Robust autonomous flight in cluttered environment using a depth sensor

Liang Lu, Alexander Yunda, Adrian Carrio and Pascual Campoy

Abstract
This paper presents a novel collision-free navigation system for the unmanned aerial vehicle based on point clouds that outperform compared to baseline methods, enabling high-speed flights in cluttered environments, such as forests or many indoor industrial plants. The algorithm takes the point cloud information from physical sensors (e.g. lidar, depth camera) and then converts it to an occupied map using Voxblox, which is then used by a rapid-exploring random tree to generate finite path candidates. A modified Covariant Hamiltonian Optimization for Motion Planning objective function is used to select the best candidate and update it. Finally, the best candidate trajectory is generated and sent to a Model Predictive Control controller. The proposed navigation strategy is evaluated in four different simulation environments; the results show that the proposed method has a better success rate and a shorter goal-reaching distance than the baseline method.

Keywords
Aerial robot, autonomous navigation, path planning, local planner, rapid-exploring random tree

Introduction
Autonomous navigation in cluttered environments is a key challenge in the area of robot navigation. It is still an unsolved problem because of the requirements of robust perception, planning, and control algorithms when robots operate in an unconstrained environment. During the last years, this problem has attracted a lot of researchers’ attention and several methods have been proposed. In the literature, algorithms have been proposed, the learning-based strategies for autonomous navigation are described in Sampedro et al. and Carrio et al. The autonomous quadrotor aerial robots for solving indoor missions are built by Sampedro et al. and Rigter et al., while the authors of Oleynikova et al. and Cui et al. give solutions for outdoor navigation. A cooperative navigation approach using tiny flying robots is explained in McGuire et al.

In our previous works, we have proposed several localization, control, and collision avoidance algorithms for safe navigation in the unknown cluttered environments. In this work, we try to develop a robust planner and combine it together with the other algorithms to perform the task in cluttered environments. Compared with other works, our planner only uses a limited field of sensing view and can be used in unconstrained environments.

The planner we develop for the autonomous navigation system can be thought of as a local planner. First, we generate finite path candidates with rapid-exploring random tree (RRT) in a three-dimensional (3D) bounding box. Then, the optimal path from the path candidates is selected using a modified Covariant Hamiltonian Optimization for Motion Planning (CHOMP) objective function. Finally, we generate the trajectory with the optimal path and track the trajectory using a Model Predictive Control (MPC) controller.
The innovations in our work are listed below:

First, we develop a robust local planner which uses an RRT-based online path candidates generation algorithm and a modified objective function of CHOMP to select an optimal candidate. Then, the local planner is combined with the localization, perception, trajectory optimization, and control algorithm to build a fully-autonomous navigation system.

The remainder of the paper is organized as follows. The related works are covered in Section “Related works”. Section “Problem formulation” presents the problem formulation. In Section “Methodology”, the proposed methodology is introduced. We describe the experimental results and the discussion in Section “Experimental results and discussions”. And finally, Section “Conclusions and future works” concludes the paper and summarizes future research directions.

Related works

A lot of methods have been proposed during the last decades addressing the problem of autonomous navigation in cluttered environments. Fang et al.13 developed a two-layer fusion framework and combined it with an online motion planning algorithm to build an autonomous navigation system. The experimental results show their system can go through a very narrow doorway in an environment in complete darkness. Sampedro et al.3 developed a fully-autonomous aerial robotic solution for indoor exploration and inspection. Their solution was validated in the 2016 International Micro Air Vehicle Competition. Lin et al.14 built an autonomous navigation system with only a fisheye camera, a low-cost inertial measurement unit (IMU), and heterogeneous onboard computing resources that comprised the minimum sensor suite allowing autonomous flight. The experimental results show that their autonomous navigation system can fly both in indoor and outdoor environments. Mohta et al.15 designed an autonomous navigation system that is capable of fast aerial navigation through cluttered GPS-denied environments. The experimental results show that their robot can fly without collision in a cluttered environment with speeds of up to 7 m/s. Gao et al.16 developed a framework for online generating safe and dynamically feasible trajectories directly on the point cloud. Oleynikova et al.5 created a system using a local planner architecture that could generate smooth global trajectories, locally explore unknown spaces, and deal with changes in a map by actively replanning. As the planning system is the central work of this paper, we discuss in detail the recent relevant planning algorithms in the rest of the section. Generally, the planning system is organized by a path planning system and trajectory generation system.

Path plan algorithm

Generally speaking, sampling-based algorithms and search-based algorithms are two main types of path planning algorithms used in the planning system. The most relevant sampling-based algorithms, RRT, RRT*, and their extended versions are widely used for the path planner in the planning system. In the work of Bircher et al., they use RRT to generate a finite iteration random tree and execute the first edge of best branch. The point clouds-based navigation framework designed by Gao et al.16 use a safe-region RRT* to generate safe corridors and update it following the principle of anytime RRT*. Then, the trajectory is generated using the updated corridors. The continuous-time trajectory optimization approach plans a dynamically feasible polynomial trajectory based on the path from an Informed RRT* algorithm. There also many planning systems using search-based algorithms like A*, D*, or others to plan the path. Gradient-based online safe trajectory generation strategies22 use A* to search a minimum distance path on a random-exploring graph. The navigation system using monocular visual–inertial fusion from the same group14 also applies A* to plan a path for trajectory optimization. Tordesillas et al.23,24 presented a real-time trajectory planner using Jump Point Search (JPS) to find the shortest path from the current position to the goal. The reason they chose JPS is that JPS could run an order of magnitude faster than A* while guaranteeing completeness and optimality.

Trajectory generation and optimization

Based on the path from the path planner, a trajectory optimization algorithm is used to generate a feasible trajectory. Gradient-based strategies like CHOMP use an objective function based on smoothness and safety to punish the waypoints in the path. The final trajectory is generated until the result of the objective function is less than a threshold value or the iteration times are bigger than the maximum times. Recently, a gradient-based online safe trajectory generation method proposed by Gao et al.22 and the trajectory optimization approach proposed by Oleynikova et al. also applied the gradient-based method for trajectory generation and optimization. The works of Tordesillas et al.23,24 use a control-based method. They consider the dynamic model of the aerial robot and generate the jerk-controlled primitive trajectory by solving a convex optimization problem.
The minimum trajectory generation proposed by Mellinger and Kumar shows that the trajectory generation problem can be thought of as a quadratic programming (QP) problem. Charles et al. extend the minimum trajectory generation problem to a numerically stable unconstrained QP problem. They combine an RRT* planner with the minimum-snap technique to perform fast, graceful flight trajectories while considering collisions. The mixed integer quadratic programming approach proposed by Mellinger et al. uses integer constraints to enforce collision avoidance. The iterative regional inflation by semidefinite programming algorithm developed by Deits and Tedrake is used to pre-compute convex regions of safe space. In recent years, the work from Zhou et al. relies on the use of a B-Spline for trajectory generation and optimization. The experiments’ results show their approach can improve the efficiency and convergence rate of trajectory optimizations.

**Problem formulation**

This section describes in detail the robot architecture and the environment within the navigation problem. The model of the robot is given at first and then the definition of the environment is explained.

**Robot architecture**

The robot used in the research is a multirotor unmanned aerial vehicle (UAV). The multirotor UAV has six Degrees of Freedom (DoF), three translation DoF, and three rotation DoF. The on-board sensors of the aerial robot are a front-looking depth (Intel RealSense d435i) and an IMU sensor.

**Environment definition**

The simulation environments used in this work are environments with multiple obstacles. The mission has a starting point \( P_s \) and goal point \( P_g \). The positions of the obstacles are totally unknown when the robot starts to move and the robot can only perceive a fraction of the environment around it. The navigation mission is moving from the starting point to a spherical region centered on the goal point without colliding with the obstacles. An example of the environment used in the paper is shown in Figure 1.

**Methodology**

There are three main parts in the navigation system, which are **localization system**, **perception system**, and **planning system**.

For the localization system, it is built based on our previous work. For the perception system, we use a depth/RGB-D sensor get point cloud and voxblox Robot Operating System (ROS) package to build the occupancy map. With the occupancy map, we can build the 3D Euclidean Signed Distance Fields (ESDFs) which provide the nearest distance to the occupied voxel and the gradient of each voxel for planning. A scheme describing that shows the perception system can be found in Figure 2.

For the planning system, a local planner is proposed to plan the trajectory in the field of view (FOV) and guide the robot moving from the starting point to the goal point. The remainder of this section will describe the planning system in detail. First, we generate the path candidates using an online path candidates generation algorithm. Then, the optimal path is selected from the candidates. Finally, we get the trajectory from the optimal path and a trajectory controller is used to move the robot.
and maximum altitude. We describe these four items by

\[
f(x, y, z), \quad f_{\text{ori}}, \quad f_{\text{ori}}, \quad f_{\text{alt}}, \quad f_{\text{alt}}
\]

respectively, and define the objective function by summing their weights

\[
f(\xi) = w_1f_1(\xi) + w_2f_2(\xi) + w_3f_3(\xi(1)) + w_4f_4(\xi)
\]

As described above, the trajectory is \(\xi\) and \(\xi(s)\) is the function mapping the trajectory length \(s\) to the robot configurations, the initial and end configurations of the trajectory \(\xi\) is \(\xi(0)\) and \(\xi(1)\), respectively. The waypoints in the trajectory \(\xi\) are the 3D points \(\{x, y, z\}\). \(w_1, w_2, w_3,\) and \(w_4\) are the weights for each objective function.

The objective functions of \(f_s, f_{\text{ori}}, f_{\text{ori}}, f_{\text{alt}}\) are the same from Fang et al.\(^{13}\)

\[
f_o(\xi) = \int_0^1 c_o(\xi(s)) \left\| \frac{d}{dt} \xi(s) \right\| ds
\]

\[
f_s(\xi) = \frac{1}{2} \int_0^1 \left\| \frac{d}{dt} \xi(s) \right\|^2 ds
\]

\[
f_{\text{ori}}(\xi(1)) = \|\xi(1) - \xi_g\|
\]

\[
c_o(\xi(s)) = \begin{cases} 
-\text{Dist}(\xi(s)) + \frac{1}{2} \epsilon, & \text{if } \text{Dist}(\xi(s)) \leq 0 \\
\frac{1}{2 \epsilon}(\text{Dist}(\xi(s)) - \epsilon)^2, & 0 < \text{Dist}(\xi(s)) \leq \epsilon \\
0, & \text{otherwise}
\end{cases}
\]

\(\text{Dist}(\xi(s))\) is the distance from the waypoints in the trajectory to the nearest obstacle. We can get the \(\text{Dist}(\xi(s))\) from the 3D ESDFs built by the perception system.

The altitude objective function \(f_{\text{alt}}(\xi)\) is learned from Zucker et al.

\[
f_{\text{alt}}(\xi) = \int_0^1 c_{\text{alt}}(\xi(s)) \left\| \frac{d}{dt} \xi(s) \right\| ds
\]
The altitude cost $c_d(\xi(s))$ in the objective function is defined as:

$$c_d(\xi(s)) = \begin{cases} (a_{min} - Alt(\xi(s)))^2, & Alt(\xi(s)) \leq a_{min} \\ (Alt(\xi(s)) - a_{max})^2, & Alt(\xi(s)) \geq a_{max} \\ 0, & \text{otherwise} \end{cases}$$

(7)

$Alt(\xi(s))$ is the altitude of the waypoints of the trajectory $\xi$, $a_{min}$ and $a_{max}$ are the predefined minimum and maximum altitude values, respectively.

We calculate the cost of every path candidate by using the proposed objective function and select the path candidate with the minimum cost as the optimal path.

**Trajectory generation and controller**

After getting the optimal candidate, we use it as an initial path for the CHOMP planner to generate the final path.

After getting the path, a polynomial micro aerial vehicle trajectory generation approach is used to generate the trajectory with the final path, and then a MPC is used to track the trajectory.

**Experimental results and discussions**

**Experimental setup**

In this subsection, we introduce the experimental setup for the simulation experiments.

We run the simulation experiments using the ROS on top of Ubuntu 18.04. The Gazebo simulator is used to provide the environment model and the RotorS simulation is used to provide the physical model parameters of robot. All the experiments run on a laptop with Intel Core i7-8750H at 2.2 GHz. The simulation of the proposed navigation system is integrated into our open source framework Aerostack (www.aerostack.org). The robot in the simulation is the AscTec hummingbird, which is equipped with a depth sensor with a FOV of 135° and an odometry sensor. The size of the voxels used in the simulation is 0.2 m. The depth sensor is used for sampling the environment.

![Figure 3. The office and three dense forest environments used for simulation experiments. The large figure corresponds to a top-view of the environment. The smaller top figures show the mesh maps and the bottom ones the occupancy maps.](image)
Simulation experiments

In order to evaluate the proposed planning system, we test it in four different environments. These environments are an office environment and three dense forest environments with different obstacle densities. The three dense forest environments have an obstacle density of 0.02, 0.05, and 0.1 $\text{obstacles/m}^2$ respectively. The sensing range used in the office environment and the dense forest environments are 5 m and 10 m respectively. The step length and the maximum number of the nodes of the RRT-based online candidate generator are 0.5 m and 40 respectively.

Figure 3 shows the simulation environments used in the paper. The trajectories of one successful run of each environment in the mesh and occupancy map are also shown.

We run the proposed algorithm 50 times in each environment. For each run, the starting point and goal point are randomly generated in a prior defined range. The starting point in the office environment is generated from $(5, 21, 0.5)$ m to $(5, 22, 0.5)$ m and the goal point is generated from $(21, 10, 0.5)$ m to $(21, 18, 0.5)$ m. The starting point in the dense forest environment is generated from $(0, 15, 0.5)$ m to $(0, 35, 0.5)$ m and the goal point is generated from $(50, 15, 0.5)$ m to $(50, 35, 0.5)$ m. After finishing the runs in one environment, the average velocity, the maximum velocity, the time to reach the goal, the trajectory length of every running and the success rate are calculated. The results of the simulation experiments can be seen in Table 1.

| Scenario | SR (%) | TL (m) | TG (s) |
|----------|--------|--------|--------|
| Office   | 90     | 19.2 ± 3.3 | 29.4 ± 5.5 |
| DF (0.02 $\text{obstacles/m}^2$) | 96 | 76.2 ± 15.2 | 106.1 ± 19.5 |
| DF (0.05 $\text{obstacles/m}^2$) | 94 | 78.0 ± 13.3 | 108.4 ± 17.0 |
| DF (0.1 $\text{obstacles/m}^2$) | 90 | 76.7 ± 10.1 | 93.5 ± 12.3 |

| Scenario | AV (m/s) | AMV (m/s) |
|----------|----------|----------|
| Office   | 0.65 ± 0.01 | 1.01 ± 0.07 |
| DF (0.02 $\text{obstacles/m}^2$) | 0.72 ± 0.01 | 1.29 ± 0.09 |
| DF (0.05 $\text{obstacles/m}^2$) | 0.72 ± 0.01 | 1.28 ± 0.09 |
| DF (0.1 $\text{obstacles/m}^2$) | 0.82 ± 0.02 | 1.49 ± 0.16 |

DF: dense forest; SR: success rate; TL: length of the trajectory; TG: time to reach the goal; AV: average velocity; AMV: average maximum velocity.

Figure 4. Trajectory length obtained by the proposed method and the baseline method in the dense forest and office simulations. The values of covered distance are computed for the cases that reach the goal.

RRT: rapid-exploring random tree.
found in Table 1. The DF means dense forest scenario. The SR, TL, TG, AV, and AMV mean: the success rate, length of the trajectory, the time to reach the goal, average velocity, and average maximum velocity, respectively.

We compare the proposed algorithm with a baseline method. We build the baseline algorithm using RRT* to generate candidates and chooses the candidate based on a goal-biased objective function. For the comparative experiment, We run the baseline algorithm 50 times in the office and in three different dense forest environments. Then, the success rate and the trajectory lengths are obtained for evaluation. The trajectory lengths are computed for the cases in which the goal is reached.

A video showing one fail and success running in the office and DF (0.1 obs/m²) has been made available in: https://vimeo.com/405905039.

Discussions
As can be seen from Table 1, the proposed planner has good successful rates in all the simulation environments. The success rates are 90%, 96%, 94%, and 90% in the office, DF (0.02 obs/m²), DF (0.05 obs/m²), and DF (0.1 obs/m²) environments, respectively. The average velocities are between 0.65 m/s and 0.82 m/s in all the simulation environments. In all the environments, the aerial robot reaches more than 1 m/s.

Figure 4 shows that the proposed strategy can plan a shorter trajectory to reach the goal. The average distance values obtained by our planner are 19.2 m in the office environment and between 76.2 m and 78.0 m in the dense forest environments. The average distance values obtained by the baseline method are 24.03 m in the office environment and between 102.62 m and 109.05 m in the dense forest environments. As can be seen from Figure 5, the proposed method has much better performance than the baseline method. The success rates of baseline method can only reach 58%, 48%, 62%, and 72% in the office, DF (0.1 obs/m²), DF (0.05 obs/m²), and DF (0.02 obs/m²) environments, respectively.

The main differences between the proposed method and the baseline strategy are that it designs an objective function for best candidate selection and updates the selected candidate using the CHOMP objective function before sending it to the controller. The experiment results show that the proposed approach helps planner find a shorter trajectory and improve the success rate to reach the goal.

Conclusions and future works
In this paper, a method for aerial robot autonomous navigation in cluttered environments has been presented. First, the depth sensors are used to sense the environment around the robot. Second, the occupied map is built using the point clouds captured with the sensors. Finally, a local planner is developed to plan a feasible trajectory and send it to a MPC controller. The local planner uses RRT to generate path candidates and a modified CHOMP objective function to select the best candidate. In order to improve the quality of the selected candidate, the CHOMP objective function is used to update it before sending it to the controller. The experiment results show that the proposed method has a better success rate and covers a shorter distance to reach the goal.

In the future, our work will focus on improving the success rate and high-speed navigation in cluttered and dynamic environments. We will combine the search-based algorithm like A* with the safe flight corridor to plan the trajectory for aggressive flight. A real flight will also be implemented in the future.

Declaration of conflicting interests
The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding
The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: The work reported in this paper is sponsored by the Chinese Scholarship Council (CSC) and by the project “COMplex Coordinated Inspection and Security missions by UAVs in cooperation with UGV”, funded by the
Spanish Ministry of Economy and Competition (RTI2018-100847-B-C21).

**ORCID iDs**

Liang Lu [https://orcid.org/0000-0002-8078-8760](https://orcid.org/0000-0002-8078-8760)

Alexander Yunda [https://orcid.org/0000-0003-0498-9656](https://orcid.org/0000-0003-0498-9656)

Adrian Carrio [https://orcid.org/0000-0001-6711-7279](https://orcid.org/0000-0001-6711-7279)

Pascual Campoy [https://orcid.org/0000-0002-9894-2009](https://orcid.org/0000-0002-9894-2009)

**References**

1. Sampedro C, Bavle H, Rodriguez-Ramos A, et al. Laser-based reactive navigation for multirotor aerial robots using deep reinforcement learning. In: 2018 IEEE/RSJ international conference in intelligent robots and systems (IROS). Madrid, Spain, 2018, pp.1024–1031. Madrid, Spain: IEEE.

2. Carrio A, Sampedro C, Rodriguez-Ramos A, et al. A review of deep learning methods and applications for unmanned aerial vehicles. J Sens 2017; 2017: 1–13.

3. Sampedro C, Bavle H, Rodriguez-Ramos A, et al. A fully-autonomous aerial robotic solution for the 2016 International Micro Air Vehicle competition. In: 2017 international conference in unmanned aircraft systems (ICUAS), Miami, FL, USA, 2017, pp.989–998. Miami, FL: IEEE.

4. Rigter M et al. An autonomous quadrotor system for robust high-speed flight through cluttered environments without GPS. In: 2019 IEEE/RSJ international conference on intelligent robots and systems (IROS), Macau, China, 2019, pp. 5227–5234. Macau, China: IEEE.

5. Oleynikova H, Lanegger C, Taylor Z, et al. An open-source system for vision-based micro-aerial vehicle mapping, planning, and flight in cluttered environments. arXiv:1812.03892, 2019.

6. Cui JQ, Lai S, Dong X, et al. Autonomous navigation of UAV in forest. In: 2014 international conference in unmanned aircraft systems (ICUAS), Orlando, FL, USA, 2019, pp.1024–1031. Orlando, FL: IEEE.

7. McGuire KN, De Wagt C, Tuyls K, et al. Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment. Sci Rob 2019; 4: 1–14.

8. Bavle H, Sanchez-Lopez JL, De la Puente P, et al. Fast and robust flight altitude estimation of multirotor UAVs in dynamic unstructured environments using 3D point cloud sensors. Aerospace 2018; 5: 1–21.

9. Sanchez-Lopez JL, Pestana J, De la Puente P, et al. Open-source system architecture for the fast designing and prototyping of autonomous multi-UAV systems: simulation and experimentation. J Intell Robot Syst 2016; 84: 779–797.

10. Lu L, Sampedro C, Rodriguez-Vazquez J, et al. Laser-based collision avoidance and reactive navigation using RRT* and signed distance field for multirotor UAVs. In: 2019 international conference on unmanned aircraft systems (ICUAS), Atlanta, GA, USA, 2019, pp.1209–1217. Atlanta, GA: IEEE.

11. Carrio A, Vemprala S, Ripoll A, et al. Drone detection using depth maps. In: 2018 IEEE/RSJ international conference on intelligent robots and systems (IROS), Madrid, Spain, 2018, pp.1034–1037. Madrid, Spain: IEEE.

12. Zucker M, Ratliff N, Dragan AD, et al. CHOMP: covariant Hamiltonian optimization for motion planning. Int J Rob Res 2013; 32: 1164–1193.

13. Fang Z, Yang S, Jain S, et al. Robust autonomous flight in constrained and visually degraded shipboard environments. J Field Robotics 2016; 34: 25–52.

14. Lin Y, Gao F, Qin T, et al. Autonomous aerial navigation using monocular visual inertial fusion. J Field Robotics 2018; 35: 23–51.

15. Mohla K, Watterso P, Mulgaonkar Y et al. Fast, autonomous flight in GPS denied and cluttered environments. J Field Robotics 2018; 35: 101–120.

16. Gao F, Wu W, Gao W, et al. Flying on point clouds: online trajectory generation and autonomous navigation for quadrotors in cluttered environments. J Field Robotics 2019; 36: 710–733.

17. Lavalle SM and Kuffner JJ Jr. Randomized kinodynamic planning. Int J Rob Res 2001; 20: 378–400.

18. Karaman S and Frazzoli E. Sampling based algorithms for optimal motion planning. Int J Rob Res 2011; 30: 846–894.

19. Otto M and Frazzoli E. RRTX: asymptotically optimal single-query sampling-based motion planning with quick replanning. Int J Rob Res 2016; 35: 797–822.

20. Bircher A, Kamel M, Alexis K, et al. Receding horizon “Next-Best-View” planner for 3D exploration. In: 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, 2016, pp.1462–1468. Stockholm, Sweden: IEEE.

21. Oleynikova H, Burri M, Taylor Z, et al. Continuous-time trajectory optimization for online UAV re-planning. In: 2016 IEEE/RSJ international conference on intelligent robots and systems (IROS), Daejeon, South Korea, 2016, pp. 5332–5339. Daejeon, South Korea: IEEE.

22. Gao F, Lin Y and Shen S. Gradient-based online safe trajectory generation for quadrotor flight in complex environments. In: 2017 IEEE/RSJ international conference on intelligent robots and systems (IROS), Vancouver, BC, 2017, pp. 3681–3688. Vancouver, BC: IEEE.

23. Tordesillas J, Lopez BT, Carter J, et al. Real-time planning with multi-fidelity models for agile flights in unknown environments. In: 2019 international conference on robotics and automation (ICRA), Montreal, QC, Canada, 2019, pp. 725–731. Montreal, QC: IEEE.

24. Tordesillas J, Lopez BT and How JP. FASTER: fast and safe trajectory planner for flights in unknown environments. In: 2019 IEEE/RSJ international conference on intelligent robots and systems (IROS), Macau, China, 2019, pp.1934–1940. Macau, China: IEEE.

25. Oleynikova H, Taylor Z, Siegwart R, et al. Safe Local Exploration for Replanning in Cluttered Unknown Environments for Micro-Aerial Vehicles. IEEE Rob Autom Lett 2018; 3: 1474–1481.

26. Mellinger D and Kumar V. Minimum snap trajectory generation and control for quadrotors. In: 2011 IEEE
international conference on robotics and automation (ICRA), Shanghai, China, 2011, pp.2520–2525. Shanghai, China: IEEE.

27. Charles R, Adam B and Nicholas R. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In: Inaba M and Corke P (eds) Robotics research. Springer tracts in advanced robotics, vol 114, Berlin, Germany: Springer, 2016, pp.649–666.

28. Mellinger D, Kushleyev A and Kumar V. Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. In: 2012 IEEE international conference on robotics and automation, Saint Paul, MN, 2012, pp. 477–483. Saint Paul, MN: IEEE.

29. Deits R and Tedrake R. Efficient mixed-integer planning for UAVs in cluttered environments. In: 2015 IEEE international conference on robotics and automation (ICRA), Seattle, WA, 2015, pp.42–49. Seattle, WA: IEEE.

30. Zhou B, Gao F, Wang L, et al. Robust and efficient quadrotor trajectory generation for fast autonomous flight. IEEE Rob Autom Lett 2019; 4, pp.3529–3536.

31. Oleynikova H, Taylor Z, Fehr M, et al. Voxblox: incremental 3D Euclidean signed distance fields for on-board MAV planning. In: 2017 IEEE/RSJ international conference on intelligent robots and systems (IROS), Vancouver, BC, 2017, pp.1366–1373. Vancouver, BC: IEEE.

32. Kamel M, Burri M and Siegwart R. Linear vs nonlinear MPC for trajectory tracking applied to rotary wing micro aerial vehicles. arXiv preprint arXiv: 1611.09240, 2016.

33. Morgan Q, Ken C, Brian PG, et al. ROS: an open-source Robot Operating System. In: Proc. Open-source software workshop of 2009 international conference on robotics and automation (ICRA), 2009, Kobe, Japan: IEEE.

34. Howie Choset. Robotic motion planning: RRT’s, www.cs.cmu.edu/~motionplanning/lecture (2007, accessed 27 September 2015).