Ferguson, D., Gordon, G., Stentz, A., Thrun, S.: Anytime dynamic A*: An anytime, replanning algorithm. In: Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), pp. 262–271 (2005)
3. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006)
4. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. The International Journal of Robotics Research 20(5), 378–400 (2001)
5. Melchior, N.A., Simmons, R.: Particle RRT for path planning with uncertainty. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 1617–1624 (2007)
6. Alterovitz, R., Siméon, T., Goldberg, K.: The stochastic motion roadmap: A sampling framework for planning with Markov motion uncertainty. In: Robotics: Science and Systems, pp. 246–253 (2007)
7. Papadimitriou, C.H., Tsitsiklis, J.N.: The complexity of Markov decision processes. Mathematics of Operations Research 12(3), 441–450 (1987)
8. Prentice, S., Roy, N.: The belief roadmap: Efficient planning in belief space by factoring the covariance. The International Journal of Robotics Research 28(11-12), 1448–1465 (2009)
9. Van Den Berg, J., Abbeel, P., Goldberg, K.: LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. The International Journal of Robotics Research 30(7), 895–913 (2011)
10. Karaman, S., Frazzoli, E.: Incremental sampling-based algorithms for optimal motion planning. In: Robotics: Science and Systems (2010)
11. Bry, A., Roy, N.: Rapidly-exploring random belief trees for motion planning under uncertainty. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 723–730 (2011)
12. González-Sieira, A., Mucientes, M., Bugarín, A.: Anytime Motion Replanning in State lattices for Wheeled Robots. In: Workshop on Physical Agents (WAF), pp. 217–224 (2012)
13. Likhachev, M., Ferguson, D.: Planning Long Dynamically Feasible Maneuvers for Autonomous Vehicles. The International Journal of Robotics Research 28(8), 933–945 (2009)