# An Adaptive Multi-resolution State Lattice Approach for Motion Planning with Uncertainty

A. González-Sieira[1], M. Mucientes[1] and A. Bugarín[1⋆]

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

**Abstract.** In this paper we present a reliable motion planner that takes into account the kinematic restrictions, the shape of the robot and the motion uncertainty along the path. Our approach is based on a state lattice that predicts the uncertainty along the paths and obtains the one which minimizes both the probability of collision and the cost. The uncertainty model takes into account the stochasticity in motion and observations and the corrective effect of using a Linear Quadratic Gaussian controller. Moreover, we introduce an adaptive multi-resolution lattice that selects the most adequate resolution for each area of the map based on its complexity. Experimental results, for several environments and robot shapes, show the reliability of the planner and the effectiveness of the multi-resolution approach for decreasing the complexity of the search.

**Keywords:** motion planning, state lattice, multi-resolution

## 1 Introduction

Motion plans must consider the uncertainty due to the inaccuracy of the motion model and the measurements. As the uncertainty depends on the current state and control commands, different paths will have different uncertainties. Safety and accuracy are critical issues, specially to guarantee the integrity of the robot and to ensure the successful completion of the task. Stochastic sampling methods —like probabilistic roadmaps (PRM) [6] and rapidly-exploring randomized trees (RRT) [7]—, and deterministic sampling techniques —like state lattices [13]—, have proven to be successful approaches in the field of motion planning. With the aim to reduce the search space and obtain solutions faster, several state lattice approaches rely on multi-resolution planning techniques [12]. These proposals adjust the resolution of the state lattice, using for example high resolutions near the start and goal states, and lower resolutions in other areas of the environment.

⋆ This work was supported by the Spanish Ministry of Economy and Competitiveness under projects TIN2011-22935, TIN2011-29827-C02-02 and TIN2014-56633-C3-1-R, and the Galician Ministry of Education under the projects EM2014/012 and CN2012/151. A. González-Sieira is supported by a FPU grant (ref. AP2012-5712) from the Spanish Ministry of Education, Culture and Sports.

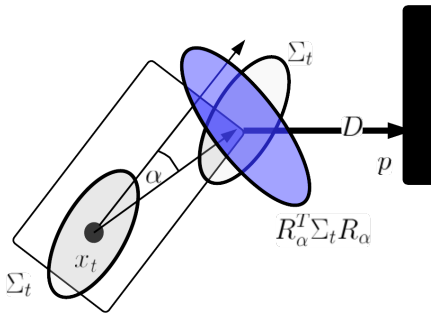



**Fig. 1.** To estimate accurately the probability of collision of the robot with an obstacle it is required to know the PDF at the point of the robot shape closest to the obstacle ($p$). The simple translation of the covariance ($\Sigma_t$) from the center of rotation ($x_t$) to $p$ generates a PDF completely different from the real one —$R_\alpha^T \Sigma_t R_\alpha$, in blue. More details are given in Sec. 3.3.

Most planners assume, that a full knowledge about the states and possible actions is available, leaving the uncertainty management to be solved by feedback controllers that execute the paths. However, in this way planners do not take into account hazardous states, e.g. when a small deviation could cause a collision. Also, the robustness of a solution lies exclusively in the ability of the controller to follow the path with precision. To solve these drawbacks, several authors have proposed different solutions for stochastic sampling motion planners [2][15]. These approaches model the uncertainty at planning time with *a-priori* probability density functions (PDFs) of the states along the paths generated by the planner. These distributions allow to evaluate the candidate paths and select the one that minimizes the probability of collision.

In this paper we present a motion planner based on a state lattice that models the uncertainty with realistic PDFs that take into account the shape of the robot, and the controller that executes the paths. Also, the proposal uses a novel multi-resolution technique to reduce the complexity of the search. The planner is able to obtain safe and optimal paths, i.e., it maximizes the probability of success while minimizes the length of the path —its traversal time. The probability of collision is evaluated in a realistic way, taking into account the uncertainty in the state and the real shape of the robot, instead of making an approximation that underestimates the probability of collision (Fig. 1). Our approach handles stochasticity in both controls and observations and takes into account the corrective effect of using a Linear Quadratic Gaussian (LQG) controller. The prediction of the PDFs along the paths is done in a realistic way without making assumptions about the measurements —i.e. we do not use maximum likelihood observations. We introduce an adaptive approach for multi-resolution planning that groups the actions used to connect the lattice according to their length and maneuvering complexity. As each group of actions allows different kind of motions, each one defines a resolution for the state lattice

—where the connectivity and the distances between neighbors change. For each region of the map, our proposal estimates the maneuvering complexity needed to avoid the obstacles and drive the robot to the goal. Depending on this estimation, it automatically selects the minimum resolution that can be used in each region and, therefore, the planner can obtain solutions faster and without a significant loss in the optimality of the path.

## 2 Related Work

Successful approaches in the field of motion planning are based in the combination of a search algorithm and random sampling —probabilistic roadmaps (PRM) [6] and rapidly-exploring randomized trees (RRT) [7]— or deterministic sampling techniques —state lattices [13]. Although random sampling methods provide a very efficient way for the exploration of the state space, the advantage of state lattices is the possibility to generate offline the set of control actions, instead of using the motion model to connect the random samples in planning time.

Motion planning with uncertainty has raised an increased attention in the last years. Stochasticity in motions arises from different sources: noise in the controls and measurements, partial information about the state and uncertainty in the map. Some approaches only take into account uncertainty in controls as [10], which presents a planner that avoids rough terrain to reduce the differences between the predictions and the result of the controls. In [1] the probability of collision avoidance is maximized using a Markov Decision Process (MDP) approach. Sensor uncertainty can be managed using Partially Observable MPDs (POMPD), as in [4], but this has scalability issues [11].

Van den Berg et. al. [15] propose an algorithm (LQG-MP) based on RRT which takes into account both the uncertainty associated to the control and the measurements and minimizes the probability of collision, but due to the stochasticity of RRT the generated paths may be non smooth. This is addressed via Kalman smoothing but it requires additional processing after planning. It also presents an approach to use LQG-MP in a roadmap with a search algorithm, but with the limitation of splitting the LQG control in the individual trajectories instead of using the same for the whole path. Bry et. al. [2] present a similar approach that uses a user-defined threshold of tolerance to the risk to define a chance-constrained search. Both methods use a bounding disk and a point to represent the robot, respectively. The safety and feasibility of the solutions obtained by the planners may be affected using these approximations, specially in cases where the shape of the robot is significantly different.

The application of multi-resolution planning techniques over state lattices was introduced by Pivtoraiko et. al. [12]. The approach uses a low resolution to represent the state space, except in three defined regions of high resolution: the initial and goal regions, and the area centered on the robot. This significantly increases the search performance, but as the robot moves along the path the resolution of the lattice changes, requiring to replan to keep the solution updated.

## 3 Planning With Uncertainty

Our motion planner obtains the path that minimizes the probability of collision and the time needed to reach the goal. The state space $\mathcal{X}$ is sampled using a state lattice, a deterministic and regular sampling that obtains a set of lattice states $x_t \in \mathcal{X}^{lat} \subset \mathcal{X}$. These states are connected by a finite set of actions $\mathcal{U}$, also called motion primitives, which are extracted from the vehicle dynamics. $\mathcal{U}$ is generated offline, and in this paper we have used an iterative optimization technique based on Newton-Raphson to obtain that set [3].

The motion ($f$) and measurement ($h$) models are assumed to be linear or locally linearizable:

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

where $x_t \in \mathcal{X} = \mathcal{X}^{free} \cup \mathcal{X}^{obs}$ is a state of the robot, $u_t \in \mathcal{U}$ is a control, $z_t$ is the measurement, $m_t$ and $n_t$ are the random motion and observation noises, and $\mathcal{X}^{obs}$ is the set of states in which there are obstacles.

Each action belonging to $\mathcal{U}$ is composed by a set of control commands ($u^{a:b}$) that drive the vehicle from $x^a \in \mathcal{X}^{lat}$ to $x^b \in \mathcal{X}^{lat}$ in a time $t^{a:b}$:

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

where the intermediate states ($x^{a:b}$) are obtained from the motion model ($f$) in absence of process noise.

Because of the regularity of the lattice, the primitives are position-independent, and the same commands can be used to connect every pair of states in $\mathcal{X}^{lat}$ equally arranged. Also, as these actions are extracted from the vehicle dynamics, it is clear that the lattice observes the kinematic restrictions. It is straightforward to translate this representation to a directed weighted graph, where the nodes are the states in $\mathcal{X}^{lat}$ and the arcs are the motion primitives $\mathcal{U}$. The solution to a planning problem is obtained executing a discrete search algorithm over the graph structure.

### 3.1 Search Algorithm

Our proposal relies on the search algorithm Anytime Dynamic A* (AD*)[9][1]. The algorithm is able to obtain sub-optimal bounded solutions varying an heuristic inflation parameter ($\epsilon$). Although AD* typically performs a backwards search, our proposal uses a forward variant to allow the estimation of the PDFs along the generated paths.

[1] We used the implementation included in Hipster4j [14].

**Algorithm 1** Path planning algorithm main loop

1: **while** solution not found **do**
2: select $\arg\min_{x^a \in \mathcal{X}^{lat}}(cost(x^{0:a}) + \epsilon \cdot e(x^a))$
3: **for all** $x^b \in successors(x^a)$ **do**
4: $x^{a:b} = uncertaintyPrediction(x^a, x^b, u^{a:b}, t^{a:b})$
5: $cost(x^{0:b}) = cost(x^{0:a}) + cost(x^{a:b})$
6: **end for**
7: **end while**

An overall approach of the operations done by the motion planning algorithm is summarized in Alg. 1. Iteratively, the most promising state ($x^a$) is selected. This is the state that minimizes the aggregated cost from the initial state —$cost(x^{0:a})$—, and the estimated cost to goal —$\epsilon \cdot e(x^a)$— scaled by $\epsilon$. The heuristic function —$e(x^a)$— is calculated using the mean of the PDFs and combines two values: the cost of the path with kinematic constraints under free space —stored in a Heuristic Look-Up Table [5]— and the cost of the path regardless the motion model but with information about the environment —calculated in a 8-connected 2D grid [8].

Then, the motion planner (Alg. 1) propagates the uncertainty from $x^a$ to its successors according to Alg. 2 (Sec. 3.2). Finally, the outgoing actions of $x^a$ are evaluated to calculate the cost of the path to each successor (Sec. 3.3).

The planner estimates the PDFs for each state along the generated paths. As both the noises and the prior probability follow a Gaussian distribution, and the motion and measurement models can be locally approximated with linear functions, the PDFs are efficiently estimated using EKF-based methods.

The prediction of the PDFs at planning time using an EKF assumes that the observations are those with the maximum likelihood. This underestimates the uncertainty and also ignores the corrective effect of using a feedback controller. To solve these drawbacks, our planner uses the EKF-based method presented in [2]. This method poses that the system will execute the paths using a LQR controller, so the true state and the estimated one are dependent reciprocally. Combining the LQR control policy and the EKF allows to analyze their joint evolution as functions of each other, so the prediction of the PDFs can be done without making assumptions about the observations in planning time.

### 3.2 Uncertainty Prediction

Alg. 2 details the estimation of the PDFs along a trajectory $x^{a:b}$. The calculation starts with the PDF at the initial state of the trajectory ($\mathcal{N}(\bar{x}^a, \Sigma^a)$), and iteratively calculates the PDFs of the intermediate states along the motion primitive that connects them. $L_t$, $A_t$, $B_t$ and $H_t$ represent respectively the gain of the LQG controller, the Jacobians of the motion model and the Jacobian of the measurement model. The PDFs are calculated based on the state estimations given by an EKF —lines 3, 4, 6 and 7— which return the distribution $P(x_t|\tilde{x}_t) = \mathcal{N}(\tilde{x}_t, \widetilde{\Sigma}_t)$, where $\bar{x}_t$ and $\bar{\Sigma}_t$ are the mean and covariance of

**Algorithm 2** *uncertaintyPrediction*$(x^a, x^b, u^{a:b}, t^{a:b})$

1: $\bar{x}^0 = \bar{x}^a$; $\Sigma^0 = \Sigma^a$; $\Lambda^0 = \Lambda^a$; $x^{a:b} = \emptyset$
2: **for all** $t \in [1, t^{a:b}]$ **do**
3: $\quad \bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + M_t$
4: $\quad K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + N_t)^{-1}$
5: $\quad \Lambda_t = (A_t + B_t L_t) \Lambda_{t-1} (A_t + B_t L_t)^T + K_t H_t \bar{\Sigma}_t$
6: $\quad \bar{x}_t = \widetilde{x}_t = f(\bar{x}_{t-1}^{a:b}, u_t^{a:b}, 0)$
7: $\quad \widetilde{\Sigma}_t = (I - K_t H_t) \bar{\Sigma}_t$
8: $\quad \Sigma_t = \widetilde{\Sigma}_t + \Lambda_t$
9: $\quad x^{a:b} = \left\{ x^{a:b} \cup x_t \sim N(\bar{x}_t, \Sigma_t) \right\}$
10: **end for**
11: **return** $x^{a:b}$

the predicted state of the EKF, and $\tilde{x}_t$ and $\widetilde{\Sigma}_t$ are the mean and covariance of the true state of the EKF. The algorithm also calculates $P(\tilde{x}_t) = \mathcal{N}(\bar{x}_t, \Lambda_t)$, which models the uncertainty due to the estimation of the state without the real observations —lines 4 and 5. These two distributions are used to obtain the joint distribution of the real state of the robot and the true state of the EKF, $P(x_t, \tilde{x}_t) = P(x_t|\tilde{x}_t)P(\tilde{x}_t)$. This distribution allows to obtain the PDF of the real state, $P(x_t) = \mathcal{N}(\bar{x}_t, \widetilde{\Sigma}_t + \Lambda_t)$, which can be used by the motion planner as:

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

### 3.3 Path Evaluation

The planner minimizes the probability of collision and the minimum length (represented by the traversal time). The cost function returns a vector with three elements: a safety cost ($c_s$) related to the probability of traversing the path without collisions, the traversal time of the path ($t^{a:b}$), and the uncertainty at the goal ($\Sigma^b$). Each element of the cost defines an objective to minimize in the search. To prioritize the different objectives, the algorithm compares the cost of two paths in the following way:

$$\begin{aligned} cost(x^{0:a}) < cost(x^{0:b}) \Leftrightarrow (c_s^{0:a} < c_s^{0:b}) \vee \\ (c_s^{0:a} = c_s^{0:b} \wedge t^{0:a} < t^{0:b}) \vee \\ (c_s^{0:a} = c_s^{0:b} \wedge t^{0:a} = t^{0:b} \wedge \Sigma^a < \Sigma^b) \end{aligned} \tag{4}$$

We developed a new method to obtain the probability of successfully traversing a path without collisions. The method avoids the sampling of the PDFs — which allows to our planner to have a deterministic behavior— and takes into account the shape of the vehicle. Alg. 3 describes the procedure to evaluate the cost of the path between the states $x^a$ and $x^b$, $x^{a:b}$. Given a PDF estimated by our planner, we use an efficient approximation of the probability of collision based on the number of the standard deviations that the vehicle can afford before colliding at the stage $t$ of the path [15]. The algorithm obtains from the PDF $x_t \sim N(\bar{x}_t, \Sigma_t)$ the eigenvalues and eigenvectors of $\Sigma_t$, and calculates the

**Algorithm 3** $cost(x^{a:b})$

1: $c_s = 0$
2: **for all** $t = 1 : t^{a:b}$ **do**
3: $d_m = \infty$
4: **for all** obstacles in confidence region **do**
5: $D =$ distance between vehicle at $x_t^{a:b}$ and obstacle
6: $d_m = \min(d_m, D^T (R_\alpha^T \Sigma_t R_\alpha)^{-1} D)$
7: **end for**
8: $c_s = c_s - \log(\Gamma(1, d_m/2))$
9: **end for**
10: **return** $\left[c_s, t^{a:b}, \Sigma^b\right]$

2D region of the map for which the PDF has an accumulated probability close to the 100%. Then, it obtains the minimum Mahalanobis distance ($d_m$) between the vehicle and the obstacles. $d_m$ is calculated taking into account the real robot shape —as $D$ is the vector and the border of the robot with minimum $d_m$ (Fig. 1)— and the relative angle $\alpha$ between the point in the border and the heading of the vehicle —which requires the transformation of the covariance with matrix $R_\alpha$, the Jacobian of the transformation function of the point in the border respect to the state. For a multivariate Gaussian distribution, a lower bound for the probability of a sample being within $m$ standard deviations is given by $\Gamma(n/2, m^2/2)$, being $\Gamma$ the regularized Gamma function, $n$ the dimensions of the distribution and $m$ the number of standard deviations. The probability of successfully traversing the path is the accumulation of all the probabilities along the intermediate states (line 8).

## 4 Adaptive Multi-Resolution Planning

The computational efficiency of the motion planner is very dependent on the resolution of the state lattice. Lower resolutions result in a faster search, but decreasing the diversity of maneuvers at higher resolutions. However, in some cases low resolutions could generate paths with higher costs. Multi-resolution planning techniques can manage this trade-off between computational efficiency and performance. Our proposal uses an adaptive selection of the resolution depending on the complexity of each area of the environment. Those regions that require complex maneuvers should be explored with a higher resolution, whereas if simple motions can drive the vehicle to the goal, the resolution could be reduced without a significant loss of performance.

The aim of the proposed adaptive multi-resolution technique is to represent the state lattice with low resolution in those regions where the absence of obstacles allows to drive the vehicle to the goal using long and simple motion primitives. When the search algorithm —Alg. 1, line 3— selects the next state to be expanded ($x^a \in \mathcal{X}^{lat}$), it must obtain its neighborhood. The neighborhood of $x^a$ depends on its resolution. Alg. 4 describes the generation of the successors of

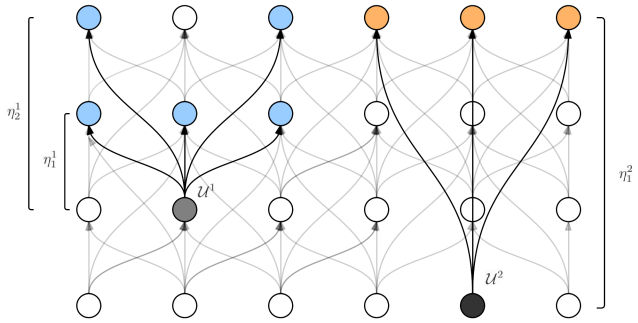



**Fig. 2.** A multi-resolution state lattice. $\mathcal{U}$ is divided in $\mathcal{U}^1$ and $\mathcal{U}^2$ to define two resolutions. Applying $\mathcal{U}^1$ on the gray state generates the successors in blue, and applying $\mathcal{U}^2$ on the black one generates the successors in orange.

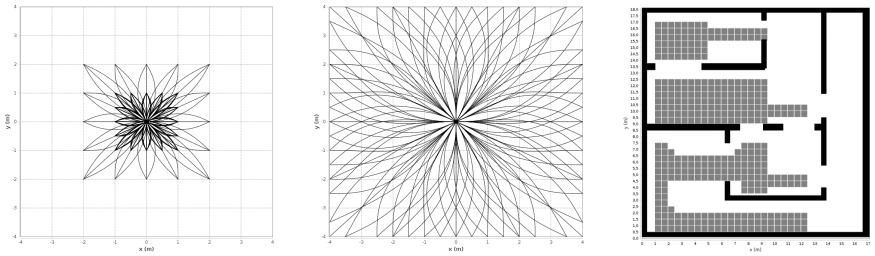
a state $x^a$ through the selection its resolution. $\mathcal{U}$ contains the motion primitives sets for the $\gamma$ different resolutions of the lattice. Each set —$\mathcal{U}^k$, the lower the $k$ the higher the corresponding resolution— has several motion primitives subsets for different neighborhoods ($\mathcal{U}(\eta_j^k)$), where $\eta_j^k$ is the $j$-th neighborhood for the $k$-th resolution (see Fig. 2 for an example).

The resolution of a state $x^a$ is obtained by searching in a 2D grid —where the positions match the states belonging to the highest resolution of the lattice— the path in the direction given by the heading. This allows to check the presence of obstacles in the moving direction —Alg. 4, lines 1-3). Given the path for the 2D grid, the method counts the number of complex maneuvers for each of the resolutions using the closer neighborhood for that resolution —it checks if there is a complex maneuver (based on the turning angle) in the next $\eta_1^i$ steps of the 2D grid path. The algorithm ends when there is a complex maneuver at a given resolution. Finally, it selects the motion primitives of the previous resolution, and generates the neighbor states of $x^a$ using the corresponding primitives ($\mathcal{U}^{i-1}$).

This approach allows to work with several resolutions which are automatically selected depending on the estimated motions. The 2D search is an optimistic

**Algorithm 4** *successors*$(x^a)$

**Require:** $\mathcal{U} = \{\mathcal{U}^1, ..., \mathcal{U}^\gamma\}$
where $\mathcal{U}^k = \{\mathcal{U}(\eta_1^k), ..., \mathcal{U}(\eta_\delta^k)\}$
1: $x^{a:g} = path2D(x^a, x^g)$
2: **for all** $i = 2 : k$ **do**
3: $\beta = \sharp complexMove(x^{a:g}, \eta_1^i)$
4: **if** $\beta > 0$ **then**
5: **return** $\{x^{b_1}, x^{b_2}, ..., x^{b_n}\}$ given by $\mathcal{U}^{i-1}$
6: **end if**
7: **return** $\{x^{b_1}, x^{b_2}, ..., x^{b_n}\}$ given by $\mathcal{U}^k$
8: **end for**



(a) High resolution primitives ($\mathcal{U}_H$).

(b) Low resolution primitives ($\mathcal{U}_L$).

(c) Resolution map (gray is low) when heading is $\theta = 0$.

**Fig. 3.** Different sets of actions ($\mathcal{U}$) are used for each resolution.

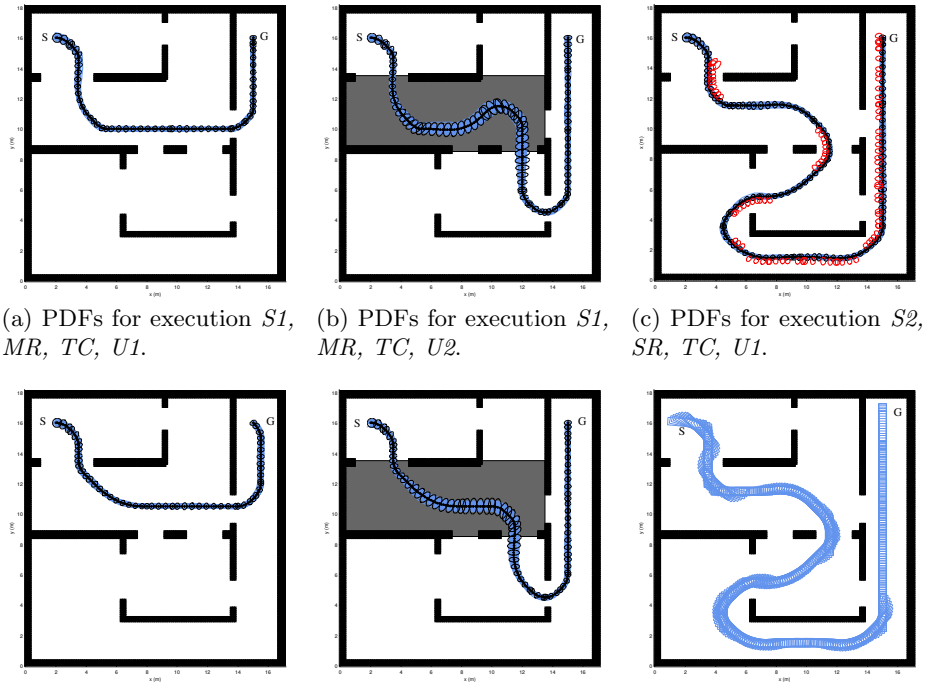
approximation to the real path, as it does not take into account the motion model or the real shape. This may lead to pick a resolution higher than necessary. Nevertheless, the opposite is not possible; this ensures that the adaptive selection of the resolution does not underestimate the complexity of the maneuvers in that area of the path. Our proposal reuses the 2D heuristic described in Sec. 3.1. This search is executed once before the planning process, and whenever a change in the environment occurs. Therefore, the adaptive multi-resolution technique does not require any new operations at planning time.

## 5 Experimental Results

This proposal has been tested with a differential drive robot, which state is a 5-dimensional variable described by the pose at its center, the linear and the angular speeds: $x_t = (x_x, x_y, x_\theta, x_v, x_\omega)^T$. Commands are described by the linear and the angular velocity controls, $u_t = (u_v, u_\omega)^T$. The trajectories of the motion primitives ($\mathcal{U}$) were generated for neighborhood distances: 1, 2, 4 and 8 —each unit is 0.5 $m$— resulting in trajectories between 0.5 $m$ and 4 $m$ long. The primitives have been divided in two resolutions ($\gamma = 2$) as seen in Fig. 3: high ($\mathcal{U}_H$) and low ($\mathcal{U}_L$) containing trajectories up to 2 $m$ and longer, respectively.

We tested different configurations for the planner: with and without multi-resolution —*MR* and *SR*—, with and without estimating the covariance at the border of the robot —*TC* and *NC*—, two different shapes —*S1* and *S2*— and two uncertainty conditions —*U1* and *U2*. Detailed results are given in Table 1: number of expansions of the search algorithm, number of nodes evaluated (insertions), planning time, cost of the solution path (traversal time), probability of collision estimated by the planner and the real one. This one was obtained simulating the execution of the planned path 100 times using a LQG controller and introducing stochastic noise in controls and observations with covariances:

$$M = diag\left(0.1^2, \frac{\pi^2}{60^2}\right), N = diag\left(0.1^2, 0.1^2, \frac{\pi^2}{60^2}\right), N' = diag\left(0.3^2, 0.3^2, \frac{\pi^2}{9^2}\right)$$




(a) PDFs for execution *S1, MR, TC, U1.*

(b) PDFs for execution *S1, MR, TC, U2.*

(c) PDFs for execution *S2, SR, TC, U1.*

(d) PDFs for execution *S1, SR, TC, U1.*

(e) PDFs for execution *S1, SR, TC, U2*

(f) Trail for execution *S2, SR, TC, U1.*

**Fig. 4.** Planning results for different shapes and uncertainty conditions. Executions of the planned paths are in blue, the PDFs of the rotation center in black, and their transformation to the border of the robot in red. Regions with higher uncertainty in observations are in gray—lower localization precision. Environment is 17 $m \times 18$ $m$.

A comparison of the planning results for different uncertainty conditions and robot shapes is shown in Fig. 4. Performing several simulations for the planned path also allows to validate the good performance of the a-priori PDFs (in black) to the paths obtained in execution time (in blue). Translating the PDFs from the rotation center to the border of the shape does not reflect the real distribution in that point. Figure 4(c) stands out those situations in which the PDF estimated at the border provides a more realistic estimation of probability of collision, while the last two rows in Table 1 compare the results of the planner with and without using the correct PDFs at the border. Our method to predict the real PDFs at the border achieves a better performance when turning near obstacles at the sides of the robot. This results in a lower cost of the solution and a better estimation of the probability of collision along the planned path.

The experiments also show the reliability of our planner to work with several robot shapes. While *S1* can safely cross the first door (Fig. 4(a)), for *S2* this becomes unfeasible and the planner selects the longest —but safest path in the

**Table 1.** Summary of the tests. *MR* and *SR* mean multi and single-resolution, while *TC* indicate the estimation of the PDF at the border of the robot. *U1* and *U2* are different uncertainty conditions. *S1* dimensions are $0.5x0.5$ $m$ and *S2* 2.5x0.5 $m$.

| **Execution** | **Expansions** | **Insertions** | **Time (s)** | $p_{col}$ **(est)** | **Cost (s)** | $p_{col}$ **(real)** |
|---|---|---|---|---|---|---|
| *S1, MR, TC, U1* | 422 | 2326 | 13 | 0.0 | 50.82 | 0.0 |
| *S1, SR, TC, U1* | 1566 | 8356 | 40 | 0.0 | 49.50 | 0.0 |
| *S1, MR, TC, U2* | 1164 | 5376 | 32 | 0.0 | 77.88 | 0.0 |
| *S1, SR, TC, U2* | 3591 | 19489 | 39 | 0.0 | 71.28 | 0.0 |
| *S2, MR, TC, U1* | 1990 | 9627 | 97 | 8.099E-9 | 117.81 | 0.0 |
| *S2, SR, TC, U1* | 4285 | 23068 | 180 | 1.404E-12 | 111.87 | 0.0 |
| *S2, SR, NC, U1* | 4423 | 23433 | 519 | 1.738E-11 | 113.19 | 0.0 |

environment (Fig. 4(c)). The coverage of shape *S2* when executing the maximum likelihood path in this experiment is shown in Fig. 4(f). Moreover, Fig. 4(a) and Fig. 4(b) also show how the planner selects a different path for shape *S1* to avoid crossing the first door because of the high probability of collision when adding uncertainty to the environment.

Fig. 3(c) shows the resolution assigned to each region of the environment when the robot heading is $\theta = 0$. The complexity in each state $x^a$ is estimated counting the number of turnings of the 8 first steps —maximum neighborhood distance in the set of primitives $\mathcal{U}$— of the 2D path in the direction given by the heading. Figs. 4(a), 4(b), 4(d) and 4(e) compare the paths obtained with and without multi-resolution. The adaptive selection of the resolution decreases the average neighborhood of the lattice and reduces the branching factor of the search —related to the number of insertions. It also decreases the operations required to propagate the uncertainty along the candidate paths. As table 1 shows, our multi-resolution approach obtains a significant reduction in the number of nodes managed by the search algorithm —62% in expansions and 66% in insertions, on average—, while the cost of the planned paths increases only in a 5.6%. Also, it does not affect to the estimation of the probability of collision, which is similar to the one obtained from the executions of the paths.

## 6 Conclusions

We introduced an adaptive multi-resolution state lattice approach for motion planning. The proposal selects automatically the resolution of each state according to the estimated complexity of the maneuvers in the next steps. The planner also predicts at planning time the motion uncertainty due to the imprecision in controls and observations and takes into account the corrective effect of executing the paths with a LQG controller. Moreover, the algorithm evaluates the probability of successfully traversing the path by taking into account the real shape of the robot, i.e., the PDF is calculated for the point of the robot closer to the obstacles. The experiments show a reduction in the number of explored states due to the adaptive multi-resolution technique. Also, the prediction of the PDFs

along the paths has proved to be a very good approximation to the real PDFs. As future work, we expect to use the information of a multi-resolution map in the selection of the lattice resolution in order to achieve a better reduction of the search space. This will significantly reduce the planning time.

## References

1. 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)
2. 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)
3. 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)
4. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial intelligence 101(1), 99–134 (1998)
5. Knepper, R., Kelly, A.: High Performance State Lattice Planning Using Heuristic Look-Up Tables. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) pp. 3375–3380 (2006)
6. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge, U.K. (2006)
7. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. The International Journal of Robotics Research 20(5), 378–400 (2001)
8. Likhachev, M., Ferguson, D.: Planning Long Dynamically Feasible Maneuvers for Autonomous Vehicles. The International Journal of Robotics Research 28(8), 933–945 (Jun 2009)
9. 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)
10. 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)
11. Papadimitriou, C.H., Tsitsiklis, J.N.: The complexity of Markov decision processes. Mathematics of operations research 12(3), 441–450 (1987)
12. Pivtoraiko, M., Kelly, A.: Differentially constrained motion replanning using state lattices with graduated fidelity. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 2611–2616 (2008)
13. 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)
14. Rodriguez-Mier, P., Gonzalez-Sieira, A., Mucientes, M., Lama, M., Bugarin, A.: Hipster: An open source java library for heuristic search. In: Information Systems and Technologies (CISTI), 2014 9th Iberian Conference on. pp. 1–6. IEEE (2014)
15. 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)