Trajectory Generation for Aerial Multicopters

Mark W. Mueller · Raffaello D'Andrea

Received: date / Accepted: date

Abstract The ability to generate feasible and safe trajectories is crucial for autonomous multicopter systems. These trajectories can ideally be generated on low-cost, embedded computational hardware, and exploit the system's full dynamic capabilities while satisfying constraints. As operations increasingly focus on operation at high speeds, or in dynamically changing environments, strategies are required that can rapidly plan and re-plan trajectories. This article reviews typical approaches for trajectory generation of aerial robots, with a focus on multicopters, and discusses various approximations that may be used to make the problem more tractable. The strategy of planning in higher derivatives of the vehicle position (such as acceleration, jerk, and snap) is discussed in depth. We also discuss the related issue of expressing system limitations and constraints in these derivatives. Finally, possible future directions are discussed.

Keywords Multicopters · UAV · Planning · Differential flatness

1 Introduction

Aerial robots are increasingly routinely used in everyday operations, accomplishing tasks such as remote sensing, surveillance, and delivery of goods and people. Continuing technological development, especially of energy storage, and development of the regulatory environment means that such systems may soon be commonplace. A crucial requirement for their autonomous operations is the ability to plan motions to achieve goals, especially trajectories moving them through space. Due to their mechanical simplicity, and the ability to hover in place when required, the most common aerial robots are multicopters, especially quadcopters.

A typical trajectory generation problem consists of moving a single multicopter from an initial state (typically described as a position, velocity, orientation, and angular velocity) to a final state. The final state may be as detailed as the initial state, or it may only specify some components (such as an emergency stopping trajectory that requires simply zero final velocity). The resulting trajectory must respect the system dynamics, and potentially avoid additional constraints, such as collisions. The satisfaction of system dynamics and constraints is to be understood as satisfying some sufficiently accurate approximation of the true system capabilities (that is, sufficiently accurate for the problem at hand). Moreover, the available resources (e.g., computational power and computation time) are typically important considerations, as is the ability to accurately model the system and disturbances.

This article focuses on model-based approaches, that is, generation methods that rely on first-principle models of the multicopter systems to generate trajectories. An additional focus is on low computational complexity, so that trajectories may be computed on constrained hardware.

Mark W. Mueller (corresponding author)
University of California, Berkeley
E-mail: mwm@berekley.edu

Raffaello D'Andrea
ETH Zurich
E-mail: rdandrea@ethz.ch

Published in *Encyclopedia of Systems and Control* , 2019
https://doi.org/10.1007/978-1-4471-5102-9_100037-1

2 Dynamics and low-level control

We briefly review the equations of motions governing the motion of a multicopter, using the example of a quadcopter with propellers located symmetrically about the center of mass, with fixed, parallel axes of rotation. The model may be easily extended to other classes of multicopter, such extensions are briefly discussed.

The vehicle body's six degrees of freedom are captured in the position vector $\boldsymbol{x} = (x_1, x_2, x_3)$, expressed in an inertial frame, and the rotation matrix $\boldsymbol{R}$ that relates the body-fixed and inertial frames. The compact notation $\boldsymbol{x} = (x_1, x_2, x_3)$ is used to express the elements of a column vector. These evolve over time as functions of the translation velocity $\boldsymbol{v}$ and angular velocity $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3)$ as

$$\dot{\boldsymbol{x}} = \boldsymbol{v}, \quad \dot{\boldsymbol{R}} = \boldsymbol{R}\,\boldsymbol{S}(\boldsymbol{\omega}) \tag{1}$$

where $\boldsymbol{S}(\boldsymbol{a})$ is the skew-symmetric matrix version of the cross product, so that $\boldsymbol{a} \times \boldsymbol{b} = \boldsymbol{S}(\boldsymbol{a})\,\boldsymbol{b}$ for any vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.

Considering for simplicity a quadcopter, as illustrated in Fig. 1: the vehicle is actuated by four propeller forces f_i acting at displacements $\boldsymbol{r}_i$ from the center of mass. The propellers point along the body-fixed $\boldsymbol{e}_3$ direction, and produce a pure moment τ_i about their axes of rotation in addition to the force. This moment is typically assumed to be proportional to the force produced. The governing differential equations are then given as below

$$m\boldsymbol{a} = \boldsymbol{R}\,\boldsymbol{e}_3 f_\Sigma + m\boldsymbol{g} + \boldsymbol{f}_{\mathrm{aero}} \tag{2}$$
$$\boldsymbol{J}\dot{\boldsymbol{\omega}} = -\,\boldsymbol{S}(\boldsymbol{\omega})\,\boldsymbol{J}\boldsymbol{\omega} + \sum_i \left(\boldsymbol{S}(\boldsymbol{r}_i)\,\boldsymbol{e}_3 f_i + \boldsymbol{e}_3\tau_i\right) + \boldsymbol{\tau}_{\mathrm{aero}} \tag{3}$$

with $\boldsymbol{a}$ the acceleration of the vehicle, $\boldsymbol{g}$ the acceleration due to gravity as expressed in the inertial coordinate frame, and $\boldsymbol{f}_{\mathrm{aero}}$ aerodynamic forces in addition to the simple propeller model (such as propeller in-plane forces). Furthermore, $\boldsymbol{J}$ is the mass moment of inertia, and $\boldsymbol{\tau}_{\mathrm{aero}}$ any aerodynamic torques acting on the system in addition to those captured by the propeller model. We use the short-hand $f_\Sigma = \sum_i f_i$ for the total thrust force.

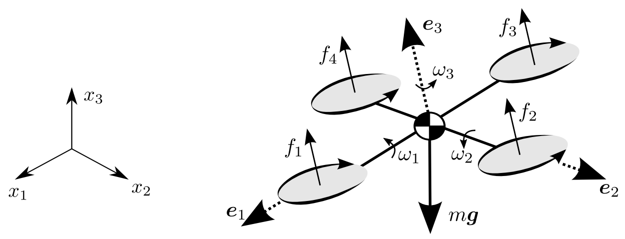

Fig. 1 Dynamic model of a quadcopter, acted upon by gravity $\boldsymbol{g}$, a thrust force f pointing along the (body-fixed) axis $\boldsymbol{e}_3$; and rotating with angular rate $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3)$, with its position in inertial space given as (x_1, x_2, x_3).

Multicopters with more than four propellers with parallel thrust directions will have the same equations as above, except that the summation will be over more than four propellers. Vehicles with thrust directions that are not parallel also exist, and such designs may have various benefits. For quadcopters, this may improve control authority and energy efficiency (Holda et al. 2018). When using more than four propellers, this reduces (or completely eliminates) the system's underactuation, see e.g., the hexacopter of Jiang & Voyles (2014) with improved disturbance rejection, or the omnidirectional vehicle of Brescianini & D'Andrea (2018). Vehicles with fewer than four propellers can also be constructed, but require special considerations especially with regards to attitude control, for a discussion see e.g. Mueller & D'Andrea (2016).

The motor forces vary as a function primarily of the respective propeller angular velocity, and may be accurately modelled as varying proportionally to the angular velocity squared, though more detailed models exist considering, e.g., relative airspeed. The forces are typically assumed to vary instantaneously, though the dependence on angular velocity of the propellers implies that an instantaneous change is impossible for actual systems with finite motor torques.

The individual motor forces are typically constrained as $f_i \in [f_{\min}, f_{\max}]$, with the upper limit $f_{\max}$ following from the maximum torque characteristics of the motors, and the lower limit $f_{\min}$ following, e.g., from the capacity of the speed controllers driving the motors to operate at low rotational speeds, so that usually $f_{\min} > 0$. For brushed motors, as is common on low-cost and inexpensive systems, typically $f_{\min} = 0$, while for systems with variable pitch or reversible propellers we may have $f_{\min} < 0$.

Closed-loop control is commonly achieved with nested control, especially by considering the angular velocity of the body as a high-bandwidth subsystem, and treating a target angular velocity as instantaneously achievable. This simplification, similar to the assumption of instantaneously achievable propeller forces, is here justified by noting that the angular velocity is fully actuated, so that the nonlinear terms may be compensated by feedback linearization, and that the achievable angular accelerations are very large due to the vehicle's low mass moment of inertia (as mass is typically concentrated at the center) and high torques (due to large moment arms).

3 Trajectory generation

For a general dynamic system, the goal of trajectory generation is a control input sequence, which would move it from an initial state in order to achieve a goal. This is typically posed as an optimization problem, minimizing some cost function. As the system dynamics are expressed as differential equations, the problem is most directly stated in continuous-time searching over the space of functions. Alternatively, the system dynamics may be approximated in discrete time, so that the optimization is now over a finite dimensional space. In both cases closed-form solutions are difficult to solve for, and general problems typically can only be solved through numerical approaches. As these generic problems tend to be non-convex and non-linear, brute-force numerical solutions tend to struggle, especially when computational power is limited. However, by exploiting the dynamic characteristics of multicopters, various transformations are possible that make the problem more easily solvable.

A key factor in the difficulty of finding a solution is the level of abstraction applied to the system dynamics. This is often implicitly done through low-level control loops, where certain states are treated as responding instantaneously to track desired values. The lowest level inputs typically considered are the electric signals to the motors, e.g., as a voltage level applied over a motor. When the aerial vehicle is equipped with high-performance motor speed controllers, the motor speeds (equivalently, the propeller forces) may be used as input; the next level up assumes sufficiently fast angular dynamics to allow the angular velocity to be used as input. Coarser models still consider the system as a double integrator, with acceleration as input (assuming, effectively, instantaneous changes in attitude), or most crudely as a single integrator with translational velocity as input. Increasingly coarse models have multiple advantages: they reduce the dimension of the planning system's state space (leading to smaller problems that are easier and faster to solve), they may hide nonlinearities (e.g., treating angular acceleration as input rather than torque avoids the nonlinearities in the angular velocity dynamics), and they may hide constraints (e.g., low-level input constraints are not visible if planning is done at higher levels).

A very successful approach for multicopters is to plan in some derivative of the position and consider each translational axis independently, thus solving three trajectories, each substantially less complicated. An early example of this kind of approach is given in Kalmár-Nagy et al. (2004), applied to a surface robot. The differential flatness properties of multicopters means that the actual inputs and states can be recovered straightforwardly from these position derivatives, though for uniqueness a rotation about the thrust axis is also required. A discussion on the differential flatness of multicopters is given in Mellinger & Kumar (2011), while general discussion on flatness may be found in Fliess et al. (1995) or Murray et al. (1995). Due to the superlinear complexity scaling of most optimization problems in the problem size, this spatial decomposition typically leads to a dramatic reduction in the required computational time: an example of quadcopter trajectory generation using this approach coupled with time discretization is given in Mueller & D'Andrea (2013). An obvious concern with spatial decoupling is the difficulty of decoupling constraints acting on the system – the decoupling of input constraints is discussed below, and obstacles are unlikely to present as neatly decoupled along user-specified directions.

3.1 Decoupled planning and constraints

When a decoupling approach is used, it is necessary to relate the planning states to the allowable inputs. A first requirement may be that the total thrust along the trajectory does not exceed the maximum available value. From (2), we have

$$|f_\Sigma|^2 = \|m\boldsymbol{a} - m\boldsymbol{g} - \boldsymbol{f}_{\text{aero}}\|^2 \tag{4}$$

Assuming, for simplicity, that the aerodynamic disturbances are negligible, the thrust bound, through (4), may be encoded per-axis by specifying conservative box constraints: Let, e.g., $\bar{a}_i$ be such that $\left(\sum_i \left(\bar{a}_i^2\right)\right)^{\frac{1}{2}}$ is less than the maximum allowed thrust, then the following decoupled bounds guarantee total thrust feasibility:

$$-\bar{a}_i \leq \boldsymbol{e}_i^T \boldsymbol{a} \leq \bar{a}_i \tag{5}$$

Ensuring that the trajectory does not violate lower-bounds on the thrust (relevant when $f_{\min}$ is strictly positive) is more difficult. This may be achieved by specifying an additional lower bound on the vehicle's acceleration along the gravitational direction:

$$\boldsymbol{e}_3^T \boldsymbol{a} \geq -\|\boldsymbol{g}\| + \frac{1}{m}\sum_i f_{\min} \tag{6}$$

This constraint is very conservative, as it effectively limits the tilt of the vehicle's thrust axis and does not, for instance, allow the vehicle to rotate its thrust axis to point downwards.

Limiting the trajectory jerk $\boldsymbol{j} = \dot{\boldsymbol{a}}$ allows for applying bounds on the the angular velocity and change in total thrust along the motion, but this relationship is substantially more complex than the acceleration. Taking the time derivative of the translational dynamics (2) and neglecting for simplicity changes in the aerodynamic forces yields

$$m\boldsymbol{R}^T\boldsymbol{j} = -\boldsymbol{S}(\boldsymbol{e}_3)\,\boldsymbol{\omega} f_\Sigma + \boldsymbol{e}_3\dot{f}_\Sigma = \begin{bmatrix}-\omega_2 f_\Sigma \\ \omega_1 f_\Sigma \\ \dot{f}_\Sigma\end{bmatrix} \tag{7}$$

From this equation it is clear that a simple bound on the jerk (either per-axis, or total) is insufficient to bound the angular velocity along the trajectory unless additionally a minimum bound on the thrust is specified.

To make statements regarding individual motor forces, an additional derivative of position is required, the snap $\boldsymbol{s}$. Taking the derivative of (7) gives

$$\begin{aligned} m\boldsymbol{R}^T\boldsymbol{s} =& \boldsymbol{S}(\boldsymbol{\omega})^2\,\boldsymbol{e}_3 f_\Sigma + \boldsymbol{S}(\boldsymbol{\omega})\,\boldsymbol{e}_3\dot{f}_\Sigma - \boldsymbol{S}(\boldsymbol{e}_3)\,\dot{\boldsymbol{\omega}} f_\Sigma \\ &+ \boldsymbol{S}(\boldsymbol{\omega})\,\boldsymbol{e}_3\dot{f} + \boldsymbol{e}_3\ddot{f}_\Sigma \end{aligned} \tag{8}$$

where the angular dynamics of (3) can be substituted for $\dot{\boldsymbol{\omega}}$ to relate motor forces to snap. As before, a bounded (per-axis or total) snap does not imply that the system's states or inputs remain bounded – one example is that the angular acceleration component parallel along the body's $\boldsymbol{e}_3$ axis may become unbounded whilst the snap remains bounded. A more subtle example is of a vehicle starting at rest at time $t = 0$, with acceleration trajectory $\boldsymbol{a}(t) = \boldsymbol{g}t$. In such a motion, the vehicle starts at rest with $f_\Sigma = m\,\|\boldsymbol{g}\|$, at $t = 1$ is in free-fall, and at $t = 2$ is thrusting downwards with $f_\Sigma = m\,\|\boldsymbol{g}\|$. For this motion the total thrust requirements are benign (ranging between zero and the hover thrust), the jerk is constant at $\boldsymbol{j} = \boldsymbol{g}$, and the snap is identically zero. However, this is a physically impossible trajectory, as the thrust direction at $t = 1$ must rotate instantaneously from pointing vertically upwards to pointing downwards. Thus, naïvely planning trajectories with bounded position derivatives may result in motions requiring unbounded motor forces, or unbounded angular velocities. Ensuring that the forces remain bounded would require simultaneously considering all axes, thereby negating the main benefit of the original decoupling. However, the above issues may not be common in many applications, and planning separately in position derivatives has been very successful with wide application in the literature.

3.2 Closed-form solutions

Under certain circumstances, closed-form solutions can be found for trajectories. Such solutions are typically computationally inexpensive to generate, but are restricted to work in specific circumstances. These kinds of trajectories naturally lend themselves to being used as motion primitives, which can be concatenated together to achieve complex goals. Moreover, if they are sufficiently inexpensive to compute, they may be used in sampling-based approaches where their computational complexity is traded off against their specialized nature.

An example of exploiting computationally cheap primitives is given in Mueller et al. (2015): here, a quadcopter trajectory primitive is proposed for state-interception trajectories (that is, with fixed final times) that minimize the average jerk along the motion. From (7) it can be seen that minimizing jerk along a motion can be interpreted as reducing the product of angular velocity and total thrust; intuitively leading to qualitatively "nice" trajectories. The closed-form solution of the trajectories reduces to a simple affine transformation of the initial/final states; and feasibility (interpreted here as collisions with planar objects in space; minimum/maximum total thrust values; and maximum angular velocity) is evaluated using a recursive scheme. In total, a motion primitive can be generated and evaluated for feasibility in approximately 1μ s on a laptop computer, or 100 μ s on a simple micro-controller. This computational speed then immediately lends itself to searching over complex spaces, and operation in very dynamic situations.

3.3 Numerical solutions

Other approaches to trajectory generation rely on numerical optimization tools. Such numerical solutions can be posed directly on the nonlinear aerial robot dynamics, and thus avoid the complexities of spatial decomposition, etc. A disadvantage of such approaches tends to be substantially higher computational loads, though tools such as the ACADO toolbox (Houska et al. 2011) allow for MPC implementations that can be run on embedded hardware.

Because the problems can be posed in a generic fashion, it is easy to include other objectives into the trajectory generation problem. One example of this is to include knowledge of the multicopter's sensors, and generate motions that not only satisfy feasibility constraints related to inputs and collisions, but also ensure that motions are informative to the system's sensors. An example that optimizes for visibility of features to assist in visual localization is given by Falanga et al. (2018), or for optimizing exploration gain in Papachristos et al. (2017).

An example using a specialized numerical solver to compute minimum time trajectories is given in Hehn et al. (2012), specifically used to argue about fundamental dynamic limits of quadcopters.

3.4 Computational speed and receding horizon control

If computing a trajectory requires a substantial time, compared to the system dynamics, the trajectory is typically used as a reference trajectory tracked by a separate feedback controller. However, if the computation is sufficiently fast, trajectories may be continuously replanned as the system moves. Such receding horizon implementations have multiple performance advantages, and allow systems to naturally react to disturbances, or to respond in changes in objective (e.g., the sudden appearance of an obstacle) – see, e.g., Chen et al. (2016) and Nägeli et al. (2017).

4 Summary and future directions

Trajectory generation for multicopters benefits from the same technological advancements that enable autonomy for an ever-increasing set of robots, primarily increases in computational power (and a reduction in mass, and power requirements, for computation), and improvements in generic numerical optimization tools. When sufficient computational power is available, this allows solving relatively straight-forward encodings of problems, including constraints on inputs and states. Where computational power is constrained, or time is of extreme importance, it is often useful to create methods that rely on specific characteristics of multicopters, such as the spatial decoupling discussed earlier.

A relatively open area of future work is to more tightly couple the trajectory planning component of autonomy to the sensing systems. Typical state-of-the-art approaches decouple the planning problem from the estimation problem: for example, a map of obstacles may first be created, and this map is then used for planning in a second step. This approach is very good at isolating complexity, and follows typical engineering practise to ‘divide and conquer’ complex problems. However, much like designing a state feedback controller independently of an estimator to regulate a plant, such a decomposed approach may have a substantial performance penalty compared to a unified approach when uncertainty is taken into account (consider $\mathcal{H}_\infty$ output-feedback control vs. LQG control – see, e.g., Zhou & Doyle (1998)). This may take the form of planning trajectories directly based on the sensors; e.g., planning motions in a laser point-cloud. If combined with low computational complexity methods, this may lead to very responsive behavior that naturally takes uncertainty in the sensors and system into account.

Another area of relevance for future work is to plan trajectories that take energy and power consumption into account. This is of particular importance for multicopters, as a primary constraint to their widespread use is limitations in range and payload due to modest energy budgets. Examples exploring this include Ware & Roy (2016) and Tagliabue et al. (2019). Trajectories for multiple vehicles operating simultaneously are also challenging, due to increasing size of the state space and mutual collision avoidance; for large collections of vehicles this remains an open problem. Another useful direction would be strategies that allow an operator to directly specify safety objectives (e.g., by parametrizing the risk to third parties from certain maneuvers).

5 Cross-References

– Differential Geometric Methods in Nonlinear Control
– Iterative Learning Control
– Model Predictive Control
– Underactuated Robots
– Unmanned Aerial Vehicles

6 Recommended reading

A good overview of convex optimization can be found in Boyd & Vandenberghe (2004), and a discussion of tools specifically for embedded application is given by Ferreau et al. (2017).

A good overview of multicopter modeling and control is given by Mahony et al. (2012), with more specialized models used for e.g. fault tolerance in Mueller & D’Andrea (2016).

Learning may be applied to aid in trajectory tracking, some examples of this use frequency based methods Hehn & D’Andrea (2014) or deep neural nets Zhou et al. (2017).

References

Boyd, S. & Vandenberghe, L. (2004), *Convex optimization*, Cambridge university press.

Brescianini, D. & D’Andrea, R. (2018), ‘An omnidirectional multirotor vehicle’, *Mechatronics* **55**, 76–93.

Chen, J., Liu, T. & Shen, S. (2016), Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments, *in* ‘2016 IEEE International Conference on Robotics and Automation (ICRA)’, IEEE, pp. 1476–1483.

Falanga, D., Foehn, P., Lu, P. & Scaramuzza, D. (2018), PAMPC: Perception-aware model predictive control for quadrotors, *in* '2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 1–8.

Ferreau, H. J., Almér, S., Verschueren, R., Diehl, M., Frick, D., Domahidi, A., Jerez, J. L., Stathopoulos, G. & Jones, C. (2017), 'Embedded optimization methods for industrial automatic control', *IFAC-PapersOnLine* **50**(1), 13194–13209.

Fliess, M., Lévine, J., Martin, P. & Rouchon, P. (1995), 'Flatness and defect of non-linear systems: introductory theory and examples', *International journal of control* **61**(6), 1327–1361.

Hehn, M. & D'Andrea, R. (2014), 'A frequency domain iterative learning algorithm for high-performance, periodic quadrocopter maneuvers', *Mechatronics* **24**(8), 954–965.

Hehn, M., Ritz, R. & D'Andrea, R. (2012), 'Performance benchmarking of quadrotor systems using time-optimal control', *Autonomous Robots* **33**, 69–88.

Holda, C., Ghalamchi, B. & Mueller, M. W. (2018), Tilting multicopter rotors for increased power efficiency and yaw authority, IEEE, pp. 143–148.

Houska, B., Ferreau, H. & Diehl, M. (2011), 'ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization', *Optimal Control Applications and Methods* **32**(3), 298–312.

Jiang, G. & Voyles, R. (2014), A nonparallel hexrotor uav with faster response to disturbances for precision position keeping, *in* '2014 IEEE International Symposium on Safety, Security, and Rescue Robotics (2014)', IEEE, pp. 1–5.

Kalmár-Nagy, T., D'Andrea, R. & Ganguly, P. (2004), 'Near-optimal dynamic trajectory generation and control of an omnidirectional vehicle', *Robotics and Autonomous Systems* **46**(1), 47–64.

Mahony, R., Kumar, V. & Corke, P. (2012), 'Aerial vehicles: Modeling, estimation, and control of quadrotor', *IEEE Robotics & Automation Magazine* **19**(3), 20–32.

Mellinger, D. & Kumar, V. (2011), Minimum snap trajectory generation and control for quadrotors, *in* 'IEEE International Conference on Robotics and Automation (ICRA)', pp. 2520–2525.

Mueller, M. W. & D'Andrea, R. (2013), A model predictive controller for quadrocopter state interception, *in* 'European Control Conference', pp. 1383–1389.

Mueller, M. W. & D'Andrea, R. (2016), 'Relaxed hover solutions for multicopters: Application to algorithmic redundancy and novel vehicles', *The International Journal of Robotics Research* **35**(8), 873–889.

Mueller, M. W., Hehn, M. & D'Andrea, R. (2015), 'A computationally efficient motion primitive for quadrocopter trajectory generation', *IEEE Transactions on Robotics* **31**(6), 1294–1310.

Murray, R. M., Rathinam, M. & Sluis, W. (1995), Differential flatness of mechanical control systems: A catalog of prototype systems, *in* 'ASME International Mechanical Engineering Congress and Exposition'.

Nägeli, T., Alonso-Mora, J., Domahidi, A., Rus, D. & Hilliges, O. (2017), 'Real-time motion planning for aerial videography with dynamic obstacle avoidance and viewpoint optimization', *IEEE Robotics and Automation Letters* **2**(3), 1696–1703.

Papachristos, C., Khattak, S. & Alexis, K. (2017), Uncertainty-aware receding horizon exploration and mapping using aerial robots, *in* '2017 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 4568–4575.

Tagliabue, A., Wu, X. & Mueller, M. W. (2019), Model-free online motion adaptation for optimal range and endurance of multicopters, *in* 'IEEE International Conference on Robotics and Automation (ICRA)'.

Ware, J. & Roy, N. (2016), An analysis of wind field estimation and exploitation for quadrotor flight in the urban canopy layer, *in* 'Robotics and Automation (ICRA), 2016 IEEE International Conference on', IEEE, pp. 1507–1514.

Zhou, K. & Doyle, J. C. (1998), *Essentials of robust control*, Vol. 104, Prentice hall Upper Saddle River, NJ.

Zhou, S., Helwa, M. K. & Schoellig, A. P. (2017), Design of deep neural networks as add-on blocks for improving impromptu trajectory tracking, *in* '2017 IEEE 56th Annual Conference on Decision and Control (CDC)', IEEE, pp. 5201–5207.