Graduated Fidelity Lattices for Motion Planning under Uncertainty

Adrián González-Sieira[1], Manuel Mucientes[1] and Alberto Bugarín[1]

***Abstract*— In this work we present a state lattice based approach for motion planning in mobile robotics. Sensing and motion uncertainty are managed at planning time to obtain safe and optimal paths. To do this reliably, our approach estimates the probability of collision taking into account the robot shape and the uncertainty in heading. We also introduce a novel graduated fidelity approach and a multi-resolution heuristic which adapt to the obstacles in the map, improving the planning efficiency while maintaining its performance. Results for different environments, shapes and motion models are reported, including experiments with real robots.**

I. Introduction

Motion planning in mobile robotics has been successfully addressed using stochastic and deterministic sampling strategies [1]. Among the latter, state lattices are noteworthy for their regularity and for having the structure of a graph in which the discrete states are connected by motions extracted from the dynamics model. Thus, optimal paths satisfying the kinematic constraints can be found with a search algorithm.

Managing motion and sensing uncertainty is also important, because the safety of the planned paths is crucial in real world applications. Doing so at planning time allows selecting the best path according to its probability of collision, which should be reliably estimated taking into account the robot dimensions and the uncertainty in heading.

The fidelity of the lattice is the resolution of the sampled states. Decreasing the fidelity improves the planning efficiency, but it affects the optimality and the capacity to obtain valid solutions. However, in a graduated fidelity lattice the resolution can vary and adapt to the obstacles in the map, managing the trade off between planning performance and efficiency. Heuristics are key for the latter, so reducing their computation time is equally relevant. Multi-resolution maps, like octrees [2], also adapt to the obstacles, so they can be a good source of information for the fidelity selection and to introduce multi-resolution techniques in the heuristics.

In this paper we present a state lattice based motion planner that manages the uncertainty at planning time. Our main contributions are: 1) a sampling strategy which reliably estimates the probability of collision of the robot taking into account its dimensions and the uncertainty in heading; 2) a novel graduated fidelity approach which, unlike prior works, adapts to the maneuverability of the robot and to the obstacles in the map; and 3) a novel multi-resolution heuristic with improved efficiency and scalability in large environments. This approach allows to reduce significantly the runtime of the planner while maintaining its performance.

This research was supported by the Spanish Ministry of Economy and Competitiveness (grants TIN2014-56633-C3-1-R and TIN2017-84796-C2-1-R), and the Galician Ministry of Education, Culture and Universities (grants GRC2014/030 and accreditation 2016-2019, ED431G/08). These grants are co-funded by the European Regional Development Fund (ERDF/FEDER program).

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

II. Related work

Sampling based techniques combined with search algorithms have been successfully applied to the field of motion planning. Random sampling —Probabilistic Roadmaps (PRM) [3], Rapidly Exploring Random Trees (RRT) [4]— and deterministic sampling techniques have been described in the literature. Among the latter, state lattices [5] benefit from their regularity to pre-compute a set of canonical actions from the dynamics model. Graduated fidelity state lattices were introduced by [6], although this approach does not take into account the dynamics in the low fidelity areas, thus making impossible managing the uncertainty at planning time. In [7] this issue was addressed using a subset of the motion primitives to connect the states within low fidelity areas. However, the approach of [7] and [8] is to obtain a pre-planned path and improve it in real time as the robot moves along the planned path, increasing the fidelity of the lattice around its position. Combining this with uncertainty management is not trivial because every time the lattice changes the uncertainty of the affected paths has to be recomputed, which is a costly operation. Instead, our graduated fidelity approach only depends on the obstacles in the map and the maneuverability of the robot, addressing these issues.

Different heuristics to improve the planning efficiency were described in the literature. In [9] they presented an admissible heuristic —FSH— which copes with the robot dynamics. Although its obtention is computationally expensive, it can be pre-computed offline and stored in a look-up table. In [7] they introduced H2D, a low-dimensional heuristic which takes into account the obstacles in the map, and which combined with FSH obtained good results. However, it is computed applying the Dijkstra's algorithm over a grid with a fixed resolution, so its efficiency when dealing with large and uncluttered environments can be improved. For this reason we introduce H2DMR, a heuristic based on that of [7], which relies on a multi-resolution grid and improves the planning efficiency and scalability.

Regarding the uncertainty management, prior works used the theory of Markov Decision Processes —MDPs and POMDPs— to deal with the inaccuracies in the controls [10] and measurements [11], although they had scalability

issues [12]. These were addressed by [13], which computes a locally-optimal solution given an initial path. However, this approach can only be applied to smooth dynamics and observations. Other works, like [14] and [15], use the Extended Kalman Filter —EKF— to deal with motion and sensing uncertainty without assuming maximum likelihood observations. [14] relies on RRT to compute a set of paths and selects among them the one with the minimal probability of collision, although this does not guarantee finding a good solution in all cases. [15] addresses this drawback growing a search tree which predicts the uncertainty for all candidate paths. Despite their good results in uncertainty management, both approaches estimate the probability of collision using simplified versions of the robot shape. [14] approximates the robot by its bounding circle, while [15] checks collisions with the predicted distributions. This affects the reliability of the planner and leads to potential failures when the shape is far from these approximations —e.g. asymmetric or irregular shapes. Our approach considers the real robot shape with a deterministic sampling strategy which computes the probability of collision from the predicted uncertainty, also taking into account the uncertainty in heading.

III. Planning on state lattices

Our motion planner relies on a state lattice, so the state space, $\mathcal{X}$, is sampled in a regular manner —a rectangular arrangement was used in this work, although others are possible. A canonical set of actions extracted from the dynamics model —$\mathcal{U}$, also called motion primitives— connects the lattice states, $\mathcal{X}_{lat}$. Due to their regular arrangement, these actions are position-independent and can be computed offline and replicated to generate the whole connectivity. Moreover, these actions respect the kinematic restrictions.

The motion primitives are obtained using numerical optimization, as detailed in [16]. The cost of the resulting actions is optimal given the constraints: the initial an final states, belonging to $\mathcal{X}_{lat}$, and the dynamics model.

Due to the graph structure of the state lattice, an informed search algorithm can be used to find the optimal path in it. These algorithms rely on heuristics, which estimate the cost between each state and the goal, to efficiently explore the state space. The planner we present uses Anytime Dynamic A* [17], AD*, due to its capability to obtain sub-optimal solutions and refine them iteratively without planning from scratch. The value of the heuristic is inflated by a parameter ϵ, which acts as boundary for the cost of the solution.

As heuristic, we combine the proposed H2DMR —which deals with the obstacles in the map, like H2D [7]— with FSH [9] —which copes with the kinematic restrictions while considering free space—, as follows:

$$\text{HEURISTIC}(x) = \max\left(\textit{h2dmr}(x), \textit{fsh}(x)\right) \qquad (1)$$

This allows estimating the cost to the goal in an accurate manner, which benefits the planning efficiency.

Alg. 1 summarizes the operations done by AD*. Inputs —x^0 and x^G— are the initial state and the goal, and the output is the path with the minimal cost connecting them. H2DMR is initialized before running the planner —Alg. 1:2—, since it depends on the location of the goal and the obstacles in the environment. The algorithm is then run iteratively, starting with $\epsilon = \epsilon_0$ and decreasing the value of the parameter until the optimal solution is found —Alg. 1:13.

Given a value for ϵ, the algorithm iteratively extracts a state x^a from the *OPEN* queue —Alg. 1:6. This is the state minimizing the added cost from the start and the estimated one to the goal —c_x and h_x, respectively. The latter is given by the heuristic and scaled by ϵ. The successors for x^a, X^b, are then obtained in Alg. 1:7 and the cost of each transition is evaluated —Alg. 1:9. Finally, each state x^b is inserted in *OPEN* after computing its heuristic —Alg. 1:10-11. A valid path is found when x^a is the goal.

To ensure the safety of the planned paths they have to be evaluated in terms of probability of collision. This requires managing the uncertainty at planning time, for which we use the approach of [15] to predict the Probability Density Functions of the robot being in each state of the path —PDFs. This method focuses on nonlinear, partially observable systems with dynamics and observations described in a discrete time manner:

$$\begin{aligned} x_{t+1} &= f(x_t, u_t) + m_t, & m_t &\sim \mathcal{N}(0, M_t) \\ z_t &= z(x_t) + n_t, & n_t &\sim \mathcal{N}(0, N_t) \end{aligned} \qquad (2)$$

where $x_t \in \mathcal{X}$, $u_t \in \mathcal{U}$ and z_t are the robot states, the controls and the measurements. m_t and n_t are Gaussian motion and observation noises with covariances M_t and N_t. The PDFs are estimated considering the influence of a Linear Quadratic Gaussian [18] controller —LQG— when executing the planned paths, using linearized versions of f and z. The uncertainty depends on that at the initial state, the controls and the accuracy of the measurements; and it is therefore different for each candidate path in the lattice.

IV. Reliable probability of collision

The distributions resulting from predicting the uncertainty are used to estimate the probability that the robot collides

Algorithm 1 Main operations of the search algorithm

Require: x^0, initial state; x^G, goal state; initially $\epsilon = \epsilon_0$

1: **function** MAIN (x^0, x^G, ϵ)
2: INITIALIZEHEURISTIC(x^0, x^G) ▷ Alg. 4
3: **while** $\epsilon >= 1$ **do**
4: $c_{x^0} = 0$; h_{x^0} = HEURISTIC(x^0); *OPEN* = $\{x^0\}$
5: **repeat**
6: $x^a = \arg\min_{x \in OPEN}(c_x + \epsilon \cdot h_x)$
7: X^b = SUCCESSORS(x^a) ▷ Alg.3
8: **for all** $x^b \in X^b$ **do**
9: $c_{x^b} = c_{x^a} +$ COST(x^a, x^b) ▷ Alg. 2
10: h_{x^b} = HEURISTIC(x^b) ▷ Alg. 4
11: *OPEN* = *OPEN* $\cup \{x^b\}$
12: **until** $x^a = x^G$
13: publish *path*(x^0, x^a) and decrease ϵ
14: **return**

when executing the paths. We introduce a novel method to do so in a reliable manner, taking into account the real shape of the robot and dealing with the uncertainty in heading.

The goal of the planner is to obtain paths minimizing the probability of collision and the traversal time. This is achieved with a cost function which evaluates a path between x^a and x^b with three objectives: a safety measure, which depends on the probability of collision; the traversal time; and the uncertainty at x^b —$c^{a:b}$, $t^{a:b}$ and Σ^b, respectively. An order of priority is introduced in these elements, so that the planner first minimizes the probability of collision, then the traversal time and finally the uncertainty at the goal.

Alg. 2 details the evaluation of a path between two states x^a and x^b. First, the PDFs of the path are predicted with the approach of Bry et. al. [15] —Alg. 2:2—, since they are needed to obtain $c^{a:b}$ and Σ^b. Conversely, $t^{a:b}$ is directly given by the motion primitives —Alg. 2:3.

The probability of collision is estimated from the PDFs with a deterministic sampling strategy, which is similar to the obtention of the sigma points in an Unscented Kalman Filter —UKF [19]. Given a n-dimensional distribution —$x_t \sim \mathcal{N}(\bar{x}_t, \Sigma_t)$— a set of samples X^S is obtained:

$$x^s_{i^+}, x^s_{i^-} = \bar{x}_t \pm \lambda \cdot \left(\sqrt{\Sigma_t}\right)_i \qquad i = 1, ..., n \tag{3}$$

where $\left(\sqrt{\Sigma_t}\right)_i$ is the i-th column of the factorized covariance matrix, and λ allows obtaining samples with different distances to $\bar{x}_t$. Then, the following samples outside the main axes of the distribution are added to X^S:

$$\begin{aligned} x^s_{i^-j^+}, x^s_{i^-j^-} &= x^s_{i^-} \pm \lambda \cdot \left(\sqrt{\Sigma_t}\right)_j \qquad i, j = 1, ..., n \\ x^s_{i^+j^+}, x^s_{i^+j^-} &= x^s_{i^+} \pm \lambda \cdot \left(\sqrt{\Sigma_t}\right)_j \qquad j \neq i \end{aligned} \tag{4}$$

By doing so we achieve a better coverage of the PDF for collision check purposes —Fig. 1.

Each sample $x^s \in X^S$ is weighted according to its probability in the PDF. The probability that the robot collides —Alg. 2:8-11— is the ratio of the weights of the colliding samples —checked with the real shape of the robot— and the total weight. Then, the collision cost of the path, $c^{a:b}$, is obtained combining the estimations of the PDFs belonging to it, assuming they are independent —Alg. 2:12. The cost due to Σ^b is the trace of the matrix —Alg. 2:13.

The elements of the cost are compared hierarchically, which allows obtaining paths that are optimal in terms of safety, in the first place, then in terms of traversal time —given the set of motion primitives used for planning—, and finally in terms of estimated final uncertainty.

Algorithm 2 Cost of a trajectory between x^a and x^b

Require: x^a and x^b, beginning and final states

1: **function** COST(x^a, x^b)
2: $P^{a:b}$ = UNCERTAINTY(x^a, x^b) ▷ PDF prediction
3: $t^{a:b} = \mathit{time}(u^{a:b})$
4: $c^{a:b} = 0$
5: **for all** $x^{a:b}_t \sim \mathcal{N}(\bar{x}^{a:b}_t, \Sigma^{a:b}_t) \in P^{a:b}$ **do**
6: $w_c = 0$; $w_t = 0$
7: $X^S = \mathit{sampling}(x^{a:b}_t)$
8: **for all** $x^s \in X^S$ **do**
9: $w_t = w_t + \mathit{pdf}(x^s, x^{a:b}_t)$
10: **if** $\mathit{collision}(x^s)$ **then** ▷ with real shape
11: $w_c = w_c + \mathit{pdf}(x^s, x^{a:b}_t)$
12: $c^{a:b} = c^{a:b} - \log(1 - w_c/w_t)$
13: **return** $\left[c^{a:b}, t^{a:b}, \mathit{tr}(\Sigma^b)\right]$ ▷ Σ^b, uncertainty at x^b

V. Efficient Motion Planning

The fidelity of the lattice has a great impact in the planning efficiency. High fidelities allow representing more precisely the state space, while lower ones benefit the runtime at the expense the cost of the paths. The trade off between them can be properly managed with a graduated fidelity lattice, specially if it adapts to the obstacles in the map.

The standpoint of the proposed method is to use high fidelity only in those areas which require complex maneuvering —i.e. near the obstacles—, selecting long trajectories whenever possible —Fig. 2(a). Thus, the state space is simplified reducing the density of lattice states and candidate paths connecting them. This is achieved grouping together similar actions in $\mathcal{U}$ and selecting only one representative of each group when generating the successors of a state —Alg. 1:7. The selected maneuver is the longest one which does not affect the probability of collision —Fig. 2(b). Each group contains those trajectories beginning and ending with the same heading and velocities, $\mathcal{U}_{(\theta_i, v_i, \omega_i, \theta_f, v_f, \omega_f)}$, so that it has maneuvers of the same kind but different length. Any two groups are disjoint, while the union of all of them is the whole set of motion primitives, $\mathcal{U}$.

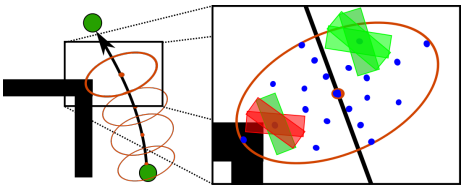
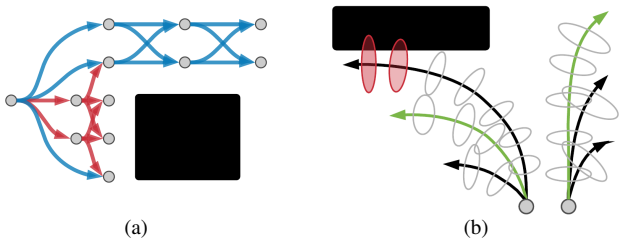


Fig. 1: We estimate the probability of collision considering the robot real shape, deterministically sampling the PDFs.



(a) (b)

Fig. 2: Graduated fidelity approach. (a) shows a lattice combining high (red) and low (blue) fidelity; (b) the fidelity selection approach: the longest collision free motion of each group is selected (green).

Algorithm 3 State successors in a graduated fidelity lattice

Require: $\mathcal{U} = \{\mathcal{U}_{(\theta_i,v_i,\omega_i,\theta_f,v_f,\omega_f)}\}, \forall(\theta, v, \omega) \in \mathcal{X}_{lat}$

1: **function** SUCCESSORS(x^a)
2: $\theta_i = x^a_\theta$; $v_i = x^a_v$; $\omega_i = x^a_\omega$; $\Gamma = \emptyset$
3: **for** $U \in \{\mathcal{U}_{(\theta_i,v_i,\omega_i,\theta_f,v_f,\omega_f)}\}, \forall(\theta_f, v_f, \omega_f)$ **do**
4: **repeat**
5: $U^c = \arg\max_{t^{a:b}}(U)$ ▷ Get longest
6: $U = U \setminus U^c$
7: **until** CHECK($U^c_{x^a}, U^c_{x^b}$) $\vee$ $U == \emptyset$
8: $\Gamma = \Gamma \cup U^c$
9: **return** Γ
10: **function** CHECK(x^a, x^b)
11: $\kappa^a = cell(x^a)$; $\kappa^b = cell(x^b)$ ▷ Get map cells
12: $s^a = size(\kappa^a)$; $s^b = size(\kappa^b)$ ▷ Get size of cells
13: $c^{a:b}$ = COST(x^a, x^b)[0] ▷ Alg. 2
14: **return** $(s^a + s^b \geqslant \|x^a - x^b\|) \wedge (c^{a:b} == 0)$

Alg. 3 details this process. The successors of a state x^a, Γ, are selected from those groups of trajectories with the same initial heading and velocities —Alg. 3:3. For each group, all candidates are evaluated in descending order by $t^{a:b}$, and the first one fulfilling the restrictions is selected —Alg. 3:4-7. These restrictions —Alg. 3:10-14— relate to the resolution of the source and destination map cells —s^a and s^b— and the probability of collision. The goal of the former is to choose maneuvers in accordance with the resolution of the map, which is a good indicator of the complexity of the environment. The latter maintains the safety of the solutions and it is obtained from the cost function —Alg. 2.

With this approach the density of states and maneuvers significantly decreases except near the obstacles, where complex maneuvering is required. This improves the planning efficiency, while the cost of the solutions is barely affected.

A. Multi-resolution heuristic

Heuristics play a key role in the planning efficiency and their obtention time should be the lowest possible. Unlike FSH, H2D cannot be computed offline, since it depends on the obstacles in the map. Therefore, we propose H2DMR, a multi-resolution version of the latter which takes advantage of the octree map to reduce its obtention time.

Dijkstra's algorithm is used to build a multi-resolution grid which stores the distance between each point and the goal considering the obstacles —Alg. 4. Collisions are checked with the inscribed circle to the real shape to maintain the optimistic nature of the estimations. The resolution of this grid depends on that of the map and the highest fidelity of the motion primitives, f^+. H2DMR uses positions instead of states, and the cost is the euclidean distance —Alg. 4:19-20. Starting in the goal, p^G, the point with the lowest cost from the start is iteratively selected —Alg. 4:5— and its successors explored. As in H2D [7], the stopping condition is reaching twice the cost between p^0 and p^G.

This approach takes advantage of the octree structure —Fig. 3. Given a point p, its successors are the center of the map cells which are adjacent to the one containing p. If the size of a cell is higher than f^+, it is split into subcells to keep the accuracy of the heuristic —Alg. 4:8-12. Otherwise a bigger cell containing it is selected —Alg. 4:13-17— to limit the highest resolution of the grid to f^+.

Algorithm 4 Obtention of H2DMR

Require: f^+, highest fidelity of the lattice

1: **function** INITIALIZEHEURISTIC(x^0, x^G)
2: $p^0 = position(x^0)$; $p^G = position(x^G)$;
3: $c(p^G) = 0$; $OPEN = \{p^G\}$
4: **repeat**
5: $p = \arg\min_{p \in OPEN} c(p)$
6: $\kappa = cell(p)$ ▷ Get map cell containing p
7: **for all** $\kappa' \in adjacent(\kappa)$ **do** ▷ Adjacents to κ'
8: **if** $size(\kappa') > f^+$ **then** ▷ Size of cell κ'
9: **for all** $\kappa'' \in subcells(\kappa')$ **do**
10: $p' = position(\kappa'')$ ▷ Center of κ''
11: $c(p') = c(p) +$ COSTH(p, p')
12: $OPEN = OPEN \cup \{p'\}$
13: **else**
14: $\kappa'' = adjust(\kappa', f^+)$ ▷ Adjust size to f^+
15: $p' = position(\kappa'')$ ▷ Center of κ''
16: $c(p') = c(p) +$ COSTH(p, p')
17: $OPEN = OPEN \cup \{p'\}$
18: **until** $c(p) > 2 \cdot c(p^0)$
19: **function** COSTH (p, p') ▷ Uses the optimistic shape
20: **if** $collides(p')$ **then return** ∞; **else return** $\|p' - p\|$

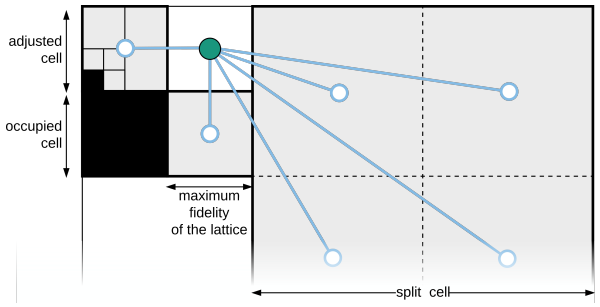


Fig. 3: Given a point (in green), its neighbors (in blue) are the adjacent map cells (in gray). Those bigger than f^+ are split, otherwise bigger ones containing them are selected.

AD* uses the costs stored in this multi-resolution grid as heuristic. However, the positions of H2DMR and the lattice states do not directly match, so the value for a state x^a is given by the position, p, contained in the cell of x^a or one of its adjacents which minimizes the sum of its cost in the grid —$c(p)$— and its distance to x^a:

$$h2dmr(x^a) = \arg\min_p(\|x^a - p\| + c(p)) \tag{5}$$

This approach outperforms H2D in the obtention time of the grid, since it requires significantly less iterations to build it. Moreover, due to the octree structure the benefits are even more noticeable in large and uncluttered environments.

VI. Results

The following experiments were run on a computer with a CPU Intel Core™ i7-4790 and 16 GB of RAM. Motion uncertainty is $M_t = 0.01 \cdot I$. Measurement noise is $N_t = 0.01 \cdot I$ or $N_t = \infty \cdot I$, depending on whether the robot is in a location denied area or not.

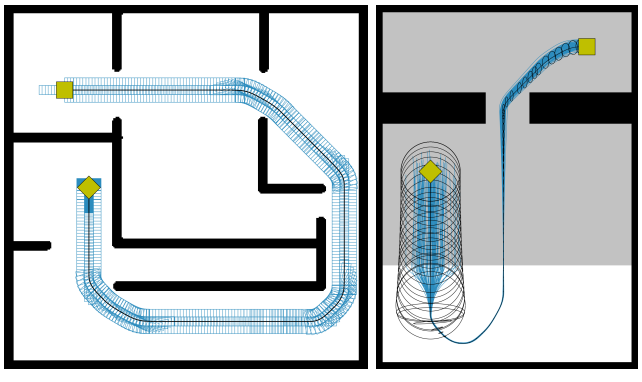
A. Reliable probability of collision

In this section we report results for a robot with 2D Ackermann dynamics and a set of 336 motion primitives with lengths between $0.5\ m$ —f^+— and $8\ m$.

Fig. 4(a) shows the planned path in a $30 \times 30\ m$ squared environment. The robot has an irregular shape, like a "T", with dimensions $3.0 \times 0.75\ m$ and $2.0 \times 0.75\ m$ —long and short edges. Our approach finds a safe path for this environment, whilst other approaches disregarding the real shape and the uncertainty in heading would fail. Approximating the robot by its inner circle would be unreliable due to underestimating probability of collision in the turnings. On the contrary, using a bounding box would be too conservative and some maneuvers would be discarded for being too risky —e.g. the turning before entering the corridor.

Fig. 4(b) shows the optimal solution and the predicted PDFs for an environment of $25 \times 35\ m$ and a rectangular robot of $3.0 \times 0.75\ m$. Due to the uncertainty, the best path in terms of probability of collision and traversal time goes outside the location denied area to receive measurements, allowing the robot to localize itself before crossing the door.

We have measured the accuracy of the estimations given by our method and compared it with that of the gamma function —used by other approaches, like [14]. To do this, we compared the output of both methods for the robot with rectangular shape, a diagonally placed obstacle, and an uncertainty of $\Sigma = I$. The error of the gamma function was inversely proportional to the distance between the robot and the obstacle, up to a 59% for distances between $0.75\ m$ and $1.0\ m$. Conversely, as our PDF sampling strategy takes into account the deviations both in position and orientations, it obtained reliable estimations for the entire range of distances —with only a 1.5% of average error. To show the reliability of the planner, we run $1,000$ simulations of the solutions of Fig. 4 with random noise, which were all collision free.



(a) (b)

Fig. 4: Motion planning results. The start is the diamond, and the goal is the rectangle. (a) shows the predicted uncertainty, where the ellipses are $3 \cdot \sigma$ of the PDFs, and the location denied areas are in gray; (b) shows the path for a robot with an irregular shape (in blue).

B. Planning efficiency

To improve the planning efficiency, the proposed graduated fidelity method reduces the number of candidate paths decreasing the fidelity, but only in the uncluttered areas of the environment. Thus, high fidelity is used only when the estimated probability of collision is greater than 0 —due to the proximity of obstacles or the amount of uncertainty. With this approach, the complex maneuvers in the control set are available where they are most needed. Similarly, using high fidelity near the goal facilitates approaching to this point.

Table I details the impact of the graduated fidelity lattice approach —GF— in the planning efficiency and performance. Columns "Iterations" and "Insertions" count the extractions and insertions in OPEN —see Alg. 1—, "Time" the runtime of the planner, while "Cost" is the traversal time of the solution. A 87.4% average reduction in the number of iterations is achieved, although more important is the reduction in the number of insertions, which is decreased a 89.7%. This caused the runtime to improve on average a 89.9%, as each insertion in OPEN involves obtaining the PDFs and the probability of collision for the new candidate path. With regard to the performance, there is an average increase of a 9.7% in the cost of the paths caused by selecting longer maneuvers in uncluttered areas, although this is almost unnoticeable when the solutions are visually compared.

Executions in Table I were made without anytime search —$\epsilon_0 = 1$— to show the planning efficiency due to the graduated fidelity, although they improved when iteratively refining the solution until finding the optimal one. For $\epsilon_0 = 1.5$ and the experiment of Fig. 4(a), the optimal plan was obtained in $1.2\ s$ and 422 iterations, for $\epsilon = 1.15$. For Fig. 4(b) it was obtained in $2.2\ s$ and 684 iterations, for $\epsilon = 1.03$.

These results outperform other approaches in the state of the art. In the environment of Fig. 4(b), [15] finds a valid solution in approximately $40\ s$ — $17,500$ iterations—, while [14] fails due to not expanding the paths going through the measurement area. [13] obtains in $3.65\ s$ a locally-optimal control policy based on an initial path, and their simulations show a 93% of probability of success. However, this approach requires obtaining a policy for each homotopy class of trajectory to guarantee that the global optimum is found, which would lead to higher planning times.

TABLE I: Impact of the graduated fidelity lattice.

		Planning			Solution
Fig.	**GF**	**Iterations**	**Insertions**	**Time (s)**	**Cost (s)**
4(a)	✓	525	1,191	1.65	133.46
4(a)	✗	2,755	10,844	11.1	130.86
4(b)	✓	1,302	2,868	2.68	108.27
4(b)	✗	15,952	42,594	45.38	95.96

TABLE II: Efficiency of H2DMR compared to that of H2D.

	Iterations			Time (ms)		
C_+	**SR**	**MR**	**Gain**	**SR**	**MR**	**Gain**
≤0.8	8,281	3,250	60.8%	176.8	81.1	52.2%
1.6	8,281	842	89.8%	123.4	33.0	73.3%
3.2	8,281	842	89.3%	119.0	30.0	74.8%
6.4	8,281	410	95.1%	117.8	20.0	83.0%
12.8	8,281	362	95.6%	116.2	19.0	83.7%

We also compared the efficiency of H2DMR and H2D in an empty environment of 50×50 m. To analyze the impact of the map structure in the results we used several octrees with a maximum resolution of 0.1 m, varying the minimum one up to 25.6 m. Table II details these results —columns "MR" and "SR" are for H2DMR and H2D, respectively. Since H2DMR adapts the resolution to that of the octree —with an upper bound of f^+—, the iterations and runtime decrease as the minimum resolution of the octree —C_+— increases. The optimistic nature H2DMR is not affected due to the multi-resolution grid. In fact, it has an increased connectivity —all points between adjacent cells are linked— and the estimations are on average a 4% lower than those of H2D.

C. Real experiments

In Fig. 5 we show an experiment done with the robot Pepper (Softbank Robotics). First, we obtained the dynamics model from $512s$ of navigation data, and then a set of 236 actions with lengths between 0.2 and 1 m. The maneuvering of the robot was that of an odometric motion model, with a maximum linear speed of 0.5 m/s.

We used a SLAM algorithm [20] to build the map, and a Monte Carlo method [21] to localize the robot. The accuracy of the localization decreases when the distance to the obstacles exceeds the range of the laser sensor —1.5 m for Pepper—, e.g. near the start or after waypoint 2. This

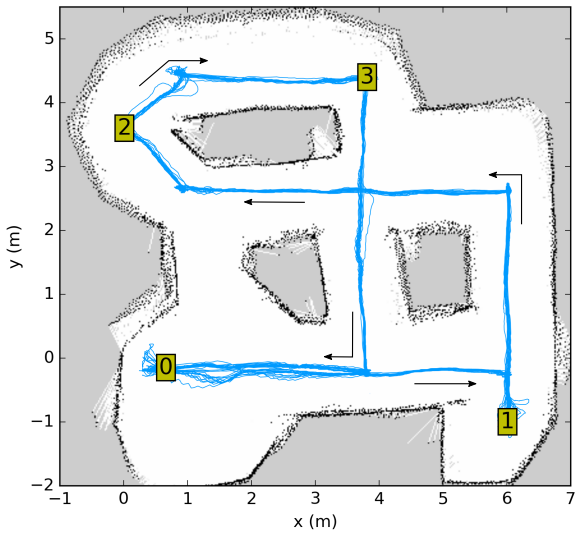


Fig. 5: Executions in a real environment with the robot Pepper —the robot starts in waypoint 0, then it goes to 1, 2, 3, and back to 0. Arrows indicate the direction of motion.

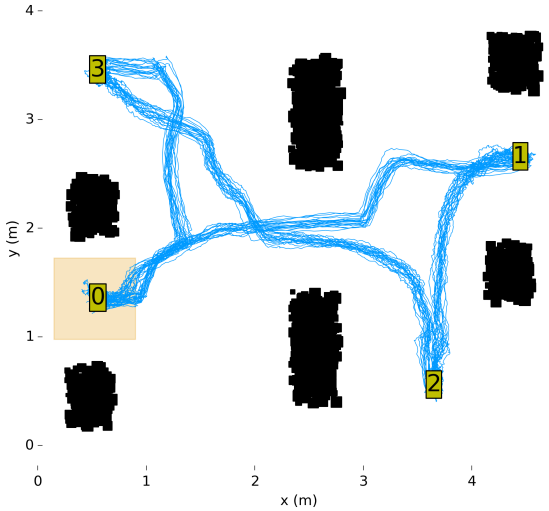


Fig. 6: Executions in a real, cluttered environment with a UAV —starting in waypoint 0, then going to 1, 2, 3, and back to 0. The UAV shape is in orange.

causes deviations that are corrected by the LQG controller. Despite this, the planned path was safe, and no collisions were reported in 20 executions. Finally, the efficiency of the planner allowed the robot to operate without noticeable delays. On average, the plans between consecutive waypoints were obtained in less than 5 seconds and 1,000 iterations due to our graduated fidelity approach and the anytime search capabilities of AD*. Also, H2DMR was obtained in 0.2 s —941 iterations. The total cost of the path was 115 s.

The experiment of Fig. 6 was done with a UAV (Asctec Pelican) localized with a motion capture system. The dynamics model was obtained from $1,458$ s of flight data, which resulted in a set of $3,316$ motion primitives with lengths between 0.2 and 0.8 m. These motions allowed the UAV to move forward and laterally with a maximum linear speed of 0.7 m/s. The maximum angular speed was $30°/s$. The environment was cluttered with obstacles, and the only way to navigate between the waypoints was to cross the narrow spaces between them. In fact, in some points there was only 11 cm of distance between the UAV and the obstacles. Despite this, the planned path was safe, and in 20 executions no collisions were reported. H2DMR was obtained in 0.1 s —216 iterations, and on average the plans between waypoints were obtained in less than 8 seconds and 2,500 iterations. The total cost of the path was 86.9 s.

VII. Conclusions

We have presented a state lattice based motion planner which manages motion and sensing uncertainty. The planner reliably obtains the probability of collision considering the real robot shape and the uncertainty in heading, and it is based on a novel graduated fidelity approach, which adapts to the obstacles in the map, and on a muli-resolution heuristic. We have reported results for simulated and real environments, with different motion models and robot shapes,

proving the reliability and efficiency of the proposed method. Future research will extend the validation of the proposed algorithms in practical applications demanding autonomous navigation for UAVs.

REFERENCES

[1] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006.

[2] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[3] L. E. Kavraki, P. Svestka, J. . Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, Aug 1996.

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

[5] M. Pivtoraiko and A. Kelly, "Efficient constrained path planning via search in state lattices," in *8th International Symposium on Artificial Intelligence, Robotics and Automation (I-SAIRAS)*, 2005.

[6] ——, "Differentially constrained motion replanning using state lattices with graduated fidelity," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008, pp. 2611–2616.

[7] M. Likhachev and D. Ferguson, "Planning Long Dynamically Feasible Maneuvers for Autonomous Vehicles," *The International Journal of Robotics Research*, vol. 28, no. 8, pp. 933–945, Jun. 2009.

[8] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *Journal of Field Robotics*, vol. 26, no. 3, pp. 308–333, 2009.

[9] R. Knepper and A. Kelly, "High Performance State Lattice Planning Using Heuristic Look-Up Tables," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3375–3380, 2006.

[10] R. Alterovitz, T. Siméon, and K. Goldberg, "The stochastic motion roadmap: A sampling framework for planning with Markov motion uncertainty," in *Robotics: Science and Systems*, 2007, pp. 246–253.

[11] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1, pp. 99–134, 1998.

[12] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of Markov decision processes," *Mathematics of operations research*, vol. 12, no. 3, pp. 441–450, 1987.

[13] J. Van den Berg, S. Patil, and R. Alterovitz, "Motion planning under uncertainty using iterative local optimization in belief space," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1263–1278, 2012.

[14] J. Van den Berg, P. Abbeel, and K. Goldberg, "LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 895–913, 2011.

[15] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 723–730.

[16] T. M. Howard and A. Kelly, "Optimal rough terrain trajectory generation for wheeled mobile robots," *The International Journal of Robotics Research*, vol. 26, no. 2, pp. 141–166, 2007.

[17] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, "Anytime dynamic A*: An anytime, replanning algorithm," in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2005, pp. 262–271.

[18] D. Bertsekas, *Dynamic programming and optimal control*. Athena scientific Belmont, MA, 1995, vol. 1.

[19] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte, "A new approach for filtering nonlinear systems," in *Proceedings of the American Control Conference*, vol. 3. IEEE, 1995, pp. 1628–1632.

[20] G. Grisettiyz, C. Stachniss, and W. Burgard, "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling," in *Proceedings of the 2005 IEEE international conference on robotics and automation*. IEEE, 2005, pp. 2432–2437.

[21] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: Efficient position estimation for mobile robots," *AAAI/IAAI*, vol. 1999, no. 343-349, pp. 2–2, 1999.