Autonomous navigation for UAVs managing motion and sensing uncertainty

Adrián González-Sieira, Daniel Cores, Manuel Mucientes and Alberto Bugarín

Version: accepted article

Available online at www.sciencedirect.com

ScienceDirect

Robotics and Autonomous Systems (2020) 1–14

Journal Logo

Autonomous navigation for UAVs managing motion and sensing uncertainty

Adrián González-Sieira[a,*], Daniel Cores[a], Manuel Mucientes[a], Alberto Bugarín[a]

[a] *Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS), Universidade de Santiago de Compostela, Santiago de Compostela, Spain*

Abstract

We present a motion planner for the autonomous navigation of UAVs that manages motion and sensing uncertainty at planning time. By doing so, optimal paths in terms of probability of collision, traversal time and uncertainty are obtained. Moreover, our approach takes into account the real dimensions of the UAV in order to reliably estimate the probability of collision from the predicted uncertainty. The motion planner relies on a graduated fidelity state lattice and a novel multi-resolution heuristic which adapt to the obstacles in the map. This allows managing the uncertainty at planning time and yet obtaining solutions fast enough to control the UAV in real time. Experimental results show the reliability and the efficiency of our approach in different real environments and with different motion models. Finally, we also report planning results for the reconstruction of 3D scenarios, showing that with our approach the UAV can obtain a precise 3D model autonomously.

Keywords: autonomous navigation, motion planning, motion uncertainty, UAVs, scene reconstruction

1. Introduction

The popularity of Unmanned Aerial Vehicles (UAVs) has significantly increased in the last years. Currently, most of the times a human operator supervises and usually controls the UAV during the flight. However, the benefit of using an autonomous navigation system is clear when it comes to help the operator to accomplish tasks that can be automated. In this sense, a motion planning approach that minimizes the probability of collision of the paths can guarantee a safe navigation, leaving the operator more time to focus on the goals or other key aspects of the mission. Examples of applications that could benefit from this are: exploration [1], surveillance [2], tracking of mobile targets [3] or traffic monitoring [4, 5]. However, the interest is even clearer in other contexts in which human intervention is hazardous or too expensive, like in infrastructure inspection [6] or the reconstruction of 3D scenarios [7].

In order to reliably estimate the safety of the planned paths, it is essential to take into account the dynamics of the UAV. This allows obtaining plans which comply with its kinematic constraints. However, it is equally important to consider that, under certain circumstances, a significant amount of motion uncertainty may arise during the flight. This is due to the imprecisions in the UAV motions and the noisy or incomplete measurements of the on-board sensors, which make the uncertainty to be different for each path. Estimating it at planning time allows selecting the best path according to its safety. This would allow obtaining solutions adapted to the uncertainty conditions —e.g. the accuracy of the localization measurements—, this way fulfilling the flight safety requirements.

Planning efficiency is also relevant to allow operating the UAV without noticeable delays. Although there are motion planning approaches based both on stochastic and deterministic sampling strategies, the latter —e.g. the state lattices [8]— allow distributing the sampled states in a regular manner, which results in a very efficient representation of the state and action spaces. Moreover, their structure resembles a graph in which the vertices and the edges are the discrete states and the motions connecting them, respectively. Thus, optimal paths between states can be found with an informed search algorithm, using heuristics to improve the search efficiency. Despite this, managing motion uncertainty at planning time can be computationally expensive, so fluent autonomous navigation can only be achieved by using strategies to improve

*Corresponding author.

Email addresses: adrian.gonzalez@usc.es (Adrián González-Sieira), daniel.cores@usc.es (Daniel Cores), manuel.mucientes@usc.es (Manuel Mucientes), alberto.bugarin.diz@usc.es (Alberto Bugarín)

the planning efficiency. On the one hand, multi-resolution maps can be helpful to get a better representation of the environment, since they adapt the resolution to the obstacle cluttering, a strategy which is also applicable to obtain the heuristics faster. In this sense, octree based maps [9, 10] can deal with large environments very efficiently. On the other hand, the efficiency of state lattice based motion planners can be boosted by reducing the fidelity —the resolution of the sampled states—, although this cannot be done without sacrificing the optimality and feasibility of the planner —the capacity to obtain valid solutions given the constraints. Notwithstanding, graduated fidelity lattices [11] can manage the trade-off between planning efficiency and performance, since they vary the resolution of the sampled states in different areas of the map. This way the uncertainty can be managed while keeping reasonable runtimes, making this strategy suitable for navigating autonomously in a safe manner.

The contributions of this paper are: (i) a motion planning algorithm adapted to the autonomous navigation of UAVs which manages the motion and sensing uncertainty with the goal of obtaining optimal solutions in terms of probability of collision and traversal time; and (ii) the integration of the proposed motion planner in a real system, which was applied to the autonomous reconstruction of 3D scenes with a RGB-D camera. The uncertainty is managed at planning time, in such a way that the probability of collision of each path is estimated before the UAV starts moving. Thus, the best path in terms of probability of collision and traversal time is obtained, so that the UAV can navigate autonomously in a safe manner, even in cluttered environments.

The motion planner is based on the algorithms detailed in [12], which in this paper are presented as a whole system for planning and navigation of UAVs that includes flight control and state estimation. In this regard, this work does not consist in a mere 3D extension of the existing planner, which would result in unsuitable planning times that would prevent its integration in real systems. Instead, an adaptation of the underlying algorithms that addresses the increased complexity due to the higher dimensionality of the problem is presented. In this context, we propose a deterministic sampling method to obtain the probability of collision reliably, taking into account the real dimensions of the UAV and also the uncertainty in heading. Moreover, to improve the efficiency of the motion planner, we propose H3DMR, a novel heuristic which takes advantage of the multi-resolution map to efficiently estimate the cost to the goal in 3D environments. This heuristic, together with the graduated fidelity lattice of the motion planning algorithm, allows obtaining solutions fast enough to operate the UAV in real time.

2. Related work

As mentioned before, the efficiency of the motion planning algorithms is essential for navigating autonomously. In order to achieve this, some techniques obtain collision free routes disregarding the dynamics model of the UAV, and then rely on a PD [13] or PID [14] controller to track the paths focusing on the stability of the flight. Other approaches use these paths as initialization for a low level planner which obtains a similar solution that observes the kinematic restrictions, like [15], which uses Potential Fields to obtain the velocity commands that would drive the robot through free space. Similarly, in [16] they presented an architecture based on a high level planner, which obtains the waypoints for navigation; and a low level planner, which deals with trajectory tracking and obstacle avoidance. However, the main issue shared by these approaches is that they cannot guarantee the feasibility of the global route once the UAV starts to move, since the initial solution disregards the kinematic restrictions. Despite this, approaches of this kind reported good results in particular domains that rely on local planning solutions, like [17].

This is addressed obtaining collision free paths taking into account the dynamics model, something that can be achieved using different techniques. In this regard, the proposal of [18] uses Potential Fields and deals with the local minimums. However, this method relies on computing the control commands in real time, which might be not possible in large environments due to scalability issues. The proposal of [19] uses the Fast Marching Square method to produce smooth paths for navigating at a fixed altitude. Despite not being explicitly considered, the kinematic restrictions of the UAV can be fulfilled if the parameters of the algorithm are properly tuned.

Sampling based methods have proved successful for dealing with global motion planning under kinematic restrictions, achieving good results in high dimensional spaces. There have been efforts for applying these algorithms to the autonomous navigation of UAVs, using both stochastic [20, 21] and deterministic sampling methods [22, 23]. In [20] they use Rapidly-exploring Random Trees (RRT) [24], to obtain a global plan very efficiently. However, the paths obtained by RRT usually have smoothness issues and require post-processing to avoid undesirable maneuvers during flight, especially when the number of samples is low. Informed RRT* [25], a variant of RRT, was used in [21] in combination with a local planner which performs trajectory optimization to deal with these smoothness issues. This local planner is constantly updating the path to deal with unseen obstacles, which allows navigating in unknown environments. Despite the good results reported, this approach uses a smooth cost function for collision avoidance which does not scale well with the size of the environment.

Among the approaches based on deterministic sampling, state lattices [26] are noteworthy for their good results when combined with graph search algorithms, like Anytime Dynamic A* [26]. In [22] and [23] they reported results for multi-rotor autonomous navigation in partially known environments. In addition, the latter proposal deals with robots with non-circular shapes, which is very interesting for navigating in cluttered environments. However,

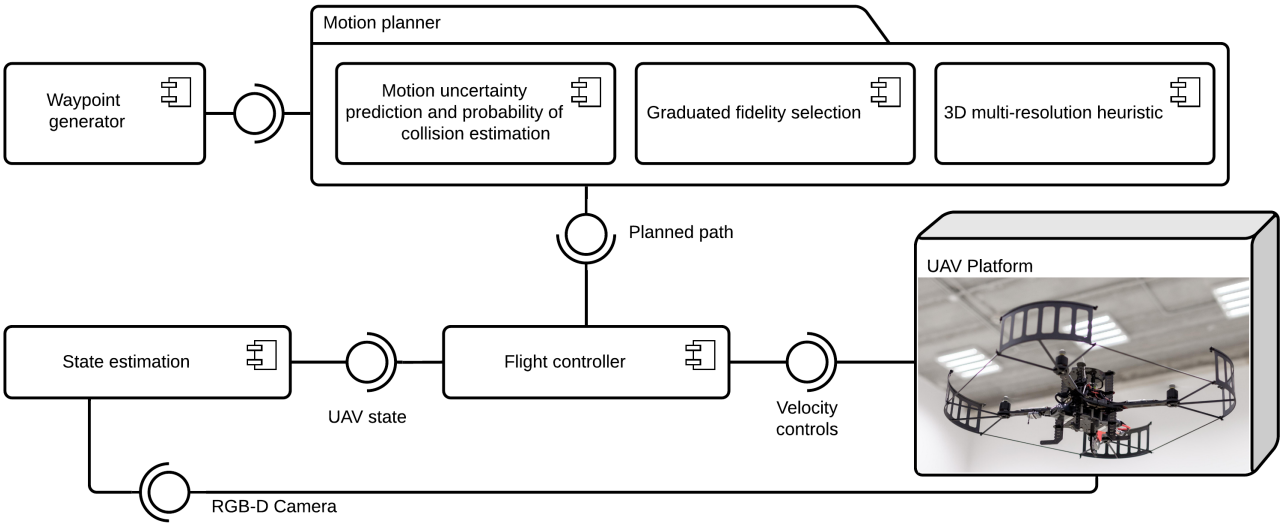


Figure 1. Architecture of the autonomous navigation system. The modularity of the different software components allows running the system with different configurations.

all the approaches mentioned above rely on feedback controllers to track the planned path accurately, a strategy which does not guarantee that the planned paths will be collision free under all circumstances.

In regards to motion planning under uncertainty, some authors pointed out the suitability of taking it into account for UAV navigation [27] in order to address the motion imprecisions or the sensing inaccuracies that might affect the reliability of the solutions. In this context, the state of the art for dealing with the uncertainty at planning time are the approaches in [28] and [29]. Both methods work in a similar manner, which consists in predicting the uncertainty throughout all the candidate paths —considering the inaccuracies in the controls and the observations— that the robot would have in execution time with the goal of retrieving the one that maximizes a safety criterion. The former achieves this in such a way which minimizes the probability of collision using RRT, although due to this method not being asymptotically optimal [30] there is no guarantee that the path with the lowest cost will be retrieved. The latter addresses this issue using RRT* instead, albeit in this case with the goal of obtaining the path which minimizes the amount of uncertainty.

In spite of the suitability of considering the uncertainty at planning time, the high complexity of this problem and the limited computing power of the on-board computers have cause very high planning times, which make it impossible to integrate these algorithms in real systems. Hence, most prior research in the field of UAV navigation has so far focused on algorithms capable of computing fast solutions [13, 15, 16, 21], relying on multi-level architectures that deal with the risky situations in execution time, e.g. how to approach the obstacles or when a blocking situation occurs. Thus, the initial plan is assumed to be collision free and it is locally modified in execution time to keep a security distance with the obstacles. While these approaches might still be key to deal with certain situations, they cannot guarantee neither the feasibility nor the optimality of the paths before the UAV starts moving. Instead, our proposal manages the uncertainty at planning time and selects the best path in terms of probability of collision and traversal time. To achieve this while retaining reasonable planning times, our proposal relies on a state lattice with graduated fidelity which is able to deal with the overhead caused by taking into account the uncertainty. Moreover, in this work we present a strategy for reliably estimating the probability of collision of the candidate solutions taking into account the real dimensions of the UAV, as well as a multi-resolution heuristic which adapts to the obstacles in the map. The efficiency and reliability of our approach makes it suitable for autonomous navigation purposes.

3. System overview

Our system has a modular and extensible architecture, detailed in Fig. 1, in which each component works independently and can be developed or replaced without affecting the rest. Concretely, it is formed by several independent software modules which work in parallel and communicate between them: the waypoint generator, the motion planner, the UAV state estimation and the flight controller. For the implementation we used the widely extended Robotic Operating System —ROS— framework

[31], which facilitates the communications between the different software modules and the use of different hardware configurations. Moreover, the modular and flexible architecture allows adapting the different components without affecting the rest.

The state estimation module combines the UAV pose with the information of the inertial unit. The pose can be obtained from an external source, like a GPS, or be estimated by other means, such as a visual odometry algorithm which uses an on-board camera. Any localization method can be used with our system, provided that its error is bounded in time and it can be represented by a Gaussian distribution. Independently of the source, an Extended Kalman Filter combines the localization and the inertial unit measurements to improve the reliability of the estimated state. Moreover, this enhances the robustness of the system when dealing with inaccuracies and partial information.

The controller tracks the planned paths, minimizing the deviations that might occur during the flight. To achieve this, it adjusts the velocity commands in real time according to the estimated state of the UAV. The design of our system is not tied to a specific vehicle. For this reason, our approach assumes that a low level controller calculates the suitable throttle for each motor given the velocity commands.

3.1. Motion planner

The input for planning is the set of waypoints that the UAV has to visit. These waypoints can be obtained in different manners depending on the nature of the mission: for autonomous navigation purposes only the current location and the goal are needed, while for scene reconstruction a set of waypoints allowing to collect data without leaving unseen areas is required.

The motion planner obtains the path that minimizes the probability of collision and the flight time between consecutive waypoints. The result is the list of velocity commands that the robot has to execute to follow the plan, which are sent to the flight controller. Our approach: (i) manages the motion and sensing uncertainty at planning time using the method of [29], predicting the probability distributions of the robot being in each state of the path during its execution; (ii) estimates the probability of collision for each path given the predicted uncertainty; (iii) uses a state lattice and graduated fidelity strategy which adapts to the obstacles in the map.

The lattice states are connected by a set of motion primitives, that are obtained from the dynamics model of the UAV, and that can be computed offline for the sake of efficiency. This way of representing the state space can be seen like a graph, so the optimal path between any pair of lattice states can be found with a search algorithm. This path, due to the construction of the state lattice, is formed by the set of feasible actions connecting them. As search algorithm we used Anytime Dynamic A* (AD*) [26, 32], which allows obtaining sub-optimal solutions faster than the optimal one. The parameter ϵ acts as boundary for the cost of the sub-optimal solutions, which are refined iteratively with the information previously calculated. ϵ can be initially set to any sufficiently high value, which depends on the desired upper bound to the cost of the sub-optimal solutions, in order to compute the first path quickly.

Algorithm 1 Main operations of the motion planner

Require: x^0 and x^G, initial and goal states, and ϵ_0

1: **function** MAIN (x^0, x^G, $\epsilon = \epsilon_0$)
2: INITIALIZEH3DMR(x^0, x^G) ▷ See Alg. 2
3: **while** $\epsilon >= 1$ **do**
4: $OPEN = \{x^0\}$
5: **repeat**
6: $x^a = \arg\min_{x \in OPEN}(c_x + \epsilon \cdot h_x)$
7: **for all** $x^b \in$ SUCCESSORS(x^a) **do**
8: h_{x^b} = HEURISTIC(x^b) ▷ See Alg. 2
9: $c_{x^b} = c_{x^a} +$ COST(x^a, x^b) ▷ See Alg. 3
10: $OPEN = OPEN \cup \{x^b\}$
11: $OPEN = OPEN - \{x^a\}$
12: **until** $x^a = x^G$
13: publish $path(x^0, x^a)$ and decrease ϵ
14: **return**

While this planning approach is based on [12], this prior work was proposed for planning under uncertainty for 2D mobile robots. Its applicability to 3D systems is not straightforward due to the increased computational complexity caused by the higher dimensionality of the problem. In this regard, Sec. 4 details: (i) a novel multi-resolution 3D heuristic, and (ii) a new algorithm for the estimation of the probability of collision of the UAV. These modifications allow applying the motion planning algorithm to the autonomous navigation of UAVs while retaining reasonable planning times.

Algorithm 1 outlines the main steps of the motion planner —a more detailed explanation can be found in [12]. AD* explores the lattice states in an iterative manner, for which a list containing the states yet to be explored is maintained —*OPEN*. The states in this list are ordered according to their current cost from the start and their estimated cost to the goal, which is given by the heuristics.

Our motion planner uses two different heuristics, combined by their maximum, to estimate the cost between each state and the goal. The first one, FSH (Free Space Heuristic) [33], copes with the kinematic restrictions of the UAV considering free space, while the second one is a low-dimensional heuristic which copes with the obstacles in the map, H3DMR. Since this heuristic depends on the map, it has to be initialized every time the planner is run —Alg. 1:2—, so its efficiency is key for obtaining low runtimes. This is achieved using multi-resolution techniques which adapt the resolution of this heuristic to the obstacles in the map and reduce its obtention time to the lowest possible, as detailed in Sec. 4.1.

The inputs of the search algorithm are the initial state of the UAV, x^0, and the goal, x^G. AD* is initialized processing x^0 in the first place —Alg. 1:4. Then the algorithm processes other states iteratively until the goal is found. Firstly, the state x^a for which the sum of its cost and heuristic —c_x and h_x, respectively— is minimum, is pulled from *OPEN* —Alg. 1:6. As the heuristic solely informs about the traversal time, only this term is considered for combining c_x and h_x, and the rest of the information is taken unchanged from c_x. Secondly, in Alg. 1:7 the successors of x^a are obtained from the motion primitives, using the graduated fidelity strategy that was described in [12]. For each successor, x^b, its heuristic and cost are calculated —Alg. 1:8-9—, and the state is inserted in *OPEN* to be explored in subsequent iterations. The optimal solution for the current value of ϵ is available after reaching the goal, x^G, after which the value of ϵ is decreased —Alg. 1:13. This process is repeated to refine the current plan until the optimal one is found —this is, $\epsilon = 1$. Although there is no rule for decreasing the value of ϵ, better results are achieved if it is done at small steps [34].

4. Motion planning for UAVs

In this section we describe the contributions of this paper, which adapt the motion planning algorithm introduced in [12] to the autonomous navigation of UAVs. In Sec. 4.1 we introduce a novel multi-resolution heuristic which copes with the obstacles in the map, and in Sec. 4.2 we describe a method to estimate the probability of collision of the UAV in a reliable manner taking into account its real dimensions and the uncertainty in heading.

4.1. Multi-resolution 3D heuristic

H3DMR is a multi-resolution heuristic for motion planning in 3D environments. It estimates the cost to the goal taking into account the obstacles in the map, for which a 3D low-dimensional grid containing the estimated cost between each position in the map and the goal is built. To increase the efficiency of the heuristic and improve its scalability in large environments, the resolution of the grid adapts to the obstacles in the map. Since the precision of the heuristics affects the efficiency and the performance of the motion planner, the resolution of the H3DMR also takes into account the fidelity of the lattice used for planning. Moreover, as already mentioned in Sec. 3, this heuristic is combined with FSH, which improves the precision of the estimations due to coping with the kinematic restrictions of the UAV.

Alg. 2 details how H3DMR is calculated. The inputs are the beginning position and the goal state of the UAV, q^0 and q^G; and the maximum fidelity of the state lattice, f^+. The grid is obtained applying the Dijkstra's algorithm, starting in q^G. Generating the grid backwards allows storing the estimated cost between every position and the goal, which is easily queried later and used as heuristic

Algorithm 2 H3DMR

Require: f^+, highest fidelity of the state lattice
Require: q^0 and q^G, initial and goal positions of the UAV
1: **function** INITIALIZEH3DMR(q^0, q^G)
2: $c_{q^G} = 0$; $c_{q^0} = \infty$
3: $q = q^G$
4: **while** $c_q > 2 \cdot c_{q^0}$ **do**
5: $q = \arg\min_{q \in OPEN}(c_q)$
6: $OPEN = OPEN - \{q\}$
7: $\kappa = cell(q)$
8: **for all** $\kappa' \in adjacent(\kappa)$ **do**
9: **if** $size(\kappa') > f^+$ **then**
10: INSERTSUBCELLS(q, κ')
11: **else**
12: $\kappa'' = adjust(\kappa', f^+)$
13: $q' = position(\kappa'')$
14: **if** $collisionBetween(q, q')$ **then**
15: INSERTSUBCELLS(q, κ'')
16: **else**
17: $c(q') = c_q + \|q' - q\|$
18: $OPEN = OPEN \cup \{q'\}$
19: **function** INSERTSUBCELLS(q, κ)
20: **for all** $k'' \in subcells(\kappa')$ **do**
21: **if** κ'' not adjacent to κ **then**
22: **continue**
23: $q' = position(\kappa'')$
24: $c_{q'} = c_q + \|q' - q\|$
25: $OPEN = OPEN \cup \{q'\}$
26: **function** H3DMR(x)
27: $\kappa = cell(x)$
28: $K = \{\kappa \cup adjacent(\kappa)\}$
29: $Q = \{q$ inside $K \mid q$ position of the grid $\}$
30: **return** $\arg\min_{q \in Q}(\|x - q\| + c_q)$
31: **function** HEURISTIC(x)
32: **return** $\max(\text{H3DMR}(x), \text{FSH}(x))$

for planning. As stopping condition we used the double of the cost between the starting position, q^0, and the goal —Alg. 2:4. This is done for efficiency purposes, since the areas with higher costs are unlikely to be interesting for planning.

The algorithm stores all the positions to be explored in the OPEN queue, and iteratively it selects the position with the lowest cost from q^G —Alg. 2:5-6— to generate its neighbors, which are processed in subsequent iterations. The neighbors of a position q are obtained from the free space cells adjacent to κ, the one containing q —Alg. 2:7. This way they go in accordance with the resolution of the map. Moreover, with the goal of adapting the resolution of the grid to the maximum fidelity of the state lattice, f^+, each cell κ' adjacent to κ is processed differently depending on its size. Fig. 2 represents the two situations that are considered. On the one hand, those cells that are larger than f^+, like the one labeled as "A" in the image, are split into subcells of equal size —Alg. 2:9-10— to avoid

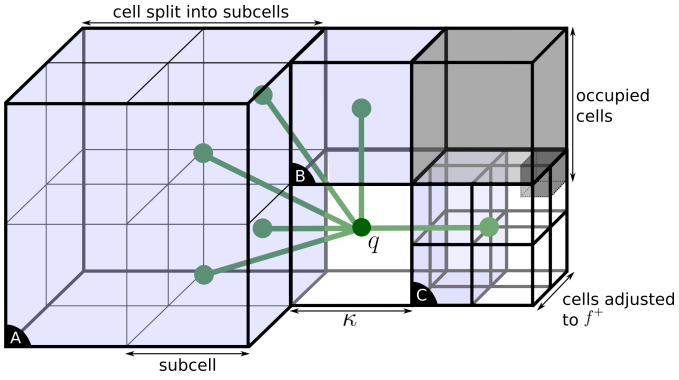


Figure 2. Neighbors of a point q of the multi-resolution grid, in green. κ is the cell which contains q, and the free adjacent cells used for obtaining the neighbor points are highlighted in blue. The large cell labeled as "A" is split into equal subcells, while "B" is used as it is, since its resolution is already adjusted to f^+. The cells on the right are smaller due to the obstacles —the occupied cells are colored in gray—, so they are adjusted to a lower resolution and represented by the larger cell containing them, labeled as "C".

obtaining neighbors at distances much greater than f^+, thus obtaining more precise estimations. For the sake of efficiency, only those subcells of κ' that are adjacent to κ are considered.

On the other hand, the resolution of the cells is limited to f^+, since obtaining neighbors at lower distances would affect the efficiency of the heuristic. Cells smaller than f^+ are not considered for the purpose of generating the grid. Instead, H3DMR adjusts the resolution according to f^+, like in cell "C" of Fig. 2, so that the neighbor point q' is obtained from the larger cells —Alg. 2:12-13. Although the general rule is to decrease the resolution whenever possible, an exception is made when there are obstacles between q and q'. In this case, faced the impossibility of using the occupied cell for generating neighbors of q, its subcells are considered instead —Alg. 2:14-15. As some of these subcells might be free space, this way the H3DMR copes with the obstacles in a more precise manner and the optimistic nature of the heuristic is retained.

Finally, each neighbor position q' is inserted into the *OPEN* queue to be explored by the search algorithm in subsequent iterations, using the Euclidean distance as the cost between q and q' —Alg. 2:17-18.

The multi-resolution grid obtained by H3DMR allows estimating the cost between each point of the map and the goal, which is used by the motion planning algorithm as heuristic. A direct match between the states of the graduated fidelity lattice and the positions of the grid may exist, but this is not guaranteed. For this reason, we obtain the heuristic of a state x from the position of H3DMR for which the estimated cost to the goal is minimal, considering those which are inside the cell containing x and the adjacent ones —Alg. 2:26-30.

The strategy for obtaining H3DMR takes into account the fidelity of the lattice used for planning and also the obstacles in the map. The cluttered areas are represented in the map with cells of lower sizes, for which the resolution of the H3DMR grid increases. Conversely, the free space areas are represented with larger cells, resulting in lower resolutions for the H3DMR grid. This allows estimating the cost to the goal very efficiently. Despite decreasing the resolution in some areas of the map, the estimations are precise due to taking into account the fidelity of the motion primitives.

4.2. Probability of collision estimation

The candidate paths are evaluated in terms of probability of collision, traversal time and the uncertainty at the final state, following the procedure detailed in Alg. 3. These elements are compared hierarchically, so the motion planner minimizes, in the first place, the probability of collision. Among the safe paths, it gets the one with the minimal traversal time, and if several alternatives exist, it chooses the one with the lowest uncertainty at the goal. While the traversal time of the path is given by the motion primitives, estimating the probability of collision —with the purpose of obtaining the best plan in terms of safety— requires to predict the probability distributions of the UAV state during the execution of that path —Alg. 3:2.

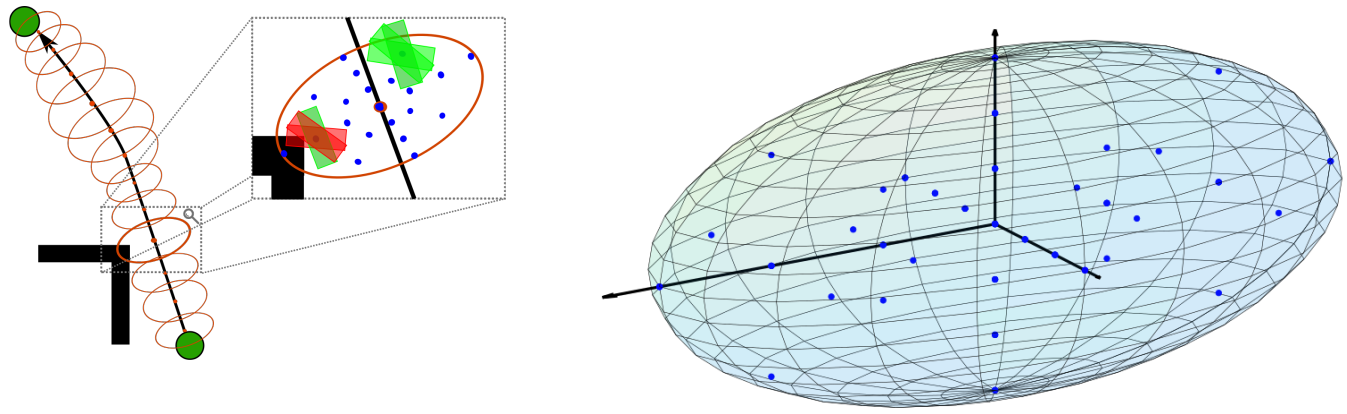
Estimating the probability of collision reliably is crucial for the robustness of the motion planner. To achieve this, from each predicted distribution $x_t \sim \mathcal{N}(\bar{x}_t, \Sigma_t)$ we obtain a set of samples —Alg. 3:4—, which we use to check collisions with the surrounding obstacles, as shown in Fig. 3(a). The strategy of sampling the distributions allows considering the uncertainty in heading, in such a way that each sample represents a different pose of the UAV. In this way, the real dimensions of the UAV are taken into account centering the real shape in each sampled pose for the purpose of checking collisions with the obstacles in the map. Then, the probability of collision is estimated calculating the ratio between the weights of the colliding samples and the total —Alg. 3:6-7. The weight of each sample $x^s \in X^s$ is obtained according to its likelihood:

$$w_{x^s} = 2\pi^{-\frac{n}{2}} |\Sigma_t|^{-\frac{1}{2}} \exp\{-\frac{1}{2}(x^s - \bar{x}_t)' \Sigma_t^{-1} (x^s - \bar{x}_t)\} \quad (1)$$

and the cost for the entire path, $c^{a:b}$, is then obtained from the individual estimations —Alg. 3:7. Although by doing so we assume that they are independent, this is reasonable for practical purposes.

Despite the benefits of sampling the distributions, the high number of stochastic samples that would be needed to represent the distributions may severely impact the efficiency of the motion planner. For this reason, our approach uses a deterministic sampling strategy that represents the distributions with a limited number of samples, making it possible to use it in a real setup.

[1] PREDICTUNCERTAINTY is explained in detail in [12]



(a) Estimation of the probability of collision of a path taking into account the real shape and the uncertainty in heading —2D view.

(b) The samples —in blue— are obtained with a deterministic strategy which distributes them regularly according to the maximum direction of spread of the distributions —3D view.

Figure 3. Probability of collision estimation.

Algorithm 3 Path evaluation

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

1: **function** COST(x^a, x^b)
2: $P^{a:b}$ = PREDICTUNCERTAINTY(x^a, x^b) ▷ See[1]
3: $c^{a:b} = 0$
4: **for all** $x_t \sim \mathcal{N}(\bar{x}_t, \Sigma_t) \in P^{a:b}$ **do**
5: X^s = SAMPLE(x_t) ▷ See Alg. 4
6: $w_c = \sum_{\{x^s \in X^s \mid collides(x^s)\}}(w_{x^s})$ ▷ See Eq. 1
7: $c^{a:b} = c^{a:b} - \log(1 - w_c / \sum_{\{x^s \in X^s\}}(w_{x^s}))$
8: **return** $\left[c^{a:b}, time(x^a, x^b), tr(\Sigma^b)\right]$ ▷ $x^b \sim (\bar{x}^b, \Sigma^b)$

Our strategy relies on, for each one of the predicted distributions, obtaining a set of samples arranged in a regular manner within the limits of a confidence region. The confidence region represents the volume of the space in which the UAV might be, with a certain probability, during the execution of the path. Arranging the set of samples following a regular pattern, as shown in Fig. 3(b), allows representing the distributions in a way which is suitable for collision check purposes, as well as retaining the deterministic nature of the planner.

Given a distribution for the state of the UAV, $x_t \sim \mathcal{N}(\bar{x}_t, \Sigma_t)$, the eigenvectors and eigenvalues of Σ_t represent the directions and magnitude of maximum spread of x_t in each dimension, which allow distributing the samples accordingly.

The volume of the confidence region depends on the probability that a random sample of x_t would fall in it. In our approach, the motion uncertainty is represented with multivariate Gaussian distributions, for which the confidence region consists in those values of x satisfying:

$$(x - \bar{x}_t)' \Sigma_t^{-1} (x - \bar{x}_t) \leq \chi_n^2(p) \tag{2}$$

where $\chi_n^2(p)$ is the quantile function for a Chi-Square distribution with n degrees of freedom, and p is the probability that the confidence region encompasses the state of the UAV. In the 3D space, this confidence region can be represented by an ellipsoid, whose implicit equation —without considering any rotation given by the eigenvectors— is:

$$\sum_i^n \frac{x_i^2}{\lambda_i^2} = \chi_n^2(p) \tag{3}$$

where λ_i is the i-th eigenvalue of Σ_t. From Eq. 3 follows that the length of each semi-axis of this ellipsoid is:

$$a_i = \sqrt{\chi_n^2(p) \cdot \lambda_i} \tag{4}$$

Our goal is to obtain a minimal set of samples which represent the distribution well enough for estimating the probability of collision reliably. We use the a_i values from Eq. 4 to obtain samples at l different levels, all equally spaced. While most of them are in the directions of maximum spread due to their likelihood, we also sample in the diagonals to avoid leaving gaps which might be relevant for collision check purposes. This is achieved by combining the direction of maximum spread in different dimensions, given by the corresponding eigenvectors. However, in order to limit the number of samples and avoid obtaining poses with very low likelihoods, we only combine two dimensions at the same time.

Alg. 4 details the sampling process for a distribution $x_t \sim \mathcal{N}(\bar{x}_t, \Sigma_t)$. First, in Alg. 4:2, we factorize the covariance matrix into a canonical form, obtaining its representation in terms of eigenvectors and eigenvalues —U_t and V_t, respectively. U_t is a $n \times n$ matrix whose i-th column —$U_{t[i]}$— is the i-th eigenvector, and V_t is a diagonal matrix containing the eigenvalues —$\lambda_i, ..., \lambda_n$. Then, in Alg. 4:3 we obtain the quantile of the chi-square distribution which satisfies Eq. 2, and we initialize the set of samples, X^S, with the mean of the distribution —Alg. 4:4.

X^S is populated iterating over the different dimensions of x_t. We obtain samples aligned with the directions of

Algorithm 4 Regular sampling of the distributions

Require: $x_t \sim \mathcal{N}(\bar{x_t}, \Sigma_t)$
Require: p, probability of the confidence region
1: **function** SAMPLE(x_t)
2: $U_t V_t U_t^{-1} = \Sigma_t$ ▷ Eigendecomposition of Σ_t
3: $\chi_p = \chi_n^2(p)$ ▷ Quantile of the chi-square
4: $X_s = \{\bar{x_t}\}$
5: **for** l in $1, ..., L$ **do**
6: **for** i in $1, ..., n$ **do**
7: $a_i = \sqrt{\chi_p \cdot \lambda_i}$
8: $d_i = l/L \cdot a_i$
9: $x_s^+ = \bar{x_t} + d_i \cdot U_{t[i]}$
10: $x_s^- = \bar{x_t} - d_i \cdot U_{t[i]}$
11: $X_s = X_s \cup \{x_s^+,\ x_s^-\}$
12: **for** j in $1, ..., n; j \neq i$ **do**
13: $a_j = \sqrt{\chi_p \cdot \lambda_j}$
14: $d_j = l/L \cdot a_j$
15: $X_s = X_s \cup \{x_s^+ + d_j \cdot U_{t[j]},\ x_s^+ - d_j \cdot U_{t[j]}\}$
16: $X_s = X_s \cup \{x_s^- + d_j \cdot U_{t[j]},\ x_s^- - d_j \cdot U_{t[j]}\}$
17: **return** X^s

maximum spread, given by the corresponding eigenvector, $U_{t[i]}$ —Alg. 4:7-10. These samples are combined with the direction of spread of the j-th dimension to obtain samples also in the diagonals —Alg. 4:13-16. The parameter l allows obtaining L samples with different distances to the mean —Alg. 4:8 and Alg. 4:14—, which are equally spaced in their direction axis and with the maximum distance which is given by the corresponding eigenvalue and $\chi_n^2(p)$, as shown in Eq. 4 —Alg. 4:7 and Alg. 4:13.

5. Results

In this section we report planning results for 2D and 3D autonomous navigation, and also for the generation of trajectories for autonomous scene reconstruction. The UAV is an Asctec Pelican[2] equipped with a RGB-D camera, the Orbbec Astra Pro[3], which has VGA resolution at 30 FPS for both the depth and RGB information. For collision check purposes we have approximated the UAV as a $0.5 \times 0.5 \times 0.4$ m cuboid, which includes the space occupied by the RGB-D sensor and the propellers. During the experiments we used several localization techniques with different uncertainties. In Sec. 5.1 we relied on a motion capture system, while in Sec. 5.2 and 5.3 we used the RGB-D camera and the publicly available implementation of ORB-SLAM2 [35], a visual-based SLAM system which uses loop closure to limit the long term drift and to keep the localization errors bounded. Moreover, in Sec. 5.3 KinectFusion [36] was used to obtain a dense 3D model from the data gathered with the RGB-D camera during the flight.

The UAV controller and the state estimation algorithm are run in the on-board computer, which has a CPU Intel Atom™ x5-Z8300 at 1.84 GHz, while the motion planner, the visual odometry and the 3D reconstruction algorithms are run in a laptop with CPU Intel Core™ i7-4720 at 2.6 GHz.

The state vector of the UAV contains the position of its center of rotation (x, y, z), the Euler angles (ϕ, θ, ψ) and the current linear $(\dot{x}, \dot{y}, \dot{z})$ and angular speeds $(\dot{\phi}, \dot{\theta}, \omega)$ in each axis:

$$[x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \omega] \tag{5}$$

The control vector, on the other hand, contains the variables that can be managed for a multi-rotor system. These are the desired linear speeds in each dimension (u_x, u_y, u_z), and the angular speed in the vertical axis (u_ω):

$$[u_x, u_y, u_z, u_\omega] \tag{6}$$

The UAV motions are represented with acceleration based equations. The dynamics model was obtained from 24 *min* and 18 *sec* of real flight data, recorded prior to the experiments described in this section, as described in [37]. Thus, the real behavior of the UAV to the different velocity commands was learned. Finally, the set of motion primitives used for planning purposes was obtained from the dynamics model, following the method described in [38]. In order to represent a variety of maneuverability restrictions, different motion primitives were used for each experiment despite the fact that the UAV dynamics model was the same.

5.1. 2D autonomous navigation

We tested the reliability and the precision of the autonomous navigation approach in a cluttered environment of 5×4 m with several narrow passages. For this experiment the set of motion primitives was comprised of $3,316$ actions which connected states between 0.2 m and 0.8 m. With these motion primitives the UAV was allowed to move forward and laterally at a maximum of 0.7 m/s, and rotate at a maximum of $30°/s$. Motion and sensing uncertainty were $M_t = 0.08 \cdot I$ and $N_t = 0.006 \cdot I$, respectively. The UAV state was estimated combining the information of the inertial unit with the position given by an external motion capture system. The use of the RGB-D camera for this experiment was discarded due to the limitations of the depth sensor —its optimal range is from 0.6 m to 5 m. Moreover, in this experiment the UAV navigates at a minimum distance to the obstacles of only 0.11 m, which requires a very precise localization.

There are several waypoints in the map that the UAV has to visit, ordered by number, starting and ending the route in the number 0. The maximum altitude was limited

[2] http://www.asctec.de/en/uav-uas-drones-rpas-roav/asctec-pelican/
[3] https://orbbec3d.com/product-astra-pro

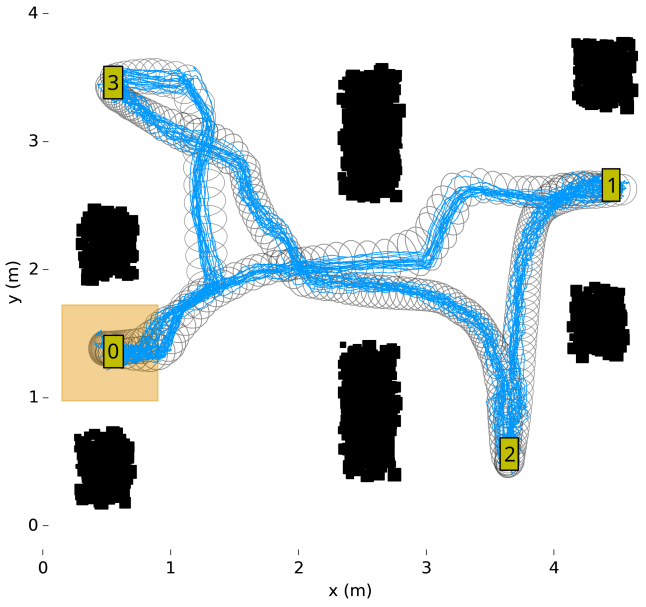


Figure 4. 20 different executions of our autonomous navigation approach in a cluttered environment. The followed routes are in blue, obstacles are in black, and waypoints are in yellow. The UAV takes off in 0, then visits 1, 2, 3, goes back to 0 and lands. Ellipses show $3 \cdot \Sigma$ of the predicted distributions.

to 1.0 m, so that the planned paths go through the obstacles instead of surpassing them from above. The controller makes a request to the motion planner to calculate the path to the next waypoint when the UAV is approaching the current goal. Due to the efficiency of the motion planning approach, these paths can be obtained on-demand during the flight without causing delays.

Fig. 4 shows the route followed by the UAV in 20 different executions of the experiment. As it can be seen, the predicted distributions adjust well to the real UAV state. The average deviation of the UAV during the 20 executions was 6.4 cm.

This scenario is challenging because of the proximity between the UAV and the obstacles. The minimum distance between them is as low as 11 cm in some points of the route, so reliable plans —in terms of estimated probability of collision— and precision in the control are required to avoid collisions. Our motion planner achieves this by reducing the speed of the UAV in those maneuvers which could affect the safety of the flight, e.g. just before passing between the obstacles in the middle of the map. This way the UAV stabilizes itself and can approach the obstacles in a safe manner, minimizing the overall probability of collision. In fact, in the 20 executions of this experiment no accidental collisions were reported. However, this would not be possible if both the kinematic restrictions and the uncertainty were not considered at planning time. In fact, in this kind of narrow passages those approaches that obtain a coarse high level solution would have to evaluate *in situ* the feasibility of the operation and either rely on their low level planner or being conservative and find an alternative path, if it exists. For both approaches the safety and cost of the solution cannot be quantified in advance, which might limit their suitability in some cases. In a similar manner, those approaches that consider the kinematic restrictions but not the uncertainty at planning time cannot offer any optimality guarantees regarding the safety of the plans. Instead, our approach is able to obtain reliable solutions that are optimal both in terms of safety and traversal time before the UAV starts moving.

Table 1. Planning results for the navigation experiment of Fig. 4. Column "Iter." is the number of iterations of AD*, "Inser." is the number of nodes inserted in OPEN, and "Time" is the planning time. "Cost" is the traversal time of the planned path, and "Min dist." is the minimum distance to the obstacles.

	Planning			Solution	
Path	Iter.	Inser.	Time[a]	Cost[a]	Min Dist.[b]
0-1	194	1,584	2.4	18.5	0.11
1-2	160	1,549	2.9	17.2	0.16
2-3	428	3,540	7.3	29.0	0.15
3-0	385	2,906	6.1	16.9	0.18
TOTAL	1,167	9,579	18.7	119.6	—

[a] In seconds.
[b] In meters.

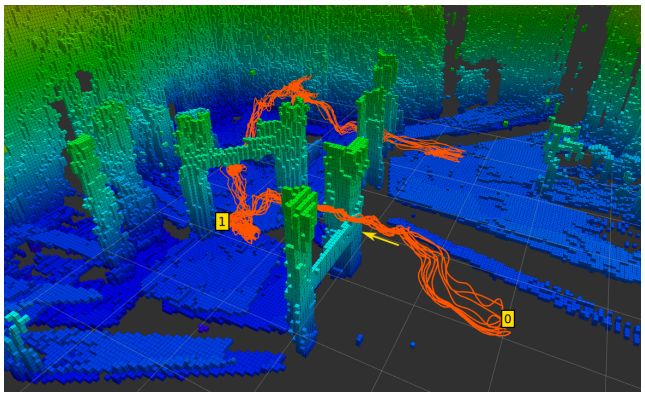
Table 1 details the planning results for this experiment. All the solutions are obtained within seconds, so the UAV can navigate between points without noticeable delays. The total cost of the route —the time the UAV is navigating between waypoints— is 119.6 s, although the execution time is slightly higher because the UAV stabilizes itself for a couple of seconds at each waypoint[4].

5.2. 3D autonomous navigation

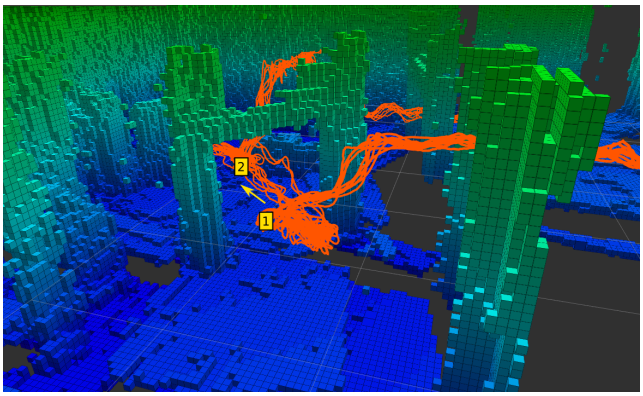
We have also tested our autonomous navigation approach in an environment of $8 \times 4 \times 2$ m with obstacles at different altitudes, which compelled the UAV not only to navigate between them, but also to change its altitude while moving. For this experiment the UAV state was estimated from the on-board RGB-D camera, using ORB-SLAM2 [35] with loop closure, combined with the inertial unit using an Extended Kalman Filter. Due to the minimum range of the RGB-D camera, the UAV cannot approach the obstacles closer than 0.6 m on its front side. For this reason, for motion planning purposes we have defined the UAV shape taking into account the additional distance which is needed for data acquisition. In this experiment the set of primitives used for planning included 1,700 motions which allowed the UAV to move forward, rotate and change its altitude. These motions connected states of distances 0.25 m, 0.5 m and 1 m, so the maximum fidelity is 0.25 m. Motion uncertainty was that of Sec. 5.1, while for this experiment sensing uncertainty was $N_t = 0.05 \cdot I$.

[4] A video recording of this experiment is available at https://youtu.be/HTCd3cwix60.

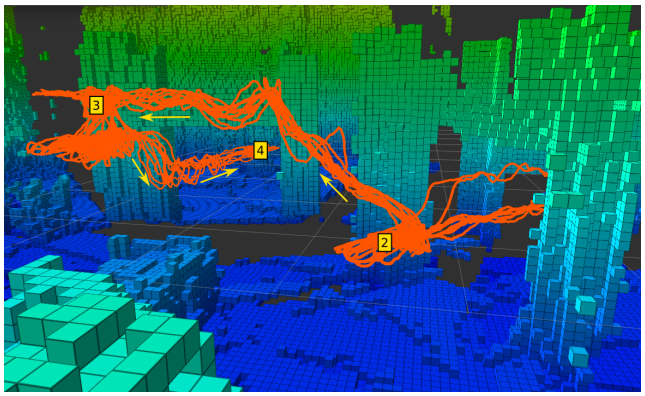
(a) 2D view of the executions of this experiment. The obstacles are in black, while the waypoints are numbered in yellow. Ellipses show $3 \cdot \Sigma$ of the predicted distributions.



(b) 3D view of the executions, in orange, focusing on the path between waypoints 0 and 1.



(c) Detail of the executions of the planned route between waypoints 1 and 2.



(d) Executions of the planned path between waypoints 2, 3 and 4.

Figure 5. Executions of the planned paths in a 3D environment, avoiding obstacles at different altitudes. The different routes followed by the UAV are in orange, while the waypoints and directions of motion are in yellow. The cells of the map are colored between blue and green, depending on their altitude. Detailed planning results are shown in Table 2.

The map of the environment was obtained beforehand from a handheld RGB-D camera, using ORB-SLAM2 and OctoMap [10] to store the result. In execution time, the map was updated at 1 Hz with the latest measurements of the RGB-D camera. Fig. 5(a) shows a 2D view of the final map and the distribution of the waypoints. Moreover, it details the results of 20 different executions of the UAV following the planned paths, and how well the predicted distributions adjust to the true UAV state. The average deviation of the UAV during these executions was 12.3 cm. There are several structures formed by vertical columns that the UAV has to cross. These columns are bonded by horizontal strips placed at different altitudes, and the planned paths have to surpass them from above or from below depending on their position and the predicted motion uncertainty.

Fig. 5(b) shows the path to the first waypoint, which avoids the horizontal strip from above to take advantage of the vertical speed acquired during the takeoff. After that, the UAV descends again to reach the first waypoint, which is located between the two first obstacles. The second path, shown in Fig. 5(c), goes through the second pair of columns, but before crossing them the UAV has to change its heading, because only motions in the direction of the camera view are allowed. These kinds of turnings are challenging because of the motion and sensing uncertainty. In fact, the UAV does not remain in a stable position while executing these maneuvers because of the inaccuracies of the estimated state, so the controller has to correct the possible deviations to avoid collisions. This is taken into account by the motion planner, which copes with this situation leaving enough time to stabilize the UAV before approaching the obstacles.

As shown in Fig. 5(d), after going through the second pair of columns, the UAV goes to the third waypoint, which is on the top of an obstacle with irregular shape. Finally, it goes to the goal, which is beyond the final structure. To achieve it, the UAV changes its heading again and then descends enough to surpass the horizontal strip from below.

Table 2 contains the detailed planning results for this experiment. The map used for planning purposes was ob-

Table 2. Planning results for the 3D autonomous navigation experiment of Fig. 5. Column "Iter." is the number of iterations of AD*, "Inser." is the number of nodes inserted in OPEN, and "Time" is the planning time. "Cost" is the traversal time of the resulting path.

	Planning			Solution
Path	Iter.	Inser.	Time[a]	Cost[a]
0-1	24	297	2.78	15.08
1-2	23	569	4.63	15.13
2-3	55	841	6.70	15.97
3-4	86	1,569	9.4	20.98
TOTAL	188	3,276	23.51	67.16

[a] In seconds.

tained *a priori* and updated in real time at 1 Hz while the UAV was moving. There were several waypoints that the UAV had to visit consecutively, and each path was obtained before reaching its starting point. The reported planning times allowed the UAV to execute all the routes without delays[5].

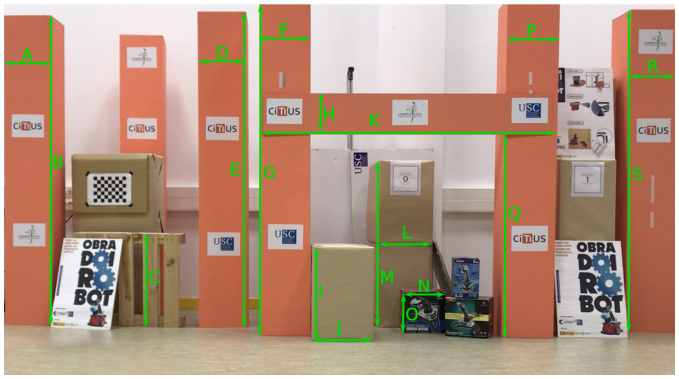
5.3. Autonomous navigation for scene reconstruction

In this section we report results for the generation of trajectories to achieve the autonomous reconstruction of a real scene. The goal is to obtain an accurate 3D model without manually controlling the UAV. Like in the experiment of Sec. 5.2, the UAV state was estimated with the on-board RGB-D camera, using ORB-SLAM2 with loop closure, and the inertial unit. For the scene reconstruction we used the KinectFusion [36] algorithm. The uncertainty conditions were the same as in the experiment of Sec. 5.2.

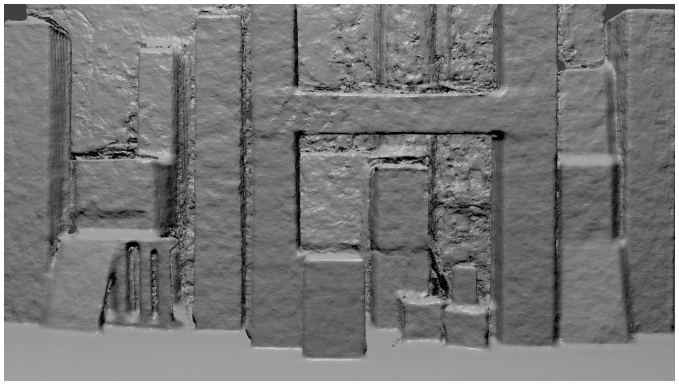
The autonomous navigation module takes as input the dimensions of the scene to be reconstructed and obtains a set of waypoints for the data acquisition. These waypoints guide the flight of the UAV in horizontal layers from one side of the scene to another, increasing the altitude at regular intervals. Like in the other experiments, the paths between waypoints are planned on demand, so the next one is obtained just before arriving to the current goal. However, in this case the navigation is done without prior knowledge of the map. The occupancy data, stored in the multi-resolution map, is updated in real time and used not only for planning purposes, but also to check collisions in the current path. If this is the case, the motion planner would obtain a new collision free plan taking into account the new obstacles.

The UAV has to move in a smooth and consistent manner to successfully obtain the reconstruction. This is because KinectFusion [36], like many other reconstruction algorithms, relies on Iteratively Closest Point (ICP) techniques —matching the current depth scan with the previous ones to incrementally build a consistent model. Our autonomous navigation approach is key to guarantee a successful reconstruction as, unlike other strategies, it

[5] A video recording of this experiment is available at https://youtu.be/xephfR35PYo.



(a) Real environment for the autonomous reconstruction experiment. The green labels are the distance measurements to show the accuracy of the obtained model, as detailed in Table 3.



(b) Render of the point cloud resulting from the dense reconstruction algorithm applied to the real environment in a).

Figure 6. Autonomous dense reconstruction of a scene with a UAV and a RGB-D camera.

manages the uncertainty, the kinematic restrictions and the controller all together, generating paths that ensure smooth camera motions.

This experiment consisted in the autonomous reconstruction of an environment of 2.5×0.75 m with objects of different sizes, which is shown in Fig. 6(a). The reconstruction algorithm was able to obtain a dense model with a high level of detail, shown in Fig. 6(b), with no holes in the surfaces that were directly seen by the RGB-D camera. The accuracy of the reconstruction is shown in Table 3, which compares some distance measurements of the objects in the real scene —labeled in Fig. 6(a)— to their counterparts in the obtained model. The error was, on average, of 1.08% with a standard deviation of 0.71%. The overall accuracy of the reconstruction is very high, with errors as low as 1 mm. The highest error was measured in the height of the orange columns —which were overestimated between 2.5 and 4 cm, which is around a 2% of their real size. This was caused by limiting the altitude of the flight, which prevented the RGB-D camera from taking measurements from above the columns.

The set of motion primitives of this experiment was comprised of 492 actions which allowed the UAV to change

Table 3. Distance measurements taken in the real scene compared to their counterparts in the reconstructed model, showing the accuracy of the 3D scene reconstruction. The highlighted measurements are those with the minimum and maximum error.

Distance	Reconstruction[a]	Real[a]	Error
A	30.9	31.0	0.32%
B	203	200.5	1.25%
C	59.7	60.0	0.50%
D	31.1	31.0	0.32%
E	204.5	200.5	2.00%
F	29.6	30.3	2.31%
G	203.7	200.5	1.60%
H	23.7	24.0	1.25%
I	54.3	53.2	2.07%
J	34.1	34.6	1.45%
K	177.3	177.5	0.11%
L	33.6	34.3	2.04%
M	107.8	106.5	1.22%
N	24.9	25.0	0.40%
O	24.8	25.0	0.80%
P	30.2	30.3	0.33%
Q	130.4	129.8	0.46%
R	30.8	31.0	0.65%
S	203.5	200.5	1.50%

[a] In centimeters.

its altitude, move forward and laterally. The speed was limited to 0.3 m/s, which was suitable for the purpose of scene reconstruction. Several routes were planned to acquire the data for the reconstruction. Concretely, the UAV went from side to side three times and changed its altitude twice. Planning results are detailed in Table 4. The total cost of the obtained routes was 34.09 s, although the duration of the flight is longer because of stabilizing the UAV at the end of each path[6].

Something to take into consideration is that the heuristic H3DMR has to be re-calculated every time the map changes, although operating in real time is not an issue due to the reduced planning times. In fact, due to its efficiency, H3DMR is obtained on average in 94 iterations and less than 40 ms.

As mentioned before, a smooth flight is key for getting a good 3D model. The consistency of the reconstruction benefits from a stable flight, so we measured this in the experiment. We have recorded the state of the UAV, discarding those measurements which did not correspond with the planned paths —the takeoff, the landing and the moments in which the UAV is holding its position in the air. The distribution of the real speed during the execution of the planned routes is shown in Fig. 7. The average speed is 0.289 m/s, almost in the upper bound of 0.3 m/s. More importantly, the standard deviation is only of 0.053, resulting in a navigation which is smooth enough for obtaining a consistent model.

[6] A video recording of this experiment is available at https://youtu.be/1UAwEXGYBOA.

Table 4. Planning results for the autonomous reconstruction environment of Fig. 6. Column "Iterations" is the number of iterations of AD*, "Insertions" is the number of nodes inserted in OPEN, and "Time" is the planning time. "Cost" is the traversal time of the resulting path.

	Planning			Solution
Path	Iter.	Inser.	Time[a]	Cost[a]
1	6	120	0.73	9.89
2	3	113	0.52	2.21
3	6	125	0.52	9.89
4	3	116	0.62	2.21
5	6	125	0.49	9.89
TOTAL	24	599	2.88	34.09

[a] In seconds.

A drawback of using visual odometry to estimate the state of the UAV is the increased delay with respect to other localization systems. Since the LQG controller uses these estimations to adjust the controls and minimize deviations from the expected state, there might be convergence issues in certain situations. However, our motion planner takes into account the dynamics model of the UAV, obtained from navigation data which is acquired with this localization system. Thus, the delay in the estimations is implicitly taken into account, overcoming this issue and guaranteeing the stability of the controller.

6. Conclusions and future work

In this work we have presented a motion planning algorithm for UAV autonomous navigation that manages motion and sensing uncertainty at planning time. Due to

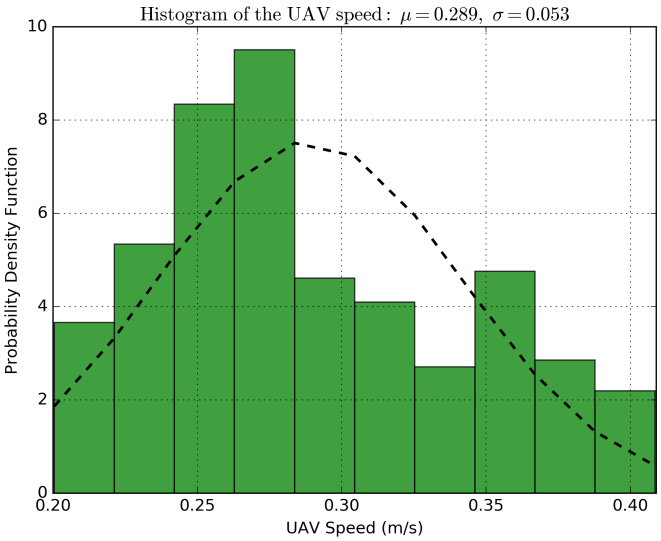


Figure 7. Histogram of the UAV linear speed during the execution of the autonomous 3D scene reconstruction experiment. Speed target was 0.3 m/s. The mean was $\mu = 0.289$, and the standard deviation was $\sigma = 0.053$, which resulted in a stable flight.

taking into account the real dimensions of the UAV and the uncertainty in heading, the probability of collision is estimated reliably, and optimal paths in terms of safety, traversal time and uncertainty at the goal are selected. We have tested our proposal in 3 different experiments and with different maneuvering limitations to show the safety of the planned paths, repeating the experiments several times to validate the estimation of the probability of collision. No collisions were reported during the execution of the planned routes, and the LQG controller was able to produce a stable flight while tracking the planned routes with accuracy.

The detailed planning results showed the efficiency of the planning approach, which allowed the UAV to navigate autonomously in real time, and to reconstruct 3D scenes. This is due to our heuristic H3DMR, which deals with the obstacles in the map, and the graduated fidelity lattice. Videos for all the experiments are available, showing the autonomous navigation capabilities on a real platform. Finally, we also reported results for the autonomous reconstruction of a 3D scene. The UAV collected data during the flight, which was used to obtain a precise dense model. The average measured error of the reconstruction was of 1.08%, with a standard deviation of 0.71%.

A subject of future work would be extending this approach to use sensors of other kinds. The proposed architecture is flexible enough to allow doing so without introducing substantial changes in the controller and planning modules. Thus, using stereo or monocular cameras would allow operating the UAV in environments where the use RGB-D cameras is constrained due to their limited range, the interference of natural light or the excessive power consumption.

Acknowledgments

This research was funded by the Spanish Ministry for Science, Innovation and Universities (grant TIN2017-8479 6-C2-1-R) and the Galician Ministry of Education, University and Professional Training (grants ED431C 2018/29 and "accreditation 2016-2019, ED431G/08"). These grants were co-funded by the European Regional Development Fund (ERDF/FEDER program).

References

[1] F. Ropero, P. Muñoz, M. D. R-Moreno, Terra: A path planning algorithm for cooperative UGV–UAV exploration, Engineering Applications of Artificial Intelligence 78 (2019) 260 – 272.

[2] E. Semsch, M. Jakob, D. Pavlicek, M. Pechoucek, Autonomous UAV surveillance in complex urban environments, in: Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 02, 2009, pp. 82–85.

[3] M. Shirzadeh, H. J. Asl, A. Amirkhani, A. A. Jalali, Vision-based control of a quadrotor utilizing artificial neural networks for tracking of moving targets, Engineering Applications of Artificial Intelligence 58 (2017) 34 – 48.

[4] A. Puri, A survey of unmanned aerial vehicles (UAV) for traffic surveillance, Department of computer science and engineering, University of South Florida (2005) 1–29.

[5] B. Coifman, M. McCord, R. G. Mishalani, M. Iswalt, Y. Ji, Roadway traffic monitoring from an unmanned aerial vehicle, in: IEE Proceedings-Intelligent Transport Systems, Vol. 153, 2006, pp. 11–20.

[6] R. Almadhoun, T. Taha, L. Seneviratne, J. Dias, G. Cai, A survey on inspecting structures using robotic systems, International Journal of Advanced Robotic Systems 13 (6) (2016) 1–18.

[7] F. Nex, F. Remondino, UAV for 3D mapping applications: a review, Applied Geomatics 6 (1) (2014) 1–15.

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

[9] D. Meagher, Geometric modeling using octree encoding, Computer graphics and image processing 19 (2) (1982) 129–147.

[10] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, W. Burgard, OctoMap: An efficient probabilistic 3D mapping framework based on octrees, Autonomous Robots 34 (3) (2013) 189–206.

[11] M. Pivtoraiko, A. Kelly, 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.

[12] A. González-Sieira, M. Mucientes, A. Bugarín, Motion planning under uncertainty in graduated fidelity lattices, Robotics and Autonomous Systems 109 (2018) 168 – 182.

[13] S. Zingg, D. Scaramuzza, S. Weiss, R. Siegwart, MAV navigation through indoor corridors using optical flow, in: IEEE International Conference on Robotics and Automation (ICRA), 2010, pp. 3361–3368.

[14] I. Sa, H. He, V. Huynh, P. Corke, Monocular vision based autonomous navigation for a cost-effective MAV in GPS-denied environments, in: IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), 2013, pp. 1355–1360.

[15] M. Nieuwenhuisen, D. Droeschel, M. Beul, S. Behnke, Obstacle detection and navigation planning for autonomous micro aerial vehicles, in: International Conference on Unmanned Aircraft Systems (ICUAS), 2014, pp. 1040–1047.

[16] F. J. Perez-Grau, R. Ragel, F. Caballero, A. Viguria, A. Ollero, An architecture for robust UAV navigation in GPS-denied areas, Journal of Field Robotics 35 (1) (2018) 121–145.

[17] T. Nägeli, L. Meier, A. Domahidi, J. Alonso-Mora, O. Hilliges, Real-time planning for automated multi-view drone cinematography, ACM Transactions on Graphics (TOG) 36 (4) (2017) 132.

[18] M. Iacono, A. Sgorbissa, Path following and obstacle avoidance for an autonomous UAV using a depth camera, Robotics and Autonomous Systems 106 (2018) 38–46.

[19] V. González, C. A. M. Micharet, L. Moreno, C. Balaguer, UAVs mission planning with flight level constraint using Fast Marching Square Method, Robotics and Autonomous Systems 94 (2017) 162–171.

[20] H. Yu, R. Beard, A vision-based collision avoidance technique for micro air vehicles using local-level frame mapping and path planning, Autonomous Robots 34 (1-2) (2013) 93–109.

[21] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, E. Galceran, Continuous-time trajectory optimization for online UAV replanning, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 5332–5339.

[22] M. Pivtoraiko, D. Mellinger, V. Kumar, Incremental micro-UAV motion replanning for exploring unknown environments, in: IEEE International Conference on Robotics and Automation (ICRA), 2013, pp. 2452–2458.

[23] B. MacAllister, J. Butzke, A. Kushleyev, H. Pandey, M. Likhachev, Path planning for non-circular micro aerial vehicles in constrained environments, in: IEEE International Conference on Robotics and Automation (ICRA), 2013, pp. 3933–3940.

[24] S. M. LaValle, J. J. Kuffner, Randomized kinodynamic planning, The International Journal of Robotics Research 20 (5) (2001) 378–400.
[25] J. D. Gammell, S. S. Srinivasa, T. D. Barfoot, Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2014, pp. 2997–3004.
[26] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, 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.
[27] N. Dadkhah, B. Mettler, Survey of motion planning literature in the presence of uncertainty: Considerations for UAV guidance, Journal of Intelligent and Robotic Systems 65 (1-4) (2012) 233–246.
[28] J. Van den Berg, P. Abbeel, K. Goldberg, LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information, The International Journal of Robotics Research 30 (7) (2011) 895–913.
[29] A. Bry, 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.
[30] S. Karaman, E. Frazzoli, Incremental sampling-based algorithms for optimal motion planning, Robotics Science and Systems VI 104.
[31] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, Ros: an open-source robot operating system, in: ICRA workshop on open source software, Vol. 3, 2009, p. 5.
[32] P. Rodriguez-Mier, A. Gonzalez-Sieira, M. Mucientes, M. Lama, A. Bugarin, Hipster: An open source java library for heuristic search, in: 9th Iberian Conference on Information Systems and Technologies (CISTI), 2014, pp. 1–6.
[33] M. Likhachev, D. Ferguson, Planning Long Dynamically Feasible Maneuvers for Autonomous Vehicles, The International Journal of Robotics Research 28 (8) (2009) 933–945.
[34] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, S. Thrun, Anytime search in dynamic graphs, Artificial Intelligence 172 (14) (2008) 1613–1643.
[35] R. Mur-Artal, J. D. Tardós, ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras, IEEE Transactions on Robotics 33 (5) (2017) 1255–1262.
[36] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, A. Fitzgibbon, KinectFusion: Real-time dense surface mapping and tracking, in: 10th IEEE international symposium on Mixed and augmented reality (ISMAR), 2011, pp. 127–136.
[37] P. Abbeel, Apprenticeship learning and reinforcement learning with application to robotic control, Stanford University, 2008.
[38] T. M. Howard, A. Kelly, Optimal rough terrain trajectory generation for wheeled mobile robots, The International Journal of Robotics Research 26 (2) (2007) 141–166.